



Provided by the author(s) and University of Galway in accordance with publisher policies. Please cite the published version when available.

Title	Improving flow in large software product development organizations: A sensemaking and complex adaptive systems perspective
Author(s)	Power, Ken
Publication Date	2019-08-23
Publisher	NUI Galway
Item record	http://hdl.handle.net/10379/15498

Downloaded 2024-04-25T17:29:41Z

Some rights reserved. For more information, please see the item record link above.





**Improving Flow in Large Software Product
Development Organizations: A Sensemaking and
Complex Adaptive Systems Perspective**

by

Ken Power

Research Supervisor: Professor Kieran Conboy

A thesis submitted in fulfillment of the requirements for the degree of

Doctor of Philosophy

of the

National University of Ireland

Department of Business Information Systems

J.E. Cairnes School of Business & Economics

College of Business, Public Policy, & Law

National University of Ireland, Galway

Submission Date: August 2019

Table of Contents

Table of Contents	i
List of Figures	v
List of Tables	vii
Declaration	viii
Abstract	ix
Acknowledgements	xi
Chapter 1 Introduction	1
1.1 Introduction and Chapter Layout.....	1
1.2 Background and Context	1
1.2.1 Lean Product Development.....	1
1.2.2 Contemporary Software development	4
1.2.3 Flow-Based Software Development.....	6
1.2.4 Impediments to Flow	9
1.2.5 Sensemaking and Complex Adaptive Systems.....	11
1.2.6 Focus of this Research Study.....	14
1.3 Research Objective.....	16
1.4 Research Method.....	17
1.5 Thesis Structure.....	19
1.6 Summary	20
Chapter 2 Contemporary Software Development	22
2.1 Introduction and Chapter Layout.....	22
2.2 Lean Product Development.....	22
2.2.1 The Toyota Production System	22
2.2.2 Principles of Lean Product Development.....	25
2.2.3 Contemporary Lean Thinking.....	29
2.3 Contemporary Software Development.....	30
2.3.1 The Emergence of Agility in Software Development	30
2.3.2 Continued Evolution of Software Development.....	33
2.3.3 Extending the Feedback Loop.....	34
2.3.4 Characteristics of Contemporary Software Development.....	35
2.4 Flow-Based Software Development.....	37
2.4.1 Considerations for Flow-Based Software Development	37
2.4.2 Flow Units.....	42
2.4.3 Flow States	42
2.5 Impediments to Flow	44
2.5.1 Identifying Impediments to Flow.....	46
2.5.2 Contributing Factors to Impediments to Flow.....	52

2.5.3	Effects of Impediments.....	57
2.6	Summary	59
Chapter 3	Metrics for Understanding Flow and Impediments.....	63
3.1	Introduction and Chapter Layout.....	63
3.2	Selecting Metrics for this Study.....	63
3.3	Metrics for Understanding Flow and Impediments	64
3.3.1	Throughput.....	65
3.3.2	Cycle Time	67
3.3.3	Cumulative Flow.....	67
3.4	Summary	68
Chapter 4	Software Development Organizations	71
4.1	Introduction and Chapter Layout.....	71
4.2	Complexity in Software Development Organizations	71
4.2.1	Organizations Operate in Complex Environments	71
4.2.2	Organizations are Complex Adaptive Systems	72
4.2.3	Properties of Complex Adaptive Systems	73
4.2.4	Dynamics at Different Levels of a System	77
4.2.5	Organizations are Impermanent Social Structures	78
4.2.6	Culture in Software Development Organizations is a CAS.....	80
4.3	Sensemaking in Software Development Organizations.....	83
4.3.1	Using Sensemaking to Understand Complex Systems	83
4.3.2	Properties of Sensemaking.....	85
4.3.3	Distributed Sensemaking across a Value Stream	88
4.3.4	Using Micronarrative as Cues for Sensemaking.....	89
4.3.5	Making Sense of Experiences in Organizations.....	94
4.3.6	The Cynefin Sensemaking Framework.....	98
4.4	Resolving Impediments in a Complex Adaptive System	102
4.4.1	Interventions in a Complex System.....	102
4.4.2	Approaches to Dealing with Change in Complex Systems	103
4.4.3	The Role of Experiments in Software Organizations.....	104
4.4.4	Experiments in the <i>Complex</i> Sensemaking Domain	105
4.4.5	System Improvement Goals	106
4.4.6	Interventions Shape the Patterns in a System	107
4.5	Summary	108
Chapter 5	Research Methods	109
5.1	Introduction and Chapter Layout.....	109
5.2	Research Objectives.....	109
5.3	Epistemological Stance.....	110
5.3.1	Philosophical Perspectives.....	110

5.3.2	Logic	113
5.3.3	Theory	115
5.3.4	Epistemological Stance for This Study.....	118
5.3.5	Self-Reflexivity.....	118
5.4	Research Methodologies.....	119
5.4.1	Quantitative and Qualitative Research Methods	119
5.4.2	Focus Group Research	121
5.4.3	Narrative Research.....	124
5.4.4	Case Study Research.....	127
5.4.5	Triangulation.....	130
5.5	Research Approach for This Study	130
5.5.1	Research Context.....	130
5.5.2	Research Phases	135
5.5.3	Literature Review	136
5.5.4	Use of Research Methods in this Study.....	137
5.6	Data Collection Methods.....	143
5.6.1	Collection of Focus Group Data	144
5.6.2	Collection of Micronarrative Data.....	145
5.6.3	Collection of Case Study Data.....	150
5.7	Data Analysis Methods	151
5.7.1	Analysis of Focus Group Data.....	152
5.7.2	Analysis of Micronarrative Data	152
5.7.3	Analysis of Case Study Data	153
5.7.4	Content Analysis.....	157
5.7.5	Coding Process Used in This Study.....	157
5.7.6	Purposive Sampling.....	161
5.8	Validity and Reliability.....	162
5.9	Summary	166
Chapter 6	Findings.....	162
6.1	Introduction and Chapter Layout.....	162
6.2	R01: Impediments to Flow.....	162
6.2.1	Categories of Impediments	163
6.2.2	Contributing Factors to Impediments.....	174
6.2.3	Effects of Impediments.....	187
6.3	R02: Impediment Management Framework.....	192
6.3.1	R02(i): Analyze System Patterns	193
6.3.2	R02(ii): Identify Impediments to Flow	213
6.3.3	R02(iii): Analyze Ways to Make Sense of Impediments	218
6.3.4	R02(iv): Explore Approaches to Resolving Impediments.....	229

6.4	Summary	250
Chapter 7	Discussion and Conclusions	252
7.1	Introduction and Chapter Layout.....	252
7.2	R01: Impediments to Flow.....	253
7.2.1	Categories of Impediments	253
7.2.2	Contributing Factors to Impediments.....	256
7.2.3	Effects of Impediments.....	257
7.3	R02: A Framework for Managing Impediments	258
7.3.1	R02(i): Analyze System Patterns	258
7.3.2	R02(ii): Identify Impediments to Flow	267
7.3.3	R02(iii): Analyze Ways to Make Sense of Impediments	276
7.3.4	R02(iv): Explore Approaches to Resolving Impediments.....	280
7.4	Contributions of this Study.....	288
7.4.1	Contributions to Research.....	289
7.4.2	Contributions to Practice	294
7.5	Limitations and Reflections	301
7.5.1	Limitations of the Study.....	301
7.5.2	Limitations of the Research Approach.....	303
7.6	Trajectories for Future Research	305
References	308

List of Figures

<i>Figure 1-1 Funnel showing narrowing of topics towards the research focus area</i>	15
<i>Figure 1-2 Relationship between research objectives and conceptual framework</i>	17
<i>Figure 2-1 A typical visualization of flow states</i>	43
<i>Figure 2-2 Workflow states represented as a Kanban board with WIP limits</i>	43
<i>Figure 2-3 Contributions to the conceptual framework from chapter 2</i>	62
<i>Figure 3-1 Industry trends in flow metrics usage from 2015 to 2019</i>	65
<i>Figure 3-2 Flow metrics and the levers through which to influence them</i>	70
<i>Figure 3-3 Contributions to the conceptual framework from chapter 3</i>	70
<i>Figure 4-1 The Cynefin sensemaking framework</i>	99
<i>Figure 4-2 Contributions to the conceptual framework from chapter 4</i>	109
<i>Figure 5-1 Validated conceptual framework for this study</i>	110
<i>Figure 5-2 Relationship between the three research methods</i>	121
<i>Figure 5-3 Case study design showing contexts and embedded units of analysis</i>	131
<i>Figure 5-4 Relationship of research methods, objectives, and conceptual framework</i> ..	138
<i>Figure 5-5 General approach for data collection in focus groups</i>	145
<i>Figure 5-6 Example triad used in micronarrative collection instrument</i>	147
<i>Figure 5-7 Example dyad used in micronarrative collection instrument</i>	148
<i>Figure 5-8 NVivo structure highlighting top-level coding structure</i>	160
<i>Figure 5-9 Complete set of codes used for data analysis</i>	161
<i>Figure 6-1 Chapter sections relating findings to the conceptual framework</i>	162
<i>Figure 6-2 Value Stream Map for ORG-A</i>	194
<i>Figure 6-3 Value Stream Map for ORG-E</i>	195
<i>Figure 6-4 Visualizing organization productivity using a throughput diagram</i>	196
<i>Figure 6-5 Value demand and failure demand as percentage of throughput</i>	196
<i>Figure 6-6 Visualizing organization responsiveness with cycle time</i>	197
<i>Figure 6-7 Visualizing organization predictability using CFDs</i>	198
<i>Figure 6-8 A CFD showing smooth flow of work and high organization productivity</i>	198
<i>Figure 6-9 A CFD showing too much WIP and low organization productivity</i>	199
<i>Figure 6-10 Cultural patterns identified through micronarrative sensemaking</i>	200
<i>Figure 6-11 Emotional sentiment associated with people’s experiences</i>	201
<i>Figure 6-12 Emotion of BU1 experiences with impediments and contributing factors</i> .	202
<i>Figure 6-13 Emotion of BU2 experiences with impediments and contributing factors</i> .	202
<i>Figure 6-14 Emotion of BU3 experiences with impediments and contributing factors</i> .	203
<i>Figure 6-15 People’s reasons for sharing their specific experiences</i>	204
<i>Figure 6-16 Impediments associated with experiences shared for celebration</i>	204
<i>Figure 6-17 Impediments associated with experiences that impact morale</i>	205
<i>Figure 6-18 Impact on use of resources, quality, and morale</i>	205
<i>Figure 6-19 Patterns of how recently the experiences occurred</i>	207

<i>Figure 6-20 Patterns of how frequently these kinds of experiences occur</i>	<i>208</i>
<i>Figure 6-21 Frequency of experiences with impediments and contributing factors</i>	<i>208</i>
<i>Figure 6-22 Impediments associated with experiences that occur all the time.....</i>	<i>209</i>
<i>Figure 6-23 Frequency of occurrence of experiences involving impediments</i>	<i>209</i>
<i>Figure 6-24 Understanding context of impediments through people's experiences</i>	<i>210</i>
<i>Figure 6-25 Impediments identified via micronarrative-based sensemaking analysis..</i>	<i>214</i>
<i>Figure 6-26 Visualizing the co-occurrence of multiple impediments.....</i>	<i>216</i>
<i>Figure 6-27 Understanding the relative impact of impediments.....</i>	<i>218</i>
<i>Figure 6-28 Impact of impediments and who needs to contribute to resolving them</i>	<i>219</i>
<i>Figure 6-29 Understanding who needs to be involved to resolve impediments.....</i>	<i>224</i>
<i>Figure 6-30 Roles in experiences involving impediments and contributing factors.....</i>	<i>225</i>
<i>Figure 6-31 Roles involved in positive and strongly positive experiences</i>	<i>225</i>
<i>Figure 6-32 Roles involved in negative and strongly negative experiences</i>	<i>226</i>
<i>Figure 6-33 Emotion of experiences involving different roles.....</i>	<i>226</i>
<i>Figure 6-34 Making sense of impediments using Cynefin domain mapping</i>	<i>228</i>
<i>Figure 6-35 Artifacts used to record continuous improvement goals.....</i>	<i>231</i>
<i>Figure 6-36 CDE interventions per sensemaking domain</i>	<i>247</i>
<i>Figure 7-1 Chapter sections relating discussion to the conceptual framework.....</i>	<i>252</i>
<i>Figure 7-2 Relative proportions of impediments coded in this study's findings</i>	<i>253</i>
<i>Figure 7-3 Summary of impediment categories impacting flow in 4 organizations</i>	<i>254</i>
<i>Figure 7-4 Contributing factors to impediments to flow in the organizations studied ..</i>	<i>256</i>
<i>Figure 7-5 Impacts of impediments to flow in the organizations studied</i>	<i>258</i>
<i>Figure 7-6 Categories of system patterns</i>	<i>260</i>
<i>Figure 7-7 Presence of system patterns when identifying specific impediment types</i>	<i>273</i>
<i>Figure 7-8 Comparing the impacts of impediments in two different companies</i>	<i>277</i>
<i>Figure 7-9 Influencing system patterns in ordered and unordered systems.....</i>	<i>288</i>

List of Tables

<i>Table 2.1 Evolution of the principles of lean software development</i>	40
<i>Table 2.2 Principles and practices of the Kanban Method</i>	41
<i>Table 2.3 Evolving categories of waste from TPS to lean software development</i>	47
<i>Table 2.4 Relating impediments to lean principles and steps to enable flow</i>	60
<i>Table 2.5 Common categories of impediments, contributing factors, and effects</i>	61
<i>Table 3.1 Summary of core metrics to understand flow and impediments</i>	69
<i>Table 5.1 Profile of the organizations that are the units of analysis for this study</i>	133
<i>Table 5.2 State of Lean Product Development principles in the organizations studied.</i>	134
<i>Table 5.3 Summary of the three research phases of this study</i>	135
<i>Table 5.4 Summary of focus groups conducted as part of this study</i>	139
<i>Table 5.5 Data collection tools used in this study</i>	144
<i>Table 5.6 Data analysis tools used in this study</i>	152
<i>Table 6.1 Relating the findings in each research phase to RO2 sub-objectives</i>	193
<i>Table 6.2 Examples of “more experiences like these”</i>	212
<i>Table 6.3 Examples of “fewer experiences like those”</i>	213
<i>Table 6.4 Summary of impediments identified from value stream mapping</i>	215
<i>Table 6.5 Summary of impediment categories that map to Cynefin domains</i>	229
<i>Table 6.6 Prioritization of categories of impediments</i>	229
<i>Table 6.7 Using continuous improvement goals to resolve impediments</i>	231
<i>Table 6.8 System improvement goals to resolve ORG-F impediments</i>	240
<i>Table 6.9 System improvement goals to resolve ORG-G impediments</i>	241
<i>Table 6.10 Container interventions in the complicated domain</i>	242
<i>Table 6.11 Difference interventions in the complicated domain</i>	242
<i>Table 6.12 Exchange interventions in the complicated domain</i>	243
<i>Table 6.13 Container interventions in the complex domain</i>	244
<i>Table 6.14 Difference interventions in the complex domain</i>	244
<i>Table 6.15 Exchange interventions in the complex domain</i>	245
<i>Table 6.16 Focus of CDE interventions in the complicated and complex domains</i>	248
<i>Table 7.1 Roles that impact flow and impediments in different states</i>	271
<i>Table 7.2 Relationship between impediments and system patterns</i>	273
<i>Table 7.3 Summary of focus areas for resolving impediments in each organization</i>	282
<i>Table 7.4 Impediment-resolving approaches, impediments, and Cynefin domains</i>	284
<i>Table 7.5 Impediment-resolving approaches, system patterns, and Cynefin domains</i> ...	286

Declaration

I hereby certify that this material, which I now submit for assessment on the program of study leading to the award of Doctor of Philosophy, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signature

Date

Abstract

Software development organizations operate in an environment of ever-increasing volatility, uncertainty, complexity, and ambiguity. The pace of change is accelerating, business and technology complexity is growing, and organizations are struggling to keep pace. The software development industry has a \$3 trillion productivity problem, according to one study. Value is not flowing as it should. Flow-based software development is part of the continued evolution of contemporary software development approaches contributing to addressing this problem. It builds on agile and lean software development approaches and incorporates lessons from Deming's management method, the Toyota Production System, Lean Product Development, Theory of Constraints, Operations Management, and other influences. Flow-based development is foundational to modern systems approaches, including DevOps, Continuous Delivery, Site Reliability Engineering, and more. Creating and sustaining flow in organizations is a challenging problem. Despite this, there is a lack of rigorous research on the topic of impediments to flow.

Stephen Hawking famously predicted that the 21st century would be the century of complexity science. Software development organizations are complex adaptive systems (CAS). Sensemaking provides a means of exploring and understanding complex systems through making sense of people's lived experiences in organizations. This study provides a sensemaking and CAS theory-informed understanding of product development flow in large software development organizations. To achieve its objective, this research studies ten large, globally distributed, software development organizations. Using the SenseMaker® software suite this study collects and analyzes micronarratives from people in these organizations, thereby providing insights into the patterns of organization culture that influence flow and impediments. The research methods for this qualitative study include a combination of focus group research, narrative research, and case study research. The unit of analysis is the software development organization.

The main contribution of this study is the development of an impediment management framework, related to which this study presents four sets of findings. First, this study provides an analysis of the system patterns that contribute to impediments to flow in organizations. In particular, this study analyzes system patterns identified from value streams, flow metrics, and organization culture. Second, this study explores factors related to how organizations identify impediments to flow. This study identifies eight common types of impediments that affect flow. The study further identifies nine common contributing factors to impediments and six common effects of impediments. Third, this study analyzes how people in organizations make sense of impediments. This research explores the use of the Cynefin sensemaking framework to make sense of the context of impediments in order to inform how to resolve them. Fourth, this study explores four approaches used by organizations to resolve impediments. In particular, this study explores the use of continuous improvement, A3 problem solving, experiment design, and system improvement goals in the context of resolving impediments and improving flow in organizations. This study employs the CDE (*containers, differences, exchanges*) framework to understand better the interventions that organizations make in order to resolve impediments and improve flow. The analysis shows how these interventions relate to the sensemaking domains of the Cynefin framework.

Ultimately, this study shows impediments continually emerge to affect flow in organizations. Some impediments are problems to be solved; others are indicators of patterns that need to shift in the organization. The quality of flow and the nature of the organization's response to impediments reflects the organization's culture. Improving flow requires attending to the patterns, identifying impediments, making sense of the impediments in the context of the system patterns, and taking context-appropriate action in the form of interventions to resolve impediments. This study adds to the body of knowledge of contemporary software development approaches, flow-based development, management, leadership, and sensemaking in organizations. The framework can guide practical approaches to improving flow in organizations and serves as a foundation for future research.

Acknowledgements

A PhD doesn't happen in a vacuum; there's a life within which it has to find a place. That feels particularly true for a part-time PhD student. I could not have crossed the finish line without the support of a large group of people who believed in and supported me throughout this epic adventure.

First, thank you to my wife Tina, and our three daughters, Kayleigh, Hannah, and Gráinne for making everything worthwhile, for all of the unconditional support and love, and coffee breaks and hugs, throughout the years of this PhD journey, and for helping me to find balance. I could not have done any of this without you! And, of course, special thanks to Bouncer, our Bichon, who kept me company on many early-morning writing sessions.

Huge thanks to Kieran Conboy, my PhD supervisor, for support, guidance, advice, high standards, patience, friendship, lots of coffee, and for accommodating the unpredictable schedule of a part-time, industry-based PhD student with a growing family, a demanding day job, and a full life. Thank you to my Graduate Review Committee at NUI Galway for all the support, guidance, advice, and encouragement over the years, particularly Lorraine Morgan, Tom Acton, Willie Golden, and Eoin Whelan. Thanks to the Lero group (Irish Software Research Center) in NUI Galway, led by Kieran, for being an inspiring and supportive incubator of ideas and great research, and for many PhD support coffee mornings over the years. Particular thanks to Denis Dennehy for helpful conversations and collaboration on flow workshops with Lero, and Eoin Cullina and Roger Sweetman for all those PhD writing support sessions that helped get me unstuck. Thanks to Linda Wulff for an excellent job of proofreading this thesis; any remaining errors are mine.

I had the opportunity to apply what I was learning during this study to my day job, and vice versa, which had benefits for both. Thank you to Tom Williams, my manager throughout this study, for always encouraging me, and helping me find ways to combine a part-time PhD with a demanding

fulltime day-job, and for approving my travel expenses ☺. Thanks to Tom and our leadership team, especially David Murray and Yvette Kanouff, for trusting me with a rare opportunity to be part of something big, and at the same time supporting my PhD journey. Thank you to our awesome team of Julie Kaefer, Eldad Yehoshafat, Jeff Beasley, Yosi Kossowsky, Joe Ira, John Brickell, Vic Morrella, and Tammy Holm. Our extended team of friends, collaborators, and co-conspirators Salah Jarrad, Ken Meagher, Steve Williams, Marc Dumonte, Corinne Lemercier, Cyril Boucher, Pierre Baleyrier, Pascal Butel, and Russell Bailey. Thanks to Tom Lambert and Todd Weber for being there when this all kicked off. Thank you all for the journey of learning we shared, the support along the way and for making work fun, interesting, and worthwhile.

Many people helped me and helped shape my thinking on these topics through inspiring conversations, support, friendship, encouragement, and collaboration throughout this journey. Particular thanks to Rebecca Wirfs-Brock, Esther Derby, Tony Quinlan, Olaf Lewitz, Don Gray, Diana Larsen, Karl Scotland, Don Reinertsen, Jean Tabaka (RIP), Stephanie Sackman, Heather Kanser, Elad Sofer, Ilan Kirschenbaum, Lior Friedman. Thanks to Glenda Eoyang and Dave Snowden for contributing valuable work to the world, and then teaching me about it. Your work on complex adaptive systems, Human Systems Dynamics, sensemaking, Cynefin, and more forms a core part of the foundation of this study, and I hope I do it justice. Thanks to Johanna Rothman, Jutta Eckstein, and Rod Nord for shepherding several of my Agile Alliance experience reports and helping me improve at this writing thing.

Many thanks to the companies and software development organizations that participated in this study and shared your experiences. Thank you to the thousands of people I had the pleasure of meeting, knowing, and working with around the world during this PhD journey. You all challenged me, taught me, supported me, and helped me to grow. Thanks to the agile, lean, and software architecture communities, particularly the XP, Agile, Lean-Kanban, ALE, SATURN, Global Scrum Gathering, and OOP conference

series. I have been attending and participating in these conferences for several years, presenting and refining the ideas in this thesis. The organizers and participants are among the brightest, generous, and open-minded people I have encountered anywhere. These conferences helped develop my thinking and understanding around software development, agile, lean, leadership, flow, complexity, metrics, and much more. Thank you to the anonymous reviewers of papers I submitted over the years, even the papers that got rejected. All of your feedback helped me to grow.

Finally, thank you to my parents (RIP), John and Joan, who always encouraged and supported me, and instilled the importance and joy of learning that started me on this journey long before I dreamed about doing a PhD.

Ken Power, August 2019.

Chapter 1 Introduction

1.1 Introduction and Chapter Layout

This chapter provides a high-level overview of what this research study is about, the context within which this research occurs, and some pointers towards the research findings. Section 1.2 introduces the background and context that motivates this research. Section 1.3 presents the research objectives that this study aims to address. Section 1.4 summarizes the research method used to conduct this study. Section 1.5 describes the structure of the remainder of this thesis, briefly summarizing each of the remaining chapters. Finally, section 1.6 provides a summary of this chapter.

1.2 Background and Context

This section traces the progression from lean product development and its major influences (section 1.2.1), to contemporary software development (section 1.2.2), and flow-based development (section 1.2.3). Section 1.2.4 discusses impediments to flow. Section 1.2.5 discusses sensemaking and complex adaptive systems theory and their relevance to this study. Finally, section 1.2.6 summarizes the focus for this study, informed by the preceding discussion.

1.2.1 Lean Product Development

Deming's work with top industry managers from Japanese manufacturers, starting from 1950, is widely credited with contributing to Japan's rise from post-war crisis to world leaders in manufacturing (Deming, 2013). Deming credits a systems perspective as "*the spark that turned Japan around*" (Deming, 2013). Toyota emerged from this post-war period to become not just the most successful automotive manufacturer in the world, but the world's most successful industrial enterprise (Womack et al., 2007). Toyota credit the Toyota Production System (TPS) with its success. TPS, developed by Taichi Ohno, is a formalization of the practices and culture of Toyota (Ohno, 1988, Ohno, 2013, Shingo, 1989). TPS is sometimes referred to as

“*Thinking People System*” to emphasize the role of people in continually improving their work in order to add value to the product or service they are creating (Dickson et al., 2009).

Ohno encouraged everyone in the organization to think in terms of identifying and removing obstacles to their work, making the flow of work more efficient, and to focus on delivering customer value. People initially had trouble thinking in these terms and seeing obstacles, so Ohno characterized seven types of waste or *muda* (Ohno, 2013, Ohno, 1988). Ohno (1988) defines waste as “*all elements of production that only increase cost without adding value*”; Shingo (1989) defines waste as “*any activity that does not contribute to operations*”; Liker (2004) defines waste as “*activities that lengthen lead times, cause extra movement to get parts or tools, create excess inventory, or result in any type of waiting*”. These definitions are complimentary for the purposes of this study. Ohno (1988) explained the number seven comes from an old Japanese expression “*He without bad habits has seven,*” which Ohno used to reinforce the point that “*even if you think there’s no waste you will find at least seven types.*” The seven types are *overproduction, time on hand (waiting), waste in transportation, waste of processing itself, stock on hand (inventory), movement, and making defective products*. These are discussed in more detail in section 2.5.1. He also described overburden (*muri*) and unevenness in process or demand (*mura*) as the main contributors to waste (Ohno, 1988, Ohno, 2013). The goal was to improve flow by eliminating all forms of inefficiency or waste that did not add value to the product (Modig and Ahlstrom, 2013). Liker (2004) added an eighth waste, *unused employee creativity*, resulting in what has become known as the eight wastes of the TPS, or the eight wastes of lean. Shingo developed the idea of kaizen, or continuous improvement, as part of TPS to have a systemic process of developing a culture of improvement (Shingo, 2007). Deming introduced many concepts to Toyota, including the Plan-Do-Check-Act (PDCA) (Shewart, 1931) improvement process (Womack et al., 2007). Toyota developed the A3 process as a concrete structure to implement PDCA-based management as part of TPS (Sobek and Smalley, 2008, Sobek and

Jimmerson, 2006). This study explores the use of *continuous improvement* and *A3 problem solving* in the context of contemporary software development organizations working to improve flow.

Lean production was the name given to a detailed study of the new techniques developed at Toyota and other Japanese manufacturers (Womack et al., 2007). It is viewed as the next step in the evolution of manufacturing from craft production to mass production to lean production. Lean product development is about evolving a system of management that focuses on growing healthy, profitable, sustainable organizations through respecting people and nurturing relationships across the entire value stream. Lean product development focuses on optimizing the system of work and the environment in which the work is done through a relentless pursuit of continuous improvement (Womack et al., 2007). This focus is summarized in the five principles of lean: *specify value, identify the value streams, make the value-creating steps flow, pull, and perfection* (Womack and Jones, 2003). Section 2.2.2 describes these principles in more detail.

The linear nature of the product development process is a characteristic of early lean implementations. By the mid-1980s, the world of commercial new product development had become “*fast-paced, fiercely competitive*” and “*speed and flexibility*” were essential (Takeuchi and Nonaka, 1986). Companies were “*increasingly realizing that the old, sequential approach to developing new products simply won't get the job done.*” Takeuchi and Nonaka (1986) noted that organizations needed to move from a sequential model of product development to a more iterative model characterized by built-in instability, self-organizing project teams, overlapping development phases, “multilearning”, subtle control, and organizational transfer of learning. They used the metaphor of a relay race to describe the outdated sequential approach that most organizations were still taking towards product development. They encouraged organizations to adopt a rugby metaphor instead, where product teams move together as a unit. Lean Product Development Flow is an evolution of lean beyond manufacturing (Reinertsen, 1997, Smith and Reinertsen, 1998, Reinertsen, 2009, Thomke

and Reinertsen, 2012). Manufacturing systems have traditionally dealt with predictable and homogeneous flows (Reinertsen, 2009). Contemporary work on lean product development flow draws from domains such as telecommunications networks and computer operating systems as examples of systems with highly-variable and heterogeneous flows (Reinertsen, 2009). Healthcare is another domain that has successfully applied lean principles (Dickson et al., 2009, Jimmerson et al., 2005, Spear, 2005, Spear, 2006). These domains have greater resonance with the variability, unpredictability, and heterogeneity found in contemporary software product development, as will be discussed in the next section. Lean product development continues to influence, directly and indirectly, many contemporary trends in software development.

1.2.2 Contemporary Software development

Software development as a profession is still relatively young. Boehm (2006b) traces the history of software engineering, from the 1950s to the present. The 1950s were characterized by a focus on hardware engineering, with software being a secondary concern (Boehm, 2006b). Margaret Hamilton coined the term “software engineering” in an attempt to place software on a parity of esteem with hardware among engineers while working on the Apollo space program (Cameron, 2018). NATO sponsored the first software engineering conference in 1968 to address the growing complexity for the defense industries (Naur and Randell, 1968). Rigorous methods were created in an attempt to ensure reliability and quality. By the 1990s, demands for reliable, dependable, and scalable software systems continued to increase. Some organizations stood out as having developed or adopted effective patterns of organizing to deliver complex products and systems (Cain et al., 1996, Coplien, 1994, Harrison and Coplien, 1996), but these patterns were not widespread. Overall, the practices and methods that had evolved were not meeting the needs of software development organizations or their customers. Software processes had become overly heavyweight and bureaucratic and were seen as stifling rather than enabling innovation (Highsmith, 2002). Agile Software Development emerged as an

umbrella term in 2001 for a set of lightweight methods that were considered an alternative approach to the overly bureaucratic, wasteful, ineffective, sequential, and plan-driven waterfall processes that were dominant at the time (Highsmith, 2002). The industry responded by creating methods that were more lightweight and adaptive. These approaches included eXtreme Programming (XP) (Beck, 2000, Beck and Andres, 2005), Scrum (Schwaber and Beedle, 2002, Sutherland and Schwaber, 2013), and the Crystal family of methods (Cockburn, 2005), among others. The patterns of effective software development organizations were a significant influence on Scrum and XP (Coplien and Harrison, 2004). Takeuchi and Nonaka (1986) influenced the creators of both XP and Scrum, with the latter taking its name from that paper (Schwaber and Beedle, 2002). The creators of many of these methods convened in 2001 to create what became known as The Manifesto for Agile Software Development, or, simply The Agile Manifesto (Beck et al., 2001).

Agile development is now commonplace in contemporary software development (CollabNet VersionOne, 2019). The intervening two decades has seen agile development evolve from a new approach used by small, co-located teams to global organizations. More methods have emerged. The Kanban Method (Anderson, 2010, Burrows, 2014) and Lean Software Development (Hibbs et al., 2009, Poppendieck and Poppendieck, 2003) brought an explicit focus on applying principles and practices from TPS and lean product development to software development organizations. Lean Startup builds on a foundation of agile development, lean product development, and Open Source to define an innovation-focused process that has become relevant not just to startups but also to established companies, large enterprises, government departments and others (Ries, 2011, Maurya, 2012). Companies have figured out how to be agile at a large scale, with increasing size in organizations, number of locations, number of people, and ever-larger projects and products (Dingsøyr et al., 2019). There are now many frameworks, techniques, methodologies, and in-house practices proposing solutions for organizations working at such scale.

Practices such as continuous delivery (Humble and Farley, 2010), DevOps (Kim et al., 2014, Davis and Daniels, 2016, Forsgren et al., 2018a, Mann et al., 2018) and site reliability engineering (SRE) (Beyer et al., 2016) have bridged the gap between product development teams and operations teams. Improved capabilities for monitoring, observability, and feedback provide software development organizations with better insight into their products after they ship and enable a tighter feedback loop between software developers and customers. Many contemporary software systems are implemented as distributed systems with complex behavior and failure modes. Many software organizations are using experimentation to verify the reliability of such systems (Basiri et al., 2016). However, even though these approaches exist, not all software development organizations are using them (Forsgren et al., 2018a, CollabNet VersionOne, 2019). Widespread adoption of agile methods has not eliminated the challenges faced by organizations. Today, software development organizations operate in an environment of ever-increasing volatility, uncertainty, complexity, and ambiguity (Bartscht, 2015, Johansen and Euchner, 2013). The pace of change is accelerating. The technical complexity is increasing, as are the demands for reliable, dependable, and scalable software. Organizations are struggling to keep pace (Forsgren et al., 2018a, Stripe, 2018, CollabNet VersionOne, 2019).

1.2.3 Flow-Based Software Development

In an effort to keep pace with these demands software development organizations are evolving towards an increasingly continuous, flow-based, approach to developing and delivering software (Fitzgerald and Stol, 2017, Forsgren et al., 2018a). Short feedback cycles, continuous integration (Beck, 2000, Duvall et al., 2007), continuous delivery (Humble and Farley, 2010), continuous deployment, and continuous operations, are all seen as essential ingredients for high-performing organizations (Forsgren et al., 2018b). Software development organizations take an increasingly customer-centric perspective on products and services, and then organize the work so that it flows as smoothly as possible from the point at which an idea is conceived or a request is made, to the point at which that idea or request is delivering

value to its intended audience. Flow is increasingly important, with 67% of respondents to one survey reporting that value stream management is important for connecting their business needs to their software development capability (CollabNet VersionOne, 2019).

The ideas behind flow-based development originated with lean product development and have inspired agile approaches such as XP (Beck, 2000, Beck and Andres, 2005), Scrum (Sutherland and Schwaber, 2013), Lean Software Development (Poppendieck and Poppendieck, 2003), and the Kanban Method (Anderson, 2010), among others. Modern agile methods focus on a continuous delivery of value, shorter cycle times, and small batch sizes (Kerievsky, 2016b). Software development organizations “*work in such a way that value is constantly flowing*” (Kerievsky, 2016a). The phrase “*from concept to cash*” (Poppendieck and Poppendieck, 2007) captures the intent of the time and was a leap forward in consciousness in 2007, but does not go far enough for the demands of today’s business context. A contemporary focus on flow means that organizations are developing better ways to understand how their products and services perform long after the cash transaction takes place. Development teams increasingly need to know how their products behave in production, employing the practices discussed in the previous section. All of the software development organizations in this study have these considerations as part of their context.

Today’s highest performing software organizations optimize for throughput and stability (Forsgren et al., 2018a). Both Little’s Law and Theory of Constraints (ToC) derive from Operations Management. They are influential in both lean product development and contemporary software development. Little’s Law, discussed in section 2.4.1, provides a way to understand throughput in a queuing system (Little and Graves, 2008). It describes how systems have limits in their capacity to process information and demonstrates that throughput is determined by the relationship between the arrival time of consecutive items into the system, and the time taken by the system to process an item. ToC (Goldratt, 1990) is a management method comprising a systemic problem structuring and problem solving

method (Balderstone and Mabin, 1998). ToC is oriented towards optimizing throughput in a system (Gupta and Boyd, 2008) and focuses on the system as a whole, rather than individual departments, or even just the confines of the organization (Goldratt and Cox, 2004). The central theme of ToC is that any system has a constraint (or a small number of constraints) which dominate the entire system. The goal is to “*manage these constraints, and the system as it interacts with these constraints, to get the best out of the whole system*” (Balderstone and Mabin, 1998). Flow attempts to balance efficiency and effectiveness. As DeMarco notes “*you’re efficient when you do something with minimum waste. And you’re effective when you’re doing the right something*” (DeMarco, 2001). Flow efficiency focuses on the amount of time it takes from identifying a need to satisfying that need (Modig and Ahlstrom, 2013). There are many ways to measure flow quality or efficiency. Chapter 3 discusses the literature on flow metrics. This study uses three primary measures as indicators of flow quality. These are *throughput*, *cycle time*, and *cumulative flow*. All three originate in lean product development and are widely used in contemporary software development. One reason this study uses these three in particular is because they provide a useful measure of the system as a whole. Throughput can be used to understand the productivity of a software development organization. Looked at from another perspective, throughput is the rate at which the system generates money through sales (Goldratt and Cox, 2004). Cycle time can be used to understand the responsiveness and predictability of a software development organization. Cumulative flow provides insight into the different states that work goes through as it moves through a value stream.

All of the approaches discussed in this and the preceding sections are about improving the culture of an organization. From Deming, TPS and lean, to agile development, lean software development, DevOps, continuous delivery, and the other contemporary approaches, a common thread is an emphasis on improving the culture of organizations so that they can deliver customer value more effectively. Iivari and Huisman (2007) have shown that organization culture has an effect on the deployment of new systems

development methodologies in organizations. Organization culture is the biggest barrier to adopting and scaling new methods in organizations (CollabNet VersionOne, 2019). Organization culture, therefore, is a determining factor in how successfully flow can be established in an organization. The findings in section 6.3.1 include insights into the culture of the organizations in this study that relate to flow.

1.2.4 Impediments to Flow

C-level executives have identified faster throughput – the ability to bring products to market faster – as the area where developers can have the biggest impact to their business (Stripe, 2018). This study uses throughput as one indicator of flow quality and hence as an indicator of the presence of impediments to flow. Cycle time is another indicator of flow quality in organizations, measuring how responsive organizations are to changes in their products. The highest performing “elite” organizations across the industry can deploy changes (from code commit to production) in less than an hour, whereas the lowest performing organizations can take one to six months (Forsgren et al., 2018a). Although Ohno initially characterized seven types of waste, in reality there are more than seven types (Ohno, 1988, Womack and Jones, 2003). Waste is best viewed as an impediment to flow (Womack and Jones, 2003). Spear (2004) found the semantic difference is significant: focusing on waste suggests the person is the problem. Deming (1986) shows that in 95% of cases, the problem is the system. Hence, this study’s perspective is on the system not the individual when seeking to understand impediments to flow. Managing the impediments that affect throughput and reduce flow efficiency is key to unlocking the potential of organizations. A review of existing literature in section 2.5.1 identifies eight common categories of impediments to flow in organizations. They are *extra features*, *handoffs*, *defects*, *delays*, *partially done work*, *task switching*, *extra processes*, and *unused employee creativity*. While frameworks such as Scrum mention an explicit focus on identifying and removing impediments (Sutherland and Schwaber, 2013), they provide no definition of what impediments are, nor do they provide guidance on

learning to see, understand or manage impediments. In general, the concept of impediments to flow in software development organizations has not been explored with much rigor.

Many factors contribute to impediments occurring in organizations. Studies show developers spend on average, 13.5 hours per week on technical debt and a further 17.3 hours per week on bad code, errors, debugging, refactoring, and modifying bad code (Stripe, 2018), all of which can either be classified as *failure demand* or the result of *failure demand*. Other studies show that technical debt can waste on average 23% of developers' time, and this wasted time negatively affects productivity (Besker et al., 2019). Ohno identified *overburden on the system* as the major contributing factor to impediments to flow in the Toyota Production System. This was closely followed by *unevenness in processing or demand*. Forsgren et al. (2018a) note that large batches and infrequent changes introduce risk to the development of today's increasingly complex systems. This study identifies nine common factors that contribute to impediments in organizations. They are *overburden*, *unevenness*, *Work in Process (WIP)*, *batches*, *queues*, *failure demand*, *technical debt*, and *complexity*. Section 2.5.2 provides a detailed discussion on these contributing factors. These are, in effect, the levers organizations use to manage and optimize flow in organizations. These contributing factors were either explicitly stated in limited or singular research studies as impeding flow in organizations or were the most commonly cited factors.

Impediments have an effect on flow, and also on other aspects of an organization. One study shows that overburden is the biggest contributing factor to developer *productivity* and *morale* (Stripe, 2018). Other factors from that study shown to have a negative impact on developer *morale* include work overload (or *overburden*), changing priorities resulting in discarded code or time wasted, not being given sufficient time to fix poor quality code, spending too much time on legacy systems, and paying down technical debt (Stripe, 2018). Section 2.5.3 summarizes six common effects of impediments to flow in organizations. These effects were either explicitly

stated in limited or singular research studies as effects of impediments or were the most commonly cited effects. The six effects cited are *productivity*, *predictability*, *quality*, *time*, *morale*, and *life-cycle profit*.

In one survey, 38% of respondents said that having “*end to end traceability from business initiative, through development, test and deployment*” would be the most valuable capability for their organization, followed by having metrics that identify disruptions in that flow (CollabNet VersionOne, 2019). This study uses the flow metrics mentioned above (and described in Chapter 3), in combination with other approaches, to identify impediments to flow in software development organizations.

1.2.5 Sensemaking and Complex Adaptive Systems

Software development organizations are losing an estimated \$300 billion annually in lost developer productivity that can be attributed to underused employee talent, failure demand, technical debt, and poor code quality (Stripe, 2018). All of the techniques, practices, and approaches discussed in the sections above are evolving to deal with the ever-increasing complexity of contemporary product development. All have the common goal of improving the feedback loop between product developers and the users of their systems in production. To enable this feedback loop, a focus on flow-based development demands a rethinking of the systems of management used in software development organizations. Quickly catching and resolving impediments is essential. At least as important is understanding the system patterns that allow impediments to emerge in the first place and developing a sensory capacity in organizations for the emergence of factors that interrupt flow. It is not enough to simply tackle the impediments. These organizations need a different approach to how they are managing their software development flows. This study proposes augmenting current approaches with management practices informed by sensemaking and complex adaptive systems theory in order to fundamentally improve the systems of work that are allowing these impediments to occur.

Dooley notes that “*the prevailing paradigm of a given era’s management theories has historically mimicked the prevailing paradigm of that era’s scientific theories*” (Dooley, 1997). Organizations are complex adaptive systems (Stacey, 1996). The complexity sciences have emerged as one of the prevailing paradigms for modern management thinking (Vasconcelos and Ramirez, 2011). Complex systems have four attributes (Page, 2015): first, they include diverse entities; second, the entities interact within a structure to form a dynamic, perpetually novel network; third, individual behaviors are interdependent, often influencing each other in subtle and imperceptible ways; fourth, the entities adapt and learn. 96% of C-level executives say it is a priority of upper management to increase the productivity of developers (Stripe, 2018). However, increasing productivity is a complex problem; from a CAS perspective, productivity is an outcome that depends on the dynamic nature of the system (Page, 2015). The most effective approach to improve productivity is to address the system patterns that are resulting in lower-than-desired performance.

Not everything in a complex system is complex (Page, 2015). Systems that possess these attributes can produce complexity, but not everything they produce is necessarily complex (Page, 2015). Contemporary software development organizations are systems capable of producing complexity. Acknowledging this distinction, Snowden and Boone (2007) distinguish between *ordered* and *unordered* systems. Ordered systems are generally stable and predictable. People in the organization can perceive cause-and-effect relationships, and they can determine “correct” answers based on available information. Unordered systems are comparatively more dynamic and unpredictable. There is no immediately apparent relationship between cause and effect, and the way forward is determined based on emerging patterns (Snowden and Boone, 2007). Both types of system are present simultaneously in contemporary software development organizations. Many activities and phenomena fall into the context of ordered systems, and many into the context of unordered systems. This is also true for impediments. When organizations encounter impediments to flow, those impediments may or may not be complex. The correct course of action is often not

immediately apparent. It is important, therefore, to understand the context of impediments so that appropriate action can be taken to resolve them. The current literature does not generally distinguish between impediments with these different characteristics. Section 7.3.3 discusses the implications of understanding these different characteristics of impediments to flow.

Resolving impediments means making changes in a complex system. The systems literature uses the term *intervention* to refer to the process of making changes in a system (Kurtz and Snowden, 2006, Holland, 2006, Eoyang, 2004, Meadows, 2008). Eoyang (2001) classifies interventions as *container*, *difference*, or *exchange* (CDE) interventions, each of which brings a specific focus to acting in a system. Section 6.3.4 presents a CDE analysis of the interventions used by product development organizations in this study to resolve impediments.

People in organizations need approaches that help them make decisions and intervene in their systems in contextually appropriate ways (Snowden and Boone, 2007). This applies to resolving impediments to flow. Sensemaking provides a way of exploring and understanding complex systems (Weick, 1995). It is a process that helps organizations to “*structure the unknown so as to be able to act in it*” (Ancona, 2012). Section 1.2.3 above discusses the influence of organization culture on flow. Culture is a result of an organization’s management system (Mann, 2014). Schein (2010) notes that culture and leadership are two sides of the same coin. To effect improvements in flow, leaders are attempting to influence the culture of their organizations. Mann (2014) notes that culture arises from experience, i.e., “*our idea of the culture of a place or organization is a result of what we experience there.*” Therefore, leaders need approaches to understand their organization’s culture that are based on understanding the collective experiences of people in the organization. Culture is a complex system (Frank and Fahrback, 1999). People’s experience of culture in their organization as they attempt to create flow and resolve impediments is “*very contextual, reflecting their lived experience within a defined work structure.*” (Dickens, 2012). This is one place where sensemaking plays a

role in this study. Using micronarrative sensemaking (see section 4.2) is an entry point to understanding these collective experiences, and how they relate to flow and impediments. Krentel et al. (2016) demonstrate how micronarrative sensemaking tools are a “*valid and effective tool to detect operational issues.*” Their research shows distributed sensemaking (section 4.2) using micronarrative tools demonstrate benefits where a more in-depth understanding is needed to improve product and service delivery. Weick (2009) shows that combining the ideas of complexity theory with those of sensemaking theory provides “*a powerful combination to understand thick, dense events that have high stakes*”. In the context of this study, impediments to flow often have high stakes for the organizations involved. There are many sensemaking frameworks that can help organizations understand their context (section 4.2.4). This study explores the use of the Cynefin sensemaking framework (Snowden and Boone, 2007) (section 4.2.4) to determine an appropriate course of action to resolve impediments by mapping them to one of five *domains* (obvious, complicated, complex, chaotic, disorder).

1.2.6 Focus of this Research Study

Each topic discussed in the preceding sections informs the focus for this study. Using a funneling approach (Creswell, 2013), this study begins with a broad overview of lean product development and contemporary software development, highlighting how the former continues to influence the latter. Both lean product development and contemporary software development form the foundation for flow-based software development. Flow-based development is a rich and growing field with many concepts; the specific focus for this study is *impediments to flow*. This study takes the perspective that organizations are complex adaptive systems. This research uses sensemaking to get a richer, more-informed understanding of flow, impediments, and the context and culture of the organizations that are experiencing impediments to flow. The organizations that are part of this study are all large software product development organizations. The focus of this study, then, narrows to managing impediments to flow in large

software product development organizations, using a sensemaking and complexity perspective. Figure 1-1 visualizes this narrowing of focus for this study.

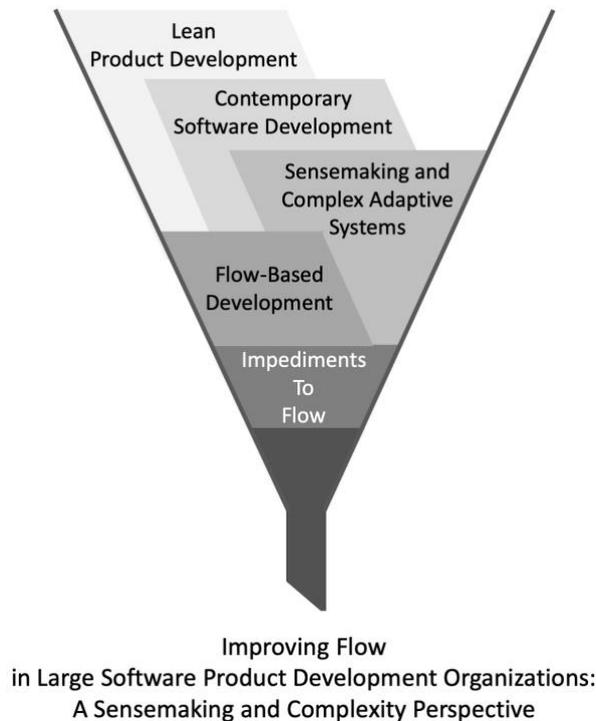


Figure 1-1 Funnel showing narrowing of topics towards the research focus area

This study, therefore, focuses first on understanding impediments, the contributing factors that lead to impediments, and the resulting effects of impediments. As far as the researcher is aware, this study is the first to collectively examine impediments together with their contributing factors and effects.

Next, this study employs approaches from lean management and complexity-informed management to help contemporary software organizations identify the patterns in their systems that are allowing the contributing factors to emerge. From there, the study explores how organizations can operate in environments of volatility, uncertainty, complexity, and ambiguity to identify impediments, make sense of impediments, and resolve them in a way that is appropriate to their context. This study is the first, as far as the researcher is aware, to employ sensemaking and complexity theory to the study of flow and impediments in software development organizations.

1.3 Research Objective

This thesis examines impediments to flow in large global software product development organizations. The objective of this research is:

Understand impediments to flow in large software product development organizations.

Through the pursuit of this research objective, the researcher aims to address the following specific sub-objectives:

RO1: Develop an understanding of the current state of research and practice regarding impediments to flow in software product development organizations.

RO2: Develop and validate an impediment management framework that is used to:

- (i) Analyze system patterns that contribute to impediments to flow
- (ii) Identify impediments to flow
- (iii) Analyze how people in organizations make sense of impediments
- (iv) Explore approaches to resolving impediments

The conceptual framework used to address these research objectives is developed throughout Chapter 2, Chapter 3, and Chapter 4. The framework is used to guide data collection and analysis. The final, validated framework from this study is shown in Figure 1-2.

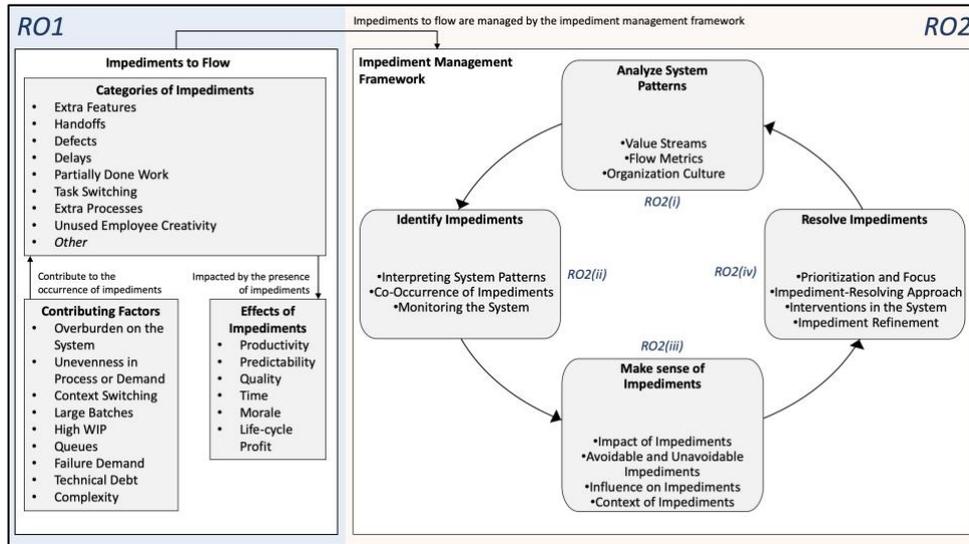


Figure 1-2 Relationship between research objectives and conceptual framework

1.4 Research Method

The epistemological stance of this qualitative research study (detailed in section 5.3) can be summarized as adopting a *constructivist* philosophical perspective (section 5.3.1), an *inductivist* logic perspective (section 5.3.2), and an *exploratory* theory perspective (section 5.3.3). The research methods for this study include focus group research (section 5.4.2), narrative research (section 5.4.3), and case study methods (section 5.4.4). This study explores three publicly traded companies responsible for delivering products and services to global markets. Each company in this study contains multiple software product development organizations. A product development organization in these companies is an organizational entity responsible for one or more products. Each product development organization contains multiple, geographically distributed teams. This study focuses on the global product development organization as the unit of analysis (Yin, 2016). As part of this study, the researcher conducted 24 in-person focus groups in 7 different globally distributed software development organizations in the USA, Ireland, UK, France, and India. Section 5.5.1 contains the details and demographics.

While researchers have attempted to understand organizations by using case study designs, these designs are only as good as the theoretical model

driving the research (Anderson et al., 2005). This study therefore follows the guidance of Anderson et al. (2005), and explores case studies through the theoretical model of complex adaptive systems theory. The combination of narrative capture with focus groups and case studies means that each approach “*provides a different way of knowing a phenomenon, and each leads to unique insights*” (Riessman, 2008). This study follows the guidance of Morgan (1997), who uses focus groups to learn about participants’ experiences rather than their opinions. Morgan (1997) emphasizes experiences and perspectives over opinion, because “*even self-reported behavior is more useful as data than opinions that have an unknown basis in behavior*”. This also fits well with the sensemaking approaches of Weick (1995) and Snowden (2010) described in section 1.2.5.

This study adopted a three-phase research approach, described in section 5.5.2. Phase 1 achieved research breadth and incorporated 10 focus groups in 3 different organizations. This phase helped inform which impediments are most impactful to flow in these organizations. This phase explored lean tools for identifying and resolving impediments. Phase 1 helped to identify gaps in the literature and in the research design. The researcher incorporated sensemaking into the study as a result of phase one. Phase 2 introduced sensemaking to this study. This phase incorporated 6 focus groups from 2 additional organizations and introduced distributed sensemaking across 3 large product development organizations. Micronarrative-based sensemaking data for distributed sensemaking was collected electronically and in person from participants around the world. Phase 3 provided a deeper exploration of complexity-informed approaches to managing flow and resolving impediments. It contained 8 focus groups across 2 additional organizations.

Analysis of data occurred following the completion of each phase. The analysis of one phase was used as input to inform the subsequent phase. Phase 1 explored impediments to flow dominantly from a lean perspective, exploring topics such as *value streams*, *continuous improvement*, and *A3 problem solving*. Phase 2 incorporated *sensemaking*, *experiment design*,

system improvement goals, and other elements to explore impediments to flow from a sensemaking and complexity-informed perspective, as well as continuing to explore lean-informed approaches. Phase 3 explored impediments to flow exclusively from a complexity and sensemaking perspective, probing deeper into topics raised during Phase 2. An overall detailed analysis was conducted after the phase 3 data were available. The researcher used the NVivo® qualitative research software to store and analyze all data related to the study.

1.5 Thesis Structure

This section describes the structure of the remaining chapters of this thesis.

Chapter 2 situates this study in the context of *contemporary software development*. This chapter begins with a history of lean management and the various sources that have influenced its evolution. It highlights how lean management has influenced contemporary software development methods. It provides an overview of relevant software development approaches, including agile and lean software development, culminating in a discussion of flow-based software development. Chapter 2 describes impediments to flow, together with their contributing factors and effects, and how they can negatively affect flow in software development organizations. Finally, it summarizes the core concepts from the chapter that form the basis of the conceptual framework for this study.

Chapter 3 discusses the literature on *metrics for understanding flow and impediments*. This section begins with a broad overview of the relevant literature, and presents the specific metrics selected for this study to understand flow and impediments. Finally, Chapter 3 summarizes the core concepts from flow metrics that contribute to the construction of the conceptual framework for this study.

Chapter 4 presents the relevant literature from sensemaking and complex adaptive systems theory in the context of studying *software development organizations*. This study uses complex adaptive systems theory (CAS) and

sensemaking processes to help understand flow and impediments in organizations. It describes the concepts and approaches to understanding change in a CAS that are relevant to this study. It describes how sensemaking and complexity contribute to the different parts of the conceptual framework developed for this study. This chapter concludes by presenting the complete conceptual framework, including all elements from Chapter 2, Chapter 3, and Chapter 4.

Chapter 5 describes the *research approach* used in this study to achieve the research objectives. It discusses the different research methodologies considered and provides a justification for the choices made. The research design is described in detail. This chapter describes the epistemological stance adopted by this study. It examines the research approaches used, including focus group, narrative research, and case study methods. It describes the data collection and data analysis methods used in this study and provides a detailed context of the 10 software development organizations studied.

Chapter 6 presents the findings and analysis of the empirical data. The purpose of this chapter is to present findings from this study that help to address the research objectives. These findings, together with the literature review, contribute to understanding the current state of practice in impediments to flow. The findings presented in this chapter were selected using the purposive sampling strategy described in Chapter 5.

Chapter 7 concludes the study and discusses the findings in the context of the research objectives and the existing literature. The purpose of this chapter is to summarize the main aspects of this study. It discusses the findings from Chapter 6 and presents conclusions, contributions, limitations, and directions for future research.

1.6 Summary

This chapter presents the background and motivation for this research study, the research objectives, and the research method used to achieve the

objectives. The chapter concludes by presenting the outline structure of the remainder of this thesis. The next three chapters will explore the relevant literature in detail, beginning with an exploration of contemporary software development.

Chapter 2 Contemporary Software Development

2.1 Introduction and Chapter Layout

This chapter describes contemporary software development. Lean is a major influence on modern software development methods. Section 2.2 describes the origins of lean product development, and its emergence via the Toyota Production System (TPS) and other key influences. Section 2.3 provides an overview of contemporary software development, including agile and lean software development methods. Section 2.4 describes flow-based software development. Section 2.5 describes impediments to flow in software development organizations and discusses the commonly cited contributing factors to those impediments, as well as the commonly cited effects of impediments. Finally, section 2.6 summarizes the core concepts from this chapter that contribute to this study's conceptual framework.

2.2 Lean Product Development

This section introduces the relevant background to lean product development that is applicable to this research. First, section 2.2.1 presents a history of the TPS discussing its origins, major influences, and essential components. Second, section 2.2.2 discusses five principles of lean product development. Third, section 2.2.3 discusses contemporary lean product development, showing how lean has evolved from its early origins in manufacturing.

2.2.1 The Toyota Production System

Deming's work with top industry managers from Japanese manufacturers, starting from 1950, is widely credited with contributing to Japan's rise from post-war crisis to world leaders in manufacturing (Deming, 2013). The Deming Management Method captures his approach to transforming the management of organizations (Ackoff, 1994, Walton, 1994, Deming, 1986,

Deming, 2013). Deming credits a systems perspective as “*the spark that turned Japan around*” (Deming, 2013). Toyota emerged from this post-war period to become not just the most successful automotive manufacturer in the world, but the world’s most successful industrial enterprise (Womack et al., 2007). Toyota credit the Toyota Production System (TPS) with its success. TPS was developed by Taichi Ohno, and is a formalization of the practices and culture of Toyota (Ohno, 1988, Ohno, 2013, Shingo, 1989). As indicated in the name, TPS is a sophisticated *system* in which all parts contribute to the whole; it is not just a set of tools (Liker, 2004). Spear and Bowen (1999) highlight that one reason so many companies have trouble adopting or imitating the lessons from the TPS is that they mistake the tools and practices with the system itself, and neglect the core principles behind it (Spear and Bowen, 1999). One key to Toyota’s success has been attributed to the fact that they continually apply underlying principles rather than focus on specific tools and processes (Spear, 2004). TPS is also sometimes referred to as “*Thinking People System*” to emphasize the role of people in continually improving their work in order to add value to the product or service that they are creating (Dickson et al., 2009).

Ohno encouraged everyone in the organization to think in terms of identifying and removing obstacles to their work, making the flow of work more efficient, and to focus on delivering customer value. People initially had trouble thinking in these terms and seeing obstacles, so Ohno characterized seven types of waste or *muda*. Ohno (1988) defines waste as “*all elements of production that only increase cost without adding value*”; Shingo (1989) defines waste as “*any activity that does not contribute to operations*”; Liker (2004) defines waste as “*activities that lengthen lead times, cause extra movement to get parts or tools, create excess inventory, or result in any type of waiting*”. These definitions are complimentary for the purposes of this study. Ohno (1988) explained that the number seven comes from an old Japanese expression “*He without bad habits has seven*,” which Ohno used to reinforce the point that “*even if you think there’s no waste you will find at least seven types*.” The seven types of waste defined in TPS are *overproduction, time on hand (waiting), waste in transportation,*

waste of processing itself, stock on hand (inventory), movement, and making defective products. These are discussed in more detail in section 2.5.1. He also described overburden (*muri*) and unevenness in process or demand (*mura*) as the main contributors to waste (Ohno, 1988, Ohno, 2013). The goal was to improve flow by eliminating all forms of inefficiency or waste that did not add value to the product (Modig and Ahlstrom, 2013). Liker (2004) added an eighth waste, *unused employee creativity*, resulting in what has become known as the eight wastes of the TPS, or the eight wastes of lean. These are discussed in detail in section 2.5 below. Womack and Jones (2003) distinguish between two types of *muda*. *Type One* is the steps or activities that “*create no value, but that are unavoidable with current technologies and production assets*”; *Type Two* is those steps in a process that create no value but are immediately avoidable. The problem of waste is particularly relevant to high performing organizations, which must remain ever vigilant against waste creeping in to their system (Womack, 2011). Rothman (2009) notes that “*real-world decisions almost never affect only a single proxy variable. In reality, we must assess how much time and effort we are willing to trade for eliminating a specific amount of waste.*” A *waste matrix* is a tool for making these trade-offs visible in organizations (Rothman, 2009). It can be used to quantitatively evaluate all of the wastes identified and can help with prioritizing which ones to address, while making conscious trade-offs with the other work the organization could be investing in. Typical information tracked in a waste matrix include the project, the kind of waste, the relative importance of eliminating this waste, total number of customers impacted by this waste, the value of removing the waste, a trigger date after which the value of removing the waste will be lost, and the relative value of managing this waste as opposed to all the other projects and work that the organization could invest in (Rothman, 2009). Section 6.3.4 shows an example used by one of the organizations in this study to help them prioritize which areas to invest in.

Deming introduced many concepts to Toyota including the Plan-Do-Check-Act (PDCA) improvement process (Womack et al., 2007). PDCA was originally developed by American statistician Walter Shewart in the 1930s

(Shewart, 1931), and popularized by Deming, who introduced it to Japan. It is also known variously as the Shewart Cycle, the Deming Cycle, and the Plan-Do-Study-Act (PDSA) cycle. PDCA cycles can be nested. When teams begin the “Do” phase on a PDCA cycle they might find they need to create multiple, smaller PDCA cycles (Martin and Osterling, 2014). Toyota developed the A3 process as a concrete structure to implement PDCA-based management as part of TPS (Sobek and Jimmerson, 2006, Sobek and Smalley, 2008). According to Liker and Convis (2012), A3 problem solving “*makes the thinking process visible*”. A3 generally takes the form of a report. The name “A3” derives from the fact that each report needs to fit on one side of an A3-size sheet of paper. Shook (2009) notes specific benefits of A3 include managers mentoring others in root cause analysis and scientific thinking. As well as an effective tool for solving problems, A3 thinking is equally about developing people’s problem-solving ability (Sobek and Smalley, 2008). This study explores the use of *A3 problem solving* in the context of contemporary software development organizations working to improve flow through managing impediments.

2.2.2 Principles of Lean Product Development

Lean production was the name given to a detailed study of the new techniques developed at Toyota and other Japanese manufacturers (Womack et al., 2007). They used the term *lean production* to differentiate from *mass production*, which had been the dominant paradigm of management thinking in manufacturing at the time. Lean product development is about evolving a system of management that focuses on growing healthy, profitable, sustainable organizations through respecting people and nurturing relationships across the entire value stream. Managers focus on optimizing the system of work and the environment in which the work is done through a relentless pursuit of continuous improvement (Womack et al., 2007). This focus is summarized in the five principles of lean: specify value, identify value streams, flow, pull, and perfection (Womack and Jones, 2003). These principles of lean product development are applicable in any domain, not just in manufacturing (Womack et al., 2007). Managers

applying principles of lean thinking should not fix problems directly themselves, but instead coach others so as to develop their problem solving capabilities (Spear, 2004). This section describes each of the five principles.

Principal 1: Specify Value

Value is defined from the perspective of the customer. Womack and Jones (2003) state that value can only be defined by the ultimate customer. According to Womack and Jones (2003), value is *“only meaningful when expressed in terms of a specific product ... which meets the customer’s needs at a specific price at a specific time”*, where products can be goods or services, or both. Value is created by the producer. In the context of this study, producers are software development organizations. Shewart, as reported by Hoyer and Hoyer (2001), differentiates between four kinds of value: use, cost, esteem, and exchange. The answer to the question *“What does the customer want from this process?”* is what defines value (Liker, 2004). Womack and Jones describe the difficulty companies face in correctly defining value: *“Partly because most producers want to make what they are already making and partly because many customers only know how to ask for some variant of what they are already getting. They simply start in the wrong place and end up at the wrong destination. Then, when providers or customers do decide to rethink value, they often fall back on simple formulas – lower cost, increased product variety through customization, instant delivery – rather than jointly analyzing value and challenging old definitions to see what’s really needed”* (Womack et al., 2007). Morgan (2011) defines values as *“the perceived worthiness of a product or service, or parts of a product or service, to a target set of customers and other stakeholders”*.

Principal 2: Identify the Value Stream

The Value Stream is the set of actions required to bring the product through the system to the customer (Womack and Jones, 2003). Ohno (1988) said of TPS: *“All we are doing is looking at the timeline from the moment a customer gives us an order to the point when we collect the cash. And we*

are reducing that timeline by removing the nonvalue-added wastes.” The timeline Ohno refers to can be drawn as a value stream. Spear and Bowen (1999) state that “[t]he pathway for every product and service must be simple and direct”. Visualizing the value stream nearly always exposes large amounts of waste. Later sections in this chapter will show that visualization is a key component of the conceptual framework for managing flow and impediments. Understanding value streams is a critical aspect of understanding how customer value flows through the organization (Rother and Shook, 2003, Womack, 2006, Martin and Osterling, 2014). Martin and Osterling (2014) provide two definitions of what a value stream is. A basic definition is that a value stream is “*the sequence of activities an organization undertakes to deliver on a customer request*” (Martin and Osterling, 2014). A more comprehensive definition elaborates on this to define a value stream as “*the sequence of activities required to design, produce, and deliver a good or service to a customer, and it includes the dual flows of information and material*” (Martin and Osterling, 2014). A value stream map is a visualization of the value stream. The concept of value stream maps was developed by Rother and Shook (2003), based on their knowledge of material and information flow diagrams used by Toyota (Womack, 2006). As well as identifying the value-adding steps in a process, value stream maps are also used to identify impediments to flow in the organization. Kovach et al. (2011) study specific lean and quality techniques used for continuous improvement by US-based organizations in manufacturing, healthcare, and aerospace. Their study shows that, of the organizations that implemented lean methods effectively, approximately 72% found value stream mapping to be effective. That same study shows that, of the organizations that failed to implement lean methods effectively, approximately 56% struggled with effectively implementing value stream mapping.

Principal 3: Make the Value-Creating Steps Flow

Flow is the process of enabling the value-adding activities that allows work to flow in a steady and even way through the organization to customers

(Womack and Jones, 2003). Creating flow is dependent on the previous two principles: flow is achieved through having a deep understanding of value and through identifying the value stream for each product. Flow is discussed in more detail in section 2.4 below.

Principal 4: Pull

Once the organization begins to focus on flow, the next step in lean thinking is to implement a pull system, where work is pulled from the customer, rather than pushed (Womack and Jones, 2003). This also has implications for how organizations create the product, with work pulled from the preceding flow state rather than being pushed in. Impediments are revealed once organizations get value to flow through the system. Womack and Jones (2003) phrase it as “*the harder you pull, the more impediments to flow are revealed so they can be removed*”. The second principle of TPS summarizes these first four lean principles: “*create a continuous process flow to bring problems to the surface*” (Liker, 2004). Impediments to flow are the central theme of this study, and are discussed in detail in section 2.5 below.

Principal 5: Perfection

In lean thinking, the pursuit of perfection is achieved through a combination of both continuous incremental improvement (*kaizen*) and radical change (*kaikaku*) (Womack and Jones, 2003). *Kaizen* is the Japanese term for continuous improvement, and is generally interpreted to mean a series of small, incremental improvements (Liker and Convis, 2012). *Kaikaku* is roughly translated as radical improvement (Womack and Jones, 2003). Both *kaizen* and *kaikaku* are part of a company’s commitment to continuous improvement (Liker and Convis, 2012). One distinction is *kaikaku* refers to breakthrough goals that take the company to a new level of achievement, while *kaizen* is viewed as part of daily management and operations in the organization (Liker and Convis, 2012). Shingo developed the idea of *kaizen*, or continuous improvement, as part of TPS to have a systemic process of developing a culture of improvement (Shingo, 2007). *Kaizen* is an integral part of leadership at Toyota (Liker and Convis, 2012). This type of

continuous improvement is required in a lean system (Liker and Morgan, 2006). Liker and Morgan (2006) note that organizations new to the idea of lean fail to adopt *kaizen* as a means of spreading continuous improvement throughout the organization. Activities and processes in the TPS are constantly being pushed to higher levels of performance, which allows Toyota to continually innovate and improve (Spear and Bowen, 1999). Womack (2011) identified the particular problem that impediments that organizations have already eliminated once can return. Womack (2011) highlights the importance of paying attention to the root causes of impediments in organizations, noting many failed lean transformations can be attributed to neglecting these root causes. Therefore, a relentless focus on identifying removing impediments to flow is critical for achieving, maintaining and improving organization effectiveness. Ackoff (1994) says of continuous improvement that it “*isn't nearly as important as discontinuous improvement. Creativity is a discontinuity, a creative act breaks with the chain that has come before it. It's not continuous.*” While continuous improvement is useful and important, approaches are needed that acknowledge how change works in human systems. It is this mindset, according to Spear and Bowen (1999), not the tools and practices, that is behind the success of Toyota and TPS. This study explores the use of *kaizen* in the context of contemporary software development organizations working to improve flow through managing impediments. The findings in Chapter 6 and the discussion in Chapter 7 elaborate this in more detail.

2.2.3 Contemporary Lean Thinking

The linear nature of the product development process is a characteristic of early lean implementations. By the mid-1980s, the world of commercial new product development had become “*fast-paced, fiercely competitive*” and “*speed and flexibility*” were essential to survival (Takeuchi and Nonaka, 1986). Companies were “*increasingly realizing that the old, sequential approach to developing new products simply won't get the job done.*” Takeuchi and Nonaka (1986) noted that organizations needed to move from a sequential model of product development to a more iterative model

characterized by built-in instability, self-organizing project teams, overlapping development phases, "multilearning", subtle control, and organizational transfer of learning. They used the metaphor of a relay race to describe the outdated sequential approach that most organizations were still taking towards product development. They encouraged organizations to adopt a rugby metaphor instead, where product teams move together as a unit. Lean Product Development Flow is an evolution of lean beyond manufacturing (Reinertsen, 1997, Smith and Reinertsen, 1998, Reinertsen, 2009, Thomke and Reinertsen, 2012). Manufacturing systems have traditionally dealt with predictable and homogeneous flows (Reinertsen, 2009). Contemporary work on lean product development flow draws from domains such as telecommunications networks and computer operating systems as examples of systems with highly-variable and heterogeneous flows (Reinertsen, 2009). Healthcare is another domain where the application of lean principles have been successfully applied (Dickson et al., 2009, Jimmerson et al., 2005, Spear, 2005, Spear, 2006). Influences on second generation lean come from a variety of domains, including computer operating systems, network and telecommunications systems design, military doctrine, healthcare, and construction. These domains have greater resonance with the variability, unpredictability, and heterogeneity found in contemporary software product development, as will be discussed in the section 2.3. Lean product development continues to influence, directly and indirectly, many contemporary trends in software development.

2.3 Contemporary Software Development

2.3.1 The Emergence of Agility in Software Development

Software development as a profession is still relatively young. Boehm (2006b) traces the history of software engineering, from the 1950s to the present. The 1950s were characterized by a focus on hardware engineering, with software being a secondary concern (Boehm, 2006b). Margaret Hamilton coined the term "*software engineering*" in an attempt to place software on a parity of esteem with hardware among engineers while working on the Apollo space program (Cameron, 2018). NATO sponsored

the first software engineering conference in 1968 to address the growing complexity for the defense industries (Naur and Randell, 1968). Rigorous methods were created in an attempt to ensure reliability and quality. By the 1990s demands for reliable, dependable, scalable software systems continued to increase. Some organizations stood out as having developed or adopted effective patterns of organizing to deliver complex products and systems (Cain et al., 1996, Coplien, 1994, Harrison and Coplien, 1996), but these patterns were not widespread. Overall, the practices and methods that had evolved were not meeting the needs of software development organizations or their customers. Software processes had become overly heavyweight and bureaucratic and were seen as stifling rather than enabling innovation (Highsmith, 2002). Agile Software Development emerged as an umbrella term in 2001 for a set of lightweight methods that were considered to be an alternative approach to the overly bureaucratic, wasteful, ineffective, sequential, and plan-driven waterfall processes that were dominant at the time (Highsmith, 2002). The industry responded by creating methods that were more lightweight and adaptive. These approaches include eXtreme Programming (XP) (Beck, 2000, Beck and Andres, 2005), Scrum (Schwaber and Beedle, 2002, Sutherland and Schwaber, 2013), and the Crystal family of methods (Cockburn, 2005), among others. XP, created by Kent Beck in 1996, is “*a style of software development focusing on excellent application of programming techniques, clear communication, and teamwork*” (Beck, 2000, Beck and Andres, 2005). XP is designed to “*address risk at all levels of the development process*”. XP was derived from a set of practices that Beck refined through working on the C3 project at Chrysler (Beck, 2000). Organizations choose the product management and development practices they are using to use to deliver value to customers. Scrum is designed to help organizations to manage complex adaptive problems, and to reveal the efficacy of their product management and development practices so that they can improve (Sutherland and Schwaber, 2013). In other words, Scrum does not prescribe a process; teams and organizations define their own process. Scrum helps them understand how effective their decisions are, gives them a set of rules to incorporate and suggests tools with which to adjust. The patterns of effective software

development organizations were a significant influence on Scrum and XP (Coplien and Harrison, 2004). Takeuchi and Nonaka (1986) had earlier used the term ‘scrum’ in the context of product development. That paper influenced the creators of both XP and Scrum, with the latter taking its name from that paper (Schwaber and Beedle, 2002). Beck (2000) notes the influence of the consensus-oriented approach to evolutionary delivery, and the ideas that can be applied for scaling XP to larger numbers of programmers. The creators of many of these methods convened in 2001 to create what became known as The Manifesto for Agile Software Development, or, simply The Agile Manifesto (Beck et al., 2001). The Agile Manifesto is a set of values and principles shared by the creators of a set of methods and frameworks. The four values of the Agile Manifesto are:

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

Abrahamsson et al. (2002) conducted one of the first academic studies to understand what it means for a method to be termed “*agile*”. They concluded that a method, to be considered agile, should be incremental, collaborative, straightforward and adaptive. In a study comprising a structured literature review of agility across multiple disciplines Conboy (2009) developed a definition and taxonomy of agility, defining agility as the ability to “*rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment*”. Dingsøyr and Lassenius (2016) discuss emerging themes in agile software development, from both a research and practitioner perspective from 2006 to 2015. From a practitioner perspective, they identify a decline in interest in XP, and a rise in interest in Scrum, continuous integration and DevOps. From a researcher perspective, Scrum appears to be the dominant area of interest, increasing steadily over the period 2006 to 2015. Tripp et al. (2018) identify lean

software development as an emerging area of research that can benefit from more rigorous study. Despite lean software development being around for almost two decades, it remains difficult for organizations to implement and there is limited research in this area (Tripp et al., 2018).

2.3.2 Continued Evolution of Software Development

Agile development is now commonplace in contemporary software development (CollabNet VersionOne, 2019). The first line of the agile manifesto declares “*We are uncovering better ways of developing software by doing it and helping others do it.*” This phrasing demonstrates a commitment to continuous improvement and learning, i.e., the manifesto authors did not claim to have found and codified the definitive ‘*right way*’ to develop software. The intervening two decades since the manifesto has seen agile development evolve from a new approach used by small, co-located teams to the approach of choice in global organizations. More methods have emerged. David Anderson created the Kanban Method as a management method for incremental, evolutionary change (Anderson, 2010). The Kanban Method (Anderson, 2010, Burrows, 2014) and Lean Software Development (Hibbs et al., 2009, Poppendieck and Poppendieck, 2003) brought an explicit focus on applying principles and practices from TPS, lean product development, and Theory of Constraints to software development organizations. Efforts to apply lean to software development date back to at least 1992 when a group of practitioners and academics convened the first conference on lean software development in an attempt to apply lean thinking to software development, rather than simply transpose manufacturing concepts (Freeman, 1992). A lean mindset prompts organizations to focus on delighting customers, to understand what they really want and need, and to ensure the right products and services are provided (Poppendieck and Poppendieck, 2013). Lean Startup builds on a foundation of agile development, lean product development, and Open Source to define an innovation-focused process that has become relevant not just to startups but also to established companies, large enterprises, government departments and others (Ries, 2011, Maurya, 2012). It is

common for organizations to use more than one of these methods. Research by Conboy and Fitzgerald (2007) shows that method tailoring is a necessary response to the complex and unique nature of every organization. For example, the use of Scrum and Kanban together is sometimes called *Scrumban*, referring to a Kanban pull system operating within the time-boxed iterations and planning cycles of Scrum (Ladas, 2009). There are now many frameworks, techniques, methodologies, and in-house practices proposing solutions for organizations working at larger scales. Conboy and Carroll (2019) identify nine specific challenges associated with these large-scale agile frameworks. Companies have figured out how to be agile at a large scale, with or without frameworks, with increasing size in organizations, number of locations, number of people, and ever-larger projects and products (Dingsøy et al., 2019).

2.3.3 Extending the Feedback Loop

Practices such as continuous delivery (Humble and Farley, 2010), DevOps (Kim et al., 2014, Davis and Daniels, 2016, Forsgren et al., 2018a, Mann et al., 2018) and Site Reliability Engineering (SRE) (Beyer et al., 2016) have bridged the gap between product development teams and operations teams. Improved capabilities for monitoring, observability, and feedback provide software development organizations with better insight into their products in production environments and enable a tighter feedback loop between software developers and customers. Many contemporary software systems are implemented as distributed systems with complex behavior and failure modes. Software organizations are using experimentation to verify the reliability of such systems (Basiri et al., 2016). However, even though these approaches exist, many software development organizations are either not using them at all, or not using them effectively or to their full potential (Forsgren et al., 2018a, CollabNet VersionOne, 2019).

2.3.4 Characteristics of Contemporary Software Development

The widespread adoption of agile methods has not eliminated the challenges faced by organizations. Today, software development organizations operate in an environment of ever-increasing volatility, uncertainty, complexity, and ambiguity (Bartscht, 2015, Johansen and Euchner, 2013). The pace of change is accelerating. The technical complexity is increasing. The demands for software that is reliable, dependable, and scalable are increasing. Organizations are struggling to keep pace (Forsgren et al., 2018a, Stripe, 2018, CollabNet VersionOne, 2019). Offering some context that elaborates on these challenges, Rajlich (2016) outlines a set of properties that characterize modern software development. These include:

- **Complexity.** Rajlich (2016) notes that software systems are amongst the most complex systems ever created. Acknowledging the complexity in modern software development, the Scrum framework is specifically designed to deal with complex adaptive problems. The best way to solve complex problems in product development is through rapid iteration (Poppendieck and Poppendieck, 2013). Short iteration lengths, combined with daily feedback and steering “*give the team the ability to respond to change quickly by constantly adjusting the direction of the development process*” (Vidgen and Wang, 2006). Sutherland and Schwaber (2013) write that Scrum is a framework “*within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value*”. Anderson (2010) describes Kanban as a Complex Adaptive System for Lean, writing that “*[t]he Kanban Method introduces a complex adaptive system that is intended to catalyze a Lean outcome within an organization*”. Boehm articulates eight trends that characterize modern software engineering (Boehm, 2006a). Common threads running through these trends include a focus on systems, increased complexity and a need to adapt to rapid change. The changes in the software development landscape will continue to “*emphasize value generation*

and enable dynamic balancing of the agility, discipline, and scalability necessary to cope with the 21st century challenges of increasing rapid change, high dependability, and scalability to globally-integrated, software-intensive systems of systems” (Boehm, 2006a).

- **Invisibility.** Software is abstract and intangible. This is a significant difference between product development work and manufacturing. Work-in-Process (WIP) is invisible in software development (Thomke and Reinertsen, 2012). Because the work is invisible, the batch sizes tend to also be invisible (Thomke and Reinertsen, 2012). The problems associated with *Large Batches* and *High WIP* are addressed in section 2.5.2. Mandić et al. (2010) contrast software development processes to manufacturing processes, noting that in software development “*the true production process is a cognitive process, and as such is intangible and difficult to control*”. It is because of this property of invisibility that lean approaches such as Gemba walks (Womack, 2011), where managers go and see the work in a production environment, require supplemental approaches in software development. Value stream maps (section 2.2.2) are another way to visualize work. One of their main advantages is their ability to visualize otherwise hidden work (Martin and Osterling, 2014).
- **Changeability.** Software is easy to change, but as the system grows, it becomes more difficult to change it correctly (Rajlich, 2016). Refactoring is one example of a practice that promotes a disciplined approach to continually changing the internal software of software through small, incremental changes (Fowler, 2019).
- **Sensitivity to Environment.** The fourth property is sensitivity to environment, or what Rajlich (2016) calls “*conformity*”, which acknowledges that software exists within a larger domain that includes people and other systems. Changes to the domain often require changes to the software. Conformity in turn “*adds to the complexity of the work with software*” (Rajlich, 2016). Deming noted that a good person in a bad system will be defeated by the system (Deming, 1986).

- **Non-linearity.** Rajlich (2016) describes discontinuity, or non-linearity, nonlinearity in complex software systems, where a small change in input can have a disproportionately large change in output.

2.4 Flow-Based Software Development

2.4.1 Considerations for Flow-Based Software Development

Building on the concepts discussed in the previous sections, flow-based development emphasizes a focus on the continuous flow of value through a value stream. This section discusses six points related to flow-based software development that are relevant to this study.

Motivations for Embracing Flow

In an effort to keep pace with the demands and challenges described above software development organizations are evolving towards an increasingly continuous, flow-based approach to developing and delivering software (Fitzgerald and Stol, 2017, Forsgren et al., 2018a). Short feedback cycles, continuous integration (Beck, 2000, Duvall et al., 2007), continuous delivery (Humble and Farley, 2010), continuous deployment and continuous operations are all seen as essential ingredients for high-performing organizations (Forsgren et al., 2018b). A common focus in all cases is to take a holistic, system-wide, end-to-end view of the product development or service delivery lifecycle, focusing on the whole organization and environment in which products and services are created, extending even to suppliers, partners and customers. Software development organizations take an increasingly customer-centric perspective on products and services, and then organize the work so that it flows as smoothly as possible from the point at which an idea is conceived or a request is made, to the point at which that idea or request is in use by its intended audience. This is essentially putting the five principles of lean outlined in section 2.2.2 into practice. Flow is increasingly important to software development organizations, with 67% of respondents to one survey reporting that value

stream management is important for connecting their business needs to their software development capability (CollabNet VersionOne, 2019). A key principle in achieving flow is that “*everyone involved must be able to see and must understand every aspect of the operation and its status at all times*” (Womack and Jones, 2003). As noted in section 2.3, one of the characteristics of software development is that the work is invisible, making it more difficult to see the operation and status at all times. One way to make the work visible is to use visual control boards that show the different states of the work (Thomke and Reinertsen, 2012). Research by Rodríguez et al. (2014) finds that Kanban is increasingly used and helps to visualize and manage WIP, and create a pull (versus a push) culture. That study reports that essential ingredients for flow include continuous integration, test automation, transparency, and streamlined communication. In one survey, 38% of respondents said that having “*end to end traceability from business initiative, through development, test and deployment*” would be the most valuable capability for their organization, followed by having metrics that identify disruptions in that flow (CollabNet VersionOne, 2019). This chapter focuses on flow and impediments. Chapter 3 discusses flow metrics and how they can help to identify disruptions in flow.

Steps for Achieving Flow

A primary goal of Lean is to achieve a smooth flow from the point at which customers make a request to the point at which they take possession of a completed product. As discussed in section 2.3.3 above (*Extending the Feedback Loop*), contemporary software development organizations need to consider what happens beyond the point at which customers take possession of a completed product. Womack et al. (2007) identify three specific steps that help work to flow in organizations.

1. The first step is to focus on the actual object of work, from the beginning of the process through to completion (Womack et al., 2007). This encapsulates the lean principles of *specify value*, *identify the value stream*, and *make the value-creating steps flow*, as described in section 2.2.2.

2. The second step is to ignore traditional boundaries of jobs, careers, functions, departments, and even firms, in order to create a lean enterprise (Womack et al., 2007). Then, the goal is to remove all impediments to the continuous flow of the product. This encapsulates the lean principle of *pull*, as described in section 2.2.2.
3. The third step is to rethink specific work practices and tools to stop “*backflow, scrap, and stoppages of all sorts*” in order to enable a continuous flow of work and continuously resolve impediments to flow (Womack et al., 2007). This encapsulates the lean principle of *perfection*, as described in section 2.2.2.

Value Streams in Software Organizations

The ideas behind flow-based development originated with lean product development (section 2.2 above) and have inspired agile approaches such as XP (Beck, 2000, Beck and Andres, 2005), Scrum (Sutherland and Schwaber, 2013), Lean Software Development (Poppendieck and Poppendieck, 2003), the Kanban Method (Anderson, 2010), and others. Beck and Andres (2005) define flow in software development as “*delivering a steady flow of valuable software by engaging in all the activities of development simultaneously*”. This approach is different to traditional project management approaches that focus on managing timelines and project phases. The XP practices are “*biased towards a continuous flow of activities rather than discrete phases*”. Modern agile methods focus on a continuous delivery of value, shorter cycle times, and small batch sizes (Kerievsky, 2016b). Software development organizations “*work in such a way that value is constantly flowing*” (Kerievsky, 2016a). The phrase “*from concept to cash*” (Poppendieck and Poppendieck, 2007) captures the intent of the time and was a leap forward in consciousness in 2007, but does not go far enough for the demands of today’s business context. A contemporary focus on flow means that organizations are developing better ways to understand how their products and services perform long after the cash transaction takes place. Referencing research by Fitzgerald and Stol (2015), Dittrich et al. (2018) describe this as “*a continuous flow from business*

strategy and decision making over development, to operations and use, and finally returning user feedback and new ideas back to the business”.

Development teams increasingly need to know how their products behave in production, employing the practices discussed in section 2.3 above. All of the software development organizations in this study have these considerations as part of their context.

Flow and Lean Principles in Software Development

The principles of lean described in section 2.2.2 are relevant to software development organizations who are seeking to optimize flow. Numerous other principles have emerged in addition to this set of principles. For example, Table 2.1 shows the Poppendiecks have articulated a different set of principles of lean software development in each of their books, demonstrating how the area is still evolving.

Table 2.1 Evolution of the principles of lean software development

Poppendieck and Poppendieck (2003)	Poppendieck and Poppendieck (2007)	Poppendieck and Poppendieck (2013)
<ul style="list-style-type: none"> • Eliminate waste • Amplify learning • Decide as late as possible • Empower the team • Build integrity in • See the whole 	<ul style="list-style-type: none"> • Eliminate waste • Build quality in • Create knowledge • Defer commitment • Deliver fast • Respect people • Optimize the whole 	<ul style="list-style-type: none"> • Optimize the whole • Energize workers • Focus on customers • Deliver fast • Build quality in • Keep getting better • Eliminate waste • Learn first

Reinertsen (2009) describes 175 principles of product development flow, organized around eight key themes. The themes are *economics, queueing, variability, batch size, WIP constraints, flow control, fast feedback, and decentralization of control*. Burrows (2014) describes the 4 foundational principles and 6 core practices of the Kanban Method. These are summarized in Table 2.2 and are an updated version that supersede the earlier principals published by Anderson (2010).

Table 2.2 Principles and practices of the Kanban Method

Foundational Principles	Core Practices
1. Start with what you do now	1. Visualize
2. Agree to pursue evolutionary change.	2. Limit Work-in-Progress (WIP)
3. Initially, respect current processes, roles, responsibilities, and job titles.	3. Manage flow
4. Encourage acts of leadership in every level of your organization – from individual contributor to senior management.	4. Make policies explicit
	5. Implement feedback loops
	6. Improve collaboratively, evolve experimentally (using models and the scientific method)

Optimize for Throughput

Today’s highest performing software organizations optimize for throughput and stability (Forsgren et al., 2018a). Both Little’s Law and Theory of Constraints (ToC) come from Operations Management. They are influential in both lean product development and contemporary software development. Little’s Law provides a way to understand throughput in a queuing system (Little and Graves, 2008). It describes how systems have limits in their capacity to process information and demonstrates that throughput is determined by the relationship between the arrival time of consecutive items into the system, and the time taken by the system to process an item. ToC (Goldratt, 1990) is a management method comprising a systemic problem structuring and problem solving method (Balderstone and Mabin, 1998). ToC is oriented toward optimizing throughput in a system (Gupta and Boyd, 2008). ToC focuses on the system as a whole, not individual departments, and not just the confines of the organization (Goldratt and Cox, 2004). The central theme of ToC is that any system has a constraint (or small number of constraints) which dominate the entire system. The goal is to “*manage these constraints, and the system as it interacts with these constraints, to get the best out of the whole system*” (Balderstone and Mabin, 1998). Flow attempts to balance efficiency and effectiveness. As DeMarco notes “*you’re efficient when you do something with minimum waste. And you’re effective when you’re doing the right something*” (DeMarco, 2001). Flow efficiency focuses on the amount of time it takes from identifying a need to satisfying that need (Modig and Ahlstrom, 2013). There are many ways to measure flow quality or efficiency. Chapter 3 discusses the literature on flow metrics. This study uses three primary measures as indicators of flow

quality. These are throughput, cycle time, and cumulative flow. All three originate in lean product development and are widely used in contemporary software development. One reason this study uses these three in particular is that they provide a useful measure of the system as a whole. Throughput can be used to understand the productivity of a software development organization. Looked at from another perspective, throughput is the rate at which the system generates money through sales (Goldratt and Cox, 2004). Cycle time can be used to understand the responsiveness and predictability of a software development organization. Cumulative flow provides insight into the different states that work goes through as it moves through a value stream.

2.4.2 Flow Units

The unit of flow, i.e., the “thing” that flows through the system, will vary from one organization or company to another, depending on what is important to them and their customers. For some organizations there will be multiple different types of flow items. For example, for hospitals, the units of flow include patients. For an aircraft manufacturer, the units include wings, engines and seats. For a university research group, the units of flow might include publications and research grant applications. For agile software development teams, the units of flow might include features, defect fixes, user stories, and designs. Flow units move through different states as work flows through the system.

2.4.3 Flow States

Work can be considered to pass through different states as it moves through the organization. Typical states for many teams are “*Planned*”, “*In Progress*”, and “*Done*”. What this means is that as the work moves through three distinct states. First, it is in a *planned* state. This could be a new feature or a user story that a team plans to work on, or a defect they plan to address. The work item is said to be *In Progress* when the team actually starts to work on something. The work item moves to the *Done* state when all work is completed. Many agile teams use a “*Definition of Done*” that

clearly articulates what it means for a work item to be considered “*Done*”. Scrum prescribes that teams using Scrum must have a clear Definition of Done that helps everyone to understand what it means for something to be actually *done* (Cohn, 2010, Sutherland and Schwaber, 2013). The diagram in Figure 2-1 depicts these three states.

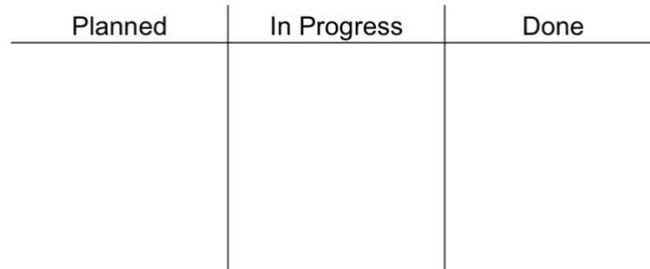


Figure 2-1 A typical visualization of flow states

Both Scrum and Kanban encourage the visualization of workflow states. Teams typically have a board that displays these states and shows the flow of work items through the states. Many teams, especially those that are also using Kanban, explicitly add Work in Process (WIP) limits to workflow states, and have explicit policies, often called entry and exit criteria. These govern when a work item can enter a state and when it can be pulled into the next state. A typical board representing these flow states is represented in Figure 2-2.

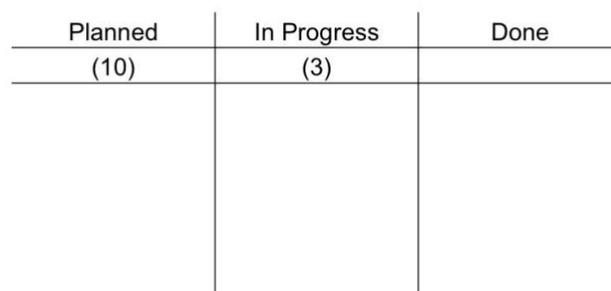


Figure 2-2 Workflow states represented as a Kanban board with WIP limits

The states used by the teams in this research project are described in section 6.3.1. Note that this is the most basic version of a Kanban board, incorporating WIP limits. More advanced Kanban visualizations incorporate additional elements including classes of service for distinguishing different

types of work, an expedite lane for processing unplanned urgent work, and visual indicators that show the status of work items within the flow states.

2.5 Impediments to Flow

There has been little research to date in understanding impediments to flow in software engineering organizations. This study contributes to addressing that gap. Ohno (1988) wrote that achieving flow necessitates removing any waste from the system. Definitions of waste in software development such as those in Poppendieck and Poppendieck (2003) have for the most part simply reused the TPS and lean production definitions, emphasizing waste as anything that does not add value from the perspective of the customer. However, waste is best viewed as an impediment to flow (Womack and Jones, 2003), a perspective missed in most studies of lean and flow in software development. The translations of the lean concept of waste in the software development literature to date have focused on an almost literal translation of the wastes of manufacturing production. These translations are inconsistent and lack a coherent presentation in the context of modern knowledge work, including software development. The waste metaphor does not translate comfortably from its origins in automobile manufacturing to modern knowledge work (Anderson, 2010). Anderson (2013) writes that *“a focus on flow, rather than a focus on waste elimination, is proving a better catalyst for continuous improvement within knowledge work activities such as software development”*. This thesis proposes that a more appropriate perspective on the lean concept of waste for the complexity of 21st century organizations of knowledge workers is to instead focus on understanding impediments to flow. Spear (2004) found the semantic difference is significant: focusing on waste suggests the person is the problem. Deming (1986) shows that in 95% of cases, the system is the problem. Hence, this study’s focus is on the system rather than the individual when seeking to understand impediments to flow. This is one of the primary reasons for choosing the product development organization as the unit of analysis, as discussed in section 1.4. Ohno (1988) referred to waste as obstacles and impediments to flow. With reference to *pull*, the fourth principle of lean

product development in section 2.2.2 above, Womack and Jones (2003) note that when attempting to create a smooth flow “*the harder you pull, the more impediments to flow are revealed so they can be removed*”. Removing impediments to flow is critical to improving an organization’s processes (Shalloway et al., 2010). Ohno asserts that eliminating impediments completely can improve the operating efficiency of an organization by a large margin (Ohno, 1988). Understanding impediments in the context of software product development can be valuable in helping organizations achieve significant improvements in operating efficiencies. Another example of where the perspective of impediments is more useful than that of waste comes into play when considering variability. TPS emphasizes removing variability from the manufacturing process through eliminating waste (Ohno, 2013). Variability in product development, on the other hand, is something to be acknowledged and even leveraged to gain a competitive advantage (Reinertsen, 2009). The two perspectives are not in conflict, they are simply reflecting different domains.

All five of the lean product development principles described in section 2.2.2 above can be summarized from the central perspective of flow and impediments. It can be seen from this summary that the first three principles relate to understanding what needs to flow, and how it needs to flow. The fourth and fifth principles are about managing and resolving impediments to flow.

1. **Specify Value:** What do customers want and need? What is it the organization is producing to meet that need?
2. **Identify the Value Stream:** What value-creating steps produce what customers want?
3. **Flow:** How do organizations make the value-creating steps flow?
4. **Pull:** What impediments to flow emerge as customers pull value from the system?
5. **Perfection:** How do organizations continually remove impediments to flow?

Referencing the work of Shewart, Ackoff (1994) said that “*when you get rid of something you don’t want, you don’t necessarily get something you do want. And so, finding deficiencies and getting rid of them is not a way of improving performance of a system.*” He went on to add that “[a]n improvement program must be directed at what you want, not at what you don’t want. And determining what you want requires you redesigning the system, not for the future but for right now.” (Ackoff, 1994). What organizations want is a smooth and continuous flow of value to customers. A focus on identifying and resolving impediments is essential for creating and maintaining a continuous flow. While frameworks such as Scrum mention identifying and removing impediments (Sutherland and Schwaber, 2013), they provide no definition of what impediments are, nor do they provide guidance on learning to see, understand or manage impediments. In general, the concept of impediments to flow in software development organizations has not been explored with much rigor. A review of existing literature identifies eight common categories of impediments to flow in organizations (section 2.5.1), nine common factors that contribute to impediments in organizations (section 2.5.2), and six common effects of impediments in organizations (section 2.5.3). These were either explicitly stated in limited or singular research studies or were the most commonly cited. These are discussed in detail in the following sections.

2.5.1 Identifying Impediments to Flow

Ohno (1988) originally described seven types of impediment in manufacturing to help people learn to see those things that were impacting their work. To resolve impediments, one first must be able to recognize them, and recognizing impediments has proven to be difficult (Rother and Shook, 2003). Table 2.3 shows the evolution of various popular categories of waste, starting with the original set from Ohno (1988).

Table 2.3 Evolving categories of waste from TPS to lean software development

TPS	TPS	Lean Software Development	Lean Software Development
Ohno (1988)	Liker (2004)	Poppendieck and Poppendieck (2003)	Poppendieck and Poppendieck (2007)
<ul style="list-style-type: none"> • Overproduction • Time on hand (waiting) • Waste in transportation • Waste of processing itself • Stock on hand (inventory) • Movement • Making defective products 	<ul style="list-style-type: none"> • Overproduction • Waiting (time on hand) • Unnecessary transport or conveyance • Over-processing or incorrect processing • Excess inventory • Unnecessary movement • Defects • Unused employee creativity 	<ul style="list-style-type: none"> • Partially done work • Extra processes • Extra features • Task switching • Waiting • Motion • Defects 	<ul style="list-style-type: none"> • Partially done work • Extra features • Relearning • Handoffs • Task switching • Delays • Defects

This study takes the perspective advocated by Ohno (1988) and Poppendieck and Poppendieck (2007) that categories are not the point, but they are useful when starting to see impediments to flow. Having a set of categories helps reinforce the habit of seeing them (Poppendieck and Poppendieck, 2007). Therefore, rather than re-invent a new set of impediment categories, this study takes these existing categories as a starting point. They are also helpful in analyzing and understanding what kind of impediments are impacting organizations. Section 6.2.1 presents findings that show an analysis of impediments, with examples, from the organizations in this study. A review of existing literature, including those in Table 2.3, identifies eight common categories of impediments to flow in organizations. These are *extra features*, *handoffs*, *defects*, *delays*, *partially done work*, *task switching*, *extra processes*, and *unused employee creativity*. Impediments from each of these categories is present to some degree in organizations.

Extra Features

A Standish group report shows that approximately 45% of features in a typical system are never used, with 19% rarely used (Poppendieck and Poppendieck, 2003). These *extra features* are an example of unnecessary work. *Extra features* impede the flow of work through the system by

consuming time and effort that could otherwise be spent on more value-adding work. They later prove to add no value for customers or delay the delivery of more valuable features. There are also maintenance costs associated with *extra features*. Beyond the initial costs to develop the feature, there are hidden costs including time invested in maintaining the feature, possibly in multiple branches, time invested in Failure Demand related to the feature, e.g., fixing defects, refactoring, or managing technical debt, and the motivation of the people that developed the unused features. Liker refers to this as *overproduction* (Liker, 2004). Beck and Andres (2005) note that software development “*is full of the waste of overproduction*”, including “*elaborate architectures that are never used*” and “*documentation no one reads until it is irrelevant or misleading*”. Many organizations do not consider the hidden economic costs of adding features that customers don’t want, or for which there is not enough demand. Adding extra features significantly slows down feedback and revenue generation, as the product could be released sooner with fewer features. The opportunity costs associated with time lost on these other costs means the company could have been investing the time and money in something more valuable.

Handoffs

Handoffs occur whenever incomplete work must be handed over from one person or group to another (Poppendieck and Poppendieck, 2007). *Handoffs* impede the flow of work through the system by adding delays, requiring more people, or losing knowledge as work is handed over from one person or group to another. *Handoffs* are also referred to in the literature as hand-offs. Ward (2007) argues that *handoffs* are the most fundamental waste in companies because they separate knowledge, responsibility, action and feedback. Poppendieck and Poppendieck (2013) describe a case study from Ericsson that illustrates the cost of *handoffs*: “*handovers of information between functions tended to be inefficient; both knowledge and time were lost in every handover. As the number of handovers increased, the problems tended to escalate nonlinearly. Furthermore, workers in each function were assigned to multiple projects, causing severe multitasking that increased*

inefficiencies. The inefficiencies of handovers and multitasking showed up as decreased speed, and therefore slower time to market.”

Defects

Poppendieck and Poppendieck (2007) identify defects as an impediment to flow. Many *handoffs* in a value stream generates *defects* (Modig and Ahlstrom, 2013). Weinberg (2014) distinguishes between faults and failures: A failure “*is the departure of the external results of program operation from requirements.*” A fault “*is the defect in the program that, when executed under particular conditions, causes a failure.*” Many organizations measure project success in terms of defects to production, defects over time, and defect resolution (CollabNet VersionOne, 2019). Defect rate is a measure of the reliability of a software development organization (Putnam and Myers, 2003). Forsgren et al. (2018a) show that organizations are spending 10-20% of their time working on defects identified by end users, i.e., just the defects that were found later in the value stream and that impacted customers.

Delays

Delays impede the flow of work through the system by adding to the overall cycle time from request or idea to delivered product or service (Poppendieck and Poppendieck, 2007). This impediment is also referred to as *waiting* or *time on hand* (Liker, 2004). Poppendieck and Poppendieck (2003) note that “*one of the biggest wastes in software development is usually waiting for things to happen*”. *Delays* prevent the organization from delivering value to the customer as quickly as possible (Poppendieck and Poppendieck, 2003). The ability of a team or organization to respond to an idea or request is directly related to the delays in the system. Reinertsen (2009) asserts that 85% of product development organizations do not understand the cost of *delay* associated with their products. He argues that understanding *delay* is so critical that, if organizations were to quantify just one thing, they should quantify the cost of delay, which he codifies as “*The Principle of Quantified Cost of Delay*” (Reinertsen, 2009). Brooks noted the “*severe financial, as*

well as psychological, repercussions” of delays discovered late in a project (Brooks, 1995). *Delays* can have a cumulative effect. Brooks (1995) further points out that the secondary costs incurred by other projects waiting on the delayed project can far outweigh all other costs.

Partially Done Work

Incomplete work in software development is analogous to inventory in manufacturing (Poppendieck and Poppendieck, 2007). It is work that is not yet complete, and, therefore, does not yet provide any value to the business or the customer. Too much incomplete work impedes the flow of work through the system by slowing down the flow of work for individual work items and delaying the point at which value can be realized. Poppendieck and Poppendieck (2003) translated the TPS waste of “*inventory*” to “*partially done work*”. Beck and Andres (2005) provide examples of waste resulting from excess incomplete work, including “*requirements documents that rapidly grow obsolete*”, and “*code that goes months without being integrated, tested, and executed in a production environment*”. Other examples include starting lots of projects or work items but taking a long time to finish anything. A common motivator behind too much incomplete work is organizations that measure progress in terms of perceived activity rather than delivered value.

Task Switching

Authors translating from TPS to software translate *motion* to *task switching* (Poppendieck and Poppendieck, 2003, Poppendieck and Poppendieck, 2007, Hibbs et al., 2009). Task switching between multiple activities or projects impedes flow and wastes significant time in software development (Poppendieck and Poppendieck, 2003, Poppendieck and Poppendieck, 2007). Task switching can often be the result of *delays*: “*having to switch focus while you wait for someone else kills progress, not to mention enthusiasm for the task at hand*” (Mann et al., 2018).

Extra Processes

Extra processes generate extra work that consumes time and effort without adding value (Poppendieck and Poppendieck, 2003). *Extra processes* are also referred to as *overprocessing* or *incorrect processing* (Liker, 2004). Poppendieck and Poppendieck (2003) translate “extra processing” to “Extra Processes”. Hibbs et al. (2009) translate *overprocessing* to “*unneeded processes*”. Examples include meetings and metrics reviews that add no value, paperwork and documentation that add no value, time spent pursuing an unreasonable level of certainty in estimating projects or features, manual tasks that could be automated (Hibbs et al., 2009), and forced conformance to centralized process checklists of “quality” tasks (Larman and Vodde, 2009). Extra processes have a demotivating impact on people who are forced to comply with non-value adding processes.

Unused Employee Creativity

Deming wrote that the “*greatest waste in America is failure to use the ability of people. Money and time spent for training will be ineffective unless inhibitors to good work are removed*” (Deming, 1986). Liker (2004) identified *unused employee creativity* as the eighth waste of the TPS and lean. Unmet human potential refers to the waste of not using or fostering people’s skills and abilities to their full potential. Unmet human potential impedes the flow of work through the system in many ways. Generally, there is an opportunity cost through failing to reach the potential capability of the system. The flow of work, and the associated value created, is neither as effective nor as efficient as it could be. Poppendieck and Poppendieck (2007) describe the problem of not engaging people in the development process. Research into motivation has shown that engagement through a sense of purpose, combined with the opportunity to develop one’s skills and abilities, are essential ingredients in fostering intrinsic motivation (Pink, 2010).

Other

This study remains open to identifying other categories of impediments during data collection and analysis. Although Ohno initially characterized seven types, in reality there are more than seven (Ohno, 1988, Womack and Jones, 2003). For example, Womack and Jones (2003) added *the design of goods and services which do not meet users' needs*. Poppendieck and Poppendieck (2010) describe *policy-driven waste*. Other researchers derive a similar but different set of impediments from those in this study. For example, Sedano et al. (2017) identify nine types: building the wrong feature or product, mismanaging the backlog, rework, unnecessarily complex solutions, extraneous cognitive load, psychological distress, waiting/multitasking, knowledge loss, and ineffective communication. However, these specific examples are not widespread in the literature so will not be discussed further here.

2.5.2 Contributing Factors to Impediments to Flow

This section reviews the most commonly cited factors that contribute to impediments occurring in organizations. This study identifies nine common factors that contribute to impediments in organizations. These are *overburden*, *unevenness*, *work-in-process (WIP)*, *batches*, *queues*, *failure demand*, *technical debt*, and *complexity*. These are, in effect, the levers organizations use to manage and optimize flow in organizations. These contributing factors were either explicitly stated in limited or singular research studies as impeding flow in organizations or were the most commonly cited factors. These contributing factors can multiply and reinforce each other. For example, *queues* are a direct cause of *high WIP* (Reinertsen, 2009). *Complexity* and *technical debt* lead to *failure demand* and *context switching* (Poppendieck and Poppendieck, 2010). *Failure demand* contributes to *unevenness in process or demand* and *overburden* (Seddon, 2005, Anderson, 2010, Poppendieck and Poppendieck, 2010). Note that other studies take a different perspective by researching sources of impediments. For example, Ikonen et al. (2010) articulate specific sources for each of seven categories of impediments observed in a university

laboratory setting. Graebisch et al. (2007) study waste “drivers” in the context of information flows. Oppenheim (2004) identifies nine “common-knowledge” reasons for waste in product development.

Overburden on the System (*muri*)

Ohno (1988) identified *overburden on the system* as the major contributing factor to impediments to flow in the TPS. Overburden in the system of work refers to activities that place unreasonable stress on people or equipment. Spear (2004) suggests that the semantic difference is significant: focusing on waste suggests that the person is the problem, whereas focusing on overburden places emphasis on the design of the system of work. Spear (2004) describes a case study where starting with focusing on overburden rather than on waste helped get buy-in from people. Lapinski et al. (2006) show that identification, elimination and sequencing of overburdening activities have been proven to reduce bottlenecks and improve process flow.

Unevenness in Process or Demand (*mura*)

Overburden is closely followed by *unevenness in processing or demand* as the next major contributor to impediments (Ohno, 1988). *Mura* refers to unevenness in processes or demand not caused by a customer. *Mura* is the uneven flow of work through the system, caused by the system of work within which organizations operate. Highlighting the potential impact of unevenness, Womack (2006) notes that with unevenness, or, the absence of basic stability, “*it is more likely that there will be no flow rather than continuous flow*”.

Context Switching

Context switching occurs when people or teams divide their attention between more than one activity at a time (Reinertsen, 2009). Context switching impedes the flow of work through the system by adding to the overall cycle time from request or idea to delivered product or service, and by causing failure demand and relearning. In human systems the time that gets wasted is significant, but there are other costs associated with context

switching. These include the opportunity cost associated with the interruption, as well as morale costs. People have reported dissatisfaction with repeated context switching because it does not allow them to properly engage with the work, prevents them from contributing their best work, prevents them from developing mastery of their skills, and contributes to feelings of guilt because they feel that they are letting down team members by not completing tasks or taking longer than they committed.

Large Batches

Working with batch sizes is a great starting point for lean product development (Reinertsen, 2009). Ideally, there would be just a single item in a batch. In reality, there are good economic and practical reasons for having more than one item in a batch. Large batches, where “large” is a relative term considered to be higher than the optimal batch size, impede the smooth flow of work through the system because a batch cannot flow through the system from one state to the next until all of its constituent items are ready to move. Forsgren et al. (2018a) note that large batches and infrequent changes introduce risk to the development of today’s increasingly complex systems. Examples of batches in the context of software development include the features in a product release, the user stories in a release backlog, the user stories in a Sprint backlog, multiple features deployed simultaneously in a Cloud environment, or a set of features comprising a minimum viable product (MVP).

High WIP

WIP limits are an important element for achieving flow in a software development context (Rodríguez et al., 2014). The path to enable work to flow to customers is impeded by having a high amount of work in process. The mantra “*stop starting, start finishing*” evolved in the Kanban community (Burrows, 2014), and emphasizes the point that organizations should limit WIP by focusing on completing active work before starting new work. Product development organizations are largely unaware of the problems of high WIP, and so do not measure it (Reinertsen, 2009). High

WIP leads to longer cycle times, which leads to slower throughput, which means an overall slower pace of innovation and delivering customer value (Reinertsen, 2009). One reason that product development organizations do not measure WIP is because the inventory in software development is invisible, as discussed in section 2.3.

Queues

Queues are the single biggest cause of poor product development performance, yet most product development organizations remain unaware of them (Reinertsen, 2009). Conceptualizing the value stream states as *queues* provides a rich set of tools for understanding flow. Ideally, there would be no work ever waiting in *queues*. In reality, there are multiple points in a value stream where *queues* form. These *queues*, by definition, impede the smooth flow of work through the system. Large *queues* are also an indicator of *overburden*. *Queues* increase cycle time, risk, and variability (resulting in increased *mura*, or *unevenness in process or demand*), while decreasing *efficiency*, *quality*, and *morale* (Reinertsen, 2009). From a management perspective, a flow perspective encourages managing *queues* (within a guiding economic framework) rather than managing time lines (Reinertsen, 2009).

Failure Demand

Failure demand refers to the demand placed on systems (including organizations) and is “*demand caused by a failure to do something or do something right for the customer*” (Seddon, 2005). It is the opposite of value demand, where the demand on systems is driven by value-adding work. Poppendieck and Poppendieck (2010) define it as “*the demand on the resources of an organization caused by its own failures*”. It impedes flow by consuming time and effort that could be spent on value-adding work. Failure Demand includes what TPS calls rework (Ohno, 1988). Anderson (2010) refers to *failure demand* as “*Failure Load*”. *Failure demand* in many organizations can initially be as high as 50-80%, meaning that up to 80% of an organization’s capacity can be consumed by work related to errors,

mistakes, poor quality, poor usability, poor service, or requests that come through customer support organizations (Seddon, 2005). Examples include defects, forced rework, technical debt, incomplete features, incorrect features, poor customer service, poor design, and poor or insufficient documentation. Eliminating *failure demand* has a large economic benefit. In the financial services sector *failure demand* can vary from 20% to 45% of demand (Seddon, 2005). Impediments occur when a support team places demands on a development team. Products that are difficult to integrate, deploy, or configure all create large amounts of failure demand. If the software “*gives operations and support organizations problems, both you and they are wasting valuable time.*” (Poppendieck and Poppendieck, 2010). Removing *failure demand* can lead to enormous productivity improvements. Studies show developers spend, on average, 13.5 hours per week on technical debt and a further 17.3 hours per week on bad code, errors, debugging, refactoring, and modifying bad code (Stripe, 2018), all of which can either be classified as *failure demand* or the result of *failure demand*.

Technical Debt

Poppendieck and Poppendieck (2010) identify technical debt as a contributor to impediments. Technical debt is mostly invisible, but manifests in at least two ways: difficulty, and additional cost in evolving the system (Ozkaya et al., 2019). *Technical debt* can waste on average 23% of developers’ time, and this wasted time negatively affects productivity (Besker et al., 2019). There are many forms of technical debt, including architecture, code, and infrastructure debt (Ozkaya et al., 2019). Technical debt makes software more difficult to change (Poppendieck and Poppendieck, 2010), which in turn leads to *context switching* and *failure demand*. Technical debt can also result in more *defects* and *extra processes* (Guo et al., 2016), as well as impacting on *quality, morale, productivity, predictability* (Besker et al., 2019), *time*, and *profits* (Ozkaya et al., 2019).

Complexity

Poppendieck and Poppendieck (2007) identify complexity as a significant contributor to impediments, noting the cost of complexity in software development does not increase linearly. Complexity and its associated costs contribute to *defects*, *extra processes*, and *task switching*, as well as impacting on *quality* and *lifecycle profits*.

2.5.3 Effects of Impediments

Impediments have an effect on flow, and on other aspects of an organization. This study identifies six common effects of impediments in organizations. These effects were either explicitly stated in limited or singular research studies as effects of impediments or were the most commonly cited effects. The six effects cited are *productivity*, *predictability*, *quality*, *time*, *motivation*, and *lifecycle profit*.

Productivity

Ohno (1988) describes the effects of impediments on the productivity of organizations as they try to achieve a smooth flow along a value stream. Shewart (1931) was one of the early observers to note that there is a direct correlation between quality and productivity. Improved productivity is a goal for adopting agile methods for 51% of respondents in one survey, while “accelerated software delivery” is a goal for 74% of respondents (CollabNet VersionOne, 2019). Note that this study is concerned with the productivity of the software development organization, not the productivity of individuals.

Predictability

Ohno (1988) further observes the effects of impediments on the predictability of flow in a value stream. The contributing factors described in section 2.5.2 also impact on predictability. For example, Ohno (1988) describes the effects of unevenness in process or demand, while *Besker et al. (2019)* show that *technical debt* can impact on *predictability*. In general,

predictability is an outcome of flow through a value stream. Improved predictability is a goal for adopting agile methods for 43% of respondents in one survey (CollabNet VersionOne, 2019).

Quality

Quality in software products is “*the outcome of meeting the goals, requirements, and actual needs of the users*” (Putnam and Myers, 2003). Ackoff (1994) notes that “*quality ought to contain a notion of value, not merely efficiency. That’s the difference between efficiency and effectiveness. Quality ought to be directed at effectiveness*”. Hence, quality is a positive concept, but difficult to measure directly, so various quality attributes are used as a means of referring to the different dimensions of software quality (Barbacci et al., 1995, Chandrasekar et al., 2014, Moses, 2009, Gorton, 2011). Typical quality attributes of a software system include availability, efficiency, installability, integrity, interoperability, modifiability, expandability, performance, resilience, flexibility, maintainability, portability, reliability, reusability, robustness, safety, scalability, security, usability, and verifiability (Putnam and Myers, 2003, Pinciroli, 2016). These quality attributes cannot, in general, be ‘counted in’ they instead must be ‘designed in’ to the product” (Putnam and Myers, 2003). This fundamental philosophy of “*build quality in*” is fundamental to TPS (Ohno, 1988), lean (Womack and Jones, 2003), and contemporary software development (Poppendieck and Poppendieck, 2010). Effects of impediments can generally be felt by their impact on one or more of these quality attributes. Hoyer and Hoyer (2001) examine the definitions of quality from eight quality gurus, including Crosby, Deming, Feigenbaum, Ishikawa, Juran, Pirsig, Shewart, and Taguchi. In their study, the authors conclude that Shewart’s perspective on quality is the most useful and practical. In his definition of quality, Shewart acknowledges both the subjective and objective nature of quality. He acknowledges the relationship between value and quality, and the difficulty in defining value (discussed in section 2.2.2 above). Improved software quality was a benefit of adopting agile methods for 47% of respondents in one survey (CollabNet VersionOne, 2019).

Time

Ohno (1988) said of TPS: “*All we are doing is looking at the timeline from the moment a customer gives us an order to the point when we collect the cash. And we are reducing that timeline by removing the nonvalue-added wastes.*” One focus for identifying waste in TPS was to counter their effects on the overall timeline (Ohno, 1988). All of the impediments identified in section 2.5.1 have the effect of increasing the timeline from the start of the value stream to the end.

Morale

Liker (2004) noted the effect of *unused employee creativity* (section 2.5.1) on *morale*. One study shows that overburden is the biggest contributing factor to developer *productivity* and *morale* (Stripe, 2018). In addition, other factors from that study are shown to have a negative impact on developer morale, including work overload (*overburden*), changing priorities resulting in discarded code or time wasted, not being given sufficient time to fix poor quality code, spending too much time on legacy systems, and paying down technical debt (Stripe, 2018). Improved team morale was a benefit of adopting agile methods for 64% of respondents in one survey, making it the fourth-highest of 13 benefits articulated (CollabNet VersionOne, 2019).

Life-Cycle Profit

Reinertsen (2009) discusses the impact of impediments on *life-cycle profit*. *Life-cycle profits* are what product development organizations really care about, but find it difficult to measure directly, so they substitute proxy variables such as cycle time (Reinertsen, 2009).

2.6 Summary

Section 2.2.1 describes the origins of the Toyota Production System and its influence on lean product development. Section 2.2.2 describes five core principles of lean and section 2.2.3 discussed how lean has evolved from a focus on manufacturing and linear process flows and is now embraced by

multiple domains including healthcare, construction, and software development. Section 2.3 then reviews the current state of contemporary software development, noting the impact of agile and lean methods on the evolution of software development. Section 2.4 describes the current trend in contemporary software development as being increasingly towards flow-based development, which has its roots in TPS and lean. Many contemporary approaches, including continuous delivery, DevOps, Lean Startup, Chaos Engineering, and Site Reliability Engineering are all rooted in lean and embrace flow as a means of managing the ever-increasing complexity of software development. Next, section 2.5 discusses the concept of impediments to flow. The lean principles (section 2.2.2) inform the steps to enable flow (section 2.4), which in turn help understand flow and impediments (section 2.5). This relationship is summarized in Table 2.4.

Table 2.4 Relating impediments to lean principles and steps to enable flow

Lean Principles <i>(Womack and Jones, 2003)</i> (Section 2.2.2)	Steps to Enable Flow <i>(Womack et al. 2007)</i> (Section 2.4)	Understanding Flow and Impediments (Section 2.5)
Specify Value	[Step 1]: Focus on the actual object of work, from the beginning of the process to completion.	What do customers want and need? What is it the organization is producing to meet that need?
Identify the Value Stream		What value-creating steps produce what customers want?
Make the Value-Creating Steps Flow		How do organizations make the value-creating steps flow?
Pull	[Step 2]: Ignore traditional boundaries of jobs, careers, functions, departments, and even firms, to create a lean enterprise. Then, remove all impediments to the continuous flow of the product.	What impediments to flow emerge as customers pull value from the system?
Perfection	[Step 3]: Rethink specific work practices and tools to stop “backflow, scrap, and stoppages of all sorts” to enable a continuous flow of work and continuously resolve impediments to flow.	How do organizations continually remove impediments in order to enable an even flow?

Table 2.5 is a summary of the common categories of impediments (section 2.5.1), common contributing factors to impediments (section 2.5.2), and effects of impediments (section 2.5.3).

Table 2.5 Common categories of impediments, contributing factors, and effects

Common Categories of Impediments to Flow (section 2.5.1)	
Extra Features	Poppendieck and Poppendieck (2007)
Handoffs	Poppendieck and Poppendieck (2007)
Defects	Poppendieck and Poppendieck (2007)
Delays	Poppendieck and Poppendieck (2007)
Partially Done Work	Poppendieck and Poppendieck (2007)
Task Switching	Poppendieck and Poppendieck (2007)
Extra Processes	Poppendieck and Poppendieck (2003)
Unused Employee Creativity	Liker (2004)
Contributing Factors to Impediments (section 2.5.2)	
Overburden on the system (muri)	Ohno (1988)
Unevenness in process or demand (mura)	Ohno (1988)
Context Switching	Reinertsen (2009)
Large Batches	Reinertsen (2009)
High WIP	Reinertsen (2009)
Long Queues	Reinertsen (2009)
Failure Demand	Seddon (2005)
Technical Debt	Poppendieck and Poppendieck (2007)
Complexity	Poppendieck and Poppendieck (2007)
Effects of Impediments (section 2.5.3)	
Productivity	Ohno (1988)
Predictability	Ohno (1988)
Quality	Ohno (1988)
Time	Ohno (1988)
Morale	Liker (2004)
Lifecycle Profit	Reinertsen (2009)

This chapter has started to develop the conceptual framework for this study that will help to address the research objectives from section 1.3. Each chapter of the literature review adds detail to the impediment management framework. Figure 2-3 shows the core contributions to the conceptual framework from this chapter, including a reference to the relevant sections from this chapter where the concept is discussed. The primary concepts from this chapter that contribute to the conceptual framework are:

- The concept of flow and impediments to flow, including common factors that contribute to impediments, and commonly cited effects of impediments.
- The initial version of an impediment management framework that proposes to improve flow by addressing impediments and their contributing factors.

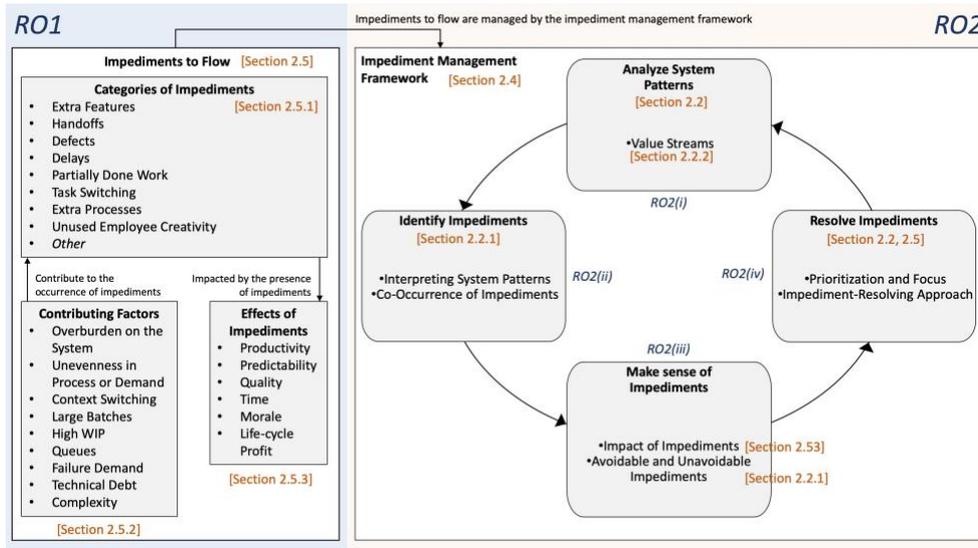


Figure 2-3 Contributions to the conceptual framework from chapter 2

Chapter 3 and Chapter 4 will complete this framework. Next, Chapter 3 reviews the literature related to metrics for understanding flow and impediments in organizations.

Chapter 3 Metrics for Understanding Flow and Impediments

3.1 Introduction and Chapter Layout

This chapter presents the core metrics used in this study to understand flow and impediments. Section 3.2 discusses the criteria for selecting metrics. Section 3.3 describes throughput, cycle time, and cumulative flow in detail. Finally, section 3.4 summarizes the core concepts drawn from this chapter that contribute to the conceptual framework for this study.

3.2 Selecting Metrics for this Study

The goals of software development organizations employing flow-based development described in section 2.4 have remained largely unchanged through the development of TPS, lean product development, and contemporary software development. What has changed is the context within which organizations are operating. These goals can be summarized as increased *productivity*, improved *predictability*, and faster *responsiveness*. These goals are highlighted as important factors in multiple surveys (CollabNet VersionOne, 2019, Mann et al., 2018, Forsgren et al., 2018a, Stripe, 2018). (Petersen and Wohlin, 2010) show that increased *throughput* and reduced *cycle time* result in higher responsiveness to customer needs. Section 2.5 described how impediments to flow negatively affect the organization's ability to achieve these goals.

Emphasizing the importance of timely metrics and the need for responsiveness, Smith and Reinertsen (1998) note that “[i]n general we need metrics that produce information quickly while the team that must act on them is still in existence. A metric that gives you a score months after the team has disbanded is less useful than once that gives results while the team can still do something to change things”. Measurements in lean software development support different types of decisions, including requirements prioritization, staff allocation, planning, and software process improvement

(Petersen and Wohlin, 2010). Decision makers need metrics that are “good enough” and that don’t depend on absolute accuracy to be useful contributors to decision making and action (Meneely et al., 2012). Basili et al. (1994) recommend determining appropriate metrics based on the goals of the organization.

Reinertsen (2009) describes a number of categories of metrics for understanding flow-based development. These include metrics for understanding queues, batch size, cadence, capacity utilization, feedback, flexibility and flow (Reinertsen, 2009). Feyh and Petersen (2013) conducted a systematic mapping of measures used in lean software development. Of the metrics used in this study, they identify WIP, throughput, and cycle time. They refer to cumulative flow diagrams as a visual indicator, combine WIP and queue size as a single measure, and do not mention batch size. Their study does not include work from contemporary lean product development such as those described by Smith and Reinertsen (1998) or Reinertsen (2009).

When considering these metrics, organizations must define the system boundary, i.e., the points at which the flow process starts and ends (Modig and Ahlstrom, 2013). This boundary is generally the entire value stream (section 2.2.2) but there may be reasons why organizations want to measure a specific segment of the value stream. While metrics can provide rich input to discussions, they do not replace discussion among people. Maurya notes that metrics “*can help you identify where things are going wrong, but they can’t tell you why. You need to talk to people for that.*” (Maurya, 2012).

3.3 Metrics for Understanding Flow and Impediments

This study seeks straightforward metrics that provide insight into the working of the product development organization’s value stream as they work to achieve these goals of increased *productivity*, improved *predictability*, and faster *responsiveness*. Throughput (section 3.3.1), cycle time (section 3.3.2), and cumulative flow (section 3.3.3) meet these needs. Observing and monitoring patterns in these measures offers insights into the

quality of flow in software development organizations, and indicates where in the system impediments might be occurring. These metrics are used to varying degrees in software development organizations. One annual survey shows the prevalence of usage of several metrics in software development organizations (CollabNet VersionOne, 2019, CollabNet VersionOne, 2018, VersionOne, 2017, VersionOne, 2016, VersionOne, 2015). Figure 3-1 shows the usage of *throughput* (noted in the survey as ‘*business value delivered*’), *cycle time*, and *cumulative flow* in software development organizations over a five year period.



Figure 3-1 Industry trends in flow metrics usage from 2015 to 2019

Figure 3-1 shows that the usage of *throughput* has more than doubled in this period from 2015 to 2019. The usage of *cycle time* remains relatively flat. The usage of *cumulative flow* remains relatively low, reflecting the view of Reinertsen (2009) that most product development organizations remain unaware of the benefits of managing queues. Section 6.3.1 presents examples of *throughput*, *cycle time*, and *cumulative flow* from organizations in this study. In particular, section 6.3.1 shows how patterns in these metrics over time were used to detect impediments to flow.

3.3.1 Throughput

Throughput is a measure of the productivity of a software development organization. Section 2.4.1 describes how the highest performing software organizations optimize for throughput (Forsgren et al., 2018a). Software

development organizations need to understand how much value is flowing through their system and reaching customers. They need to understand the outcomes of their product development processes, in terms of how they are delivering value. Productivity is a competitive advantage (Forsgren et al., 2018a); the higher the *throughput* the more productive the system of work. A *throughput analysis* reveals the rate of the flow of work through the system over time. Following the lean principle of focusing on value through a value stream, *throughput* should focus on value delivered to customers, not on activities performed. A *throughput analysis* reveals the rate of the flow of work through the system over time. Anderson (2010) suggests tracking failure demand (or *failure load*) as a separate metric. Combined with demand analysis, this shows how much work is *value demand* vs. *failure demand* (see section 2.5.2). *Throughput* and *demand analysis* can be measured separately, but there is value in using them together. Combining them into a single view of the system shows how much work is flowing through the system over a period of time, and gives insights into whether the focus of the organization is dominated by adding value or responding to problems in the system. A lower-than-desired *throughput* is an indicator of the presence of impediments in the system that result in lower amounts of value delivered to customers. Studies show *throughput* can be used to identify bottlenecks in a software development process (Kupiainen et al., 2015, Staron and Meding, 2011). Ikonen (2011) highlights the relationship between *quality* and *throughput*, noting that improvements in *quality* lead to improved *throughput*. A high *failure demand* is an indicator of the presence of impediments in the system that results in poor quality work, which in turn results in the capacity of the system getting used for non-value adding work. These effects are discussed in section 2.5.3. *Failure demand* that is decreasing over time is an indicator that organizations are resolving impediments. Another perspective on throughput is how frequently an organization can deploy code and how fast it can move from committing code to deploying it (Forsgren et al., 2017). Forsgren et al. (2018a) find that *throughput* enables *stability* in organizations, and vice versa. Another study found that team stability correlates with higher throughput (CA Technologies, 2017). They show an approximate 2:1 difference in

throughput between teams where there is 95% or greater stability in membership, compared to teams where there is 50% or less.

3.3.2 Cycle Time

Cycle time is a measure of the responsiveness of an organization (Nicolette, 2015). It measures how long it takes for individual work items to move through a value stream. It can be measured as “*the time it takes to process an order from the request till the delivery*” (Petersen, 2010). *Cycle time* is an important enabler for competitive innovation; Reinertsen (2009) observes that “*when cycle times are long, innovation occurs so late that it becomes imitation*”. Petersen (2010) notes the need for short cycle times in order to have “*first-mover advantage and to be able to react to changes [in] a fast-paced market*”. *Cycle time* is usually measured in days. The goal is for the end-to-end cycle time to be as short as possible. Much of the lean literature refers to lead time and cycle time, sometimes treating them as interchangeable terms. The distinctions would not contribute to the research objective of this study, so for simplicity, this study will refer to *cycle time*, and not draw distinctions between the two. Anderson (2010) has shown that cycle time is a useful metric for comparing proposed and ongoing process improvements, in part because it is easy to capture and is applicable to all processes. Mann et al. (2018) use continuous integration capability as a leading indicator of whether a software development organization is high performing, noting that feedback cycle times are the most important thing to optimize for because “*when feedback cycle times are short, more iterations can occur, and so quality improves*”. The practices discussed in section 2.3, including DevOps, continuous integration, and automation, all play a role in improving the predictability of software development organizations (Mann et al., 2018).

3.3.3 Cumulative Flow

Cumulative flow shows the cumulative history of work as it moves through each of the states in a value stream. A cumulative flow diagram is “*an area graph that depicts the quantity of work in a given state*” (Anderson, 2010).

Cumulative Flow describes the behavior of queues, and has its roots in queuing theory (Smith and Reinertsen, 1998, Little and Graves, 2008, Little, 1961). It is “cumulative” in the sense that it shows how the changes build on each other over time. It is useful to think of workflow states as queues (section 2.5.2). Cumulative flow helps people in organizations to understand and measure the behavior of queues, and shows trends in queue behavior over time (Reinertsen, 2009). It shows the amount of work in each of the defined workflow states and can incorporate a view of the sizes of all identified queues in the system. A desirable cumulative flow is one that has a smooth flow of work, showing an increasing delivery of value, with short transitions between flow states. *Cumulative flow* is helpful in understanding the effects of process improvement efforts (Nicolette, 2015). Cumulative flow diagrams can be used to guide the identification of root-causes to explain the reasons for the observed bottlenecks (Ali et al., 2016, Petersen et al., 2014, Pernstål et al., 2013).

3.4 Summary

Section 3.3 describes throughput, cycle time, and cumulative flow and discusses their usefulness in understanding flow and impediments in software development organizations. As discussed in section 2.5.2, *queue size*, *batch size* and *WIP* can be considered levers that influence flow and impediments through their effect on *throughput*, *cycle time*, and *flow*. Summarizing from the literature review in the previous sections, the following points articulate the relationships between these six concepts, and how they ultimately influence flow and impediments:

- Higher *batch sizes* lead to longer *queues*
- Longer *queues* lead to higher *WIP*
- Higher *WIP* leads to longer and unpredictable *cycle times*
- Longer *cycle times* lead to slower *throughput* and slower *flow*
- Unpredictable *cycle times* lead to unpredictable *throughput* and uneven *flow*
- Uneven *throughput* leads to uneven and slower *flow*

Table 3.1 summarizes these contributing factors from section 2.5.2, and the flow metrics from this chapter. *Throughput*, *cycle time*, and *cumulative flow* are measures of the efficacy of a software development organization's processes. None of these three can be changed directly, i.e., organizations cannot simply request or demand an increase in *throughput*, a reduction in *cycle time*, or an improvement in *cumulative flow*. In order to effect change in these outcomes, managers must change something about the patterns in the organization that are producing these outcomes. Identifying patterns in flow metrics is a useful way of understanding flow quality and the likely presence or absence of impediments to flow. Figure 3-2 summarizes these relationships.

Table 3.1 Summary of core metrics to understand flow and impediments

Section	Metric	Goal	Flow Relevance	Impediment Indicators
2.5.2	Batch size	Lower is preferred	Refers to (a) the size of an individual flow item, (b) the quantity of items in a given flow state, or (c) an overall count of flow items, e.g., for a project	A high number means a large batch size, which in turn indicates a slow, uneven, discontinuous flow of work, and a high failure demand
	Queue size	Lower is preferred	The number of flow items in a given flow state	A high number indicates a large queue
	Work in Process (WIP)	Lower is preferred	The number of flow items actively being worked on in a given flow state	A high number, with items in the same state for a long time, indicates work is not getting completed
3.3.1	Throughput	Higher is preferred	Quantifies the productivity of the system as the quantity of flow items moving through all the flow states in a given time period	A low number indicates less work is getting through the system; Outliers can also be indicators of impediments
3.3.2	Cycle Time	Shorter is preferred	Elapsed time to complete a work item, usually measured in days or hours; Quantifies the predictability of the system as the time it takes for a work item to pass through all the defined flow states over time	A high number indicates the system is not very responsive; A high variation in cycle times indicates the system is not very predictable; Outliers can also be indicators of impediments
3.3.3	Cumulative Flow	Smooth incline, narrow "in process" band	Cumulative number of work items per queue over time; Shows the degree of smoothness of the flow of work as it moves through the different flow states	A jagged, uneven or discontinuous flow; A wide "in process" band; "In process" queues growing; "done" queues plateauing; Any deviations in the pattern from a smooth continuous flow are indicators of potential impediments

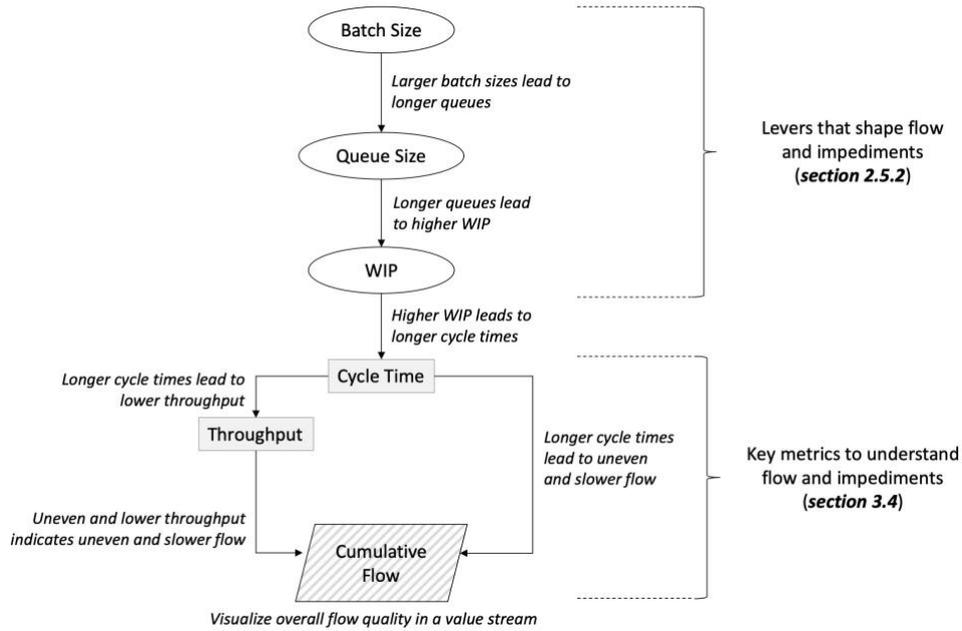


Figure 3-2 Flow metrics and the levers through which to influence them

Figure 3-3 shows the core contributions to the conceptual framework from this chapter and references the relevant sections where the concept is discussed.

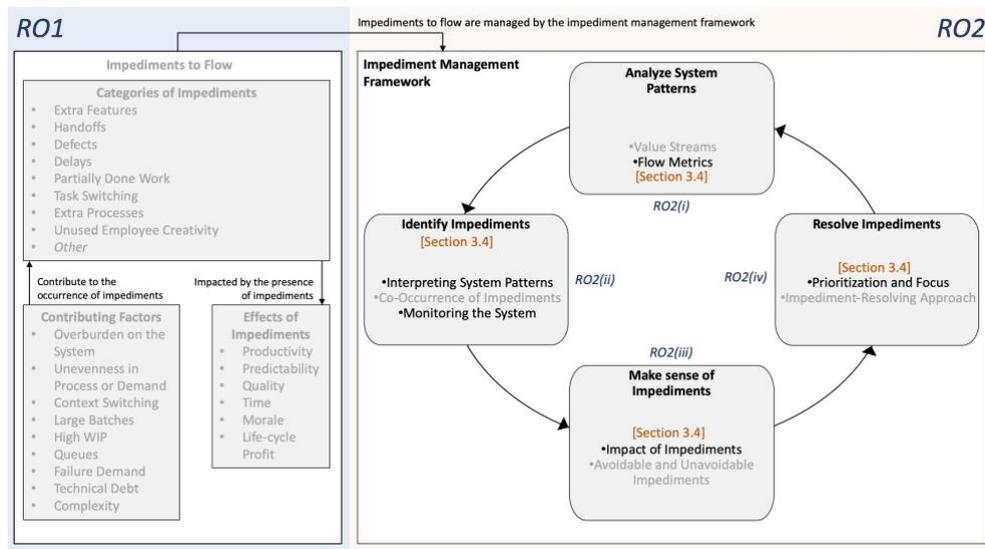


Figure 3-3 Contributions to the conceptual framework from chapter 3

Next, Chapter 4 discusses software development organizations and includes a discussion on complex adaptive systems and sensemaking in the context of this study.

All of the techniques, practices, and approaches discussed in the preceding chapters are evolving to deal with the ever-increasing complexity of contemporary product development. All have the common goal of improving the feedback loop between product developers and the users of their systems in production. A focus on flow-based development demands a rethinking of the systems of management to enable this feedback loop. Catching and resolving impediments quickly is important. At least as important is understanding the system patterns that allow impediments to emerge in the first place and developing a sensory capacity in organizations for detecting the emergence of factors that contribute to impeding flow. It is not enough to simply tackle the impediments. Organizations need a different approach to how they are managing their software development flows.

4.2.2 Organizations are Complex Adaptive Systems

Organizations are complex adaptive systems (Stacey, 1996). A CAS is a “*collection of individual agents who have the freedom to act in unpredictable ways, and whose actions are interconnected such that they produce system-wide patterns*” (Eoyang, 2013, Eoyang and Holladay, 2013). Dooley notes that “*the prevailing paradigm of a given era’s management theories has historically mimicked the prevailing paradigm of that era’s scientific theories*” (Dooley, 1997). Stephen Hawking predicted that the 21st century would be the century of complexity (Chui, 2000). Indeed, the complexity sciences have emerged as one of the prevailing paradigms for modern management thinking (Vasconcelos and Ramirez, 2011). In their work on organizations, Arrow et al. (2000) treat groups as “*adaptive, dynamic systems that are driven by interactions both among group members and between the group and its embedding contexts.*” To fully understand software product development organizations, they are best understood as groups that are “*complex, adaptive, and dynamic systems*” (Arrow et al., 2000). In human complex adaptive systems, the human agents are affected by “*emotion and aspiration, inspiration and anxiety, compassion and avarice, honesty and deception, imagination and curiosity*” (Stacey, 1996). They have their own individual purposes rather than a

shared one, they struggle with the dynamic of conformity and individualism, they are impacted by power differentials between agents, and they are capable of systems thinking, i.e., “*of observing, reflecting upon and altering behaviour according to their perceptions of the operation of the whole system of which they are a part*” (Stacey, 1996).

Meadows (2008) defines a system as “*an interconnected set of elements that is coherently organized in a way that achieves something*”. Systems can be understood as *ordered* and *unordered* (Snowden and Boone, 2007). Ordered systems are generally stable and predictable. People can perceive cause-and-effect relationships, and they can determine “correct” answers based on available information. Unordered systems are comparatively more dynamic and unpredictable. There is no immediately apparent relationship between cause and effect, and the way forward is determined based on emerging patterns (Snowden and Boone, 2007). Many activities and phenomena fall into the context of ordered systems, and many into the context of unordered systems. This is also true for impediments. When organizations encounter impediments to flow, those impediments may or may not be complex. It is often not immediately apparent what is the correct course of action. This study takes the position that it is important, therefore, to understand the context of impediments so that appropriate action can be taken to resolve them. The current literature does not distinguish between impediments with these different characteristics. Section 7.3.3 discusses the implications of understanding these different characteristics of impediments to flow.

4.2.3 Properties of Complex Adaptive Systems

Page (2015) describes four attributes of complex systems: first, they include diverse entities; second, the entities interact within a structure to form a dynamic, perpetually novel network; third, individual behaviors are interdependent, often influencing each other in subtle and imperceptible ways; fourth, the entities adapt and learn.

- **Patterns.** Patterns emerge in self-organizing systems (Dooley and Ven, 1999). In human systems, patterns can be “*ways of doing things*” or

patterns of behavior. Patterns are information in a CAS; rather than static information exchange between individual agents in the system, information takes the form of “*dynamics of patterns*” in the system (Mitchell, 2009). Software product development organizations are “*self-organizing systems in which global patterns emerge from local action and structure subsequent local action*” (Arrow et al., 2000). More than one pattern exists simultaneously, and no single pattern is so dominant that other patterns disappear (Eoyang and Holladay, 2013). This can present a challenge in identifying patterns in complex systems.

- **Coherence.** Coherence in a CAS is the degree to which the patterns across the system are similar, “*without losing the richness of difference at the same time*” (Eoyang and Holladay, 2013). In systems that have high coherence, organizations (and the individuals in those organizations) that comprise the system experience similar patterns across the system, while they might also see small, subtle differences. In systems with low coherence, agents experience different, and perhaps even contradictory, patterns across the system.
- **Constraints and boundaries.** MacIntyre (2007) writes that “*we are always under certain constraints. We enter upon a stage which we did not design and we find ourselves part of an action that was not of our making.*” (Kurtz, 2014). In a CAS, the agents and the system constrain one another, and evolve together over time (Snowden and Boone, 2007). Gatlin (1972) distinguishes between *context-free* constraints and *context-sensitive* constraints (Juarrero, 1999). Context-free constraints are necessary for a system to actually do any work or transmit information (Juarrero, 2009). While diversity is necessary in a complex system (Page, 2007), context-sensitive constraints are necessary as a balancing force so that the system does not become overwhelmed by the variety of available options and can achieve points of equilibrium (Juarrero, 2009). Examples of constraints include planning constraints (Van Beurden et al., 2011), specifications or ways of working (Rikkilä et al., 2013), time (Kurtz, 2014), technology (Martin, 2017),

environmental constraints (Jansson and Axelsson, 2017), budget, rules, processes, and regulations.

- **Emergence.** Emergence is a phenomenon of complex adaptive systems whereby “*well-formulated aggregate behavior arises from localized, individual behavior*” (Miller and Page, 2007). Emergence happens in complex systems in part because such systems are sensitive to initial conditions, i.e., a small fluctuation in one part of the system can bring unexpected, even profound and unpredictable, changes to other parts of the system (Plowman et al., 2007). A key role of leaders is to create the conditions necessary for the emergence of a desired change (Plowman et al., 2007) and to monitor and influence the patterns that emerge (Kurtz and Snowden, 2006). This approach allows organizations to evolve to a future that is more appropriate to their needs, but that was unpredictable or unknowable in advance (Kurtz and Snowden, 2006). Emergence is explicitly referenced in the 11th principle of the agile manifesto, which states that “[*t*]he best architectures, requirements, and designs emerge from self-organizing teams” (Beck et al., 2001). In other words, the best architectures, requirements, and designs (the ones that are often unpredictable or unknowable in advance, but are contextually more appropriate to present needs when they are arrived at) emerge from careful attention to the patterns in the system, and carefully managing the system of work so those conditions are conducive to the best architectures, requirements, and designs emerging.
- **Self-Organization.** Meadows (2008) defines self-organization as “*the ability of a system to structure itself, to create new structure, to learn, or diversify*”. Self-organization is “*the spontaneous generation of order in a complex adaptive system*” (Eoyang, 2001). It is explicitly referenced in the 11th principle of the agile manifesto, which states that “[*t*]he best architectures, requirements, and designs emerge from self-organizing teams” (Beck et al., 2001). There are specific conditions necessary for self-organization to occur. Three conditions serve as meta-variables that shape the “speed, path, and outcomes of self-organizing processes in human systems” (Eoyang, 2001). First, *containers* bound the system of

focus and constrain the probability of contact among agents. Second, significant *differences* establish the potential for change within the system. Third, transforming *exchanges* connects agents to each other through a transfer of information, energy, or material (Eoyang, 2001). Therefore, to enable sustainable self-organization, organizations must pay careful attention to the design of containers, differences, and exchanges in the system.

- **Attractors.** Attractors are points of stability (which can be fixed, periodic, or chaotic/strange) in complex adaptive systems (Mitchell, 2009). They are called *attractors* because any initial condition in the system will eventually be ‘*attracted*’ to it (Mitchell, 2009). They are formed when trajectories converge on typical patterns (Juarrero, 1999). Attractors can arise when small stimuli and probes resonate with people (Snowden and Boone, 2007). Attractors provide structure and coherence in a system as they gain momentum (Snowden and Boone, 2007). From an organization perspective, Thietart and Forgues (1995) note that the fractal nature of attractors means that similar structure patterns are found at organizational, unit, group, and individual levels.
- **Edge of Chaos.** The edge of chaos (Lewin, 1992) lies between order and chaos and is a compromise between structure and surprise (Vidgen and Wang, 2006). Systems at the edge of chaos “*can spontaneously self-organize into new structures*” (Dooley, 1997).
- **Nonlinearity.** Meadows (2008) defines a nonlinear relationship as “*one in which the cause does not produce a proportional effect*”. Snowden and Boone (2007) note that the interactions in a CAS are nonlinear, and minor changes can produce disproportionately major consequences.
- **Obliquity.** Change in a complex system is enacted both directly and indirectly. Obliquity is the process of indirectly achieving complex objectives, or solving complex problems (Kay, 2011). Agents in a system effect change through “*taking direct actions that alter one another’s worlds and indirect actions ... that set the stage for the future*” (Miller and Page, 2007). Regarding objectives, goals, states, and actions in a complex environment, Kay (2011) notes that they “*evolve together*

because learning about the nature of high-level goals is a never-ending process". "Boundaries and constraints can be set, and tension can be created, but influencing to sensitivity and path-dependence can be done only indirectly by influencing the underlying mechanisms." (Rikkilä et al., 2013). 96% of C-level executives say that it is a priority of upper management to increase the productivity of developers (Stripe, 2018). However, increasing productivity is a complex problem; from a CAS perspective, productivity is an outcome that depends on the dynamic nature of the system (Page, 2015). The most effective approach to improve productivity is to address the system patterns that are resulting in lower-than-desired performance.

From these properties, Arrow et al. (2000) define a group such as a software product development organization as "*a complex, adaptive, dynamic, coordinated, and bounded set of patterned relations among members, tasks, and tools*". Snowden and Boone (2007) have shown that "*most situations and decisions in organizations are complex because some major change - a bad quarter, a shift in management, a merger or acquisition - introduces unpredictability and flux*". However, not everything in a complex system is complex (Page, 2015). Systems that possess these attributes can produce complexity, but not everything they produce is necessarily complex (Page, 2015).

4.2.4 Dynamics at Different Levels of a System

Dynamics occur at different levels in a complex system (Eoyang, 2001, Page, 2015). Mitchell (2009) refers to them as "macroscopic" and "microscopic". Page (2015) refers to these broadly as "macro" and "micro" levels. This micro-macro link plays a central role in some contemporary approaches to understanding organizations (Little, 2012). Little (2012) might characterize the product development organization as a "meso" level abstraction. This study is concerned in particular with the level of the product development organization as the meso level of focus, with people as the social actors at a micro level, and its containing entity (company or business unit) as the macro level. It is important to be deliberate about the

level of focus when working with systems (Mitchell, 2009). Other levels exist above these (community, geographical, political, governmental) and below these (work group, team, individual) (Little, 2012). As discussed in section 1.4, this study selects the product development organization as the unit of analysis (Yin, 2016), so for this purpose, it serves as a meso-level abstraction. These levels are also referred as the *part, whole, and greater whole* in the context of analyzing patterns in a CAS (Eckstein, 2015, Eoyang and Holladay, 2013). Events at lower levels lead to higher-level changes, and those emergent higher-level structures subsequently constrain lower-level activity (Arrow et al., 2000). In the context of this study, the software development organization is the *whole* that is under study; the people, teams, and other entities that comprise the software development organization are the *parts*; the company or business unit of which it is part is the *greater whole*. This study shows that while impediments are generally encountered by product development organizations at this meso level, they often originate at the macro level. Understanding the system patterns at a macro level helps to identify the contributing factors that lead to impediments. Identifying system-wide patterns at the macro level also helps identify system-wide impediments that might be impacting multiple product development organizations. Making sense of impediments includes understanding these patterns and contributing factors as part of the impediments' context. This contributes to designing context-appropriate and system-aware interventions to resolve impediments.

4.2.5 Organizations are Impermanent Social Structures

No business today is static (McCord, 2017). Organizations themselves are impermanent structures (Chia, 2003, Weick, 2009). Value streams exist temporarily within and across these impermanent structures. Some organizations are designed to be impermanent, e.g., emergency response teams or flight crews (Jennex, 2007, DeVita et al., 2004, Smith, 1994). The Silicon Valley venture capitalist culture is one where “*venture-financed companies are built to sell, not to last*” (Burlingham, 2015). Other organizations have impermanence thrust upon them. The average lifespan of

today's global multinational Fortune-500 sized companies is 40-50 years (Goodburn, 2015). The average lifespan of a company listed in the S&P 500 index of leading US companies has decreased from 67 years in the 1920s to just 15 years in 2012 (Gittleston, 2012). Globally, 96% of businesses fail within 10 years (Carmody, 2015), and 8 out of 10 businesses fail in the first 18 months (Wagner, 2013). Impermanence also comes in the form of mergers and acquisitions, which are increasing worldwide, both in quantity and value (IMAA, 2019). More than \$2.5 trillion in mergers were announced in the first half of 2018 alone, putting 2018 on track to surpass the largest total on record (Grocer, 2018). The pace continued in the second half of 2018 *“as companies race to remake themselves in the face of sweeping technological change”* (Platt, 2018). Japan has some exceptions to this global trend, having more than 20,000 companies that are more than 100 years old, and some that are more than 1,000 years old (Gittleston, 2012). There are some reasons cited for their longevity. They tend to be small, mostly family-run businesses; their focus is not solely tied to profit; and their corporate culture has avoided the mergers and acquisitions that are common with Western companies. Building a company that can last for 100 years requires managers taking a long-term perspective that considers the people who will come after them and *“bequeathing them a culture, a management system, and an organizational structure that are set up for the long haul and not reliant on any one individual”* (Burlingham, 2015). Even for companies that last a comparatively long time there is ongoing change within and around the organizations that are part of these companies. Business units are created and concluded; product lifecycles are started and ended; teams are formed and disbanded; people join and leave organizations. This study's approach to understanding flow and impediments in organizations is to acknowledge this impermanence, and to consider organizations as being impermanent rather than fixed and unchanging.

4.2.6 Culture in Software Development Organizations is a CAS

Culture is a complex adaptive system (Frank and Fahrback, 1999, Page, 2015, Schein, 2010, Schein, 1996). Page (2015) says that organization culture exists at the macro level of an organization but cannot be maintained by the individual people in the system. Culture is a complex phenomenon that operates simultaneously at three levels of visibility (Schein, 2010, Schein, 1996). The three levels are artifacts, espoused beliefs and values, and basic underlying assumptions. Flow and impediments are “*manifestations*” of the organization’s culture, according to Schein’s three levels. The impacts of impediments articulated in section 2.5.3 (productivity, predictability, reliability, quality, etc.) are also manifestations of the organization’s culture.

Culture is a result of an organization’s management system (Mann, 2014). Schein (2010) notes that culture and leadership are two sides of the same coin. To effect improvements in an organization, such as improving flow, is to attend to the culture of the organization. All of the approaches discussed in Chapter 2 relate to improving the culture of an organization. From Deming, TPS, and lean, to agile development, lean software development, DevOps, continuous delivery, and flow-based development, a common thread is an emphasis on improving the culture of organizations so that they can achieve more desirable outcomes. Iivari and Huisman (2007) have shown that organization culture has an effect on the deployment of new systems development methodologies in organizations. Deming highlighted the importance of a systems perspective on culture when he noted that good people in a bad system will always be defeated by the system (Deming, 2013, Walton, 1994). Organization culture is the biggest barrier to adopting and scaling new methods in organizations (CollabNet VersionOne, 2019). Organization culture and leadership, therefore, are significant factors in how successfully flow can be established in an organization. The findings in section 6.3.1 include insights into the culture of the organizations in this study that relate to impediments to flow.

This study takes place in the context of large and complex product development organizations where flow is dependent on the effective collaboration of many people. A focus on people is a common thread through all of the approaches discussed in this chapter, from Deming, TPS, and lean, to all of the contemporary software development approaches. TPS and lean are about developing people as much as they are about delivering products (Steers and Shim, 2013, Liker et al., 2008). Ohno recognized the essential place of people in the TPS. He coined the term *autonomation* to recognize that although automation is essential to increased productivity and quality, it can never fully replace people in the system (Ohno, 2013, Ohno, 1988). (Cockburn, 2000) notes that people are “*highly variable and non-linear, with unique success and failure modes*”; failure to account for this contributes to all sorts of problems in software development organizations. Agile methods focus on people, with the first of the four values emphasizing the importance of “people and interactions”. The manifesto emerged as a response to a set of circumstances where people were devalued as individuals, and treated as fungible “resources” (DeMarco, 2001), and companies were prioritizing heavy-weight processes and tools. Cockburn and Highsmith (2001) note that agile is “*for people*”, saying that the most important implication for managers working with agile approaches is that “*it places more emphasis on people factors in the project: amicability, talent, skill, and communication.*” This increased recognition of the centrality of people as individuals brings challenges. Modern agile approaches emphasize the need for psychological safety for people in organizations, and the influence of psychological safety on productivity and innovation (Kerievsky, 2016a, Edmondson, 1999). Research by Graziotin et al. (2014) provides a set of validating measures to show that happy developers are better problem solvers. Psychological safety has been shown to be a significant determining factor in how people occupy their roles (Edmondson et al., 2004).

The language used in organizations to discuss mistakes, failures, and errors both influences the culture and is a reflection of the culture. Managers need to change the language in order to change the culture (Schein, 2010).

Conboy et al. (2011) summarize the challenges from case studies of 17 companies into nine specific challenges, ranging from developer fear to recruitment policies. In another study Dennehy and Conboy (2018) identify “*a historical culture of fear and fear of change*” as a significant factor in one organization, contributing to that organization’s inability to create flow. The issues of psychological safety, fear, and blame are complex and nuanced. Policy making bodies and experts in medical error have called for a shift in perspective to a blame-free culture, recognizing that “*errors are largely attributable to systems rather than individuals*” (Collins et al., 2009). This is a recognition that in the vast majority of cases it is the system and not the individual that is at fault. This recognition relates to the point made in section 2.5, that in 95% of cases the problem is the system, not the person (Deming, 1986). However, Collins et al. (2009) suggest that in many cases physicians see blame as valuable because it leads to “*improving practice, accepting responsibility, forgiving others and achieving control*”. They identify three types of blame: self-blame, blame the system, and blame of others. Rather than a no-blame culture, they advocate for a “*fair-blame culture*” in which a systems approach to error is balanced by individual responsibility. Accountability is “*the willingness of team members to remind one another when they are not living up to the performance standards of the group*” (Lencioni, 2006). Recognizing that a one-dimensional perspective on blame-free cultures carries its own risks to safety, many health care organizations strive to balance “no-blame” with accountability (Wachter and Pronovost, 2009). Marx (2008) promotes the idea of a “just culture” (Dekker, 2016) differentiating between “blame-worthy” and “blameless” acts. Wachter and Pronovost (2009) note the importance of deeming some behaviors as unacceptable, with clear consequences. “*While regretting them, we must all learn ‘to treasure mistakes,’ because of what they can teach us for the future. This calls for an extremely mature organisation and equally, a mature society. It means an abandonment of the easy language of blame in favour of a commitment to understand and learn. It calls for significant leadership. It calls for practical action geared to be more open about error and mistakes.*” (Weick and Sutcliffe, 2011). This need for leadership to create an environment where the organization can learn from mistakes

without fear was emphasized in the work of Deming (Deming, 2013, Walton, 1994) and TPS (Liker et al., 2008). Although not the only source of impediments, flow can be interrupted due to mistakes and errors. How people in organizations deal with that reality is a reflection of their culture and will influence their ability to report on and respond to these impediments. The ways organizations develop expectations for how these situations will be managed is an expression of culture (Weick and Sutcliffe, 2011).

4.3 Sensemaking in Software Development Organizations

4.3.1 Using Sensemaking to Understand Complex Systems

Sensemaking provides a way of exploring and understanding complex systems (Weick, 1995). There are many theories and approaches to studying and understanding organizations. This study chooses sensemaking because of sensemaking's resonance with complex adaptive systems. People in organizations need approaches that help them make decisions and intervene in their systems in contextually appropriate ways (Snowden and Boone, 2007). Sensemaking is a process that helps organizations to "*structure the unknown so as to be able to act in it*" (Ancona, 2012). Section 4.2.5 discussed the influence of organization culture on flow. This is one aspect where sensemaking plays a role in this study. There are many sensemaking frameworks that can help organizations understand their context. This study explores the use of the Cynefin sensemaking framework (Snowden and Boone, 2007) (described in section 4.3.6) to determine an appropriate course of action to resolve impediments by mapping them to one of five *domains* (obvious, complicated, complex, chaos, disorder).

Weick (1995) popularized the term "*sensemaking*" in the context of understanding organizations. Weick (2009) shows that combining the ideas of complexity theory with those of sensemaking theory provides "*a powerful combination to understand thick, dense events that have high stakes*". In the context of this research study, impediments to flow are

examples of these events, many of which can be considered thick or dense, and which have high stakes for the organizations involved. Ancona (2012) advocates sensemaking as a core capability for organizations “*so that we can break through our fears of the unknown and lead in the face of complexity and uncertainty*”. A sensemaking approach can be used to explore impediments that occur within the product development organization, and also those that originate in the wider system, but that are impacting flow for one or more product development organizations. Maitlis and Christianson (2014) suggest connecting sensemaking to other important organization activities. One goal of this research is to connect sensemaking to the activities involved in creating better systems of flow in software development organizations in order to find better ways to deliver value to customers.

There are multiple interrelated schools of thought in the field of sensemaking. This study considers two areas that align closest with the goals of this research. The first originates primarily from the work of Karl Weick and his work in studying organizations. Weick (1995) refers to “sensemaking”. Snowden builds on the work of Weick, and earlier refers to “sense making” (Snowden, 2005b), and later to “sensemaking” (Snowden, 2010). Klein uses the term “sensemaking” referencing the work of both Weick and Snowden (Klein et al., 2006). The second area originates from the work of Brenda Dervin, who developed “Sense-Making” as a specific methodology in the field of communications (Dervin et al., 2003). Dervin’s focus on communications “*encompasses the entire spectrum of human communication*” (Foreman-Wernet, 2003). There are parallels and overlaps between the two. A third area of sensemaking is participative narrative inquiry, or PNI (Kurtz, 2014). PNI is defined as “*an approach in which groups of people participate in gathering and working with raw stories of personal experience in order to make sense of complex situations for better decision making*” (Kurtz, 2014). It draws from narrative inquiry, participatory action research, mixed methods research, oral history, cultural anthropology, folklore studies, narrative therapy, participatory theatre, complexity theory, and decision support. PNI is not as widely cited in the

literature as the other approaches noted earlier, but it does contain some useful insights for sensemaking that are discussed throughout this chapter, in particular those for working with narrative-based sensemaking.

This study identifies with the work of Weick, Snowden, Klein, and related studies of organizations, and uses the term “sensemaking” throughout. This study also draws on the work of Dervin and her Sense Making methodology. Anderson (2006) notes that while Weick’s work is influential in studies of organizations, there are few studies that are critical or involve empirical tests, which presents implications and opportunities for research.

4.3.2 Properties of Sensemaking

Weick (1995) identifies seven properties of sensemaking. The first two relate to the “sensing” part, the next five to the “making” part (Weick, 1995).

- **Grounded in identity construction.** Sensemaking begins with a sensemaker, and recognizes the many identities and perspectives that inhabit each person (Weick, 1995). Weick distills *identifying* and *knowing* into a question referred to as the ‘sensemaking recipe’: “*how can I know what I think until I see what I say?*” (Anderson, 2006). Identities are constructed and re-constructed through interactions (Weick, 1995).
- **Retrospective.** The focus on retrospect implies sensemaking derives from an analysis of people’s “*meaningful lived experience*” (Weick, 1995). This is intentionally phrased in the past tense, emphasizing that people can know what they are doing only after they have done it (Weick, 1995). There are other perspectives that focus sensemaking on “here and now” and future events (Rosile et al., 2013, Boje, 2008). The focus of this study is on retrospective sensemaking.
- **Enactive of sensible environments.** Weick (1995) uses the word “enactment” to emphasize that sensemakers, like legislators or managers, construct reality through acting in their environments. People create their own environments, and these environments then constrain

their actions. Environment is not considered to be a single thing which is separate from the sensemaker.

- **Social.** Sensemaking is a social process, involving a level of analysis that considers the system, not just the individuals in the system. It is never a solitary process because what a person does internally is dependent on others. Human thinking and social functioning are essential aspects of one another (Resnick et al., 1991, Weick, 1995). Sensemaking is also social when people coordinate their actions in some way, e.g., through equivalent meanings, distributed meanings, or overlapping perspectives on ambiguous events.
- **Ongoing.** Weick (1995) notes that sensemaking never starts because pure duration never stops; people are always in the middle of things, which become things only when they stop to focus retrospectively on past events. “*Flows are the constant of sensemaking*” Weick (1995). Sensemaking is sensitive to the ways in which people extract moments from continuous flows and extract cues from those extracted moments. It means recognizing that “*there are no absolute starting points, no self-evident, self-contained certainties on which we can build, because we always find ourselves in the middle of complex situations which we try to disentangle*” through a process of making and revising assumptions (Weick, 1995, Burrell and Morgan, 2017). “*The reality of flows becomes most apparent when that flow is interrupted*” (Weick, 1995). This is true whether the flow relates to value streams in product development organizations (Womack, 2006, Womack and Jones, 2003), or the mental state of flow necessary for creative work (Nakamura and Csikszentmihalyi, 2014, Csikszentmihalyi, 1997, Csikszentmihalyi and LeFevre, 1989). Interruption is a signal that important changes have occurred in the environment (Weick, 1995).
- **Focused on and by extracted cues.** Extracted cues are “*simple, familiar structures that are seeds from which people develop a larger sense of what may be occurring*” (Weick, 1995). A cue establishes a point of reference, which may be different for each person. Selecting a cue is a

“*consequential act*”; cues create a bound on the sensemaking by establishing a focus (Weick, 1995).

- **Driven by plausibility rather than accuracy.** Weick (1995) notes that a reasonable starting point with sensemaking is that “*accuracy is nice, but not necessary*” and provides eight reasons why this is so. The degree of accuracy is determined by different environmental factors, each of which can be sensed differently by different people. Given the time-sensitive nature of organizations, most managers favor speed over accuracy (Weick, 1995). Weick (1995) offers eight reasons why accuracy is less important in organizations:
 1. People can become overwhelmed with too much data, so they distort and filter information in order to make sense of it.
 2. “Objects” (e.g., impediments, requirements, metrics, events) have multiple meanings and significance to different people, so it is more important to get some interpretation to begin with rather than wait for “the” perfect interpretation.
 3. Fast responses shape events as they emerge, before those events harden into a single meaning. Organizations are able to adjust to unfolding events by using minimal cues to respond quickly to events in the environment.
 4. If accuracy becomes important it does so for brief periods of time and in response to specific questions.
 5. The “*interpersonal, interactive, interdependent quality of organizational life*” makes accuracy more difficult to achieve (Weick, 1995).
 6. Accuracy is specific to a context and project and is defined by instrumentality. For example, beliefs that counteract impediments and facilitate ongoing flow and project success are treated as accurate. For this study, enactment in the pursuit of flow provides the frame within which cues are extracted and interpreted (Weick, 1995).
 7. “*People who want to get into action tend to simplify rather than elaborate*”, therefore anything that distracts from an “*energetic, confident, motivated response*” gets filtered out (Weick, 1995).

Action is adaptive because it “*shapes that which is emerging*” (Weick, 1995).

8. It is impossible to tell, at the time when something is being perceived, whether this perception will be accurate. Starbuck and Milliken (1988) show that most perceptual errors “*become erroneous only in retrospect*”.

4.3.3 Distributed Sensemaking across a Value Stream

All the organizations in this study are distributed across multiple locations, with value streams often spanning multiple organizations and locations. Contrary to the “rational organization” perspective, a complexity perspective holds that organizations are loosely-connected to their social environments (Weick, 2005). Weick (2005) quotes Pfeffer and Salancik (1977) who noted that “*organizations are coalitions of various interests; organization designs are frequently unplanned and are basically responses to contests among interests for control over the organization; and organization designs are in part ceremonial.*” These organizations operate in environments of volatility, uncertainty, complexity, and ambiguity (section 4.2.1), and are themselves constantly changing and evolving (section 4.2.5). The official organization chart is often a temporary fiction. The formal depiction of an organization in charts, manuals, job descriptions, and other artefacts may not correspond to how things actually happen (Boesen, 2010). The distribution of real power in the organization is rarely codified (Boesen, 2010). Informal networks of power and influence, and informal flows of work are not uncommon. Value streams are created in this context, and a single value stream can attempt to overlay multiple such loosely connected organizations. Distributed sensemaking generally occurs at the macro level or meso level (see section 4.2.4 above) and produces an emergent view of the situation that none of the individuals involved at the micro level could produce alone (Taylor and Van Every, 2000). Viewing the value stream as a system, the implications for this study are that no single person that is working *in* a value stream possesses a view of the entire value stream; it is only through interconnections between people and groups

across a value stream that a full representation can emerge. Recalling from section 4.2.4 that value streams and software product development organizations exist at the *meso*-level in organizations, seeing the whole requires a *macro*-level perspective. Taylor and Van Every (2000) have shown that the interactive complexity in distributed organizations “*outstrips the capacity of any single individual in the network to represent and discriminate events*”. At a meso level, the patterns get more specific to the specific product development organization and provide richer localized context. At a macro level, the patterns are much broader, potentially impacting multiple product development organizations.

4.3.4 Using Micronarrative as Cues for Sensemaking

A property of sensemaking (section 4.3.2 above) is that sensemaking is focused on and by extracted cues. As defined in section 4.3.2 above, a cue is a simple structure that acts as a seed from which people develop a larger sense of what may be occurring (Weick, 1995). In the context of this study, stories, narrative, and micronarrative fragments are examples of cues from which people in organizations develop a larger sense of what may be occurring in their organizations, and in their value streams. From a complexity theory perspective, narrative and story form part of an organization’s institutional memory, and contribute to the sensemaking ability of the organization (Boje, 2008). Rosile et al. (2013) propose a “*storytelling diamond*” that describes six paradigms of storytelling: narrativist, interpretivist, abstractionist, materialist, practice, and living story. Boje (2008) asserts that narrative has become a centering force of control and order in modern organizations, while story constitutes a decentering force of diversity and disorder.

Story

Weick (1995) notes that story is necessary for sensemaking. As discussed in section 4.3.2, sensemaking is about plausibility, coherence, and reasonableness. Sensemaking is about accounts that are socially acceptable and credible. Story perceives plausibility and coherence, is reasonable and

memorable, embodies past experience and expectations, resonates with other people, can be constructed retrospectively and used prospectively, captures feeling and thought, allows for “*embellishment to fit current oddities*”, and can be fun to construct (Weick, 1995). Stories explain and energize. A good story shows patterns that may already exist (Weick, 1995). This research is not concerned with the familiar Aristotelian form of narrative, with its classic beginning-middle-end form from *Poetics*. There are different styles and formats that are useful in different contexts (Boje, 2008). For example, in contrast to more formal requirements engineering approaches, eXtreme Programming (XP) adopts the story as the unit of functionality (Beck and Fowler, 2001). The intention in XP is that stories are small and written in a natural language, so that they are understandable to customers, and to everyone else involved in the development of the product (Beck and Fowler, 2001). User stories were later adopted by Scrum and other methods (Cohn, 2004). Although many different perspectives and even templates have been put forward over the years, Beck and Fowler (2001) note that “*the best user story is a sentence or two that describes something important to the customer.*” The original intent was that a story represents a concept, and is not a detailed specification; it is a cue that the developers and customers will use to discuss the feature (Beck and Fowler, 2001).

Narrative

Narrative approaches have become increasingly popular in management and organization studies (Vaara et al., 2016). Narrative is the representation of an event or a series of events (Abbott, 2008). Going deeper, narrative is “*first of all a set of signs, which may involve writing, verbal or other sounds, or visual, acted, built, or made elements that similarly convey meaning*” (Squire et al., 2014, Squire et al., 2012). For a set of signs like this to compose a narrative, there needs to be some movement between the signs that generates meaning. Fundamentally, narrative is about change (Kurtz and Snowden, 2003). The best narrative for sensemaking purposes is “*a recounting of events based on emotional experience from a perspective*”

(Kurtz, 2014). Kurtz (2014) deconstructs each part of this statement as follows: “*something happened*” (an event) “*to me*” (an experience) that “*I felt something about*” (emotion) and “*I see it this way*” (perspective). This integrates many of the properties of sensemaking described in section 4.3.2. Narratives, in this context, recount events that conveys experiences, emotions, and perspectives. They shape past events into experience using coherence to achieve believability (Boje, 2008). Narrative plays a key role in expressing the values held by a society or organization (Caracciolo, 2012).

This study is concerned with the experience of organizations that want to improve the flow of value to their customers. Experience in organizations takes narrative form (Kociatkiewicz and Kostera, 2016). Some researchers argue that narrative needs to have more than one event, and even a causal connection between events, though Abbott (2008) argues that the field of narrative research “*is so rich that it would be a mistake to become invested in a more restrictive definition*”. Boje (2008) identifies eight types of sensemaking patterns of narrative coherence. The interplay of these different ways of sensemaking provides organizations with a new understanding of “*complexity, strategy, organization change, and methodology*” (Boje, 2008). These different methods focus on sensemaking of the past, present, and future. This study focuses on sensemaking of past experience and current situation to inform how to act in the near future to improve flow through resolving impediments.

Polysemy is the capacity for a narrative to have multiple meanings (Squire et al., 2014). According to Juarrero-Roqué (1991), this property of polysemia makes narrative more suitable than abstract principles for articulating norms because “*it allows prescriptive texts to change their meaning so as to better map changed social understanding. There is sufficient ambiguity in any narrative to allow for the reinterpretation necessary for resilience*” (Juarrero-Roqué, 1991). This applies not just to narrative constructed to convey a point, but also to the narrative that people naturally tell each other in organizations. Recall from section 4.3.2 that

plausibility is more important than accuracy; it is expected that different people will interpret the same narrative in different ways.

Narrative is distinct from story. Story has multiple elements, including form, function, and phenomenon (Kurtz, 2014). Among other things, phenomenon deals with the context of when and where the story's events take place, and includes interpretations of narrative and narrative events (Kurtz, 2014). This is consistent with the earlier description of narrative as carrying meaning and relating to an event. Some authors on narrative and sensemaking, including Brown et al. (2014), use the terms story and narrative interchangeably. However, for something to be considered a story, it must add elements of form (communicative structure and meaning) and function (utility to thought, decision, and action) (Kurtz, 2014). Denning (2005) states the distinction more succinctly (though perhaps technically incomplete): a story is a narrative that links a set of events in some kind of causal sequence. Narratives shape past events into coherent experiences, while stories are more about disseminating events in the present or shaping the future (Boje, 2008).

Narrative is distinct from description; it may or may not be descriptive, but it always carries some specific meanings (Squire et al., 2014). Abbott (2008) provides a simple but illustrative example: “*my dog has fleas*” is a description; “*my dog was bitten by a flea*” is a narrative. Even though the event in the narrative is a very small one, it is enough to make it a narrative. A corresponding example from software development might be “*our software has defects*” (description) contrasted with “*our biggest customer found a defect in our product*” (narrative). This concept of small narratives, or micronarratives, is central to the use of narrative in this study, as discussed next.

Micronarrative

Boje et al. (2004) identify three levels of engagement with narrative: *micro*, *meso*, and *macro*. These correspond to the levels of systems described in section 4.2.4 above. Capturing narrative at the micro level, e.g., discrete

conversations or recollections, reveals patterns at the meso level through “*broader patterns and networks of organizational interaction*” (Boje et al., 2004). These meso-level narrative patterns contribute to the understanding of “*grand narratives and metadiscourses with wider social implications*” at the macro level. In the framing of this study, *micro-level* is the individuals that are part of a product development organization; *meso-level* is the product development organization; *macro-level* is the broader company and environment of which the product development organization is part. This study adopts the double meaning of micro in this context. The first meaning relates to the level of the system from which the narrative comes, as described above. The second refers to the size of the narrative. This study explores the small fragmented, imperfect micronarrative that while retrospective in nature, shapes the emergent present (Boje, 2008). Micronarratives are similar to the idea of *minimal* narrative (Labov, 2008, Labov and Waletzky, 1997). Micronarratives capture the anecdotes or “small-stories” people tell in their day-to-day social interactions, and reflect how people make sense of the world (Van Der Merwe et al., 2019). Collections of these micro fragments from multiple people can reveal motivations and attitudes in an organization (Van Der Merwe et al., 2019). From the perspective of this study, these small, imperfect, micronarrative fragments provide insight into how an organization manages flow, and deals with impediments, and how these micronarrative fragments shape what the organization is becoming in the emerging present.

Sensemaking is about the enlargement of small cues (Weick, 1995). In this context, a micronarrative can serve as a cue (section 4.3.2). Knowledge extracted from these micronarratives can provide insight into what is informing decisions, interests, principles, and actions in organizations (Caracciolo, 2012, Van Der Merwe et al., 2019). Fragmented narratives relating to historical events have been shown to contrast with official grand narratives (Mannava, 2014). An important advantage of working with micronarrative is that it does not constrain the respondent to provide information within a tightly prescribed framework of questions and answer options (Krentel et al., 2016). Micronarratives are more useful than “big

stories” for sensemaking in organizations (Van Der Merwe et al., 2019). Sensemaking using micronarrative as cues “*provides a mechanism to explore both expected and unexpected themes, using the respondent’s personal experience as the reference point for subsequent closed questions*” (Krentel et al., 2016). These personal micronarrative fragments can be more authentic, providing a richer context that helps people make sense of past events even in difficult circumstances: “*as citizens of a fragmented society we have fallen back onto micro-narratives each having a limited context to understand our world better*” (Mannava, 2014). Boje (2008) notes a shortage of research in retrospective sensemaking using micronarratives (or narrative fragments), particularly research that relates to “*the kinds of disrupted, interrupted, and socially distributed communication that occurs in complex organizations*”. This study contributes to addressing that gap.

4.3.5 Making Sense of Experiences in Organizations

Mann (2014) notes that culture (section 4.2.6 above) arises from experience, i.e., “*our idea of the culture of a place or organization is a result of what we experience there.*” Therefore, leaders need approaches to understand their organization’s culture that are based on understanding the collective experiences of people in the organization. People’s experience of culture in their organization as they attempt to create flow and resolve impediments is “*very contextual, reflecting their lived experience within a defined work structure.*” (Dickens, 2012). This section describes specific aspects of sensemaking that help make sense of people’s experiences in organizations.

Emotion associated with the experience

Emotion is an intrinsic part of the human experience of which sensemakers in organizations are trying to make sense. Section 2.2.2 shows that creating flow along a value stream requires the cooperation of many people. Emotions play a significant role in people’s capacities to share and cooperate with others (Dervin, 1998). The relation between sensemaking, emotion, and the interruption of work can be understood by realizing that a necessary condition for emotion is “*arousal*” or discharge in the autonomic

nervous system which is triggered by interruptions of ongoing activity (Weick, 1995, Kelley et al., 1983, Mandler, 1984). This is the general case of what happens with impediments to flow and their contributing factors discussed in section 2.5: they signal an interruption of ongoing activity in the value stream. Emotion in this context is a signal that warns of something in the system that demands attention; perceiving these triggers is a “*rudimentary act of sensemaking*” (Weick, 1995). A key event for emotion is the “*interruption of an expectation*”; emotion is a reaction to the world no longer being the way it was (Weick, 1995). An impediment to flow, therefore, is an event that signals an interruption of expectation to how the value stream ought to operate. The resulting emotion triggered by this interruption to flow can serve as information that helps to make sense of what is happening. Weick (1995) notes that emotion “*is what happens between the time that an organized sequence is interrupted and the time at which the interruption is removed, or a substitute response is found that allows the sequence to be completed.*” Creating flow along a value stream is an example of such an ordered sequence. Thus, this can be stated in terms of this study as follows: *emotion is what happens between the time that flow along a value stream is interrupted by an impediment and the time at which the impediment is resolved, or a substitute response is found that allows the flow to be completed.* Emotions affect sensemaking because recall and retrospect tend to be congruent with mood (Weick, 1995, Snyder and White, 1982). People reconstruct past events as narrative explanations in the present “*not because they look the same, but because they feel the same*” (Weick, 1995).

Emotions can be positive or negative. Negative emotions are likely when the interruption is unexpected and is interpreted as “*harmful or detrimental*”: “*If there is no means to remove or circumvent the interruption, the negative emotion should become more intense, the longer the interruption lasts.*” (Weick, 1995). Positive emotions are likely when either there is a sudden and unexpected removal of an interrupting stimulus, or events occur that suddenly and unexpectedly accelerate the completion of a plan or behavioral sequence. In other words, negative emotions are likely

when flow is impeded, and the effects are harmful or detrimental; the longer the impediment remains unresolved the more intense the negative emotion. Positive emotions are likely if an impediment or its contributing factors are resolved, or there is an unexpected alternative found to complete the work. Weick (1995) shows that occasions for positive emotions decline over time, but the occasions for negative emotions remain consistently high. Relationships in organizational settings may be short-lived, close, intense, and interdependent so the likelihood of unexpected interruption is higher. Organizational settings, therefore, have the potential to generate stronger emotions, both positive and negative, than other settings. None of the conditions for positive emotions are plentiful in organizational settings. First, people have little control over the occurrence or resolution of interruptions. Second, people tend to experience more rather than fewer interruptions over time. Third, achieving plans is more often delayed than accelerated. Organization culture, as described in section 4.2.6 above, has an influence on all three of these (Weick, 1995, VanMaanen and Kunda, 1989).

Maitlis and Christianson (2014) discuss sensemaking as an emotional process in organizations. They demonstrate that there is a recursive relationship between emotional dynamics of team discussions, and how teams make sense of issues they face. At an organization level, it has been shown that shifting emotions shape people's experiences through sensemaking. Dervin (1998) says that emotions play a role in several important ways in organizations, including acting as a measure of outcomes for human beings, and that "*systems ignore this at their peril.*" Emotion can act as an indicator of psychological safety in organizations, as discussed in section 4.2.5. Dervin (1998) notes that "*people work best when they feel good about themselves and being fearful because of differences in viewpoints usually leads to counterproductive emotions and actions*".

Weick et al. (2005) and Boje (2008) call for more research to include emotion in sensemaking, so that researchers can explore how multiple ways of sensemaking interplay. Since then, there has been a growing interest in understanding emotion as part of sensemaking, at both individual and

collective level (Maitlis and Christianson, 2014). Maitlis and Christianson (2014) call for more research in this area at the organization level. This study answers these calls. Section 6.3.1 presents findings that include emotion as one of several aspects in the context of impediments to flow in organizations. Section 7.3.2 discusses the interplay of these findings with other aspects of sensemaking.

The frequency of occurrence of this type of experience

Analysis of the frequency of occurrence of experiences and their associated emotional content can provide insight into how often interruptions to flow are occurring (Weick, 1995). For this study, these interruptions include impediments to flow.

People and groups involved in the experience

The “*social setting*” is intrinsic to sensemaking; people’s experiences in organizations are often influenced by other people (Weick, 1995). Networks of people in organizations (including value streams) are “*cross-cutting, embedded in each other, and heavily intertwined*” (Schraagen, 2017). MacIntyre (2007) writes about narrative among people: “*I am part of their story, as they are part of mine. The narrative of any one life is part of an interlocking set of narratives.*” (Kurtz, 2014). From the perspective of this study, therefore, it can be useful to know what people, roles, or groups are influencing the experience of people in a value stream.

The recency of the experience

As discussed in section 4.3.2, this study focuses on retrospective sensemaking. When people deem an experience significant, it can be useful to know how recently that experience occurred.

Understanding the impact on the work

Weick (1995) uses the term ‘project’ in a very general sense to refer to the thing that is the focus of people’s work. Sensemaking provides an opportunity to gain insight into the impacts of events on people’s work.

4.3.6 The Cynefin Sensemaking Framework

Maitlis and Christianson (2014) encourage researchers to draw on a wide range of methods to study sensemaking, noting that novel methods have emerged since the inception of sensemaking that allow examining of sensemaking questions. These methods are often embodied in sensemaking frameworks. Using micronarrative sensemaking is an entry point to understanding these collective experiences, and how they relate to flow and impediments. Krentel et al. (2016) demonstrate how micronarrative sensemaking tools are a “*valid and effective tool to detect operational issues.*” Their research shows distributed sensemaking (section 4.3.3) using micronarrative tools have demonstrable benefits where a more in-depth understanding is needed in order to improve product and service delivery.

The Cynefin Framework is a sensemaking framework with five *domains* that helps people to make sense of their environment so they can take appropriate action (Snowden and Boone, 2007). The five domains are *obvious, complicated, complex, chaotic, and disorder*. Figure 4-1 shows the Cynefin framework, adapted from Snowden and Boone (2007), with each of its five domains labeled. Referencing the types of systems discussed in section 4.2 above, the Cynefin Framework divides *ordered* systems into *obvious* and *complicated* and divides unordered systems into *complex* and *chaotic*.

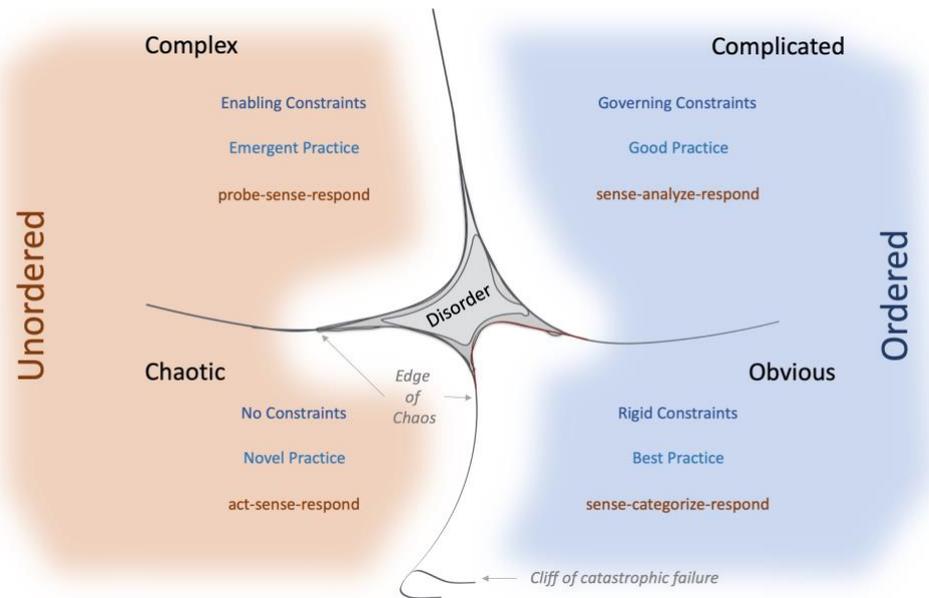


Figure 4-1 The Cynefin sensemaking framework

The *Obvious* Domain. In the *obvious* domain there is usually one “correct” or obvious solution to a problem (or response to a situation), and so the term “best practice” is appropriate for problems in the *obvious* domain (Snowden and Boone, 2007). The way of dealing with impediments in the *obvious* domain is *sense-categorize-respond*: sense the problem, categorize into one of a known set of types, such as the impediment categories identified in section 2.5.1, and then respond appropriately. Note that this domain was originally labeled “*simple*”. It has since evolved to be called the “*obvious*” domain, though some publications still use the old name.

The *Complicated* Domain. The *complicated* domain is the domain of governing constraints. There can be multiple solutions to a problem. An appropriate solution is found through research or consulting experts. This is the domain where subject matter expertise is relevant. The way of dealing with problems in the *complicated* domain is *sense-analyze-respond*: sense the problem, investigate or analyze several options, and then respond appropriately. Because there can be more than one “correct” answer, “good practice” is more appropriate here than “best practice”.

The *Complex* Domain. The *complex* domain is the domain of enabling constraints. A significant distinction between *complex* and *complicated* is that complicated systems are reducible, whereas complex systems are not

(Miller and Page, 2007). How a problem was solved, or impediment removed, can only be understood in retrospect, not predicted in advance. The way of dealing with problems in the *complex* domain is *probe-sense-respond*: conduct safe-to-fail experiments (or, simply, try different approaches), get a sense of the patterns that emerge, and then respond appropriately (Snowden and Boone, 2007). Section 4.4.4 below describes the concept of experiments in relation to managing change in complex adaptive systems, and specifically in relation to managing complex impediments to flow. Snowden and Boone (2007) note that most changes in organizations are complex because “*some major change...introduces unpredictability and flux*”.

The Chaotic Domain. The *chaotic* domain is characterized by an absence of constraints. Unlike the *complex* domain, there are no manageable patterns in the *chaotic* domain. The way of dealing with problems in the *chaotic* domain is *act-sense-respond*: act to establish order, get a sense of where stability is present or absent, and then respond to change the situation from *chaotic* to *complex* (Snowden and Boone, 2007). Thietart and Forgues (1995) note that similar actions taken by organizations in a chaotic state will never lead to the same result. This is generally the domain of emergency or crises responses.

The Disorder Domain. A problem, including impediments to flow, maps to the disorder domain when it is not clear which of the other four domains applies (Snowden and Boone, 2007). One way of dealing with problems in the *disorder* domain is to break them down into constituent parts, and map each of those parts to one of the other domains (Snowden and Boone, 2007).

A number of studies show the use of the Cynefin framework, primarily in healthcare and related domains. Kempermann (2017) apply Cynefin in biomedical research to explore the genetics of Parkinson’s Disease. Their study finds that the Cynefin framework “*provides a common reference language and conceptual framework to talk about complexity and draw the appropriate conclusions for insight, decisions and actions*” (Kempermann, 2017). Burman and Aphane (2016) used Cynefin in the context of a bio-

social HIV/AIDS risk-reduction pilot to provide an indication whether the intervention had been effective, to identify emergent opportunities and challenges, and to categorize them into appropriate decision-making domains in preparation for the next phases of the intervention. French (2015) uses Cynefin to reflect on processes of modelling and analysis in statistical, risk and decision analysis. Puik and Ceglarek (2015) use Cynefin in association with axiomatic design to characterize the relation between the quality of a design and the knowledge of its designer. Gorzeń-Mitka and Okręglicka (2014) and McLeod and Childs (2013) use Cynefin as a strategic lens through which to view electronic records management, and speculate that it might provide a “*new construct for re-perceiving the challenge in a holistic way and offers a strategic approach to taking action for change*”. Elford (2012) investigates whether Cynefin could be applied in the field of ergonomics. Van Beurden et al. (2011) show that the Cynefin Framework, as a sensemaking tool, can help practitioners in health promotion practice to “*understand the complexity of issues, identify appropriate strategies and avoid the pitfalls of applying reductionist approaches to complex situations*”. Littlejohn et al. (2010) use the Cynefin framework to understand the nature of problems causing workplace health and safety incidents. As far as the researcher is aware, this study is the first to use the Cynefin sensemaking framework in the context of software development organizations using agile and lean methods, and the first study to use the Cynefin framework to understand impediments to flow.

Constraints and boundaries were discussed, with examples, in section 4.2.3 as a property of complex adaptive systems. The Cynefin framework provides a structure for thinking about how constraints and boundaries apply when making sense of impediments to flow in systems. The *obvious* domain has *rigid* constraints that enforce repeatable “*best practices*” (Kurtz and Snowden, 2003). The *complicated* domain has *governing* constraints. Governing constraints are rules that allow action to be determined in accordance with a known set of acceptable “*good practices*” (Snowden, 2015). The *complex* domain has *enabling* constraints. Enabling constraints are often embodied as heuristics (Snowden, 2015). Research on “*fast and*

frugal heuristics” shows that simple heuristics can be successful in complex, uncertain environments and also when and why this is the case (Artinger et al., 2015). The *chaotic* domain has no constraints (Snowden and Boone, 2007). This perspective on constraints can be applied when determining how to resolve impediments to flow. Section 7.3.4 discusses constraints in the context of this study’s findings.

4.4 Resolving Impediments in a Complex Adaptive System

4.4.1 Interventions in a Complex System

Dooley (1997) says that “*the quickest way to predict the future of a complex system is to let it evolve and see what happens.*” Sensemaking helps people understand what action might be appropriate in a given context in order to evolve the system (Snowden and Boone, 2007, Weick, 1995). Kurtz (2014) notes that “[*e*]ach of our dramas exerts constraints on each other’s, making the whole different from the parts, but still dramatic ... I can only answer the question “What am I to do?” if I can answer the prior question “Of what story or stories do I find myself a part?””. Narrative, as discussed in section 4.3.4, helps people make sense of what is happening so they can determine an appropriate course of action.

Resolving impediments means making changes in a complex system. The systems literature uses the term *intervention* to refer to the process of making changes in a system (Kurtz and Snowden, 2006, Holland, 2006, Eoyang, 2004, Meadows, 2008). Eoyang (2001) classifies interventions as *container*, *difference*, or *exchange* (CDE) interventions, each of which brings a specific focus to acting in a system. Section 6.3.4 presents a CDE analysis of the interventions used by product development organizations in this study to resolve impediments.

Analyzing the narratives and associated data gathered through sensemaking can help identify and refine possible interventions (Van Der Merwe et al., 2019). Van Der Merwe et al. (2019) show that identifying clusters of

narratives that illustrate desirable or undesirable patterns can identify opportunities for change that help nudge the situation in a beneficial direction by encouraging “*more stories like these, fewer stories like those*”. Section 6.3.1 shows examples from this study’s findings of narratives associated with desirable and undesirable patterns related to impediments to flow in organizations.

4.4.2 Approaches to Dealing with Change in Complex Systems

There are two fundamental approaches to designing how systems deal with change. One is a fail-safe approach, where “*fluctuations [are] minimized and explicit efforts made to minimize the probability of failure*” (Juarrero-Roqué, 1991). The other is a safe-to-fail approach, whereby the system is designed to deal with changes in a way that does not damage the system, i.e., the system remains safe under conditions of failure. Juarrero-Roqué (1991) notes that since it is not possible to develop complete knowledge of social systems, a fail-safe strategy will not work. Therefore, a safe-to-fail approach is needed. Juarrero-Roqué (1991) refers to safe-to-fail approaches as “*safe-fail*”. The absence of complete knowledge, combined with the effects of properties such as nonlinearity and emergence (section 4.2.3), means that organizations need to design safe-to-fail experiments. This notion of safe-to-fail experiments is embedded in the Cynefin sensemaking framework, described in section 4.3.6. For problems in the complex domain, Snowden and Boone (2007) suggest that leaders in the organization need to allow the way forward to emerge through safe-to-fail experiments. These experiments act as probes that shift the patterns in the system, hence the response style of probe-sense-respond. Eoyang and Holladay (2013) assert that the only way to minimize risk in a CAS is to “*try something, quickly and carefully assess how the system responds to your action, and take another action in mutual response*”. Dervin (1998) notes that one of the “*deadly sins*” of sensemaking is “*failing to recognize the importance of experimentation*”. Page (2007) notes that in groups, where there is diverse ability, experimentation can lead to a better performance of individuals and

the group as a whole. Deciding how much experimentation to do depends on the context. Page (2007) suggests that organizations should experiment more when the cost is lower, and the importance of the problem is higher. Page (2007) also notes the connectedness between problems, stating that groups should experiment more when a problem is connected to other problems – the higher the degree of connectedness, the more experiments should be run. Hence, for impediments in the *complex* domain, organizations remove them through a series of safe-to-fail experiments.

4.4.3 The Role of Experiments in Software Organizations

The way an organization solves problems or removes impediments is an expression of culture (section 4.2.6). The relationships are non-linear so the effect or outcome is not proportional to the cause (Meadows, 2008). Looking for a single root cause in a complex space is unlikely to be successful. Instead, decision makers that are responsible for designing interventions to remove impediments benefit from understanding the interactions of agents at many levels. While some causes and their effects can be perceived when dealing with organizational culture, it is impossible to predict what will happen and provide enough understanding to build models and establish causality. Therefore, exploratory approaches are necessary (French, 2013).

A role of managers in TPS is to construct work as experiments that yield continuous learning and improvements, and to help others also structure work in this way (Spear, 2004). Spear and Bowen (1999) describe how experimentation is part of the “DNA” of the TPS. People are continuously conducting experiments, where work activities have in-built hypotheses that are tested in operations. At least as far back as the 1980s, the need for experimentation in software engineering has been recognized as an important contributor to solving problems (Basili et al., 1986). Research by Basili (1996) shows that “*problem solving ability evolves over time. The evolution is based upon the encapsulation of experience into models and the validation and verification of those models based upon experimentation, empirical evidence, and reflection*”. In agile development, a “spike

solution” is a “*small experiment to research the answer to a problem*” (Shore and Warden, 2007). Ries (2017) suggests that instead of asking customers directly what they want, product developers should “*design experiments that allow you to observe it*”. As an alternative to requirements-driven software development, (Melegati et al., 2019) propose a set of practices for handling hypotheses that guide experiments that result in features users actually want. As well as its relevance to experimentation, this approach has the potential to reduce the impediments related to unwanted *extra features*, as described in section 2.5. Google uses experiments to understand the impact of features in their products. For example, in 2010 alone they ran 8,157 A/B tests (experimenting with two different versions of a feature) and 2,800 1% tests (rolling out a feature to just 1% of users) (Bock, 2015). Every day in 2010 they ran more than 30 experiments just for the search product (Bock, 2015). Research by Fabijan et al. (2017) asserts that controlled experimentation is “*becoming the norm in the software industry for reliably evaluating ideas with customers and correctly prioritizing product development activities*”. Experimentation applies not just to solving problems and developing features, but also to developing the organization itself. Section 4.2.6 discussed the importance of culture in organizations. McCord (2017) describes creating a culture as an evolutionary process. Referring to the how Netflix evolved its culture, she says “*think of it as an experimental journey of discovery. That was how we thought about building the culture at Netflix.*” Google used experiments to try different organization structures and find the right management structure to support its engineers (Garvin, 2013).

4.4.4 Experiments in the Complex Sensemaking Domain

Snowden and Boone (2007) discuss the concept of experimentation more generally as being part of a leader’s job when faced with *complex* scenarios. A characteristic of the *complex* domain (section 4.3.6) is that people can understand why things in this domain happen only in retrospect. However, patterns (section 4.2.3) can emerge if people conduct experiments (Snowden and Boone, 2007). Bock (2015) notes that “*a thoughtfully designed*

experiment, and the patience to wait for and measure the results, will reveal reality to you". Instead of attempting to impose a course of action, leaders must patiently allow the path forward to reveal itself through probing the system with small, safe-to-fail experiments (Snowden and Boone, 2007). As a result of this probing, managers sense patterns that emerge from the system and respond in ways that either encourage or discourage undesirable system behavior (Van Der Merwe et al., 2019). Each response leads to new information which determines the next experiment, in an ongoing adaptive cycle. This is similar to the adaptive action cycle described by Eoyang and Holladay (2013). In a complex space "*we cannot sense and respond, but must first probe the space to stimulate pattern understanding or formation, then sense the patterns and respond accordingly*" (Snowden, 2002). When designing experiments for the complex domain, people manage three things: the attractors around which patterns form, the boundaries and constraints that contain the formation of those patterns, and the interactions of the agents in the system (Snowden, 2005a). Leaders in complex scenarios "*help organizations take the moment and make the best of it, without knowing what is going to happen next*" (Plowman et al., 2007).

4.4.5 System Improvement Goals

Kaikaku, discussed in section 2.2.2, is roughly translated as radical improvement (Womack and Jones, 2003) or radical transformation (Fitzgerald and Stol, 2017). Sometimes organizations undergoing change will have a "kaikaku phase", where necessary macro changes are made (Womack and Jones, 1996). Examples include rethinking an entire value stream and flow (Womack and Jones, 1996). Other examples include a software development organization's decision to transition to agile methods or redesigning an entire system architecture (Fitzgerald and Stol, 2017), or rethinking what value means to the organization (Womack and Jones, 2003). Fitzgerald and Stol (2017) note that there are times when incremental change and *kaizen* are insufficient to deal with the challenges faced by software development organizations. In such cases, *kaikaku*, or radical change in the system may be a more effective approach than *kaizen*, or

continuous improvement. These macro changes are focused on improving the organization or system and align with the macro level of systems discussed in section 4.2.4 above. As such, this study uses the term *system improvement goal* to refer to these macro-level system changes that organization seek to make.

4.4.6 Interventions Shape the Patterns in a System

Research in complex adaptive systems show that three factors shape patterns in self-organization: *containers*, significant *differences*, and transforming *exchanges* (Eoyang, 2001, Olson and Eoyang, 2001). This study uses *containers*, *differences*, and *exchanges* (CDE) to understand interventions to resolve impediments to flow in a CAS.

- **Container.** A *container* sets the boundary for self-organizing systems (Eoyang, 2001). Examples of containers include a value stream, the software product development organization, teams within the product development organization, or the company of which the product development organization is part. The container's purpose is "to hold the system together, so relationships between and among agents can be established" (Eoyang, 2001). A system may be contained by an external boundary, by some central attracting force, or by one-to-one forces between agents in the system – what Eoyang (2001) refers to as a *fence*, *magnet* or *affinity container*, respectively. Eoyang (2001) notes that any human system can contain multiple containers simultaneously, and the agents in the system can be part of multiple containers simultaneously.
- **Difference.** A *difference* represents the potential for change in a *container*, and is a necessary condition for self-organization to occur (Eoyang, 2001). Differences determine the patterns that emerge in self-organizing systems. Examples of significant differences include power, levels of expertise, quality, cost, gender, race and educational background (Olson and Eoyang, 2001). Page (2007) demonstrates that teams with a higher degree of diversity outperform teams of experts at solving problems. Juarrero-Roqué (1991) argues that diversity is important in uncertain environments and becomes less important in

environments where there is a high degree of certainty. There is, therefore, a positive correlation between diversity and resilience in software development organizations (Juarrero-Roqué, 1991).

- **Exchange.** Agents in a system communicate by sending and receiving signals (Holland, 2006). A *transforming exchange* is the interdependence between agents in a CAS, and is critical to the ability of the agents to self-organize (Eoyang, 2001). The value of *exchanges* is reflected in the first of the four values of the Agile Manifesto “*individuals and interactions over processes and tools*” (Beck et al., 2001). Considering agents and their interactions is a high leverage point for improvement in the system. Information exchanges (or “*flow of information*”) hold a system together and also play a role in determining how systems operate (Meadows, 2008).

Section 7.3.4 discusses the interventions undertaken by organizations in this study and analyzes those interventions in terms of the CDE model.

4.5 Summary

This study of impediments to flow is situated in the context of software product development organizations. Section 4.2 shows that software development organizations are complex adaptive systems. Culture is also a CAS, and culture in software development organizations is a significant determinant of whether approaches such as flow, and identifying and resolving impediments to flow, will be successful.

Sensemaking provides a way of exploring and understanding complex systems (section 4.3). It helps people in software development organizations to make sense of their context so they can take context-appropriate action to resolve impediments and therefore improve flow. Micronarratives act as a seed from which people develop a larger sense of what may be occurring. Culture is a reflection of people’s lived experience in organizations. Using micronarrative sensemaking as cues (section 4.3.4) is an entry point to understanding these experiences, and how they relate to flow and impediments.

Making changes in complex systems (section 4.4) involves designing interventions to manage the patterns in the system, so that beneficial patterns can emerge. A key role of leaders is to create the conditions necessary for the emergence of desired change and to monitor and influence the patterns that emerge. Resolving impediments in a complex space leads to a continuous process of designing probes or experiments to stimulate patterns and new information. Interventions constrain the system patterns that emerge. How interventions shape the emergent patterns can be understood through analyzing how the interventions modify the containers, differences, and exchanges in the system.

These elements come together to complete the conceptual framework for this study, as shown in Figure 4-2.

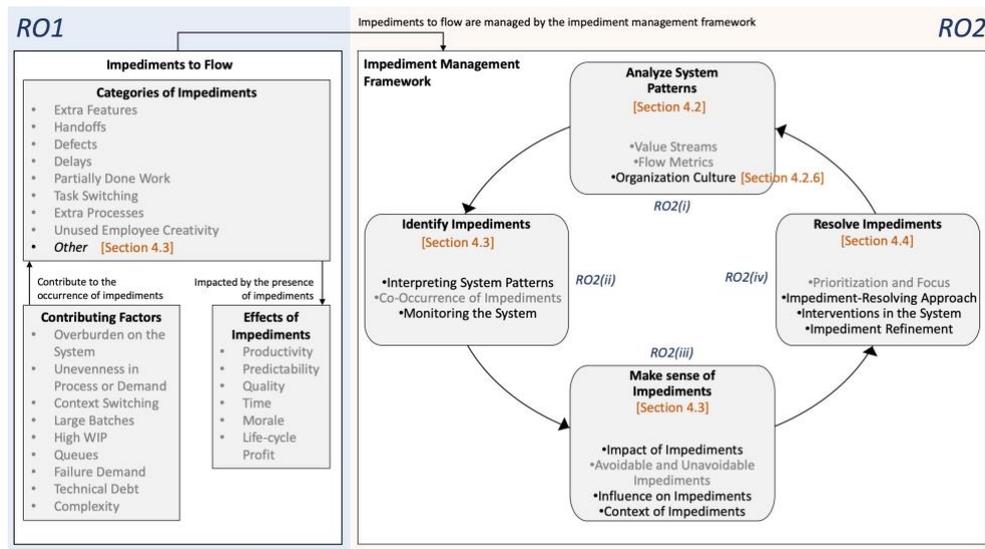


Figure 4-2 Contributions to the conceptual framework from chapter 4

Next, chapter 5 discusses the research methods used to conduct this study.

Chapter 5 Research Methods

5.1 Introduction and Chapter Layout

This research is about understanding impediments to flow in large software product development organizations. Section 5.2 summarizes the research objectives for this study and presents the completed conceptual framework that guides this study. Section 5.3 describes the epistemological stance adopted by this study. Section 5.4 outlines the research approaches used to conduct this study. Section 5.4 provides a discussion on research methodologies and includes a discussion on the three research methods used in this study. It describes how this study integrates data from focus group research, narrative research, and case study research, and shows how each map to the conceptual framework, and thus contribute to achieving the research objectives of this study. Section 5.5 describes the data collection methods used in this study. Section 5.7 describes the data analysis methods used in this study. Section 5.8 address validity and reliability concerns. Finally, section 5.9 summarizes this chapter's contribution to this study.

5.2 Research Objectives

This thesis examines impediments to flow in large, global software product development organizations. The specific objective of this research is:

Understand impediments to flow in software product development organizations.

Through the pursuit of this research objective, the researcher aims to address the following specific sub-objectives:

RO1: Develop an understanding of the current state of research and practice regarding impediments to flow in software product development organizations.

RO2: Develop and validate an impediment management framework that is used to:

- (i) Analyze system patterns that contribute to impediments to flow
- (ii) Identify impediments to flow
- (iii) Analyze how people in organizations make sense of impediments
- (iv) Explore approaches to resolving impediments

The conceptual framework used to address these research objectives is developed throughout Chapter 2, Chapter 3, and Chapter 4. The summary section of each of those chapters summarizes that chapter's contribution to the framework. The framework is used to guide data collection and analysis. The final, validated framework from this study is shown in Figure 5-1.

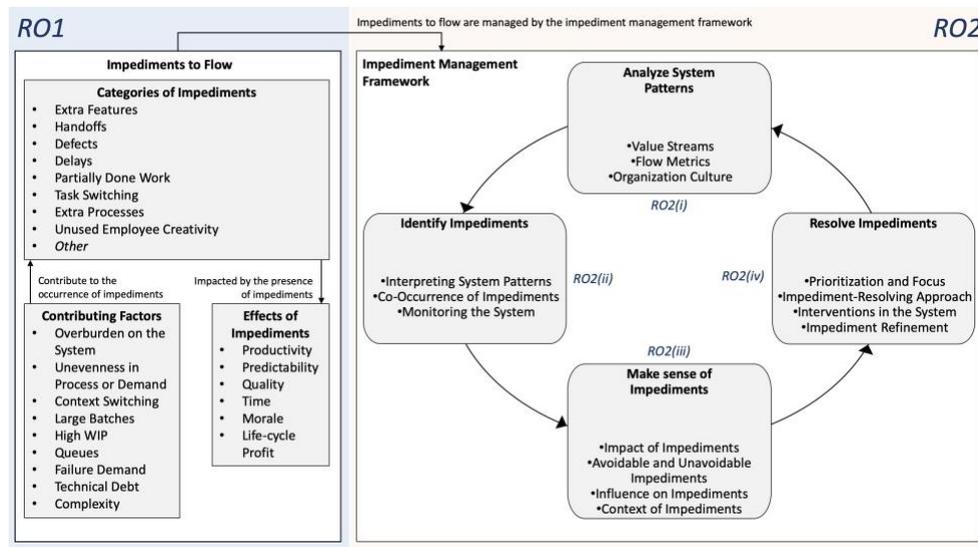


Figure 5-1 Validated conceptual framework for this study

5.3 Epistemological Stance

This section describes common philosophical and logical perspectives that underpin research studies and presents the epistemological stance for this study. Different philosophical perspectives embody ideas about reality (*ontology*) and how we gain knowledge of reality (*epistemology*) (Maxwell, 2013).

5.3.1 Philosophical Perspectives

Positivism is the view that science is based on universal truths, and that the role of research is to uncover these truths (Yin, 2016). From a positivist

perspective, there is a deterministic relationship between causes and effects or outcomes (Creswell, 2014). A positivist perspective assumes an objective physical and social world that exists independently of humans (Morgan, 2011). Coyle (2013) notes that “*the assessment and impact of emotions or thoughts would fall outside the remit of a positivist researcher*”. *Postpositivism* acknowledges that researchers cannot be positive about claims of knowledge when studying humans (Creswell, 2014). Creswell (2013) notes that postpositivist researchers do not believe in strict cause and effect but recognize that cause and effect is a probability that may or may not occur. A postpositivist perspective has the characteristics of being “*reductionist, logical, empirical, cause-and-effect oriented, and deterministic, based on a-priori theories*” (Creswell, 2013).

Interpretivist researchers acknowledge that people have potentially widely-varying perceptions of the same phenomenon, and that knowledge is a social product (Miles et al., 2014).

Constructivism is a perspective where individuals seek understanding of the world in which they live and work, and is typically associated with qualitative research methods (Creswell, 2014), as described in section 5.4.1. A constructivist perspective has the following characteristics (Creswell, 2014):

- Meanings are subjective, multiple, and varied.
- Researchers look for the complexity of views, rather than attempting to narrow meanings into a few categories or ideas.
- The research relies on the participants’ views of a situation.
- Open-ended questions are preferred, with the researcher listening for what people do or say in real settings.
- Meanings are subjective, and are negotiated socially through history and interactions, and through historical and cultural norms.
- Researchers often address the process of interaction among individuals.
- The research focuses on specific contexts in which people live and work.

Constructivist researchers recognize that their own backgrounds shape their interpretation, make sense of, or interpret, the meanings others have about the world, and generate or inductively develop a pattern of meaning, rather than start with a theory.

A *critical realist perspective* combines a realist ontology with a constructivist epistemology (Creswell, 2013). These two perspectives have implications for how researchers perceive the world. From an ontological realism perspective, there is a real world “*that exists independently of our perceptions and theories*”; the world does not accommodate to our beliefs (Maxwell, 2013). From an epistemological constructivist perspective, our understanding of the world is our own construction, not a purely objective perception of reality, and “*no such construction can claim absolute truth*” (Maxwell, 2013). Though any discussion can be only “*a partial description of possibilities*”, a review of several major interpretive frameworks can provide a sense of options (Creswell, 2013). Maxwell (2013) advocates that a critical realist perspective is a useful perspective when used with other perspectives, and not as the single correct perspective for a qualitative study.

Wang describes how a *critical perspective* brings into focus “*the reality of interdependence of parts with the whole, and to that organizations can not be studied in isolation of the context within which they operate, and which they in part constitute*” (Wang, 2007). This is particularly relevant in the context of complexity thinking, and for this research, since this thesis uses CAS theory as a lens through which to understand flow and impediments in organizations. Citing Orlikowski and Baroudi (1991), Wang (2007) notes that a critical realist perspective “*reminds us of the constantly changing potential of humans who need not be confined by their immediate circumstances. Researchers do not merely accept the self-understanding of participants, but also critically analyze it through the particular theoretical framework which they adopt to conduct their work*”. The purpose of a research study in this context, then, is to “*bring to consciousness the*

restrictive conditions of the status quo, thereby initiating change in social relationships and practices”.

A *pragmatist perspective* offers researchers the opportunity to use multiple methods, different worldviews, and different assumptions (Creswell, 2014). Pragmatism derives from the work of Peirce who viewed research as a recursive process of “*moving back and forth between a set of observations and a theoretical generalization*” (Tavory and Timmermans, 2014). A pragmatist perspective allows for different forms of data collection and analysis, to best suit the research problem (Creswell, 2014). A pragmatist perspective has the following characteristics (Creswell, 2014):

- It is not committed to any one system of philosophy and reality.
- Researchers are free to choose whatever methods, techniques, and procedures that best meet their needs and purpose.
- Truth is not absolute and is whatever works at the time.
- Researchers use both qualitative and quantitative data to provide the best understanding of a research problem.

5.3.2 Logic

Deductive and inductive approaches reflect different ways of shifting between data and concepts (Yin, 2016). An abductive approach helps researchers understand how to deal with surprising observations that do not neatly fit existing theories or hypotheses (Tavory and Timmermans, 2014). This section describes the main characteristics of deductive, inductive, and abductive approaches.

With a *deductive approach*, the researcher starts the study with preconceptions before doing any fieldwork (Yin, 2016). The researcher tests or verifies a theory by examining hypotheses or questions derived from the theory (Creswell, 2014). Deductive logic is commonly associated with quantitative research methods (Creswell, 2014). However, as Yin (2016) notes, a deductive approach can still aid a qualitative research study, e.g.,

when faced with a large amount of data it can help to start the analysis with some initial concepts or theories.

An *inductive approach* builds theory from the data (Creswell, 2014). Inductive logic is commonly associated with qualitative research methods (Maxwell, 2013). Inductive approaches “*tend to let the data lead to the emergence of concepts*” and allows events in the field to drive the later development of categories, propositions, and meaning (Yin, 2016).

Charles Peirce introduced the term “abduction” into logical theory in the 19th century (Gabbay and Woods, 2006). *Abductive logic* has applications in formal logic and artificial intelligence (e.g., diagnostic reasoning, text understanding, case-based reasoning, and planning), as well as in the fields of cognitive science and philosophy of science (Meheus and Batens, 2006). Tavory and Timmermans (2014) describe an approach to abductive analysis, which builds on the pragmatist perspective of Peirce, as described in 5.3.1.

Abductive logic examines “*an array of possibilities, in order to select the most likely, plausible, or best possibility*” (Saldana and Omasta, 2017). As noted in Chapter 4, in a CAS there is no clear link between cause and effect; there can be many causes for an occurrence, or no discernable cause. Sometimes researchers will observe an event for which there is no obvious or discernable cause. At other times, researchers will encounter surprising evidence that does not neatly fit existing theories or hypotheses. In such cases, abductions allow the researcher to form hypotheses or conjectures based on pragmatic assumptions. Abduction happens when researchers “*encounter observations that do not neatly fit existing theories*” and the researcher must “*speculate about what the data plausibly could be a case of*” (Tavory and Timmermans, 2014).

Abductive logic is a creative, inferential process that aims to produce new theories and hypotheses in the face of such surprising research evidence (Tavory and Timmermans, 2014). Abduction allows researchers to form hypotheses about likely causes for events, even when they don't have access to all the information related to the event. This helps researchers to orient

themselves towards the surrounding environment through a hypothesis. In these situations, the process of abduction provides justifications for the use of such hypotheses; it does not provide evidence of truth (Gabbay and Woods, 2006). Researchers employing abductive logic look at all the available data, knowing it is incomplete, and consider different possibilities for interpretation (Saldana and Omasta, 2017). Researchers must be wary of the ‘*post hoc fallacy*’ (Finocchiaro, 1981), or *post hoc ergo propter hoc*, which warns that just because event A precedes event B, does not mean event A caused event B (Grouse, 2016). While the possibility exists that the hypothesis is inaccurate or incorrect, it is still useful as a starting point for creating understanding and for taking action (Gabbay and Woods, 2006). From a complexity theory perspective, an *abductive* perspective can deal with unexpected or surprising data that seems not to fit neatly within the conceptual framework, as well as with situations where it is necessary to make assertions or conclusions without having access to all information on a topic.

5.3.3 Theory

Maxwell (2013) distinguishes between quantitative and qualitative research from a theory perspective, distinguishing between variance theory and process theory, respectively. Quantitative researchers look for statistical relationships between different variables; qualitative researchers employ process theory to study people, events, situations, and the processes that connect them (Maxwell, 2013). Maxwell (2013) uses the metaphor of a spotlight to make the point that theory illuminates what the researcher sees. Different theories draw attention to different events or phenomena, and shed light on “*relationships that might otherwise go unnoticed or misunderstood*” (Maxwell, 2013).

Gregor (2006) examines the structural nature of theory in the information systems discipline and presents a taxonomy of five theory types. The five types are summarized here (Gregor, 2006):

1. **Analysis.** Say what is. Does not extend beyond analysis. No causal relationships between established phenomenon, and no predictions are made.
2. **Explanation.** Say what is, how, why, when, and where. Provides explanations but does not aim to predict with any precision. There are no testable propositions.
3. **Prediction.** Say what is, and what will be. Provides predictions and has testable propositions. Does not have well-developed causal explanations.
4. **Explanation and prediction.** Say what is, how, why, when, where, and what will be. Provides predictions and has both testable propositions and causal explanations.
5. **Design and action.** Say how to do something. Gives explicit prescriptions (e.g., methods, techniques, principles of form and function) for constructing an artifact.

The subjects of this study are software development organizations, which are complex adaptive systems, as described in section 4.2.2. There is often no linear relationship between cause and effect in a CAS. This perspective eliminates *prediction theory* (Gregor, 2006) and also *explanation and prediction theory* (Gregor, 2006) as suitable theory types for this study. Runeson et al. (2012) distinguish between *explanatory* and *exploratory* studies. An *explanatory* study seeks an explanation for a situation or problem, mostly, though not necessarily, in the form of a causal relationship. This definition aligns with the definition of explanatory theory from Gregor (2006). An *exploratory* study, on the other hand, aims to find out what is happening through seeking new insights and generating ideas and hypotheses for new research. “How” questions imply explanatory approaches, whereas “what”, “in what ways” imply exploratory approaches (Saldana and Omasta, 2017). Creswell (2013) outlines a set of questions that are relevant to exploratory research. These include questions such as “*What happened? Who was involved ...? What themes ... emerged ...? What theoretical constructs helped us understand ..., and what constructs were unique to this case?*” These types of exploratory questions are relevant to

this study and informed the structure of the research objectives. These definitions of exploratory theory align with the definition of analysis theory from Gregor (2006), where the theory does not extend beyond analysis and description; this study does not specify causal relationships among phenomena and makes no predictions.

Yin (2014) notes that some types of “*what*” questions are a justifiable rationale for conducting an exploratory study. A goal of this study is to develop relevant hypotheses and propositions for further inquiry on the topic of impediments to flow. Morgan (2019) highlights the relevance of focus groups to exploratory research, noting that focus groups maximize the ability to hear about the participants’ perspectives and experiences, which was a goal in designing the focus groups for this study. Squire et al. (2014) note that qualitative narrative research is often used in an exploratory way, as is the case in this study to “illuminate” the circumstances of people in software development organizations. As per Yin (2014), this exploratory study covers the issues or problem being explored (impediments to flow in large software product development organizations, detailed in Chapter 1, Chapter 2, Chapter 3, Chapter 4), the methods of exploration (Chapter 5), the findings from the exploration (Chapter 6), and the conclusions for future research (Chapter 7).

The exploratory perspective complements the constructivist philosophical perspective described in section 5.3.1. In the constructivist tradition, as related by Creswell (2013), this study incorporates the paradigm assumptions of “*an emerging design, a context-dependent inquiry, and an inductive data analysis*”. Furthermore, while case studies can lend themselves to either exploratory or explanatory studies, an explanatory study typically has characteristics consistent with deductive logic (Runeson et al., 2012). The exploratory perspective complements the inductive logic perspective adopted by this study and described in section 5.3.2, which lends itself to an “*open-ended investigation with minimal assumptions, leaving one’s self open to emergent leads, new ideas, exploratory discovery, and decision making*” (Saldana and Omasta, 2017).

5.3.4 Epistemological Stance for This Study

Any view is shaped by the social and theoretical perspectives adopted by the study, and by the “lens” of the researcher (Maxwell, 2013). The epistemological stance of this study adopts a primarily constructivist philosophical perspective, an inductive logic perspective, and an analysis and exploratory theory perspective.

5.3.5 Self-Reflexivity

The issue of reflexivity arises whenever one observes human systems (Yin, 2016), such that the researcher’s presence will always have an unknown influence on the system. Maxwell (2013) refers to the influence of the researcher on the setting as “*reactivity*”. Following Yin’s recommendation (Yin, 2016), this section presents the “*important reflexive conditions*” that are present in the research study that could potentially “*affect the conclusions from the study*”. Self-reflexivity is about more than acknowledging the researcher’s biases and values. It is also about acknowledging how the researcher’s background can shape the direction of the study (Creswell, 2014). While it is important for the researcher to be aware of and to acknowledge their potential to influence the research environment, it is also important to understand that “*trying to minimize your influence is not a meaningful goal for qualitative research*” (Maxwell, 2013). This section presents a brief summary of how the researcher might influence aspects of this research.

- **Experienced industry practitioner.** The researcher is an experienced industry practitioner with more than twenty years’ experience in software engineering. He is currently a Principal Engineer / Engineering Director and has held several roles in multiple companies and domains including software engineer, technical leader, manager, and architect.
- **International agile and lean community.** The researcher is an active member of the international agile and lean community. He is a regular speaker at the major international agile and lean conferences. He has authored and presented numerous papers. He has served as a member of

multiple conference program committees and is a regular reviewer for international publications. He is a Fellow of the Lean Systems Society.

- **Relevant qualifications and training.** The researcher is a trained, professional coach in Co-Active Coaching and Organization and Relationship System Coaching. This provides significant advantages in asking and framing questions, as well as working with groups. He is also trained in complexity methods and tools, with training in the Cynefin framework and sensemaking, and is a certified Human Systems Dynamics professional.
- **Perspectives.** The researcher is a proponent of complexity thinking, complex adaptive systems theory, sensemaking, and CAS practice. This background influenced the selection of the theoretical lens for the research, where organizations are complex adaptive systems.
- **Domain knowledge.** The researcher has a thorough knowledge of the problem domain and the technology domain in which the research takes place. While this is an advantage in terms of understanding and relating to participants, it can also introduce unintentional bias.

5.4 Research Methodologies

5.4.1 Quantitative and Qualitative Research Methods

Quantitative research tests objective theories “*by examining the relationship among variables*” (Creswell, 2014). The variables can be measured so that “*numbered data can be analyzed using statistical procedures*” (Creswell, 2014). Quantitative methods reflect positivist and postpositivist values, as described in section 5.3.1.

Qualitative research is a diverse field with multiple definitions. Creswell (2014) defines qualitative research as “*an approach for exploring and understanding the meaning individuals or groups ascribe to a social or human problem*”, where the researcher typically collects data in the participant’s setting, analyzes data inductively, and interprets the meaning of the data. Qualitative research, unlike quantitative research, considers the context of the study, i.e., “*the social, institutional, and environmental*

conditions within which people's lives take place" (Yin, 2016). Rather than try to provide a single definition, Yin describes five attributes of qualitative research (Yin, 2016), which this research aims to use:

1. Studying the meaning of people's lives under real-world conditions
2. Representing the views and perspectives of the people
3. Covering the contextual conditions under which people live
4. Contributing insights into existing or emerging concepts that may help explain human social behavior
5. Striving to use multiple sources of evidence rather than a single source alone

Klein and Myers (1999) propose a set of principles for conducting and evaluating interpretive field research. These principles guide the method selection, data collection strategies, and data analysis approaches used in this study.

The research methodology for this study is qualitative. This study uses three primary research methods: *focus group* research, *narrative* research, and *case study* research. The relationships between these three are shown in Figure 5-2.

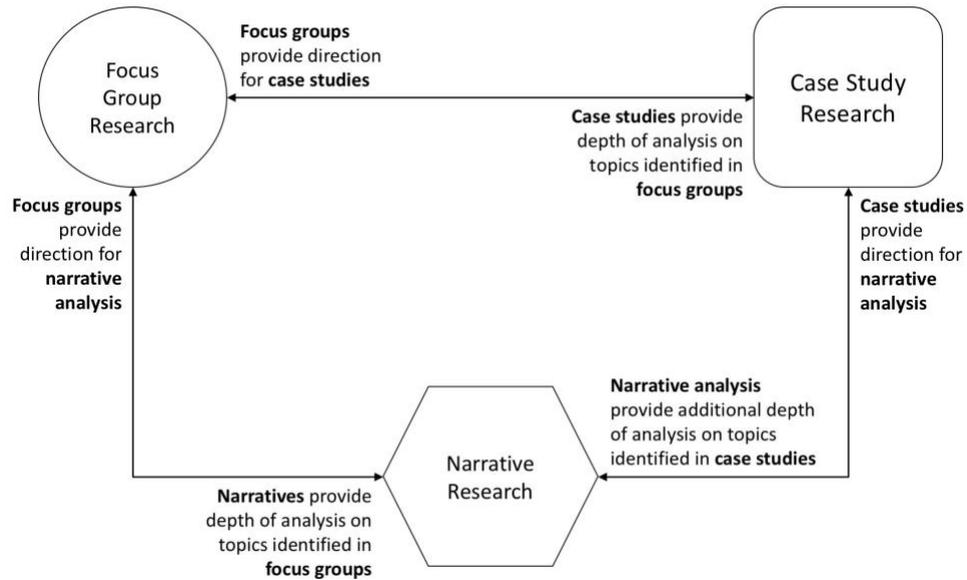


Figure 5-2 Relationship between the three research methods

Each of these three sections describes the use of the approach in research, its suitability as a method, associated challenges, and how the approach is used in this study. Note, this section discusses ‘suitability’ of the method, rather than ‘strengths’. This section also discusses ‘challenges’ with the using the method, rather than ‘weaknesses’ of the method. The concepts of ‘strengths’ and ‘weakness’ are not attributes of a research approach, but rather judgments that researchers make about a research approach (Sandelowski, 2012, Sandelowski, 2014). Researchers must still defend the choice of methods and show how they meet the specific objectives of a study (Sandelowski, 2014).

5.4.2 Focus Group Research

This study uses the definition from Morgan (1997), who defines focus groups as “a research technique that collects data through group interaction on a topic determined by the researcher” and treats focus groups as a “broad umbrella” that “can include many variations”. Morgan (1997) notes that the *focus* comes from the researcher’s interest, and the data comes as a result of *group* interactions. Yin (2016), on the other hand, says that the groups are focused because they share some common experience or views. These two perspectives are not mutually exclusive. The unit of analysis is the group itself, not the individuals within the group. In a study that

involves a series of focus groups, as is the situation in this study, each focus group is one data collection unit (Yin, 2016).

Although other researchers, including Powell and Single (1996), distinguish focus groups from other forms of group interviews by their degree of formality, Morgan (1997) argues that it is not possible “*to draw a line between formal and informal group interviews in a way that defines some as focus groups and others as something else*”. Instead, Morgan (1997) notes that the degree of formality is a decision made with consideration of the needs of the research, and is determined by the goals of the researcher, the research setting, and the “*likely reaction of the participants to the research topic*”.

Uses of Focus Groups in Research

According to Morgan (1997), focus groups are used in three ways. First, focus groups are used as a self-contained method in research studies, where focus groups are the primary source of data. Second, focus groups are used as a supplementary source of data in studies that use another method as the primary method. Third, focus groups are used as one method in a multi-method study, where no single method is the primary one determining how the others are used. Fern (1982) notes that the use of focus groups include hypotheses generation, identifying and pretesting questionnaire items, and exploration of opinions, attitudes, and attributes.

McLafferty (2004) cites several sources with varying opinions on the optimal number of focus groups a researcher should conduct as part of a single study. Opinions cited vary from a minimum of three to a maximum of twelve, with some suggesting that one focus group is enough (McLafferty, 2004). Kitzinger (1998) claims the largest number of focus groups for a single study, with 52 focus groups in a study about the amount of people involved in AIDS issues (McLafferty, 2004). At the other extreme, Howard et al. (1989) justified using a single focus group because of the difficulty of arranging mutually convenient times (McLafferty, 2004). Dickens (2012) uses focus group research in his PhD study on factors that facilitate

emergent change in an organization. In that study, Dickens uses two focus groups as the qualitative part of a mixed-methods approach, with each group containing five participants.

Saturation is a consideration when considering how many focus groups are needed. McLafferty (2004) cites multiple sources that suggest saturation can be achieved with three focus groups, and that data collected after the saturation point is largely redundant.

Suitability of Focus Groups

Focus groups give the researcher an opportunity to see how participants interact with each other, at least within the context of the focus group (Saldana and Omasta, 2017). Morgan (1997) notes that focus groups give the researcher access to a wide variety of topics that might not be directly observable, and also ensure that the data will be directly targeted to the researcher's interests. Participants have the opportunity to compare each other's experiences and opinions, which can be "*a valuable source of insights into complex behaviors and motivations*" (Morgan, 1997). Focus groups can help participants to remember things that they might not have remembered on their own (Saldana and Omasta, 2017). Morgan (1997) cites the biggest strength of focus groups as their ability to produce potentially large amounts of data on topics of interest to the researcher.

Challenges of Focus Group Research

A potential challenge is that for some participants, discussing some topics in a group can influence what they say and how they say it (Morgan, 1997). The group itself may influence the nature of the data it produces (Morgan, 1997). There is ambiguity in the literature about defining the purpose, size, constitution, and execution of focus groups (McLafferty, 2004). Figure 5-5 shows the general format used in this study for data collection in focus groups. Part of that process includes allowing participants time to work in silence and capture their thoughts on sticky notes. This gives time to reflect for quieter or more introverted participants, or those who simply need more time to think. It also reduces the risk of group think.

5.4.3 Narrative Research

Narrative research is a qualitative research method in which the researcher collects and analyzes narratives about the lives of research participants (Creswell, 2013). This study uses narrative research methods as a primary method, and also to support focus groups and case studies. Narrative is described in greater detail in section 4.3.4.

Uses of Narrative in Research

Narrative research involves working with various kinds of narrative materials (Squire et al., 2014). There are two basic scenarios; either the narrative materials exist already, or they are created as part of the research. Whether pre-existing, or coming into existence as part of the research, the first step in narrative research involves collecting the narrative materials. The second step involves analyzing the narrative material with the purpose of categorizing or interpreting it. The Cynefin sensemaking framework, described in section 4.3.6, was derived from research into the use of narrative and complexity theory in organizations (Kurtz and Snowden, 2003). Kurtz and Snowden (2003) use narrative research methods to address problems in organizational knowledge exchange, decision-making, strategy, and policy making. In addition, their work uses narrative methods to enable multi-perspective understanding in complex systems.

Suitability of Narrative Research

When the medium of communication is directly between humans, “*even work-related messages tend to be narrative in style and experiential in content*” (Smith, 2012). Narrative is present in almost all human discourse (Abbott, 2008). Squire et al. (2014) identify narrative as being useful for (a) finding out about little-known phenomenon and exploring narrative voice; (b) understanding the lives of people in the research study; (c) understanding stories in relation to cognition; (d) understanding narratives in relation to social, cultural, and political contexts; (e) understanding effects of narrative on practice; and (f) understanding ‘sensitive topics’ and ‘sensitive events’. Abbott (2008) cites Barthes (1982) on the fundamental

place of narrative among humans, its universality, and the many forms it can take.

Challenges of Narrative Research

Squire et al. (2014) note that it can be difficult for researchers to interpret narrative created by participants. To overcome this difficulty, this study uses a process of self-signification as described by Snowden (2010). Through this self-signification participants assign meaning to their own micronarratives, which enables large scale explorations, reduces researcher bias, and allows for more objective analysis (Van Der Merwe et al., 2019). By embedding self-signification in a narrative capture instrument, as described in section 5.6.2, respondents provide significance and meaning at the same time that they are providing narrative.

Micronarrative fragments are, by their nature, terse. There is much significant and relevant context that is “*between-the-lines, unspoken, yet conveyed*” (Boje, 2008). The narrative research aspects of this study deal primarily with written micronarrative fragments. Acknowledging this challenge of unspoken context, the research instrument is designed to capture some meaningful context (as determined by the researcher) that provides more information about the context of impediments and flow in organizations. Sections 5.6.2 and 5.7.2 describes how this is done from a data collection and analysis perspective.

Squire et al. (2014) note additional challenges with narrative research:

- There are no strict instructions that tell researchers how to engage in narrative research. While this flexibility has resulted in several creative approaches, it does mean that the researcher has some important considerations and decisions to make when designing the narrative research approach. The decisions for this study are described in the next section.
- Dealing with ‘difficult’ narrative material is a challenge. Very often, researchers and participants can become part of a mediation process through co-creating and interpreting narratives. To mitigate this

challenge, the narrative capture instrument used in this study guides participants through a process of self-signification, where the participant is attaching interpretation and meaning to the narrative. This not only adds rich context to the narratives, but also reduces work on behalf of the researcher, and reduces the potential for researcher bias in interpreting the narratives.

- It can be challenging to combine different kinds of data and analyses when using multiple sources of narrative material. To mitigate this challenge, the primary source of narrative data for this study is the narrative capture instrument described in section 5.6.2.
- It can be challenging to write about or present narrative research. The compelling nature of the narratives themselves can make it hard to decide what to include, and how to edit the narratives and analyses so that it fits within the word count constraints of publications. Combining the narrative analysis with other data, as is done in this study, exacerbates this challenge.
- Conducting narrative research ethically is a challenge. One of the strengths of narrative research is its ability to give voice to marginalized members of a group, or those without power. However, sometimes narratives from, of, or about these groups can reinforce social exclusion. Researchers need to be aware of such ethical complexities that can arise when using narrative research.
- Narrative's relation to truth, and, in particular, understanding whether a given narrative is 'true'. Squire et al. (2014) consider three questions in this context. First, what does truth mean when applied to descriptions of human experience? Second, does the issue of truth really matter in this context? Third, how much truth is "*true enough*"?
- The ubiquity of narrative, which is also one of its great strengths, also holds the potential for criticism, with some arguing that it has become trivial and simplistic, with researchers failing to take into account the philosophical debates that should foreshadow narrative research. This chapter of this thesis, and in particular sections 5.4.3 and 5.3, explicitly address that challenge in the context of this study.

5.4.4 Case Study Research

Uses of Case Studies in Research

Case study research is a form of social science research (Yin, 2014). According to Yin (2014), the desire to “*understand complex social phenomena*” is one distinctive reason for using case study research. (Yin, 2014) provides a twofold definition of a case study. The first part of the definition relates to the scope of a case study. A case study is an empirical inquiry that “*investigates a contemporary phenomenon ... in depth and within its real world context*”. And additionally, “*the boundaries between phenomenon and context may not be clearly evident*”. The second part of the definition relates to the features of a case study. A case study inquiry acknowledges that there are “*more variables of interest than data points*”, relies on multiple sources of evidence, and benefits from prior development of theoretical propositions to guide the research (Yin, 2014). Case studies are appropriate for answering ‘*how?*’ and ‘*why?*’ questions (Yin, 2014).

Yin (2012) defines a *case* as a “*bounded entity (a person, organization, behavioral condition, event, or other social phenomenon), but the boundary between the case and its contextual conditions ... may be blurred*”. The case is the main *unit of analysis* for a study. The units of analysis for this study are organizations, not the individuals in those organizations. Recall from section 4.2.2 that organizations, and human groups in general, are complex adaptive systems. This study adopts the perspective that in a human CAS, the agents are the groups, not the individuals.

There are different types of case study. One type classification is the one that is used by Yin (2012), who describes four types of case study. One dimension for classification is the number cases, with the options being *a single case* or *multiple cases*. The second dimension is determined by whether the case study is *holistic* or *embedded*. The combination of these two dimensions gives four options for case study design: single case holistic, single case embedded, multi-case holistic, and multi-case embedded.

A study is using multiple-case design when there is more than one case (Yin, 2014). While some fields, such as anthropology and political science, consider single-case and multiple-case designs as different methodologies, Yin (2014) treats them as variations within the same methodological framework, drawing no broad distinction between them. Multiple-case designs can offer more “compelling” evidence, though can be significantly more costly in terms of time, effort, and other resources (Yin, 2014). Another consideration with multiple-case designs is whether to use a *replication strategy* or *sampling strategy* (Yin, 2014).

Suitability of Case Studies

Case studies can “*uncover subtle distinctions and provide a richness of understanding and multiple perspectives*” (Kohn, 1997). Case studies are well suited to research in software development (Runeson et al., 2012, Runeson and Höst, 2008). The primary reason for this suitability, as cited by Runeson and Höst (2008), is similar to why case studies are well suited to many fields; they study contemporary phenomena in their natural setting. Case studies provide deeper understanding of the phenomenon under study (Runeson and Höst, 2008). Software engineering research studies how the development, operation, and maintenance of software and related artifacts are conducted by software engineers and other stakeholders (Runeson and Höst, 2008). Because of the multi-disciplinary nature of software development, including associated social and political factors in organizations, case study research is suitable for addressing many research questions in the field (Runeson and Höst, 2008).

An effective way of using CAS theory with case study research is to use the CAS framework as the dominant methodology, with case study being used as the research strategy (Anaf et al., 2007). Section 4.2.3 shows that the properties of a CAS include emergence and nonlinearity, which in turn means that there is no clear link between cause and effect. Anaf et al. (2007) show how the combination of case study research and systems theory enables the researcher to consider the broader system and environment and allows for “*in-depth exploration as well as comparative analysis between*

cases in the context of the system". Case study designs can be more informative and relevant when they help reveal these properties of complex adaptive systems in cases under study (Anderson et al., 2005). Case studies combined with complexity theory allow researchers to study a system as an *"integrated whole"* (Anderson et al., 2005). Runeson and Höst (2008) note that observation is particularly useful when there is a deviation between the "official" and the "real" view. Paraphrasing one of the case descriptions in Anderson et al. (2005), *"[i]n order to understand and improve the complex contexts and interactions that lead to [impediments to flow in organizations], theoretical models and research methods are needed for understanding [software development] organizations."*

Challenges of Case Study Research

While researchers have attempted to understand organizations by using case study designs, these designs are only as good as the theoretical model driving the research (Anderson et al., 2005). Chapter 4 describes complex adaptive systems and sensemaking in the context of this study, and how organizations are complex adaptive systems. Anderson et al. (2005) show the same for health care organizations. In systems terms, a case is a bounded system, which implies that the case is subject to the principles of systems theory (Anaf et al., 2007). *"Traditional case study designs, while often helpful, have been driven by theoretical models that are not congruent with the nature of the ... organizations we study. Researchers have studied organizations as though they were mechanistic systems with straightforward cause and effect linkages and dynamics that could be predicted from historical data ... leading to case study designs focused on understanding the elements of the organization through an examination of these straightforward cause and effect linkages and predictable dynamics"* (Anderson et al., 2005). To address this potential weakness in case study design, this study uses CAS as the underlying theoretical model through which to understand flow and impediments in organizations.

5.4.5 Triangulation

The principle of *triangulation* encourages researchers to develop “*converging lines of inquiry*” about all research actions and assumptions (Yin, 2016). Yin outlines four options for triangulation, including data triangulation using different data sources, investigator triangulation using multiple investigators from the same study team, theory triangulation using multiple perspectives on the same data, and methodological triangulation using different research methods. This study achieves data triangulation, using data from multiple sources including focus groups, narrative data, management tools, documents, and observation. In addition, this study uses triangulation of research approaches, with data coming from focus group research, narrative research, and case study research. There are different contributions made by each method, and there is overlap between and two given methods, and indeed, across all three methods.

5.5 Research Approach for This Study

5.5.1 Research Context

This research studies software development organizations in three publicly traded technology companies responsible for delivering products and services to global markets. Each company contains multiple software product development organizations, where a product development organization is defined as an organizational entity responsible for one or more products. Each product development organization contains multiple, geographically distributed teams. This study focuses on the global product development organization as the unit of analysis (Yin, 2016). Figure 5-3 visualizes the case study design for this study. There are 10 units of analysis, across 3 companies. As part of this study, the researcher conducted 24 in-person focus groups with 7 different globally distributed software development organizations in the USA, Ireland, UK, France, and India. Using Yin (2014)’s modeling approach as a guide, C1, C2, and C3 are three separate contexts for the cases. ORG-A through ORG-G, BU1, BU2, and BU3 are the software development organizations that serve as the units of

analysis (Yin, 2014) for this study. BU1 and BU2 also serve as *nested contexts* (Yin, 2014) for the units of analysis.

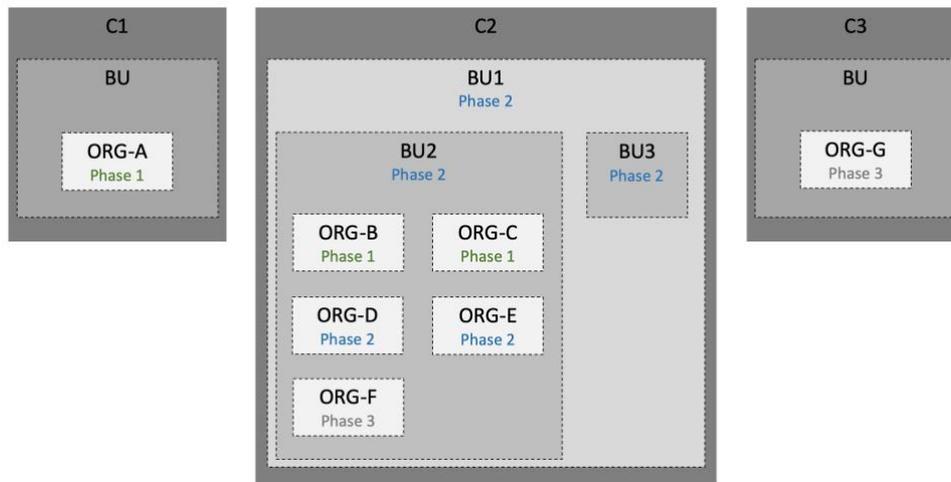


Figure 5-3 Case study design showing contexts and embedded units of analysis

(Arrow et al., 2000) take the position that groups cannot be adequately understood as collections of independently acting individuals. Instead, they focus on “*relationships among people, tools, and tasks, activated by a combination of individual and collective purposes and goals that change and evolve as the group interacts over time*”. Groups are “*adaptive, dynamic systems that are driven by interactions both among group members and between the group and its embedding contexts*”. Figure 5-3 illustrates the embedded context of the software product development organizations in this study. Software development organizations are adaptive systems that actively engage with their embedded context in two-way *interchanges* (Arrow et al., 2000) or what Eoyang (2001) calls transforming *exchanges* (section 4.2.3). The units of analysis in this study “*acquire members, projects, and tools from their embedding contexts*” (Arrow et al., 2000). This study design considers these embedding contexts and interactions to achieve a richer understanding of impediments to flow in software product development organizations. The researcher conducted a total of twenty-four focus groups with each of ORG-A through ORG-G. This study directly engages with hundreds of participants from 10 large software development organizations in 5 countries. These study participants are representative of around 6,000 people working on multiple product lines in multiple

technology domains. Approximately 190 people participated in focus groups, many of them more than once. This study collected 1,038 self-signified micronarratives; 931 of these were captured electronically using the sensemaking data gathering tools (see section 5.6.2), and another 107 captured manually during focus groups. The flow metrics analyses include artefact analysis from five value streams.

Table 5.1 profiles each of the software development organizations that serve as units of analysis for this study. Table 5.2 shows each organization in this study in relation to the principles of lean product development from section 2.2.2.

Table 5.1 Profile of the organizations that are the units of analysis for this study

Organization	HQ	Locations	Value Streams	# Employees (Approximate)	Deployment Model	Domain	Org Structure
ORG-A	Ireland	Ireland, India, USA, UK	Core products, customizable per customer	200	Hosted Cloud-based solution	Finance; Financial Technology (FinTech)	Traditional hierarchy; separation of functions along multiple reporting lines
ORG-B	France	UK, France, Israel, India	Core product, customizable per customer; Integrated with multiple other products as part of a solution	250	Deployed on custom hardware	Technology; Computer software and hardware	Traditional hierarchy; separation of functions along multiple reporting lines
ORG-C	USA	USA, Canada, UK, France, Israel, India	Large-scale solution, multiple products, customizable per customer	1000	Hosted Cloud-based solution	Technology; Computer software and hardware	Traditional hierarchy; separation of functions along multiple reporting lines
ORG-D	India	India, Israel, UK	Core product, customizable per customer; Integrated with multiple other products as part of a solution	1500	Deployed on custom hardware	Technology; Computer software and hardware	Traditional hierarchy; separation of functions along multiple reporting lines
ORG-E	USA	USA, UK, India	Core product, customizable per customer; Integrated with multiple other products as part of a solution	300	Deployed on custom hardware	Technology; Computer software and hardware	Traditional hierarchy; separation of functions along multiple reporting lines
ORG-F	France	UK, France, Israel, India	Core product, customizable per customer; Integrated with multiple other products as part of a solution	350	Hosted Cloud-based solution	Technology; Computer software and hardware	Traditional hierarchy; separation of functions along multiple reporting lines
ORG-G	UK	Ireland, India, USA, UK	Core products, customizable per customer	300	Hosted Cloud-based solution	Finance; Financial Technology (FinTech)	Traditional hierarchy; separation of functions along multiple reporting lines
BU1	USA	USA, Canada, UK, Europe, Middle East, India, AJPAC	Responsible for multiple product development organizations and multiple lines of business	8000	Multiple Cloud-based and on-prem solutions	Technology; Computer software and hardware	Traditional hierarchy; separation of product responsibility along multiple reporting lines
BU2	USA	USA, Canada, UK, Europe, Middle East, India, AJPAC	Responsible for multiple product development organizations	5000	Multiple Cloud-based and on-prem solutions	Technology; Computer software and hardware	Traditional hierarchy; separation of product responsibility along multiple reporting lines
BU3	USA	USA, Canada, UK, Europe, Middle East, India, AJPAC	Responsible for multiple products and systems	1200	Multiple Cloud-based and on-prem solutions	Technology; Computer software and hardware	Traditional hierarchy; separation of product responsibility along multiple reporting lines

Table 5.2 State of Lean Product Development principles in the organizations studied

	ORG-A	ORG-B	ORG-C	ORG-D	ORG-E	ORG-F	ORG-G
Value	Value defined by multiple customers	Value defined internally	Value defined by multiple customers	Value defined by multiple customers	Value defined by multiple customers	Value defined by multiple customers	Value defined by multiple customers
Value Stream	Responsible for end-to-end value stream	Responsible for specific deliverables as part of a larger value stream	Responsible for end-to-end value stream	Responsible for end-to-end value stream	Responsible for end-to-end value stream; In other cases, part of a larger value stream	Responsible for end-to-end value stream; In other cases, part of a larger value stream	Responsible for end-to-end value stream
Flow	Discontinuous, not smooth	Discontinuous, not smooth	Discontinuous, not smooth	Discontinuous, not smooth	Discontinuous, not smooth	Discontinuous, not smooth	Discontinuous, not smooth
Pull	Transitioning from push to pull; revealing impediments	Transitioning from push to pull; revealing impediments	Transitioning from push to pull; revealing impediments	Transitioning from push to pull; revealing impediments	Transitioning from push to pull; revealing impediments	Transitioning from push to pull; revealing impediments	Transitioning from push to pull; revealing impediments
Perfection	Stated goal to have a system of continuous improvement; no formal system in place; some teams using retrospectives; starting org-wide improvement effort	Stated goal to have a system of continuous improvement; no formal system in place; using retrospectives at a team and org level; starting org-wide improvement effort	Stated goal to have a system of continuous improvement; no formal system in place; some teams using retrospectives; starting org-wide improvement effort; some experience with managing impediments	Stated goal to have a system of continuous improvement; a formal system in place; using retrospectives at a team and org level; enhancing org-wide improvement effort; some experience with managing impediments	Stated goal to have a system of continuous improvement; no formal system in place; some teams using retrospectives; starting org-wide improvement effort	Stated goal to have a system of continuous improvement; no formal system in place; using retrospectives at a team and org level; occasional org-level improvement focus; enhancing org-wide improvement effort; some experience with managing impediments	Stated goal to have a system of continuous improvement; no formal system in place; some teams using retrospectives; occasional org-level improvement focus; refreshing org-wide improvement effort

5.5.2 Research Phases

This study adopted a three-phase research approach. Each research phase resulted in a set of findings from a combination of focus groups, sensemaking instruments, and artifacts analysis. Table 5.3 shows each of the three phases and the primary research methods used during each phase. It also notes the organizations involved in each phase, the research objective addressed, and the primary focus of each data set collected during each phase.

Table 5.3 Summary of the three research phases of this study

Research Phase	Company	Organization	Data Set Code	Research Objective	Focus	Primary Research Method
1	C1	ORG-A	FG-A1	RO2(i)	Analyze System Patterns	Focus Group Research
1	C1	ORG-A	FG-A2	RO2(ii)	Identify impediments	Focus Group Research
1	C1	ORG-A	FG-A3	RO2(iii)	Make sense of impediments	Focus Group Research
1	C1	ORG-A	FG-A4	RO2(iii)	Make sense of impediments	Focus Group Research
1	C2	ORG-B	FG-B1	RO2(iii)	Make sense of impediments	Focus Group Research
1	C2	ORG-C	FG-C1	RO2(iii)	Make sense of impediments	Focus Group Research
1	C1	ORG-A	FG-A5	RO2(iv)	Resolve impediments	Focus Group Research
1	C1	ORG-A	FG-A6	RO2(iv)	Resolve impediments	Focus Group Research
1	C1	ORG-A	FG-A7	RO2(iv)	Resolve impediments	Focus Group Research
1	C2	ORG-B	FG-B2	RO2(iv)	Resolve impediments	Focus Group Research
2	C2	BU2	FMA1	RO2(i)	Analyze System Patterns	Artifact Analysis
2	C2	BU2	FMA3	RO2(i)	Analyze System Patterns	Artifact Analysis
2	C2	BU3	FMA2	RO2(i)	Analyze System Patterns	Artifact Analysis
2	C2	ORG-E	FG-E1	RO2(i)	Analyze System Patterns	Focus Group Research
2	C2	ORG-E	FG-E2	RO2(ii)	Identify impediments	Focus Group Research
2	C2	ORG-D	FG-D1	RO2(iii)	Make sense of impediments	Focus Group Research
2	C2	ORG-E	FG-E3	RO2(iii)	Make sense of impediments	Focus Group Research
2	C2	ORG-D	FG-D2	RO2(iv)	Resolve impediments	Focus Group Research
2	C2	ORG-E	FG-E4	RO2(iv)	Resolve impediments	Focus Group Research
2	C2	BU2	SMS1	RO2(i)	Analyze System Patterns	Narrative Research
2	C2	BU1	SMS3	RO2(i)	Analyze System Patterns	Narrative Research
2	C2	BU3	SMS2	RO2(i)	Analyze System Patterns	Narrative Research
2	C2	BU2	SMS1	RO2(ii)	Identify Impediments	Narrative Research
2	C2	BU1	SMS3	RO2(ii)	Identify Impediments	Narrative Research
2	C2	BU3	SMS2	RO2(ii)	Identify Impediments	Narrative Research
3	C2	ORG-F	FG-F1	RO2(i)	Analyze System Patterns	Focus Group Research
3	C3	ORG-G	FG-G1	RO2(i)	Analyze System Patterns	Focus Group Research
3	C2	ORG-F	FG-F2	RO2(ii)	Identify impediments	Focus Group Research
3	C3	ORG-G	FG-G2	RO2(ii)	Identify impediments	Focus Group Research
3	C2	ORG-F	FG-F3	RO2(iii)	Make sense of impediments	Focus Group Research
3	C3	ORG-G	FG-G3	RO2(iii)	Make sense of impediments	Focus Group Research
3	C2	ORG-F	FG-F4	RO2(iv)	Resolve impediments	Focus Group Research
3	C3	ORG-G	FG-G4	RO2(iv)	Resolve impediments	Focus Group Research

The study began with a literature review, which helped to define the initial research objective and conceptual framework. Phase 1 achieved research breadth and incorporated 10 focus groups in 3 different organizations. This phase helped inform which impediments are most impactful to flow in these organizations. This phase explored lean tools for identifying and resolving impediments. Phase 1 helped to identify gaps in the literature and in the research design. The researcher incorporated sensemaking into this study as a result of phase 1. Phase 2 introduced sensemaking to this study. This

phase incorporated 6 focus groups from 2 additional organizations and introduced distributed sensemaking across 3 large product development organizations. Micronarrative-based sensemaking data for distributed sensemaking was collected electronically and in person from participants around the world. Phase 3 provided a deeper exploration of complexity-informed approaches to managing flow and resolving impediments. Phase 3 contained 8 focus groups across 2 additional organizations. Phase 3 focused completely on sensemaking and CAS approaches to understanding flow and impediments in organizations.

Analysis of data (as detailed in section 5.7) occurred following the completion of each phase. The analysis of one phase was used as input to inform the subsequent phase. Phase 1 explored impediments to flow dominantly from a lean perspective, exploring topics such as *value streams*, *kaizen*, and *A3 problem solving*. Phase 2 incorporated *sensemaking*, *experiments*, *goal setting*, and other elements to explore impediments to flow from a sensemaking and complexity-informed perspective, as well as continuing to explore lean-informed approaches. Phase 3 explored impediments to flow exclusively from a complexity and sensemaking perspective, probing deeper into topics raised during Phase 2. The researcher conducted an overall detailed analysis after the phase 3 data was available.

5.5.3 Literature Review

The funnel in Figure 1-1 shows the significant areas of influence that informed this study. Each area has its own body of literature that formed part of the literature review. The literature review began in phase 1 of this study with a review of the lean and agile software development literature and literature on complex adaptive systems. The review led to literature on lean product development, TPS, Ohno, Shingo, Deming, Goldratt, Reinertsen, and others (discussed in Chapter 2). Based partly on the literature review and partly on early analysis of phase 1 data, the focus of the study evolved from a focus on waste to a focus on flow and impediments to flow. As well as research literature from the domain of

software development, phase 1 led to relevant research from multiple domains, including healthcare, manufacturing, and product development. The initial literature review of CAS (discussed in Chapter 4) focused on understanding how properties of complex adaptive systems theory relate to software development organizations and led to the researcher's understanding that software development organizations are complex adaptive systems. Phase 2 of the study added more literature on previous topics and introduced a focus on sensemaking in organizations. In particular, the works of Weick, Snowden, and Eoyang (discussed in Chapter 4) were influential in informing both the literature review and the conceptual framework. This phase also added more literature on flow metrics. The researcher chose throughput, cycle time, and CFDs (discussed in Chapter 3) because of their ability to diagnose systems. In phase 3, the literature review dove deeper into sensemaking and narrative literature, as well as reviewing additional literature on organization culture, leadership, patterns, and change in complex systems (discussed in Chapter 4). To ensure the literature review remained current with contemporary research, the researcher continued to review literature related to topics from previous phases as new relevant literature emerged, e.g., other research on flow, sensemaking, complexity.

5.5.4 Use of Research Methods in this Study

The relationships between the conceptual framework and the three research methods are shown in Figure 5-4. Yin (2014) disputes the view held by many researchers that different research methods ought to be used in a hierarchical fashion, with different methods confined to either preliminary or descriptive phases of the research. Anderson et al. (2005) concur with this perspective, noting that case studies are useful “*at any level of knowledge development*”. The combination of focus groups, narrative research, and case studies means that each approach “*provides a different way of knowing a phenomenon, and each leads to unique insights*” (Riessman, 2008).

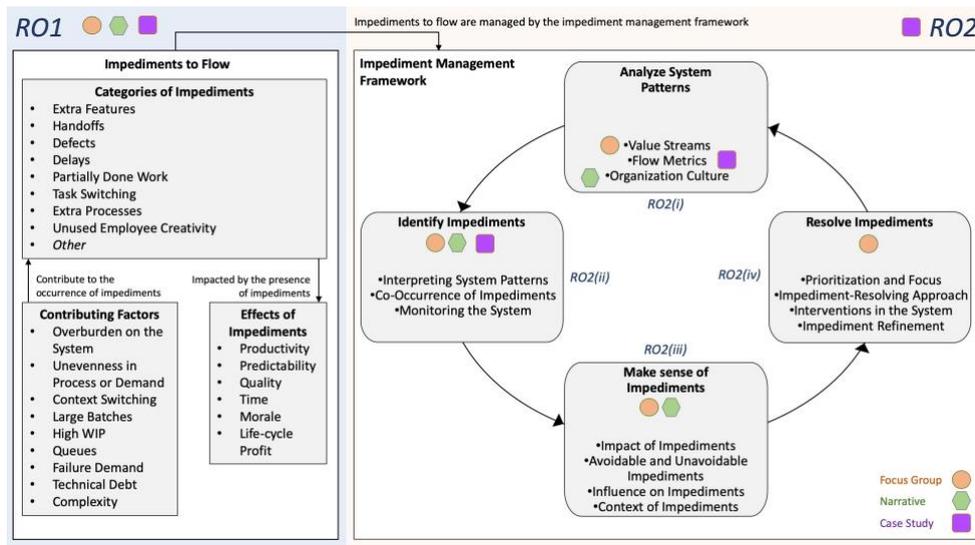


Figure 5-4 Relationship of research methods, objectives, and conceptual framework

The Use of Focus Group Research in This Study

Focus groups are used to collect data (see section 5.6.1 for more detail) that contribute to all four parts of Research Objective 2. In addition to the literature review from Chapter 2, focus groups helped to shape the researcher’s thinking on impediments and waste from a lean thinking perspective. In particular, this process helped the researcher refine the categories of impediments to flow. This approach resulted in the researcher using sensemaking approaches and narrative methods more explicitly to identify and make sense of impediments.

Focus groups are used to gather data and gain an understanding of the impediments faced by the organizations. Specifically, focus groups are used in this study to work with groups to analyze system patterns, identify impediments, understand how they make sense of impediments, and understand the approaches they take to resolve impediments. Table 5.4 shows a summary of the focus groups conducted in this study with participants from company C1. Table 5.4 also shows some of the group demographics.

Table 5.4 Summary of focus groups conducted as part of this study

	ORG-A	ORG-B	ORG-C	ORG-D	ORG-E	ORG-F	ORG-G	
Focus	System Patterns	1				1	1	
	Identify Impediments	1				1	1	
	Make sense of Impediments	2	1	1	1	1	1	
	Resolve Impediments	3	1		1	1	1	
Groups	Location of Focus Group	Ireland	France	UK	India	USA	France	Ireland
	Locations represented by participants	Ireland, India	France, UK	USA, UK, France, Israel, India	UK, Israel, India	USA, UK, India	France, UK, Israel	Ireland, UK, India
	Forum	Senior Team	Product team	Organization leadership team	Organization leadership team, Product teams	Organization leadership team, Product teams	Senior Team	Senior Team
	Roles represented	Directors, managers, team leads, technical leads, architects, business analysts	Directors, managers, engineers	Directors, managers, engineers, product managers	Directors, managers, engineers	Directors, managers, engineers	Directors, managers, engineers	Directors, managers, team leads, technical leads, architects, business analysts
	Functions represented	Engineering, product management, program management, operations, service delivery, architecture	Engineering, product management, program management, operations, service delivery, architecture					

Morgan and Botorff (2010) emphasize that there is no one right way to conduct focus groups. There are, instead, many options from which researchers select to match the needs of the project. Understanding why the research is being done, and the needs of the study, is essential to understanding and informing how to do the research (Morgan et al., 2008). This study uses focus groups whose members are practitioners in software development, coming from a wide variety of roles across multiple organizations.

Arrow et al. (2000) warn that *“when we study groups by creating ad hoc laboratory groups that have no past and no anticipated future beyond the single session, we are thereby imposing not just methodological limitations but substantive ones as well.”* The focus groups in this study are formed of participants who have a shared history and an anticipated shared future. The participants in each focus group are work colleagues from their respective organizations. This study follows the guidance of Morgan (1997), who uses focus groups to learn about participants’ experiences rather than their opinions. Morgan (1997) emphasizes experiences and perspectives over opinion, because *“even self-reported behavior is more useful as data than opinions that have an unknown basis in behavior”*. This also fits well with the sensemaking approaches of Snowden (2010) and Weick (1995), as described in section 4.3. This guidance also informs the construction of the narrative capture instrument, as discussed in section 5.4.3. This research confirms the findings from Morgan (1997) that discussion of experiences produces a livelier dynamic in the group – people are happy to share and compare experiences, while they are often reluctant to challenge someone else’s opinion.

There is no consensus in the literature on an ideal group size. The dominant concerns appear to be the moderator’s ability to control the focus, and the quality of the discussion and output. In many cases where focus groups are used, the researcher is not a skilled facilitator, and needs to acquire those skills or use a trained facilitator to help with the focus group (Morgan, 2019).

The Use of Narrative Research in This Study

Narrative capture is used to capture self-signified micronarratives about the experiences of people working in organizations, to identify impediments through those experiences, and to understand more about the context of impediments in organizations. This contributes directly to Research Objective 2.

This study is not concerned with the creation of narrative by the researcher. It is concerned with capturing and analyzing micronarrative fragments from people in the software development organizations in this study. Micronarrative and its relationship with sensemaking is discussed in detail in section 4.3.4. Squire et al. (2014) note the following basic steps that provide researchers with a coherent framework to design and conduct narrative research:

- **Situating the epistemological approach.** This study takes a constructivist approach, within which narrative itself is a topic. The alternative would be a naturalistic approach, treating narratives as resources. Section 5.3.4 further describes the epistemological position of this research.
- **Selecting the analytical model or models.** This study focuses on the context of the narratives, as well as the content. This study is not concerned with the structure of the narratives.
- **Collecting or co-constructing data to be analyzed.** Narratives can be ‘discovered’ in many different sources. This study uses a dedicated narrative collection instrument to collect narratives. The process of narrative collection is described in section 5.6.2.
- **Selecting or preparing narratives to be analyzed.** The selection of narratives for this study is predetermined by the decision to use a narrative database. All of the narratives collected are part of the analysis. Because this study takes a constructivist perspective, there is no preparation of narratives – narratives are analyzed in their natural form. In coherence with the complexity theory lens used for this study, the

researcher looks for patterns in the narratives, and in the interpretive data accompanying the narratives.

- **Analyzing narratives.** Section 5.7.2 describes the narrative analysis process used in this study. Section 5.4 describes how narrative analysis is used in conjunction with analysis of the focus group data and case study data.
- **Writing up and disseminating research.** This thesis is an example of writing up and disseminating the results of narrative research. Chapter 6 deals specifically with writing up the results of the narrative capture process and triangulates the narrative write-up with that of the focus group and case study analyses.

This study uses narrative fragments, or micronarrative, captured from participants in the field as a means towards understanding and making sense of impediments to flow, and the contextual factors surrounding impediments. As part of the overall sensemaking process, the set of signs that constitute the micronarrative are captured primarily as text fragments. It is also possible to use drawings and other visual elements, as well as audio and video elements. The narrative capture instrument is used to collect data in-person data collection and online. The process is described in section 5.6.2.

For stage two analysis, this study employs some basic quantitative methods to analyze the narrative materials, independently of their character, as suggested by Squire et al. (2014). Examples include a quantitative content analysis on specific terms, or codes (defined in section 5.7.5 below), related to flow or impediments. This study performs such an analysis. In addition, this study uses a sensemaking lens to analyze the narrative data, by looking at patterns created in the self-signified data provided by respondents. A key point here is that this analysis is possible and useful without reference to the narrative content. This will be demonstrated in detail in Chapter 6.

The Use of Case Study Methods in This Study

Case studies are used in this study to collect data about impediments (see section 5.6.3 for more detail), to collect and understand flow metrics, and to provide additional context about the organizations in this study. Case studies are used to capture and analyze flow metrics, contributing to Research Objective 2.

The area of interest of this study is the software development organizations that deliver value to customers through developing products, hence units of analyses include product organizations. Case study methods are used in this study to understand flow and impediments in such organizations. Case study researchers need to be strategic in defining the cases, as the definition of the case dictates the depth of analysis appropriate to the research, and determines “*the applicability of systems theory to strengthen the analysis of the case*” (Anaf et al., 2007). Section 5.6.3 discusses the data collection methods, and section 5.7.3 describes the data analysis approaches.

5.6 Data Collection Methods

Each of the three research methods provides different sources of data collection for this study. The researcher used several tools to support data collection in this study. These are summarized in Table 5.5.

Table 5.5 Data collection tools used in this study

Name	Description	Usage	Method
NVivo	NVivo is commercial software from QSR International.	Used to store all relevant data from this study.	• All
SenseMaker Collector	Commercial software from Cognitive Edge. It is part of a suite of tools for sensemaking and working with narratives.	Used to collect narrative fragments from participants. Narrative data was later imported into NVivo for further coding and analysis.	• Narrative
Evernote	Commercial note-taking software. The researcher uses versions for desktop, iPhone, and iPad to record field notes, observations, and other data.	Used to capture field notes during data collection; Used to collect interesting and useful information during literature review. Field notes were later exported into NVivo.	• All
Paper Notebooks	The researcher used physical paper notebooks.	Used to capture field notes. The researcher prefers Moleskine brand notebooks, and generally carries at least one everywhere. Selected notes were transcribed into Evernote and imported into NVivo.	• Case Study • Focus Group
Sticky Notes	Sticky notes that come in different colors, shapes, and sizes.	Used to capture data about impediments in focus groups; Used to collect micronarrative data during in-person sessions.	• Focus Group • Narrative
Voice Memos	Voice Memos is an iPhone voice-recording app that comes with the iPhone.	Used to record focus groups. Recordings were imported into NVivo.	• Focus Group
Camera	Camera is the built-in camera app on the iPhone.	Used to take photos of artifacts created in focus groups, data collected from focus groups; Photos of focus group participants; Used to record interesting and relevant artifacts during observations and site visits; Select photos were imported into NVivo for analysis.	• All

5.6.1 Collection of Focus Group Data

This study follows a set of generally-accepted principles for focus group design and data collection (Morgan, 2019). The general approach used by the researcher is to conduct a focus group that lasts from 45-90 minutes, depending on the number of participants and the richness of the discussion. Figure 5-5 visualizes the general format used by the researcher to collect data in focus groups.

not require a tool suite, though it does add an extra dimension of richness to the data collection and analysis for this research. Using an online tool also allowed the researcher to collect data from participants in different locations around the world.

Micronarratives are captured from participants and stored in a narrative database. Narratives are captured both in-person and using an online tool. The researcher used dedicated tools to analyze the narratives stored in the narrative database, the process of which is discussed in more detail in section 5.7.2. Finally, the results were written up, and the results are presented in Chapter 6.

Maitlis and Christianson (2014) note that sensemaking involves the active authoring of frameworks for understanding. Narrative collection frameworks are an aid to understanding the organization, the impediments, and the context within which impediments occur. The researcher employed multiple narrative frameworks for the purpose of this study (Kurtz and Snowden, 2003, Snowden, 2010). The narrative collection sensemaking instruments developed by the researcher include the following elements:

- **Prompt Question.** The prompt question is always the entry point. The prompt begins by describing a scenario, e.g., “*you have a friend who is thinking about joining your team at work*” and asks the respondent to recall a significant event at work that frustrated or delighted them. This event is often referred to as an “experience”. The seed question that invites people to recall a recent event that either frustrated or delighted them is an example of a cue, as discussed in section 4.3.2.
- **Narrative collection.** The respondent is then asked to record what they would tell their friend, and to give the experience a short title, e.g., to imagine the title as a newspaper headline or Twitter hashtag.
- **Signification using triads.** The instrument several triads. A *triad* is a triangle shape that has three labels, or signifiers. A sample triad from the narrative collection instrument is shown in Figure 5-6. The question accompanying this triad is “*Events relate to ...*”. This particular triad is designed to help understand the balance of how people, processes, and

customers contribute to outcomes in the organization, and how that correlates with flow and impediments. Respondents place their answer to a question within the triangle by placing a dot inside the triangle. Balancing three elements creates a cognitive tension that forces the respondent to find a resolution and leads to deeper reflection and insights. There is a “N/A” option if respondents feel that a particular triad is not applicable to their experience.

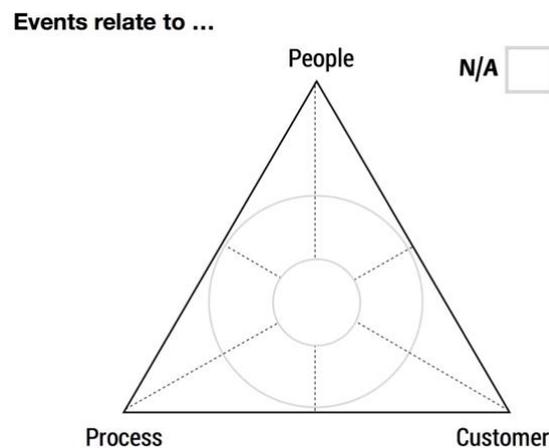


Figure 5-6 Example triad used in micronarrative collection instrument

- **Signification using dyads.** The instrument has several dyads. A dyad is also referred to as a *polarity*. Respondents place their answer to a question on a spectrum between two polar opposite positions. A sample dyad from the framework is shown in Figure 5-7. The question accompanying this dyad is “*What was valued in the experience you shared?*” This dyad is designed to help understand the balance between *working software* and *comprehensive documentation*. These two signifiers derive from one of the values of the agile manifesto, described in Chapter 2. There is a “N/A” option for cases where respondents feel that this dyad is not applicable to their specific experience.

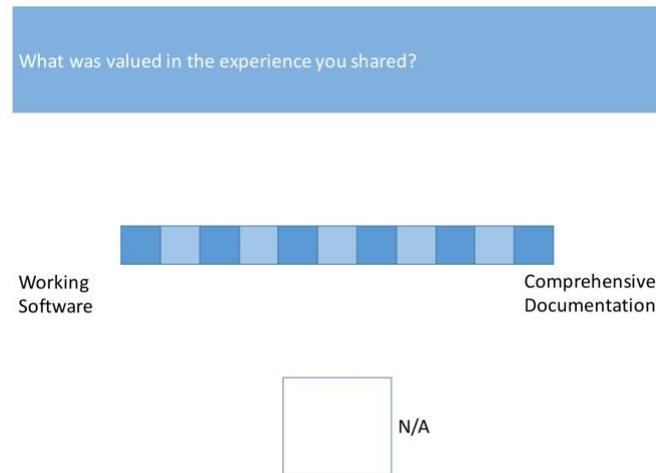


Figure 5-7 Example dyad used in micronarrative collection instrument

- **Signification using multiple-choice questions.** Data about impediments is central to the research objective of this study and is specifically captured in the form of a response to a multiple-choice question, where the respondent can select multiple options. The instrument includes a set of multiple-choice questions that capture additional data that give more detail about the surrounding context, including the emotional intensity, the frequency of occurrence, their reason for sharing this particular experience, and what roles were involved in this experience.
- **Demographics data.** The instrument includes several demographics questions, including the role of the respondent, their location, and what product or area they work on.

The micronarrative sensemaking instrument was used in two ways. The researcher conducted in-person data collection sessions using paper, and online collection using the SenseMaker Collector application. Van Der Merwe et al. (2019) describe the use of SenseMaker as a research tool. For in-person sessions, the researcher conducted a number of data-gathering sessions with participants. The narrative capture framework was drawn out on posters and placed on walls around a room. The researcher, acting as facilitator, guided participants through the process of capturing narrative, self-signifying their individual narratives, and examining the emerging patterns. For online data collection, the narrative capture framework was

implemented in a tool suite called SenseMaker. One of the tools in the SenseMaker suite is called *Collector* and is used for collecting narratives from participants. In this case, the application guides participants through the process of entering narrative and self-signifying their narratives. There is a Web version and an app version that runs on iOS and Android devices. The researcher used both the Web version and app version to collect data from participants.

As far as the researcher is aware, this is the first study that uses SenseMaker in the context of software development organizations. SenseMaker has been used in other studies, primarily in the NGO (non-governmental organization) and development sectors. The approach in this study is similar to approaches used in these other studies, i.e., using SenseMaker as an aid in micronarrative-informed sensemaking to gain insights into complex phenomena. For example, Vredeseilanden / VECO, now rebranded to Rikolto (Rikolto, 2017), is a Belgian NGO that aims to contribute to viable livelihoods for organized family farmers with a particular focus on value chain development. Deprez et al. (2012) report on their use of SenseMaker to understand the extent to which SenseMaker could be useful for VECO and its partners to support and improve value chain development interventions. Guijt (2016) provides another perspective on this same study. A number of studies use SenseMaker to understand the underlying factors contributing to child marriage among Syrian refugees in Lebanon (Dube et al., 2019, Bartels et al., 2018, Bakhache et al., 2017). Achieving “*full and productive employment, and decent work, for all women and men by 2030*” is a sustainable development goal of the United Nations (United Nations Economic and Social Council, 2019). An Oxfam report describes the use of SenseMaker to understand worker’s experiences in the context of this UN goal (Mager et al., 2018). Lynam and Fletcher (2015) use SenseMaker to gain insight into the positions and orientations of different social actors in relation to climate change. From a government practitioner’s perspective, Milne (2015) uses SenseMaker in a study to explore whether sensemaking can inform climate change adoption policy. Dunstan (2016) uses SenseMaker in a study that investigates the lived emotional experiences

(section 4.3.5) of sales people, in order to achieve insight to how the anticipatory emotions of salespeople impact upon their sales effectiveness. *Participatory market chain approach* (PMCA) is a methodology for improving the performance of poorly-coordinated value chains. Khondker et al. (2018) use SenseMaker as part of a study to assess the effectiveness of PMCA for promoting aquaculture value chain development in Bangladesh and Nepal.

5.6.3 Collection of Case Study Data

The data collection strategy in this study is designed to complement the complexity and sensemaking perspective, as described in Chapter 4, and in sections 5.3.3 and 5.4.4 above. Of the six common sources of evidence noted above (Yin, 2012), this study uses documents, archival data, observation, and interviews.

- **Documents.** The researcher was given access to multiple documents and document repositories for the organizations that either relate directly to impediments and flow or provide useful context information. The organizations produce several documents as part of their work. These documents include Word documents, PowerPoint slide decks, Excel spreadsheets, emails, and online Wikis. These provide additional supporting data and context for this study, including decisions made by the organizations concerning flow and impediments. The researcher was given access to multiple document sources used by the organizations selected for this study. Selected documents were added to NVivo and included in the coding analyses.
- **Archival Data.** In the context of this study, archival records refer specifically to tools used by organizations to manage their software development processes. Specifically, the researcher was given access to application lifecycle management tools that track the work of the software development organizations. Other archival records include spreadsheets and reports created by software development organizations to report on their work. Among other things, these tools produce flow metrics data used in this research. The tools in question have built-in

capabilities to produce reports of cycle time, throughput and cumulative flow. Batch size, WIP, and queue size data can also be readily determined from the tools. Tool reports were added to NVivo and coded with the rest of the data.

- **Observation.** Observations were captured as field notes in Evernote®, and later imported into NVivo. Sometimes field notes were written in physical notebooks, and later transcribed or scanned into Evernote. Other observational data included photographs which were also imported into NVivo.
- **Interviews.** Semi-structured and unstructured interviews were used as a source of data in this research, with the purpose of providing additional contextual data on the impediments faced by the organizations. One reason for not conducting extensive formal interviews with every individual is that the unit of analysis for this study is the group, not the individual, hence the use of focus groups, also sometimes referred to as group interviews (McLafferty, 2004). Interview data were captured in Evernote, and later imported into NVivo.

5.7 Data Analysis Methods

Table 5.6 summarizes the tools used in this study for data analysis.

Table 5.6 Data analysis tools used in this study

Name	Description	Usage	Method
NVivo	NVivo is commercial software from QSR	Used to code all relevant data from the study. Used to query the data and look for patterns.	<ul style="list-style-type: none"> • Case Study • Focus Group • Narrative
SenseMaker Explorer	SenseMaker Explorer is commercial software from Cognitive Edge. It is part of a suite of tools for sensemaking and working with narratives	Used to analyze patterns in narrative data, and create graphics based on the data.	<ul style="list-style-type: none"> • Narrative
Lifecycle Management Tools	Commercial lifecycle management tools used by the organizations in this study	Analyze flow metrics in detail. Create reports from flow metrics. Analyze additional contextual data about the organizations, and how they are working.	<ul style="list-style-type: none"> • Case Study
Tableau	Commercial data visualization software from Tableau	Analyze narrative data. Analysis dashboards were created from data exported from SenseMaker.	<ul style="list-style-type: none"> • Narrative
Excel	Commercial spreadsheet software from Microsoft	Used to analyze flow metrics data; Used to analyze impediment data and demographics data from narratives; Used to generate charts and graphs	<ul style="list-style-type: none"> • Case Study • Focus Group • Narrative

5.7.1 Analysis of Focus Group Data

The type of analysis of focus group data is dependent on the needs of the study, and the kinds of reporting that will be done based on the analysis (Morgan, 1997). One example is reporting in a study where focus groups are the sole source of data, and the intended reporting will be via a journal paper. This is different from a study where focus groups serve as one part of the research approach where the analysis is driven by the needs of the larger study (Morgan, 1997). The latter example is closer to the usage of focus groups in this study. Focus group data was coded using NVivo, using the coding process described in section 5.7.5.

5.7.2 Analysis of Micronarrative Data

Micronarrative data was analyzed using SenseMaker Explorer, Tableau, and Excel. Narrative data was also imported into NVivo, and coded using the

process described in section 5.7.5 The first point of analysis of narrative data is the self-signification performed by each participant on their own narrative, as described by Snowden (2010) and Van Der Merwe et al. (2019). For analysis purposes, the Tableau dashboards described in Table 5.6 allow for filtering based on any of the fields from the multiple-choice questions and the demographics data that are part of the narrative capture instrument, as described in section 5.6.2 above. It also allows for filtering based on selection of one or more responses in triads and dyads. This allows the researcher to explore the data from different dimensions, and to look for patterns from multiple perspectives. SenseMaker Explorer is used for deeper analyses of the narrative data. The Explorer software allows the researcher to see relationships across different elements of the data, e.g., how impediments correlate with data from triads and dyads. The SenseMaker software is designed to enable an analysis based on CAS theory. For example, it enables the researcher to look for patterns (see Chapter 4). It also enables the researcher to visualize attractors and boundaries (section 4.2.3) through fitness landscapes. Section 6.3 includes several examples based on the findings in this study.

5.7.3 Analysis of Case Study Data

Case study techniques are perfectly suited to studying systems, however *“the eventual analysis of the data will be guided by the overall relationship of parts within the system”* (Anaf et al., 2007). This study follows the guidance of Anderson et al. (2005), and uses case studies through the theoretical model of CAS theory. Anderson et al. (2005) develop a set of extensions for extending traditional ideas about the execution of case study design, by using complexity science as a “blueprint”. The extensions suggest the following:

- **Understand interdependencies.** According to Anderson et al. (2005), it is important to understand not only the elements, or agents, in a CAS, but also the interdependencies and interactions that combine to create a whole. This equates to understanding the containers, differences, and exchanges in a system, as described in section 4.2.3.

- **Be sensitive to dimensions of relationships.** According to Anderson et al. (2005), researchers who choose to use complexity theory need to have a richer understanding of relationships in their case studies. Researchers need to pay attention to the diversity in the system and need to try to understand the impact of this diversity. Chapter 4 describes diversity in the context of a CAS.
- **Focus on nonlinearities.** According to Anderson et al. (2005), it is difficult to detect nonlinearities directly, so they recommend looking for instances where small events have led to big outcomes, and vice versa. Nonlinearities are keys to understanding the system, so researchers need tools that help to notice them (Anderson et al., 2005).
- **Look for the unexpected.** Anderson et al. (2005) suggest that researchers need multiple lenses or methods to observe organizations from more than one perspective and time period. A potential weakness with traditional approaches to case studies is that they tend to focus on average events and ignore outliers. Instead, using complexity theory as a lens, researchers should pay attention to outliers as potential sources of new structural arrangements (*containers*) and patterns of behaviors (Anderson et al., 2005).
- **Examine unexpected events.** Researchers should pay attention to how organizations respond in the face of unexpected events (Anderson et al., 2005). People tend to normalize unexpected events, so researchers must be alert to attempts to normalize these (Anderson et al., 2005).
- **Focus on processes as well as events.** Anderson et al. (2005) note that traditional approaches to case studies focus on decision points as major events for understanding organizations. Instead, referencing Weick (1995), they suggest that researchers look for sensemaking properties (see section 4.3.2) to “*reveal the nature of the organization*”. Traditionally, case researchers will focus on what an organization knows, but from a CAS perspective, it is more important to understand how an organization learns (Anderson et al., 2005).
- **Recognize dynamics.** There is a formal and an informal organization (Anderson et al., 2005). The formal organization is the one described in

organization charts and formal documents and policies. The researcher must not let these hide the real nature of the organization, which is defined by the “informal organization” (Anderson et al., 2005). Observation of agents’ interactions and processes helps to explore the informal organization and its dynamics (Anderson et al., 2005).

- **Describe patterns as well as events.** According to Anderson et al. (2005), using case study methods to pay attention to patterns in a system provides richer results and more possibilities for interventions in the system. Chapter 4 describes patterns in a CAS and mentions several ways to identify patterns in a system. Deliberately noticing patterns gets qualitative researchers closer to finding answers about the human condition (Saldana and Omasta, 2017).
- **See patterns across levels.** According to Anderson et al. (2005), organizations are best understood as systems nested within a larger network of systems, in a self-similar or fractal structure. Case studies can be designed to look for self-similarity in systems when analyzing patterns. Section 4.2.3 describes the properties of self-similarity, and the fractal nature of organizations. Section 4.2.4 describes the concept of viewing systems in terms of the parts, the whole, and the greater whole.
- **Understand that patterns change.** Traditional case study design seeks to identify trends and trajectories, whereas a case study design from a CAS perspective looks for patterns, and recognizes that those patterns change over time (Anderson et al., 2005). Chapter 4 describes the concept of patterns in a CAS, and approaches to spotting patterns in systems.
- **Recognize that in any given situation different patterns may be successful.** Anderson et al. (2005), note that in healthcare, much value is placed on identifying and disseminating “best” practices, which is limiting. The same can be said of software development. Therefore, research should consider that there is more than one way of looking at situations that will lead to useful knowledge (Anderson et al., 2005).
- **Shift foreground and background.** According to Anderson et al. (2005), traditional case study approaches put more emphasis on things

like roles and formal organizational positions, which are comparatively easier to identify through such things as official organization charts. Hence, these things tend towards the foreground in terms of researcher focus. A CAS perspective can switch the focus to the “*relations, flows, and exchanges*” between formal roles and positions, bringing their focus from background to foreground.

- **Redefine observer roles.** Anderson et al. (2005) note the co-evolutionary relationship between the researcher and the system, treating the researcher not as a passive observer, but as an “intruder” in the system. How the system responds to this intrusion can reveal a lot about the system itself.
- **Learn the system’s history.** According to Anderson et al. (2005), the way a system is today is “*in large part due to what it was yesterday*”. Understanding the system’s history helps researchers to better understand its current patterns.

Chapter 6 and Chapter 7 present and discuss the findings along selections of these fifteen dimensions, using a purposive sampling strategy as described in section 5.7.6.

The *flow metrics* described in Chapter 3 are analyzed using a combination of project management tools, Excel, and NVivo. The researcher analyses the various flow metrics looking for potential indicators of impediments to flow, and to understand patterns of flow in organizations.

Documents were included in the content analysis process described in section 5.7.4, and in the coding process described in section 5.7.5. Documents are used largely to understand the context within which impediments occur, and to look for evidence of contributing factors (section 2.5.2) that lead to impediments.

Observations were captured as field notes in Evernote, and later imported into NVivo for analysis. Field notes were included in the content analysis process described in section 5.7.4, and in the coding process described in section 5.7.5.

Data and notes from the *semi-structured interviews* are included in the content analysis process described in section 5.7.4, and in the coding process described in section 5.7.5.

5.7.4 Content Analysis

Saldana and Omasta (2017) describe content analysis as a systematic approach to data analysis that examines the words and images in print and media materials, looking primarily for their “*topics, themes, content, and ideas*”. The goal is to “*examine aspects such as frequency, type, correlation, and absence in a body of data to generate manifest readings that infer latent meanings*” (Saldana and Omasta, 2017). The frequency of the use of words is considered to be an important part of a content analysis in a narrative methods study (Yin, 2016). Content analysis is applied to the body of data collected from all three research methods used in this study. It is used to analyze the data for topics, themes, content, and ideas related to flow and impediments. For example, counting the occurrence of references to impediments in the narrative data is helpful to this study.

5.7.5 Coding Process Used in This Study

Coding is a process of assigning codes, in the form of words or short phrases, to capture the meaning of a larger portion of data (Yin, 2016). The researcher decides which codes to use, and what in the data to code. This research employs two coding cycles, referred to as first cycle and second cycle coding. There is a wide variety of possible coding methods within each of these two cycles (Saldana, 2016). Saldana (2016) describes 33 distinct coding methods in first- and second-cycle coding, divided into seven categories. There is an additional cycle that Saldana calls “*first to second cycle coding*”.

First cycle coding methods happen during the initial coding of the data. Of the 33 coding methods described by Saldana (2016), 26 are categorized as first-cycle coding methods. Of those 26, 8 were selected for the purpose of this study.

- **Attribute Coding.** *Attribute coding* is used to provide “essential participant information and contexts for analysis and interpretation” (Saldana, 2016).
- **Magnitude Coding.** *Magnitude coding* supplements an analysis with qualitative, quantitative and/or nominal indicators that enhance the description of the analysis, e.g., to understand frequency and emotional sentiment of impediments.
- **Subcoding.** *Subcoding* consists of “[A] second-order tag assigned after a primary code to detail or enrich the entry” (Saldana, 2016). This study uses *subcoding* to relate parent-child hierarchies of nodes, e.g., a parent note called FLOW METRICS has a child node called THROUGHPUT.
- **Simultaneous Coding.** This study uses *simultaneous coding* where data can be coded with different meanings and contexts, e.g., a paragraph in a document that can be coded as both FLOW METRICS and IMPEDIMENTS.
- **In Vivo Coding.** This study uses *In Vivo coding* to highlight words or phrases in the data that the researcher feels are of particular interest or significance. In Vivo coding can perform a checking function to validate that the researcher has grasped what is of significance to participants.
- **Process Coding.** *Process coding* uses *gerunds*, or active verbs as nouns, to convey the notion of action. Process coding is particularly appropriate for studies such as this one that look for routines and rituals in human systems. This study uses process coding to understand when organizations are dealing with impediments and flow.
- **Concept Coding.** *Concept coding* applies intermediate or higher-level meaning to data. This study uses concept coding to create a set of codes from the conceptual framework, e.g., there are codes for FLOW, IMPEDIMENT, CONTRIBUTING FACTOR, and FLOW METRICS.
- **Emotion Coding.** *Emotion coding* labels the emotions recalled and/or experienced by participants. This study uses STRONGLY POSITIVE, POSITIVE, NEUTRAL, NEGATIVE, and STRONGLY NEGATIVE as emotion codes to understand the sentiment associated with managing impediments in organizations. Saldana (2016) notes that acknowledging

the emotional aspect of the data provides deep insight into participant's perspectives and conditions. Section 4.3.5 discusses the relevance of emotion to sensemaking, and its application to this study. This study combines emotion coding with magnitude coding to understand the range and frequency of emotion across all the impediments.

The next stage is what Saldana (2016) refers to as a "*post-coding transition*" between first cycle and second cycle coding. The goal is to cycle back to the first cycle coding effort so that the researcher "*can strategically cycle forward to additional coding and qualitative data analysis methods*" (Saldana, 2016). Saldana (2016) refers to the coding method in this stage as *eclectic* coding, which could be considered a form of *open* coding, as defined by Glaser and Strauss (1967). Eclectic coding uses a combination of two or more first cycle methods, purposively selected to serve the needs of this study (Saldana, 2016).

This study's *first cycle coding* employed a blending of 8 coding methods, which were already selected to serve the needs of this study. The researcher used the same set of methods for the *after first cycle coding* stage. The difference in this stage is the added perspective and clarity from having coded the first cycle. The researcher produced memos and other artifacts during the first cycle coding.

The main goal of second cycle coding is to "*develop a sense of categorical, thematic, conceptual, and/or theoretical organization from your array of first cycle codes*" (Saldana, 2016). This study employs *pattern coding* as a second cycle coding method. Pattern codes identify emergent themes in the data. The activities in this stage of coding are somewhat loosely defined by Saldana (2016). It is essentially a transitional phase between coding cycles and final write-up. In the context of this study, other data analysis methods are used in conjunction with coding. There are several iterations of first and second cycle coding in combination with analysis of the data.

Figure 5-8 shows the overall structure of the NVivo database used for this study, highlighting the top-level coding structure used in data analysis.

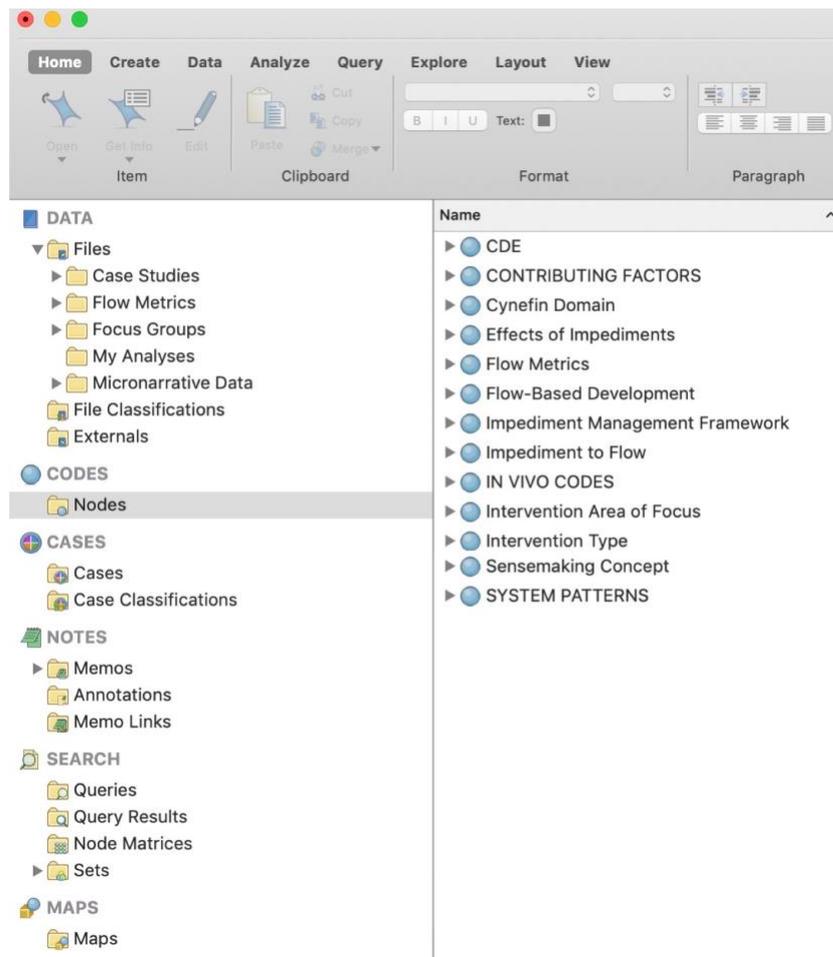


Figure 5-8 NVivo structure highlighting top-level coding structure

The majority of these top-level codes were defined *a priori* based on the outcome of the literature review and the constructs in the conceptual framework in Figure 5-1. These codes served as a “start list” of codes for data analysis (Saldana, 2016). Examples include the parent nodes *CDE*, *Contributing Factors*, *Cynefin Domains*, *Effects of Impediments*, *Flow Metrics*, *Flow-Based Development*, *Impediment Management Framework*, *Impediments to Flow*, *Intervention Type*, and *Sensemaking Concept*. Other top-level codes and several of the child nodes emerged as the researcher gathered data and analyzed the findings. Examples include everything under *In Vivo Codes* and *System Patterns*. Figure 5-9 shows the detailed coding structure used to analyze the data in this study.



Figure 5-9 Complete set of codes used for data analysis

5.7.6 Purposive Sampling

Purposive sampling involves selecting participants or sources of data for a study, based on their “*anticipated richness and relevance of information in relation to the study’s research questions*” (Yin, 2016). Yin (2016) notes that sources should be selected based on their relevance to challenge and support the researcher’s thinking.

The purpose of selecting specific samples is to use those that yield the most relevant and rich sources of data (Yin, 2016). One reason this researcher uses purposive sampling is to deal strategically with the large amount of data collected. The concept of saturation, discussed earlier in this chapter, means that there is limited return in using additional data past a certain point. Palinkas et al. (2015) provide a summary of 15 different strategies for purposive sampling, divided into 3 categories, including those that have an emphasis on similarity, those that have an emphasis on variation, and those that have a non-specific emphasis.

Another consideration is the amount of time needed to transcribe and analyze all of the data from all of the focus groups, sensemaking

micronarratives, and case studies as this would be prohibitive and largely unnecessary. In that context, sampling is not just used to determine which data to collect, but, after collecting large amounts of data, it is also used to help determine what data to include in the analysis.

5.8 Validity and Reliability

Maxwell (2013) refers to the concept of validity in qualitative research as the “*correctness or credibility of a description, conclusion, explanation, interpretation or other sort of account*”. Miles et al. (2014) distinguish between *internal* validity and *external* validity. Yin (2014) describes four tests that establish the quality of all social science research methods. These are construct validity, internal validity, external validity, and reliability.

Construct validity is concerned with ensuring the correct operational measures for the concepts being studied (Yin, 2014). Construct validity is addressed in this study in three ways. First, this study uses flow metrics to understand the quality of flow in organizations, and the impact of impediments. Flow metrics are described in greater detail in Chapter 3. Second, this study is situated in complex adaptive systems. Chapter 4 discusses change in a CAS. Third, the micronarrative sensemaking instrument employs a set of qualitative measures that provide more context about the environment in which impediments occur.

Internal validity attempts to establish causal relationships in studies (Yin, 2014). Internal validity is not applicable to this study because this study is an exploratory study, and internal validity applies to explanatory or causal studies. In addition, as noted in Chapter 4, and in several points in this chapter, this study takes place in the context of complex adaptive systems, and there are not always causal relationships in such systems.

External validity is concerned with defining the domain to which a study’s findings can be generalized (Yin, 2014). This study addresses external validity in several ways. First, this study was conducted in a number of environments, including three companies, and multiple large,

geographically distributed business units and product development organizations in those companies (section 5.5.1). The study's methods of operation are generalizable to software product development organizations. Going further, this study is generalizable to organizations that operate in complex environments, and that must deliver value through the provision of products and services.

Reliability is concerned with demonstrating that the operations of a study can be repeated with the same results (Yin, 2014). The operations, such as data collection, analysis, etc. can certainly be repeated. The description of the operations employed for data collection and analysis aid with the reliability of this study. However, repeating the operations will not lead to the same results.

Some researchers argue that coherence is a test for validity in narrative research. However, Riessman (2008) notes a shift in this perspective influenced by the “*disrupted narratives*” of people living through traumatic events in which the incoherence of narrative fragments can display evidence of a “*double existence*”, and a “*temporal rupture*” that separate recalled events from other aspects of their lives. Riessman (2008) further notes that it is often the “*needful ears*” of the researcher or reader that needs coherence in the narrative, not the person providing the narrative. Researchers can “*forge links and make connections by situating the personal narratives in social and political contexts*”. In this study, personal narratives of respondents help the researcher forge links and make connections with the broader organization in which impediments occur. Riessman (2008) argues that researchers need to appreciate the incoherence in narrative fragments, citing Langer (1991), who notes that “*we need to search for the inner principles of incoherence that make these testimonies accessible to us*”.

The bias of the researcher is a threat to the validity of this research, as it is with any study. To mitigate this, section 5.3.3 above documents these biases in the context of stating the epistemological stance for this study. As acknowledged in section 5.3.3, the domain knowledge of the researcher is a potential threat that can introduce bias. On the other hand, it is also an

advantage as the researcher's experience and knowledge contribute to the validity of the results. Maxwell (2013) identifies researcher bias and reactivity as two specific threats to validity in qualitative research. Both threats are addressed through the inclusion of a section on self-reflexivity (section 5.3.4 above).

Maxwell (2013) outlines several strategies for countering threats to validity:

- **Intensive, long-term involvement.** The researcher's involvement with the case studies and the participants in the narrative capture spanned multiple years. Advantages of this strategy cited by Maxwell (2013) and confirmed by this researcher include access to more data, diverse data, opportunity to develop and test alternative hypothesis during the research, and the ability to check and confirm observations and inferences.
- **Rich data.** Long-term involvement enables the collection of rich data that are detailed and varied and provide a more revealing picture of events.
- **Searching for discrepant evidence and negative cases.** This study aims to incorporate this strategy in two specific ways. First, through purposive sampling, as described in section 5.7.6. Second, through identifying and analyzing patterns throughout the case studies. The role of patterns in case studies is described in section 5.7.3.
- **Triangulation.** Triangulation, as described in section 5.4.5, is the process of collecting data from a diverse range of settings, using a variety of methods (Maxwell, 2013). This study achieves triangulation through using three distinct and complimentary methods.
- **Respondent validation.** This is the act of having participants check the data and interpretations. In this study, the narrative analysis, flow metrics data, and overall data analysis was shared with organizations, and validated with them. However, as Maxwell (2013) points out, participant feedback and validation is no more inherently valid than their initial responses; both should be taken simply as evidence.

- **Numbers.** The appropriate use of numbers in a qualitative study allows the research to make claims that are quantitative in nature, without incorporating quantitative methods. Examples include the ability to specify quantities, show trends, and make numerical comparisons.
- **Comparison.** Collecting and analyzing data from multiple settings, and using multiple methods, enables richer comparison of findings. Chapter 6 includes findings from each of the three research methods. Chapter 7 provides a cross-case, cross-method discussion on the findings.

The scale of this study of flow and impediments is significantly larger than previous studies, allowing for a comprehensive study of flow and impediments in software development organizations. To date, most studies of impediments study teams or projects in single development sites. For example, Murauskaite and Adomaskas (2008) interviewed 3 people from a single development site of undisclosed size about potential bottlenecks. Ikonen et al. (2010) studied waste in a co-located 13-person team using Kanban. Birkeland (2010) reports on the experience of one Scrum team moving from iterations to a flow-based development model. Mujtaba et al. (2010) interviewed 6 people from a development site of 600 about waste and lead time reduction. Petersen and Wohlin (2010) studied Cumulative Flow Diagrams where the unit of analysis consisted of two software components. Swaminathan and Jain (2012) studied flow in a 12-person team developing a Web application. Wiklund et al. (2013) used participant observation to observe a workshop with participants from several teams, and daily Scrum meetings for one Scrum team. They also used artifact inspection of team task boards and conducted semi-structured interviews with 2 people. All are from one department in one site. The study by Dennehy and Conboy (2017) includes observations and artifact analysis of 2 co-located teams from 3 companies. Their study includes interviews with 15 people. The unit of analysis for the flow metrics analysis in Petersen (2012) is the maintenance process of one group from a large company. Yet other studies of impediments and flow do not engage industry participants and settings directly, instead using techniques such as systematic mapping to understand measures and indicators (Feyh and Petersen, 2013) or structured

literature reviews to understand impediments (Carroll et al., 2018). These are all valid approaches and add to the research body of knowledge that this study builds on. The scale and scope of this study is much larger, as detailed in section 5.5.1.

5.9 Summary

This research studies software development organizations in three publicly traded technology companies responsible for delivering products and services to global markets. This chapter defines the research approach used for this study and describes how data is collected and analyzed. The epistemological stance (section 5.3) of this study adopts a primarily constructivist philosophical perspective (section 5.3.1), an inductive logic perspective (section 5.3.2), and an analysis and exploratory theory perspective (section 5.3.3). This qualitative research study uses focus group research (section 5.4.2), narrative research (section 5.4.3) and case study research (section 5.4.4) methods to achieve the research objectives outlined in section 5.2. Section 5.5 describes the research context and includes profiles of the 10 software development organizations studied and details the three phases of research undertaken as part of this study. Section 5.6 and 5.7 discuss the data collection and analysis methods, respectively. Section 5.8 describes how the research design addresses validity and reliability concerns. Next, Chapter 6 presents this study's findings.

Chapter 6 Findings

6.1 Introduction and Chapter Layout

This study included three primary research phases, as described in section 5.5.2. Each phase resulted in a set of findings from a combination of focus groups, sensemaking instruments, and case study analyses. The purpose of this chapter is to present the findings from the study that contribute to addressing the research objectives outlined in section 1.3. Each section in this chapter presents a set of findings related to an element of the conceptual framework, as shown in Figure 6-1. Section 6.2 presents findings that address RO1. These findings, together with the literature review, contribute to understanding the current state of research and practice in impediments to flow. Section 6.3 presents findings that address RO2. These findings validate the impediment management framework. Finally, section 6.4 provides a summary of this chapter.

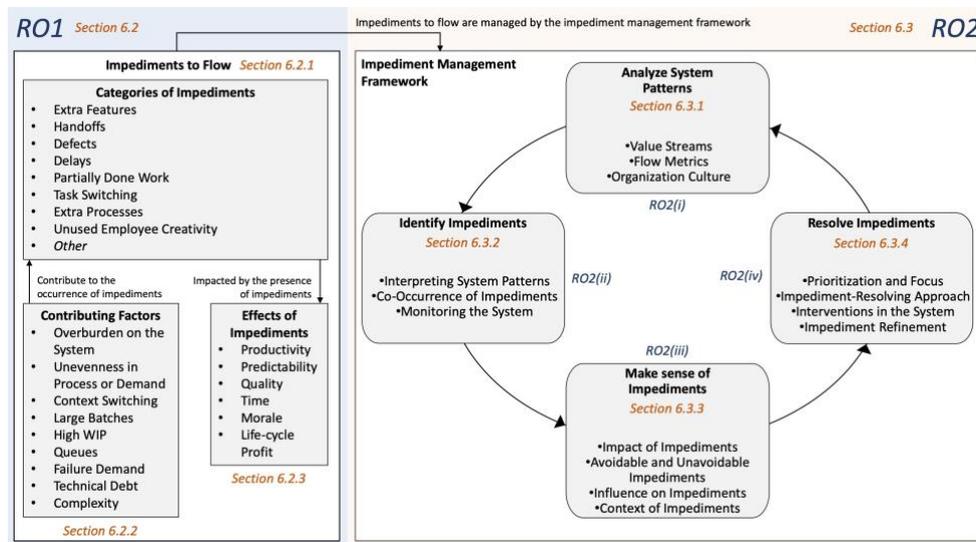


Figure 6-1 Chapter sections relating findings to the conceptual framework

6.2 RO1: Impediments to Flow

Section 2.5 summarizes the literature around impediments to flow. It articulates three primary areas of consideration: impediments to flow,

factors that contribute to impediments, and impact of impediments. This section presents the findings related to each of these areas.

6.2.1 Categories of Impediments

Extra Features

Findings from ORG-A note that “*extra features added may not always be needed*”. Focus Group FG-A2 participants noted that this results in increased planning, development, and maintenance costs for their products. Having *extra features* also contributes to delayed merges to trunk from feature branches, in relation to which ORG-A findings note increased *delays* and forced relearning. ORG-A further observed that handoffs occur as people move on and off teams which results in delays as people come up to speed. Findings show that they added new features on old code branches, even though the new features were not necessary in older versions of the product. This had the effect of “*doubling development and test effort*” for those features. ORG-A further noted that “*features started but not on trunk*” were causing problems, including one instance where 30 person-days of effort was “*wasted*”. In general, unneeded *extra features* appear to be less of a problem in ORG-D than other impediments. ORG-D cited several examples where *extra features* were not specifically the problem, but rather, at a higher level, the organization was investing in non-profitable initiatives and low-priority projects that were not aligned with the business roadmap. Developing extra features was an issue for ORG-G. Findings show a pattern of extra features contributing to experiences being more difficult.

Extra features do not just result from a lack of effort to understand customer needs; it can happen that the organization mistakenly believes that a customer needs a new feature to address a problem. A BU2 micronarrative highlights such a case where not only did the customer not need the new feature, it actually made the situation worse, and resulting in more work (*failure demand*) to resolve the problem introduced by the *extra feature*: “*I helped install an update on a customer site that resolved a problem that the customer was mistakenly told that they were affected by. Not only did the*

customer not need the expensive work done to solve a problem that they didn't have but the solution introduced a performance issue that meant that the [requested change] had to be rolled back.”

Handoffs

Handoffs are demonstrated throughout this study's findings. Approximately 11% of the impediments identified in ORG-A (section 6.3.2) were due to *handoffs*. The findings for ORG-A in section 6.3 showed that they felt that not having stable, long-lived teams leads to *handoffs* as people leave and join the team. There were cases where work handed off from one BA to another did not get prioritized until it was too late for the release. In another example, *handoffs* from development to DBAs were causing 3-4 days of delays. ORG-E decided to redefine the system integration and test activity to be simpler (section 6.3.3), and to reduce what they termed the “*handoff burden*”. ORG-F cited examples of handoffs created when product owners (POs) were not consulting developers and integration teams prior to defining user stories. Product Owners were not working collaboratively with developers, leading to situations where user stories are “*handed over*” to developers without discussions. The integration team is responsible for aggregating the output of multiple development teams. They are not “*in the loop*” on what user stories the PO is creating, which was leading to surprises for them. There is a lot of infrastructure setup required for what ORG-F refer to as “*some of the more complex technical features*”, resulting in *handoffs* between different groups.

In some cases, the findings show that the problem was not the existence of *handoffs*, but the lack of any clear process for managing *handoffs* effectively. As an example of this, ORG-A cited problems resulting from poor *handoffs* between User Acceptance Testing (UAT) and Production. The lack of a clearly defined process resulted in situations where their support team only saw some new features after some customers had already seen them. Another example where lack of a well-defined *handoff* process was causing issues was the *handoff* between the time when new features were added, and time when the customer production system went live.

ORG-A noted that *handoffs* frequently occurred between the development and production support teams.

Defects

ORG-A noted that “*complex defects*” are a problem. These complex *defects* were impacting customer satisfaction. They estimated the cost per defect at 4 person-days (1.5 days for Development, 1 day for QA, 0.5 day for BA, and 1 day for Support). ORG-A findings note that it was their customer’s perception of what constitutes *defects* that counted. As was the case with *handoffs*, discussed above, there are impediments to flow resulting from poor processes around how *defects* are managed. For example, in some cases, 50% of reported *defects* are invalid, leading to wasted time from support teams, BAs, and development teams in investigating these. The result for ORG-A was a lot of work in triaging the issues and realizing that their reporting, triaging, knowledge management, and *defect* management processes needed to be improved. Existing *defects* prevented ORG-A from meeting their desired Definition of Done (DoD). They reported a desire to have zero *defects* as part of their DoD but were dealing with an existing code base that already had *defects*. Of their current defect count, 5 *defects* originated from unplanned change requests, and 30 from issues with existing code. There are cases where fixing *defects* introduces additional *defects* because the impact of the original defect or the fix itself were not fully understood due to a lack of business knowledge. Fixing *defects* without fully understanding the context and moving on to the next *defect* was introducing cases where defects were recreated over time. In BU1 (section 6.3.1), one of the micronarratives indicated that it was “*quite common*” that “*defects impact the deadlines and delivery timelines*”. Referring to products in ORG-D, “*inadequate sprint exploratory testing leads to lot of defects being found at the end thereby impacting release commitment*”. The impact of *defects* can reach farther than the immediate project focus. Another ORG-D micronarrative explains that “[*t*]eams are not doing enough exploratory testing each sprint. This results in lot of defects being found towards the end of the project when we are approaching

a release cycle. As a result we see that teams going into death march in order to meet current release commitments. Sometimes it also has the adverse effect of impacting people working on other releases as they are also pulled in for current release support.” This also shows an example of *context switching* being a factor contributing to impediments. A micronarrative from the BU2 sensemaking survey shows that in ORG-C incoming defects slows the team down: *“We try our best to deliver on our commitments. Defects coming into team sometimes get in the way. The pipeline also slows us down.”* Findings show that *defects* are a significant factor in 29% of experiences related to flow in ORG-G (section 6.3.2).

Delays

Delays were the dominant impediment experienced by ORG-A. Moving people on and off teams results in *delays* as people come up to speed in understanding the system’s architecture. There are multiple instances of a 3- to 4-day delay per request waiting for DBA input. Customer signoff for specifications was taking anywhere from a week to a month. Delays in bringing specifications through earlier stages of the ORG-A value stream meant that *“Dev/QA time is affected”*, and that they were still expected to make previously committed customer release dates. Triage issues were regularly delayed and even not getting through to the engineering teams. Also, having distributed teams, separated by multiple time zones contributed to *delays*. For example, *“junior members blocked if senior members of team do not come online for half the day/evening.”* Getting new team members up to speed contributed to *delays* across ORG-A. Ad-hoc *delays* impacted ORG-A. Investigating and resolving “random test failures” resulted in *delays* of 3-4 days and was compounded by nobody taking responsibility. Environments could be unavailable for 1-4 hours at a time when unanticipated *“problems/issues impact testing”*.

Figure 6-12 shows that *delays* were the second-highest form of impediment impacting BU1, BU2, and BU3. The Cumulative Flow Diagram (CFD) in Figure 6-9 reveals *delays* in the value stream between the “*completed*” state and the “*accepted*” state in BU1. Delays are the biggest impediment facing

ORG-C (section 6.3.2). In BU2, delays occurred in the form of “*the flow of work from [features] thru to the teams for executions*”. Lack of “*clarity and ownership*” was leading to work being started before it was ready, which in turn resulted in “*delays interdependencies frustration*.” This was in part attributed to “*no clear ownership throughout the flow*.”

Managers turning up late for meetings was a contributor to *delays* in BU2, and also contributed to negative effects on *morale*. A BU2 micronarrative illustrates how manager behavior was having a negative impact: “*make managers be on time on meetings. Several attempts to make them understand they should give example on that behavior*.” There were cases in ORG-A where meetings needed to be deferred due to lack of availability of architects, resulting in a 3-4 day *delay* each time due to lack of signoff.

Issues with poor, inadequate, or problematic technology infrastructure, including build, CI, CD, and DevOps pipelines, is one of the major examples of *delays* across all organizations in this study. ORG-A findings note a 5-day *delay* in setting up new development environments. They often needed to do this when new people joined a team, when people switched teams, or when they needed to replicate a development environment to reproduce a reported issue from a customer with a specific version of a product. CI build times were taking more than 90 minutes, causing *delays* in builds and feedback to engineers. ORG-A were running all tests for every check-in. This was contributing to the 90 minutes *delay* per check-in. With 10-20 check-ins per day, this was resulting in *delays* of 15-30 hours and *delays* compounding and rolling into the following day. On top of this, developers had *delays* of more than 40 minutes to run test cases locally before checking in code. In some cases, committing code after a build has started running meant that developers could be waiting for 5 or more hours to confirm if all tests had passed. Manual regression test runs took 40 days. Merging from feature branches to mainline consistently experienced *delays* of 2 to 6 days. ORG-C suffered from *delays* due to their pipeline. The findings show *delays* as a result of the DevOps pipeline in particular being a source of *delay*. Remedial action to improve or fix the pipeline was not

always prioritized, which often compounded the *delays*. Similarly, ORG-G reported that *delays* are a significant impediment (section 6.3). An analysis of BU3 distributed sensemaking data shows that slow development cycle time is a major concern.

Partially Done Work

Partially done work shows up particularly in analyzing patterns in flow metrics and value streams (section 6.3.1). Table 6.4 shows that, when analyzing value streams, ORG-A identified partially done work as the third most prevalent impediment. Findings from ORG-G show a dominant pattern of there being too much *partially done work*, which negatively impacts flow. A micronarrative from BU1 refers to “*incomplete feature design*” and “*extra unplanned work*”.

However, in general in these findings there appear to be blind spots in relation to viewing *partially done work* as an impediment. The Cumulative Flow Diagrams in the findings in section 6.3.1 demonstrate that *partially done work* is a factor in these organizations. Figure 6-27 highlights that organizations in this study were less likely to identify with *partially done work* as an impediment when other impediments were also present. Despite this, Figure 6-27 shows *partially done work* is the least identified type of impediment by organizations in this study in general, along with *extra features*.

Task Switching

Table 6.4 shows that, when analyzing value streams, ORG-A identified task switching as the third most prevalent impediment. Slow build times contribute to *task switching* by interrupting people’s focus; the people switch to other tasks because the builds are slow. ORG-A identified being “*reactive not proactive*” as a factor that is contributing to *task switching*. Engineers switch on priority issues as there are not enough people to cover various customer projects. Findings show that ORG-A has different processes for user acceptance testing and production environments that result in *task switching*. ORG-A often needs up to 5 recreations per defect

due to lack of internal tools in production, with each recreation resulting in task switching. An ORG-D micronarrative referred to *task switching* for an entire team: “*suddenly changing the team to different feature in middle of current vision is bit shaky.*” Interruptions are seen as a common contributor to *task switching* in these findings. For example, in ORG-D there was “*too much of disturbance from senior management on the user stories worked on by team. in last sprint many people in senior management came to team work area and started asking the team to get the data they required.*” A micronarrative from ORG-C relates some contributing factors to *task switching* and some of its impacts in that organization: “*We use agile tools but our usual tasks and work comes directly from outside customers requests other project teams requests or a result from some meeting which engineers are not part of. This results in team members working long hours sometimes weekends and nights to deliver on both non-sprint work and also sprint committed work. Our team PO and SM try to shield the scrum team from outside sprint work but are unsuccessful for whatever reason. Our velocity and efficiency is affected which results in the team providing low output value. Multitasking is not always the best in agile.*” In this case, the teams are attempting to use a controlled process to manage the inflow of work, but this is being bypassed. This switch of focus results in interruptions, long work hours (including weekends), and there is an impact on *morale* and *productivity*.

Extra Processes

In ORG-A (section 6.3.2), the management of defects resulted in avoidable process overhead. ORG-A managers estimate that one person-day per release is “lost” by release management through managing defects in multiple tools, including Jira, spreadsheets, and home-grown tools. Considering there are 120 releases per year, this represents a significant overhead in avoidable extra processes due to defects. A recurring theme was that the tools ORG-A used for managing work did not reflect their actual workflow, and this resulted in a lot of extra processes. In some cases, as in ORG-A, they also observed that this resulted in additional human error.

They further noted that they did not have the ability to change their processes in Jira. Findings show duplication of processes and tools in ORG-A, e.g., they were using three different tools to track and log time; one for internally found defects, one for customer-found defects, and a third tool for time sheets. Communication needed to be duplicated in some cases, e.g., notes were added to issues in Jira, but not always picked up by support teams, so they also sent emails. Information duplication was common, with plans and other information recorded in Wikis (which had problems), Jira, Excel, PowerPoint, and other sources.

Analysis of micronarratives related to ORG-C show that there were unhelpful *extra processes* impeding teams getting their user stories *Done*. Examples include the inefficient execution of code review processes, often involving multiple sites: “*even if we break the stories super small it will often take a full 2 weeks just to get that tiny change through the complex process*”. Extra processes are a factor in 24% of the micronarrative experiences recorded in ORG-G.

Extra processes are generally perceived in this study’s findings as providing no worthwhile benefit, and often have a net-negative result in the organizations. The purpose of the *extra processes* is often not clear to the people who have to deal with these *extra processes*. A micronarrative from ORG-D relates how they felt the organization was creating too much process with little beneficial outcome. Their management had decided recently to switch from Scrum to Kanban, without providing any explanation: “*Recently, we have moved from scrum to kanban. No explanation provided. The experience was really frustrating and tiring. Daily spending more time in discussion and meeting rather than doing actual work.*” Lack of a clear process was causing significant process overhead. Other micronarratives showed that they have too many tracking and monitoring calls (online meetings) and processes, noting that “*each manager wants his own personal process*”. A different micronarrative also from ORG-D notes “*too many managers trying to steer a project in crisis using various means of tracking. Very difficult for first level leaders to*

manage the expectations of so many managers.” One of the micronarratives from BU1 (section 6.3.2) refers to “going through the motions” in large meetings as an impediment: “Weekly programme meeting with many key people is nothing but a status meeting. Often going into long winded detail that is better discussed outside of this forum. People ca [sic] be heard to often provide status for the sake of it purely going through the motions.”

Unused Employee Creativity

Unused employee creativity was identified as a significant impediment across BU1, BU2, and BU3, as shown in section 6.3.2. *Unused employee creativity, as identified in these findings, covers several themes, including people, processes, tools, and technical debt. For example, there are “too many defects and Technical Debt to realize creative potential” of engineers in ORG-A. ORG-A had asked for product ideas from employees, but these were not used effectively. Either no feedback was given on the product ideas submitted by employees, or there was no funding provided to implement these ideas. This also had an impact on employee morale. The ORG-A findings show there was very little downtime allowed for research and process improvement, and that “hidden skills” of employees were not utilized.*

Micromanagement is shown to have a negative effect on employee creativity. This impedes flow in several ways. A BU1 micronarrative (section 6.3.2) shows an example: *“[The engineering manager] is directly assigning defects to team members and tracking the progress in order to meet the release timelines. But this is not the Agile way of working as we end up micro-managing people and focus will shift back from team to individual. Instead we should empower teams and let them own a particular release and not disturb their focus with other support activities.”* Too many processes in ORG-D had led to *“the restriction on thinking independently or trying new ideas with too much focus on delivery.”* A micronarrative from ORG-D noted that they are *“not using the full potential of top reseources [sic] to develop features”.*

Other

The findings include examples of impediments that organizations in the study did not categorize into one of the preceding eight categories. For example, in BU2 a micronarrative identified “*unclear requirements / expectations*” as an impediment. ORG-A classified some impediments as *other*, including communication tools, deployment misconfiguration, release management errors, and delays merging to trunk. They noted that these were causing other impediments, including *delays* and *extra features*. ORG-F defined eight categories for the impediments that they identified. These included:

- ***Agile practices***. The lack of diversity in agile practices across ORG-F was seen as an impediment to productivity. The organization had a lack of rigor and discipline in adhering to their agreed development process. There were too many meetings that took too long and provided no feedback. Their teams had lost confidence in the value of retrospectives. The definitions of *Ready* and *Done* were not being met, and teams were starting new work before finishing current work (adding to *WIP*). POs were not working collaboratively with developers, leading to increasing numbers of *handovers*. The *complexity* of features led to *complexity* in infrastructure requirements and contributed to an increase in *technical debt*. Backlogs were not being managed consistently. There were lots of *extra processes* added in an effort to compensate for these other impediments.
- ***Dependencies***. Teams did not provide feedback to other teams that had dependencies on them. Dependency definitions were of poor *quality*. As well as dependencies among teams, the external dependencies were “*too much!*” Other forms of dependencies were related to infrastructure and development environments; ORG-F did not have full ownership of their development environment, including the CI system, testing environment, and other infrastructure.
- ***Managing impediments***. ORG-F had significant experience with the concept of identifying and managing impediments. However, at the time

of data collection for this study, there were several impediments related to how they were managing impediments. These included impediments that were getting identified but not resolved. They had developed an impediment removal process, but it was not efficient, and impediments were either not followed-up on or were processed far too slowly.

- **Multi-site development.** ORG-F were working with other organizations in other developed sites around the world. They identified three specific impediments related to this multi-site development model. First, it was difficult to synchronize among organization groups and among people within groups. Second, there were several cases where the PO and the team were not in the same site. Third, cross-team coordination and collaboration on large programs was challenging.
- **People issues.** The scope of work changed a lot, and this happened often. This relates to *context switching* and *unevenness*. Several other of the impediments categorized under “people issues” relate to agile development. ORG-F was having difficulty finding the balance between an “*empowered team*” and a “*team can do what it wants*”. ORG-F was split into two camps; “*pro-agile*” and “*anti-agile*”.
- **Roles.** ORG-F had identified various cases where dealing with role overlap and blurred boundaries in “agile roles” was an impediment.
- **Unclear value.** People in ORG-F felt disconnected from the value and purpose of the work. Some people did not see clear benefits from the work done in ORG-F and did not “*feel the value*” the work was providing to customers. They felt the organization had become too metrics-focused, and not value-focused.
- **Management.** The organization had adopted an organization model where groups of teams worked on related areas of the product. However, people did not feel any real alignment to the teams, and this lack of alignment was an impediment. If someone needed help, it was provided as a “side job” and not seen as a core responsibility that teams were expected to help each other. Conflicting messages from management and from within the program were also identified as impediments.

6.2.2 Contributing Factors to Impediments

The literature review in section 2.5.2 identified eight common contributing factors to impediments. These are *too much WIP*, *failure demand*, *large batches*, *context switching*, *unevenness in process or demand (mura)*, *overburden on the system (muri)*, *failure demand*, *complexity*, and *technical debt*. This section presents findings related to each of these.

Overburden on the System

Overburden in the form of *pressure* was a recurring theme across the findings. ORG-D noted a “*pressure on us not to say 'no'*”. ORG-B referred to “*pressure to deliver features*”. An analysis of BU3 micronarratives shows that architects were overloaded and feeling a lot of pressure. The dominant sentiment behind such experiences is negative, and most experiences of this type are quite common. A common theme in BU3 is that they “*end up taking and completing more than what is planned.*” Commitments were being made to customers without confirmation from the teams developing the product. This was having a negative impact on the organization: “*the team usually get's overloaded almost every sprint as most of the commits are given to the customers without confirming with team developing the functional[ity].*” In ORG-C, engineers were working additional hours “*outside of the sprint estimate in order to meet commitments.*” An ORG-C micronarrative noted that their tech leads are “*overloaded*”. Another micronarrative referred to the pressure that results when work turns out to take longer than initially thought: “*There's a lot of pressure to deliver within a 2 week time window even if during the assignment scope of requirements grows from uncovering unknowns that require attention during the time. There's often a sense of justifying doing the job the right way when the scope grows. Sometimes tasks will take longer than initially thought and there's a sense that management overlooks how much real work has been done in little time and focuses on why the assignment isn't completely checked off the todo list.*” A BU1 micronarrative refers to layoffs at the company that took away an entire solution test team, resulting in *overburden* as the remaining teams were expected to absorb the work: “*Now*

we have no end to end solutions testing. There doesn't seem to be any logic behind these decisions. Nobody assigned to take over their role. We are just to absorb their work.”

Overburden comes in the form of adding additional unscheduled work to the organization's workload, without removing other work. A BU3 micronarrative illustrates this: *“Customer deadlines are often missed because teams are not set up to accept unscheduled feature work, and maintain current workload. It is unclear if the process to schedule and scope incoming customer requests isn't followed, or does not exist.”* An ORG-C micronarrative expressed concern about overburden due to work assigned outside the normal process, and the impact this was having: *“I'm concerned that engineering managers are making the engineers do extra work on the side either out of necessity (e.g. lab work) or out of personal development goals. This can be a distraction from the committed work for the scrum team. If they factor in this extra work when they do planning then I guess it's fine. But they should be letting the SM at the very least know about this reduction in their capacity so that the planning is done right. I don't think everyone does that.”* Another micronarrative shows that even when teams try to set aside capacity for unplanned work, it hasn't worked out: *“Determine capacity of team. Set aside capacity for unexpected/unplanned plan to capacity. Get asked to take more but can't push anything out. Try to push back to take something out but usually end up having to do it anyway.”* A BU2 micronarrative further illustrates this point: *“Recently we were given a goal to port [Product] to [Public Cloud Provider] by the end of [Release] which by itself is a difficult but feasible goal. Our leadership then started adding arbitrary requirement these requirements continue to come in pulling teams away from the original goal and the teams still working on the port have to wait on deliveries from these new requirements. All the while the dev team are still delivering code that only works in the current system.”* In another example, a micronarrative titled “*watergile*” related how overcommitment contributes to overburden, and removes the ability of the organization to adapt to unforeseen circumstances: *“we often over commit at the beginning of the planning*

section but there are always some unforeseen reasons that prevents us to make all the deliverables and commitments. Like customer defects CI/CD pipeline delay or [system] issues”.

Poor or ineffective approaches to planning contribute to *overburden*. For example, not considering the capacity of the organization when planning sets the teams up to miss expectations and takes away the organization’s ability to react to changes: *“if we’re already planning to 250%, we just have no chance of being able to react”*. This point is highlighted again by another micronarrative: *“Management commits to schedule without consideration on amount of actual work effort.”* ORG-A noted from a planning perspective, treating people new to the team as being 100% available immediately contributes to *overburden*. They were not allowing for ramp-up time for onboarding, training, and other activities. They were not accounting for the time it takes for existing team members to provide training, mentoring, and onboarding. In another case, a BU2 organization released two weeks ahead of plan, but employee *morale* and product *quality* suffered from this. An analysis of micronarratives shows evidence of burnout, lack of proper planning, failure to identify obstacles and dependencies, testing not planned appropriately, *“non-functional testing”* not adequately done, known defects left open for too long, and reliance on “experts” with specialized skills. Despite releasing early, they note these factors contributed to *overburden* and made the work much more difficult.

Over-commitment and understaffing, combined with not considering everyone involved in the value stream as part of the team, leads to *overburden*. For example, technical documentation in ORG-C *“Our technical documentation commitments are all over the map. Were' not often included in Dev process until the end of the sprint. How can we be considered a true part of Dev and be included in Definition of Done. We are overcommitted and underresourced.”*

Volatility in priority changes contributes to *overburden*. An ORG-C director noted that their organization was *“spending more energy understanding volatility of priority changes”* which was a result of *overburden*. This

director wondered whether this volatility is something that can be avoided and changed, or whether it is the nature of their business, and they need better ways to manage through the volatility.

Overburden can lead to organizations taking shortcuts by not fully completing work to meet Definition of Done or neglecting it entirely. ORG-B identified several challenges in relation to *overburden* that they attribute to problems with their lack of a consistent or disciplined approach to Definition of Done (DoD). Sprints become overloaded in part because their estimates do not include “*DoD constraints*”. They noted they do not take the time to implement user stories that meet DoD. In other cases, teams in the same product development organization believe they cannot have a common DoD because of their works is very specific to them. Pressure to make a demo contributed to *overburden*. The content of the demo was not always aligned to their development plan, and they would end up piecing together incomplete features in addition to planned work. They refer to this as operating in “*demo mode*” rather than development mode. They noted there is confusion in ORG-B between demo acceptance criteria and Definition of Done. The demos are not production-quality, and so add to *overburden*, *failure demand*, *technical debt*, and *WIP*. Build environment instability was a factor here for ORG-B and is a contributor to *overburden* in ORG-C and BU3. As a result of these reasons ORG-B chose to focus on developing a common Definition of Done across all teams in an effort to mitigate the challenges that lead to *overburden* (see section 6.3.4).

The effects of *overburden* were visible beyond the resulting impediments. For example, in BU1 people who were expected to attend customer demos were not doing so due to “*over-work for customer projects*”.

Unevenness in Process or Demand

Ohno (1988) noted that unevenness in process and unevenness in demand contribute to impediments. The findings from this study show evidence of both. First, the findings show evidence of *unevenness in process*. ORG-C had a process whereby their planning cadence was every three months. They

had synchronized 2-week sprints within that 3-month cycle, and found that “*there are too many unforeseen*” demands on their teams after the planning cycles were completed. This forced their teams to modify their sprint-level plans after the sprint had begun. Despite adapting to unforeseen requests, this unevenness in process was, among other things, causing dissatisfaction and having an impact on *morale* (section 6.2.3), with one micronarrative noting “*it always feels that the team is getting dinged for not meeting the original goals.*” A similar situation arose in ORG-D, where plans changed in the middle of their release cycle. “*The change in the plan in the middle of the release was [a] little painful since the entire strategy was redefined and release was not successful.*” In that case, the changes were due to a fundamental change in strategy. Another ORG-D micronarrative estimated that teams in that organization are able to deliver their sprint commitments “*on a 50-50 basis*”. They identified three factors that contribute to this. First, the teams don’t always understand the business use cases behind the requirements. Second, requirements often change during a sprint. Third, the teams “*don’t get sufficient time*” to analyze stories for upcoming sprints because they are too busy working on the current sprint’s user stories. Unevenness in process also occurs when there is no clear process, or the process is not well understood. In one example, ORG-D had dependencies on a third-party company to deliver device drivers for specialized hardware devices: “*Team is unable to deliver as per their release commitments because of delays in driver deliveries and dependencies on drivers. There is no clear process for accepting driver releases. So team spends lot of time in validating each driver issue fix all of which causes delays and finally team is unable to deliver as per their release commitment.*” In this case, they did not have a clear process for managing the relationship with the driver vendor. This contributed to *delays* and *defects*. A BU2 micronarrative (titled “*planned vs un-planned work*”) refers to significant occurrences of unplanned work contributing to delays.

Second, the findings show evidence of *unevenness in demand*. *Unevenness in demand* occurs in the form of unforeseen work arising after starting work on a sprint or release. Dependencies show up as a significant contributor to

unevenness. In the case of ORG-C, a lack of clear understanding of the state of dependencies was leading to unevenness in demand on ORG-C. For example, as reported in a micronarrative “*Our team generally is quite sensitive to making commitments that we can delivery with somewhat high certainty. However there was a situations this past month where a commitment to deliver a feature required a dependency which at first appeared and mentioned was almost complete but after a more detailed review this dependency was not even 20% complete. After negotiating the minimum required for the dependency we proceeded to adjust our plans and commitments in face of this new reality. It was very unfortunate that the maturity of the dependency was not made clear up front.*” Another ORG-C micronarrative points towards unevenness in demand due to tasks changing mid-sprint, architecture changes that require discussion with architects, and dependencies on other teams: “*The agile method helps the team synced and kept in-line what everyone is doing. Positive experience in delivering the results. However sometimes it can be hard to finish a sprint if task requirement is changed in middle of the sprint. To solve the problem we would need more detailed and direct instructions from the Architect team. Also a task is often delayed because of its dependency on other teams in different regions and the other team may not often be patient to help us understanding their technology. [Web conference software] is great but nothing compare to physical presence in team work.*” Urgent interruptions are another contributor to *unevenness* in demand. For example, “*high priority adhoc issue coming in way of meeting sprint commitments*”. ORG-A report at least three major unplanned change requests per release that disrupt their flow. An ORG-D micronarrative expressed that refactoring, *extra processes*, and change efforts were all contributing to extra demand on the system and expressed a desire to “*stop doing just for the sake of doing.*”

Context Switching

There is evidence in the findings of *context switching* happening in different scopes. The three scopes are company, product development organization, and team or individual. These correspond to the three levels of systems

discussed in section 4.2.4. First, the context of the business itself can switch, and this has a disruptive effect on the product development organization. For example, an ORG-C micronarrative shows evidence of *context switching* related to frequent shifts in business goals and priorities, to the point where their business goal focus “*is changing every three months.*”

Second, *context switching* can happen at the level of the product development organization. BU2 micronarrative “Demo days” were a common example of *context switching*. BU2 gets multiple requests for demos from customer executives, often with little notice. They felt a pressure to comply to these requests “*due to the execs involved.*” A BU3 micronarrative relates that engineering teams are “*continually interrupted*” during sprints to “*work on emergency quality fixes for work that has been released.*” In addition to this, the sales team “*will prioritize new customer features, and team is again interrupted from current sprint only to come in past the deadline with an untested product.*” This shows examples of *context switching* contributing to *delays* and *defects* and impacts on *quality* and *morale*. BU3 exhibits a similar pattern, where poor *quality* and *defects* result in *failure demand*. In that case, customers were impacted by “*unscheduled quality related fire-drills. Engineering teams report that they are continually taken away from their development work in order to focus on quality related fire drills, resulting halting previously scheduled feature work.*” These fire drills contribute to *context switching*, which in turn contributes to *delays* and lower *throughput*. *Context switching* is a frequent problem in ORG-F, where the scope of work “*changes a lot and often*”. In ORG-C, *context switching* often contributes to over commitment “*due to many distractions*”.

Third, *context switching* happens for individuals. For example, in BU1, a micronarrative relates how they have evolved their process to work with Scrum and feature teams, where they are “*focused on delivering new vertical slices of functionality.*” However, team members with specialist knowledge are “*constantly interrupted due to their specialist knowledge*”. These interruptions come from multiple sources, including from architects

asking questions, from integration teams looking for help to investigate issues, from other teams to help with code reviews, and from program managers to help with urgent customer issues. A BU2 micronarrative relates that the person was asked to take on additional responsibilities for a specific customer, in addition to their existing responsibilities. While *overburden* was not reported in this case, it did result in *context switching* because of work in two unrelated areas. Another BU2 micronarrative (titled “*Surprise - you're late!*”) relates how lack of coordination and alignment can result in *overburden in demand* and *context switching*. The product manager had expected the architecture for a feature to be complete, yet the feature had been given a low priority. The feature got moved to a higher priority, and an architect was asked to “*begin working on it ASAP*”, switching focus from the current context to this new feature.

Large Batches

A BU3 micronarrative illustrates the difficulty organizations in this study face in determining an optimal batch size: “*user stories cannot be very small because it is not deployable. If the Platform User story has to be fully functional, it takes usually more than the sprint*”. This highlights a challenge these organizations face in developing large systems and platforms; it can be a challenge to structure user stories so that it does not take longer than a single sprint to deliver something deployable and functional.

Improvement efforts in one instance in ORG-C focused on “*splitting tasks into smaller tasks*” in order to reduce the batch size of work in sprints. ORG-C also faced challenges where the size of the batches turned out to be larger than they thought, with “*planned work items much larger than anticipated*”.

An engineering manager in ORG-D noted that engineers are unhappy due to the amount of stress created by *large batches*. Even though they had adopted agile processes, they still “*had a way to go to be agile*”. Delivering in small batches was the number one improvement suggested by this

manager. They cited working in large batches as contributing to *extra features* and *extra processes*. The contributing factors can amplify each other. For example, this engineering manager noted that *large batches* also contribute to *failure demand*.

High WIP

This study found a range of perspectives on WIP across the organizations in this study. These perceptions ranged from expressing difficulty in understanding the impact of WIP, to leveraging WIP management to improve cycle time. Noting that attempts to manage WIP in their team had proved difficult so far, one engineering manager related “*not everyone gets that too much WIP is a bad thing - some education is required*”. An engineering manager from ORG-C reported about WIP: “*I do think WIP is important. Not necessarily from the numbers perspective. It is one of the indicators of the health of our methodology. Many things in the way we work impacts WIP. WIP is an indicator. Low WIP indicates good [feature definition], good user stories, good dynamic in the teams. So, I do think it’s important.*” Another engineering manager noted the connection between WIP and cycle time by observing that “*the more we allow WIP to increase, then by definition, the cycle time will be longer. If cycle time is the thing we want to reduce, then WIP is the lever we need to use.*” An engineering director from BU2 expressed that they felt it was important to distinguish between WIP at different levels: “*It’s also important to decide what WIP we’re focusing on. [Feature] WIP, [Sub-feature] WIP, user story WIP. Each of these require a different approach and different way of dealing with it.*” In their organization, they had a structure of defining top-level features, which are composed of sub-features, which in turn are composed of user stories. Each level has a different granularity. User stories were typically completed in days, and within a sprint. Sub-features might take more than a single sprint to complete. Features might take more than a single release to complete. They expressed that it was important to consider WIP at each of these levels.

Queues

The flow metrics discussed in section 6.3.1 below show evidence of *queues* being a potentially significant contributing factor to impediments. Cumulative Flow Diagrams show evidence that work accumulates, and increasingly large numbers of workflow items spend significant time in states before moving to the next state. Despite this, none of the organizations in the study acknowledged the role of *queues* as contributing to *delays* or other impediments.

Failure Demand

As shown in section 6.3.2, ORG-A estimated that a lack of knowledge in support tools was costing the company in the region of \$2m per year. Lack of knowledge, and general lack of engineering discipline, was seen as a significant contributor to *failure demand*. Lack of business knowledge, absence of acceptance tests, and little or no automation are all contributing factors. A qualitative analysis of flow metrics for ORG-E, discussed in section 6.3.1, shows *failure demand* as a high percentage of overall *throughput*. ORG-A findings note lack of business knowledge was a factor in introducing *defects*. Even when fixing *defects*, lack of business knowledge leads to incorrect or poor resolutions, resulting in additional *failure demand*. Lack of specific knowledge about customers and their environments contributed to *failure demand* in ORG-A. Related to this, lack of adequate support processes resulted in 53% of reported defects coming through triage and support, to developers, were “*invalid*”, i.e., it was not a *defect*. They expressed that these should have been caught earlier in the support process but instead resulted in high *failure demand* impacting the development team. *Failure demand* due to existing *defects* prevented ORG-A from meeting their desired Definition of Done (DoD).

Failure demand can come in the form of responding to the demand for critical fixes to the product in production. An example from ORG-D highlights that under pressure to provide a fix, the response to *failure demand* can result in the risk of further *failure demand*, e.g., “*delivering fix*

for critical issues from field without complete technical analysis.” Another example notes that, in the rush to provide a fix in response to *failure demand*, ORG-D took short cuts with sanity testing: “*not much time frame provided for sanity testing for [recent] builds*”. *Failure demand* was a significant contributor to impediments, and part of the experience of people in the organization in ORG-G.

An instance of *failure demand* in ORG-D resulted from a release build that crashed in an unexpected runtime scenario. Relating this experience, the micronarrative also notes that “[i]t took lot of effort but it was worth it.” This experience was noted as a one-time event. It was also cited as an overall positive experience, because of how the team came together to solve the problem. *Failure demand* can originate within the organization. An example from BU2 relates an experience of poor engineering discipline resulting in *failure demand*: “*someone checked in something and all our CI tests are now failing*”.

Technical Debt

ORG-A noted that there are “*too many defects and Technical Debt*” inhibiting the “*creative potential*” of people in the organization. This resulted in *unused employee creativity* and also impacted *morale, quality* and other factors. ORG-A also identified *technical debt* as a factor in customer dissatisfaction. An engineering manager in ORG-C mentioned that their team expressed concern about how *technical debt* will be tracked after an important milestone concerning the migration of their product to a public cloud vendor. *Technical debt* was also found to be a significant contributor to impediments in ORG-G, and part of the experience of people in the organization. *Technical debt* occurs and accumulates in the organizations in this study when there is an over-focus on immediate needs and not enough focus on the bigger picture. A micronarrative from ORG-C highlights an example: “*Sometimes the big picture is ignored to address the needs of today which causes technical debt to accumulate. This is frustrating given we can never catch up because of just a few mis-steps.*”

Complexity

Complexity was noted as a contributing factor across the organizations in this study. The findings illustrate many sources of *complexity*:

- **Business complexity.** ORG-G expressed a need for approaches for managing *complexity* in a dynamic, highly regulated environment. ORG-G further noted the “*high business complexity built into system*”.
- **Product complexity.** An ORG-G micronarrative summarizes the challenge succinctly: “*The complexity of the product is what kills us*”. Another ORG-G micronarrative concurs that they have to manage a “[v]ery complex product” and a “[v]ery complex domain.” ORG-B noted that the “*complexity of what we’re working on ... has an impact on getting 'Agile' working*” in their organization. A BU2 micronarrative related the importance of explaining and discussing new and complex concepts introduced in the product before their release planning event, noting it takes time to research and absorb such complex and unfamiliar topics. The first part of the micronarrative relates how, in previous release planning events, teams “*faced difficulties to perform planning and story points assessment because of new features and concepts not familiar with. Induce partially by mis-alignment between Product and Program teams in the way to explain goal and concepts.*” The second part of that same micronarrative discussed steps they had subsequently taken to address the problem, emphasizing preparation, presence of key people in the planning event, and good interactions: “*Finally more time has been used during release planning event to clarify this and remove mis-understanding. It's the benefit of having all stakeholders during release planning event - thanks to good interactions.*”
- **Code complexity.** ORG-G noted that the *complexity* of their code is increasing over time. In addition, their engineering process was not designed to support having parallel teams in multiple countries working in the same codebase at the same time. This was resulting in “*lots of problems during the sprint. And sometimes due to same reason stories were not delivered on time.*”

- **Architecture complexity.** There is evidence that architecture complexity contributes to delays and quality issues. A BU1 micronarrative notes *“it takes forever to make small changes to the software due to complex architecture and slow/poor test environments.”* In ORG-G, they noted that product complexity and architecture complexity were increasing together. ORG-A attributed a lack of architecture governance to an increase in code complexity and defects, as well as a propagation of architecture anti-patterns. There is also evidence in the findings that organizations are proactively addressing architecture complexity, e.g., in BU1: *“What impresses me is how much architects care about what they do and how much effort they are putting into specifying systems and design guidelines that can scale and integrate with other systems under some very challenging constraints.”*
- **Process complexity.** An ORG-C micronarrative related that in an effort to reduce *batch size*, and limit *WIP*, they were attempting to split user stories into small enough units so that they could be delivered in a few days. However, this highlighted the *complexity* of their product development process: *“Even if we break the stories super small it will often take a full 2 weeks just to get that tiny change through the complex process.”*
- **Development environment complexity.** ORG-C cited development environment complexity as an issue, particularly *“stuff that’s not easy to predict; dependencies on other teams; not being able to get working [Cloud operating system instances]; this stuff is not very predictable.”*
- **Complexity of defects.** The *complexity of defects* is a factor contributing to customer dissatisfaction in ORG-A.
- **Policy complexity.** A BU2 micronarrative provides an example of how company policies can contribute to impediments. In that example, the *“cost and complexity of [C2] policies”* were making it difficult to plan for on-site activities with customers.
- **Decision making complexity.** A BU3 micronarrative noted a problem with the *complexity* of decision making in their organization.

6.2.3 Effects of Impediments

The literature review in section 2.5.3 discussed six common effects of impediments in organizations. These are *productivity*, *predictability*, *quality*, *time*, *morale*, and *life-cycle profit*. This section presents findings related to each of them.

Productivity

Impacts to productivity take several forms in this study. For example, findings from ORG-A show that build times are too slow, feedback cycles take too long, and developers encounter frequent and lengthy delays, all of which impacts productivity. Lack of automation in ORG-A also impacts *productivity*. The findings from ORG-F cite impact on *productivity* as one of the dominant patterns in their organization. Findings show a pattern of different types of impediments impacting productivity. Overall, impacts to *productivity* in ORG-F are dominated by *agile practices* impediments, *people* factors impediments, with some influence from *multi-site* impediments. Sometimes the same set of impediments in ORG-F that impacts *productivity* also impact on *quality* and *architecture*. Findings suggest that *external dependencies* and *multi-site* issues have a strong negative impact on both *architecture* and *productivity*. A product team from BU3 noted impacts to *productivity* and *morale* “*Not delivering to customers, constant churning, employee burnout*”. Frequently-changing business priorities in ORG-D were found to be a significant contributor to productivity challenges.

A BU1 micronarrative relates a positive experience of addressing technology-related impediments and the resulting improvement in productivity: “*We recently had a challenge of productivity on our scrum team. We have new members that were struggling with the new technologies. My scrum master and I discussed how we can rearrange the team to be more productive. We delivered this idea to the team and the team positively embraced the change.*” Managers were sensitive to anything that might impact on productivity, including, ironically, time needed for

improvement efforts designed to improve productivity. An ORG-C engineering manager noted that it was difficult to focus on improvement efforts because they had “*too much impact on the product velocity of my team*”.

Predictability

As discussed in Chapter 3, flow metrics can show the presence of challenges with *organization predictability* in the value stream. Section 6.3.1 illustrates how cycle time metrics reveal predictability issues in BU1, BU2, and BU3. In general, cycle time variations made it difficult to predict response times for individual work items.

ORG-A, while deciding which impediments to address, expressed that predictability was an issue for them (section 6.3.4). Section 6.2.2 recounted an example of *unevenness in process or demand* that was resulting in a pattern of ORG-D delivering 50% of their committed sprint goals. This is also a measure of that organization’s *predictability*. An engineering manager from ORG-C provided another perspective on predictability, i.e., things that make it difficult for teams to predict. These included dependencies on other teams, and unreliable testing and deployment environments.

Some organizations observed the relationship between *predictability* and *high WIP*. An engineering manager from ORG-C wondered about the relationship between predictability and WIP, asking “*is WIP the lever to get us more predictability?*” BU3 had a similar realization. In a status review they noted a problem with “*too much Work In Progress*”. They noted the impact of too much *WIP* included that “*we are not delivering working features to customers. Quality and predictability are suffering.*”

Quality

ORG-A noted manual regression testing was resulting in poor *quality*. Also, slow build times were causing significant *delays* and *task switching*, which in turn resulted in delayed testing and compressed test cycles. ORG-D noted

bespoke and custom development (equated to *extra features*, *extra processes*, and *failure demand*) was having a negative impact on product *quality*. ORG-F reported that there was some impact from impediments on a combination of *customer quality* and *productivity*. The impact on *quality* is not dominated by any single factor. The findings suggest *quality* is impacted by a combination of each of the eight types of impediment they identified (see *Other* impediments in section 6.2.1 above).

There is evidence that adding unplanned work mid-cycle impacts *quality*. This is seen through a BU3 micronarrative: “*though we plan for the tasks but we end up completing tasks on a very short note (which leads to not so excellent quality of code) due to adhoc tasks in between sprint.*”. Another micronarrative from BU3 concurs: “*Customers are impacted by unscheduled quality related fire-drills. Engineering teams report that they are continually taken away from their development work in order to focus on quality related fire-drills, resulting halting previously scheduled feature work.*”

Time

Impact on time showed up across all organizations in this study. *Delays* are the most commonly cited impediment across organizations in this study. *Delays*, by definition (see section 2.5.1), have an impact on *time*. *Time* also showed up in other ways. For example, in ORG-G: “*Time: we may not have enough time to work on this*” reflected a concern that they may not have time to work on resolving the impediments they identified, in part because of *overburden on the system*.

Morale

ORG-F findings show an awareness of the impact of impediments on *morale*. Specifically, people-related impediments, more than time or financial cost, show a dominant pattern of impacting *morale*.

A BU1 micronarrative shows the impact on *morale* resulting from a scenario where BU1 engineering teams did not feel their management

trusted them: “[Customer X] have been having stability problems in the field with [Product X]. These were largely due to the fact they ignored stability. When [Customer X] escalated the issue the assumption made by [C2] upper management was that the problem was the [C2] team. It did not cross the mind of upper management that [C2] team might be doing a good job.” This micronarrative had a strongly negative sentiment and is reflective of experiences that “*happen sometimes*” in BU1. This particular experience involved engineers, managers, and directors.

The findings show evidence that in general, the morale in organizations suffers if their work is not appreciated or their efforts are wasted. For one product organization in BU2, the volatility of the business environment was proving disruptive. A micronarrative illustrates the impact on *morale* due to fears that the product of their current work would not get released: “*We are going through a phase where the 'commitment' changes every three months with uncertainty whether what we are targeting will be demonetized in three months. [C2] is all about agile however this continuous change in strategy also brings in doubts in the minds of technical teams that what they are committing will not see the light of the day.*” An ORG-D micronarrative provides an example where this fear was realized: “*Team worked on sprint goals and delivered all the stories from the initial sprints. They were proactive and had open discussion with [X] team bringing out dependencies early in the Planning ...But due to Customer business priorities changes the entire project was descoped and all the team efforts were wasted. Entire team was disappointed and raise lot of doubts and concerns in the future work.*”

Impediments combined with issues occurring in the wider organization context impact on *morale*. Layoffs, including the threat of layoffs, had a significant impact on morale in several organizations. For example, in ORG-D “*Lay off is happening every year. Its affecting work. Not able to concentrate properly.*” Handoffs were also a factor in this particular experience. An micronarrative from an engineering manager in ORG-D relates that engineers are frustrated at times because they do not get to work

on cutting edge technologies, combined with a lack of budget: “*This impacts the morale and impacts on delivering on commitments when there has not been a very exciting hike budgets over the recent times. As an engineering manager it makes the task difficult for keeping the team motivated.*” *Delays and failure demand* were impediments noted in this experience.

Morale can also be positively impacted, despite the presence of impediments. An ORG-C micronarrative, in describing a challenge they resolved that included impediments, also noted: “*My team makes working for [C2] a lovely experience.*” Under the heading of “*Organization Culture*” and the subheading “*Impediments and Morale*”, section 6.3.1 presents additional findings related to the effects of impediments on *morale* in the organizations in this study.

Life-Cycle Profit

ORG-A noted a “*huge resource overhead in manual regression testing*”. Manual regression testing was not just expensive. It was slow (contributing to *delays*), and resulted in poor *quality*, and poor customer satisfaction. In one instance, there was an assumption made that it would take 5 days to integrate a new set of features for a customer. It ended up taking 114 days, all of which added to the cost for ORG-A and diminished their total life-cycle profit from the product. There were several contributing factors to this *delay*, including *defects*, *complexity*, *technical debt*, and *context switching* (there had not been a dedicated team focused on the integration). ORG-D noted that customer-specific projects were a significant cost for their organization, increasing their operating expenses and reducing their profitability. A micronarrative from BU2 cited “*cost and complexity*” of C2 policies for onsite activities with customers. This is an example of *extra (unhelpful) processes* contributing to extra costs and decreased profit.

Findings from ORG-F demonstrate some awareness of the impact of impediments in terms of financial cost. However, from their perspective, *morale* and *time* were more heavily impacted. The absence of evidence supporting impact on life-cycle profit does not mean it is not an issue.

Sometimes people in organizations do not know the answer. For example, in ORG-F it is common that teams do not know how the product they work on makes money for their company, or how their customers make money from the product. One of the goals they undertook (Table 6.8) was to better understand how their products make money and communicate this more broadly across their teams.

6.3 RO2: Impediment Management Framework

This section presents the findings related to RO2. Section 5.5.2 describes the three research phases for this study. Table 6.1 shows the subsection that presents the findings from each research phase that contribute to addressing each of the subobjectives from RO2. Table 6.1 serves as a guide for the remainder of this section to cross-reference subsections with the associated data gathering forum, data set, research method, research phase, and the organization from which the data came.

Table 6.1 Relating the findings in each research phase to RO2 sub-objectives

Section	Research Phase	Company	Organization	Data Set Code	Research Objective	Focus
6.3.1	1	C1	ORG-A	FG-A1	RO2(i)	Analyze System Patterns
6.3.1	2	C2	BU2	FMA1	RO2(i)	Analyze System Patterns
6.3.1	2	C2	BU2	FMA3	RO2(i)	Analyze System Patterns
6.3.1	2	C2	BU3	FMA2	RO2(i)	Analyze System Patterns
6.3.1	2	C2	ORG-E	FG-E1	RO2(i)	Analyze System Patterns
6.3.1	2	C2	BU2	SMS1	RO2(i)	Analyze System Patterns
6.3.1	2	C2	BU1	SMS3	RO2(i)	Analyze System Patterns
6.3.1	2	C2	BU3	SMS2	RO2(i)	Analyze System Patterns
6.3.1	3	C2	ORG-F	FG-F1	RO2(i)	Analyze System Patterns
6.3.1	3	C3	ORG-G	FG-G1	RO2(i)	Analyze System Patterns
6.3.2	1	C1	ORG-A	FG-A2	RO2(ii)	Identify impediments
6.3.2	2	C2	ORG-E	FG-E2	RO2(ii)	Identify impediments
6.3.2	2	C2	BU2	SMS1	RO2(ii)	Identify Impediments
6.3.2	2	C2	BU1	SMS3	RO2(ii)	Identify Impediments
6.3.2	2	C2	BU3	SMS2	RO2(ii)	Identify Impediments
6.3.2	3	C2	ORG-F	FG-F2	RO2(ii)	Identify impediments
6.3.2	3	C3	ORG-G	FG-G2	RO2(ii)	Identify impediments
6.3.3	1	C1	ORG-A	FG-A3	RO2(iii)	Make sense of impediments
6.3.3	1	C1	ORG-A	FG-A4	RO2(iii)	Make sense of impediments
6.3.3	1	C2	ORG-B	FG-B1	RO2(iii)	Make sense of impediments
6.3.3	1	C2	ORG-C	FG-C1	RO2(iii)	Make sense of impediments
6.3.3	2	C2	ORG-D	FG-D1	RO2(iii)	Make sense of impediments
6.3.3	2	C2	ORG-E	FG-E3	RO2(iii)	Make sense of impediments
6.3.3	3	C2	ORG-F	FG-F3	RO2(iii)	Make sense of impediments
6.3.3	3	C3	ORG-G	FG-G3	RO2(iii)	Make sense of impediments
6.3.4	1	C1	ORG-A	FG-A5	RO2(iv)	Resolve impediments
6.3.4	1	C1	ORG-A	FG-A6	RO2(iv)	Resolve impediments
6.3.4	1	C1	ORG-A	FG-A7	RO2(iv)	Resolve impediments
6.3.4	1	C2	ORG-B	FG-B2	RO2(iv)	Resolve impediments
6.3.4	2	C2	ORG-D	FG-D2	RO2(iv)	Resolve impediments
6.3.4	2	C2	ORG-E	FG-E4	RO2(iv)	Resolve impediments
6.3.4	3	C2	ORG-F	FG-F4	RO2(iv)	Resolve impediments
6.3.4	3	C3	ORG-G	FG-G4	RO2(iv)	Resolve impediments

6.3.1 RO2(i): Analyze System Patterns

The first step in the impediment management framework, as presented in Figure 5-1, is to analyze the system patterns in the organization. This section presents findings that help to analyze the patterns of flow in a system. A thematic coding analysis of the findings, as described in section 5.7.5, reveals themes of system patterns in the organizations in this study. The dominant themes of system patterns that emerged from an analysis of the findings are, in order: *Culture, Technology, Process, Organization, Strategy, Product, Team, Customer, Tools, People, and Architecture*. These are the dominant themes of system patterns that influence flow and impediments in the organizations in this study.

Value Stream Maps

Value Stream Maps are described in section 2.2.2, with *identify the value stream* noted as the second of five principles of lean product development. Figure 6-2 shows a value stream map for ORG-A, identifying nine flow

states as part of their value stream. They identified *requirements* and *sunset* as the two system boundary states. The seven intermediate states are *design authority*, *customer review*, *development*, *merge*, *regression*, *release*, and *production support*.

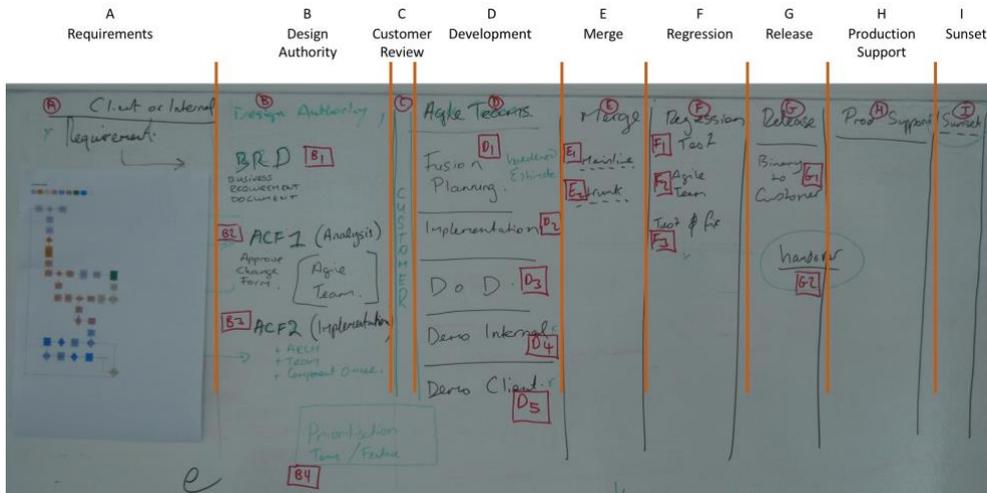


Figure 6-2 Value Stream Map for ORG-A

ORG-A already had a defined Software Development Life Cycle (SDLC) model that was part of their company’s defined process. The SDLC focused primarily on activities performed by different functional groups, the sequencing of tasks, major milestones and phase gates, and the creation of significant process artefacts. The value stream map provided a different focus. Through the value stream map, they articulated how the organization creates value from their customer’s perspective, and how that value actually flows through the organization.

ORG-E identified *concept* to *post-deployment* as the two boundary states of their value stream, as shown in Figure 6-3.

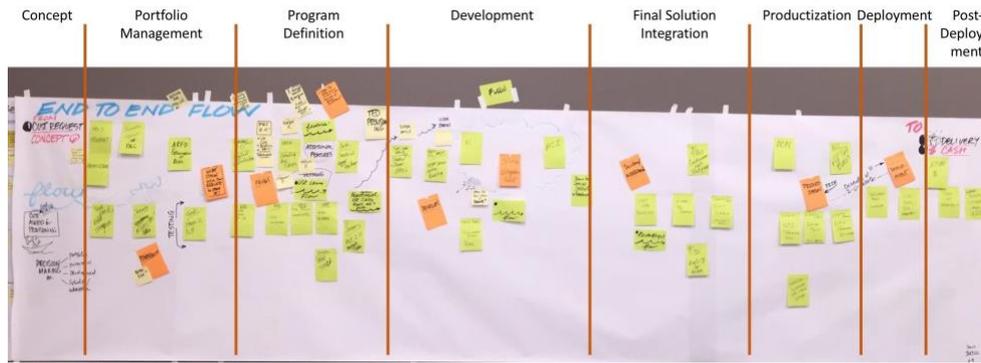


Figure 6-3 Value Stream Map for ORG-E

The ORG-E value stream has 8 flow states, including the significant activities, owners, and outputs for each state. Their flow states are *Concept*, *Portfolio management*, *Program definition*, *Development*, *Final solution integration*, *Productization*, *Deployment*, and *Post-deployment*.

Flow Metrics

As described in Chapter 3, patterns in an organization can be discerned from a qualitative analysis of flow metrics. The organizations in this study visualize throughput in a *throughput diagram* resembling a histogram. The vertical axis represents the number of flow items, and the horizontal axis represents time. Figure 6-4 shows a throughput diagram for a team that started work on a new product in November. A throughput diagram often shows a rolling window of the previous 12 months of throughput. Given the current month is July, there will have been no throughput for the July to October period. The subsequent month's throughput diagram would show the period August to July.

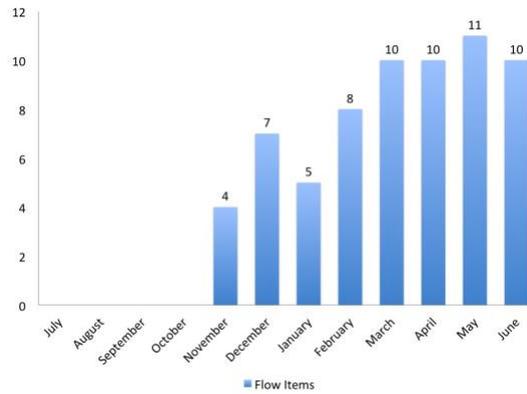


Figure 6-4 Visualizing organization productivity using a throughput diagram

Figure 6-5 shows the same system and throughput, but this time with a demand analysis applied. In this diagram, features represent value demand, while defects and technical debt represent failure demand. Where Figure 6-4 shows that throughput is the second highest that it has ever been, Figure 6-5 shows that 70% of the items represent failure demand. This means that of all the demand being placed on the capacity of the system, only 30% of the output is delivering value.

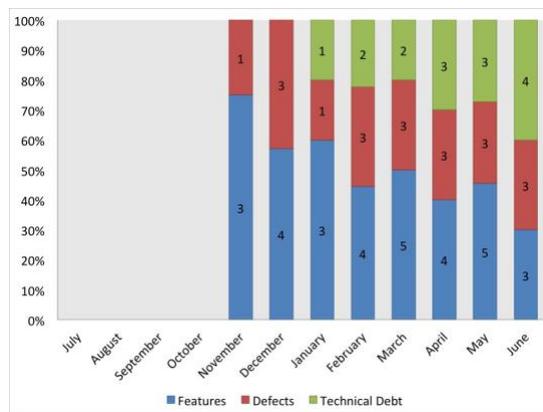


Figure 6-5 Value demand and failure demand as percentage of throughput

When combined with a Demand Analysis, throughput shows what type of demand is being placed on the system, and where the capacity of the system is being directed. Throughput Analysis trends indicate where demand will be in the near future. Patterns in previous throughput are likely to repeat in future throughput. If managers are unhappy with that pattern, it is unlikely to change unless they take steps to improve the flow.

Figure 6-6 shows two cycle time diagrams. Both diagrams show the cycle times for value demand (user stories) and failure demand (defects). The vertical axis is measured in days and shows the average cycle time in days at a point in time. The horizontal axis is time, with each point in time representing a particular date. The cycle time diagram shows trends in cycle time over a period of time. These diagrams hence show the general predictability and responsiveness of the system by showing on average how long it takes to complete a user story or resolve a defect.

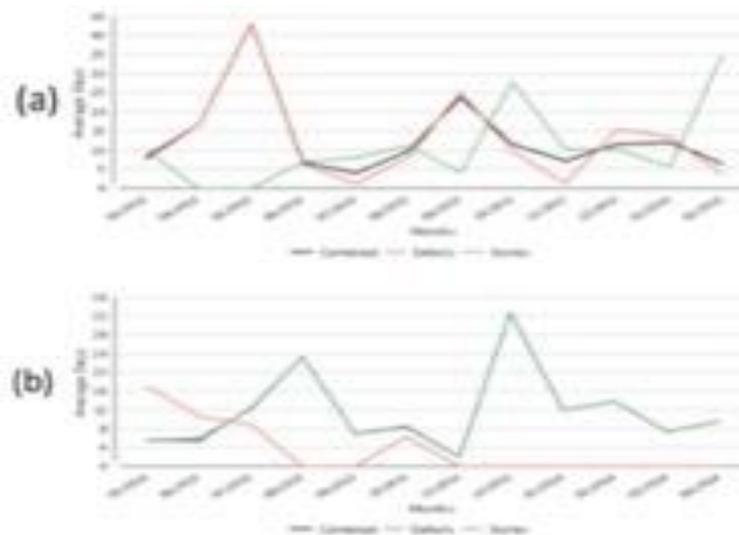


Figure 6-6 Visualizing organization responsiveness with cycle time

Predictability is observed to be a factor of variation in throughput and cycle time. Cycle time variation makes it difficult to predict response times for individual work items. Throughput variations made it difficult to predict how many, or which, features would be complete each month, or as part of an overall release with a large batch size that contains many features.

CFDs can be used as indicators of the future level of smoothness of the flow of work through the system. CFDs help in answering two fundamental questions. The first question is some variation of “*When will a given set of functionality or work be complete?*” It is straightforward to make predictions that fall within a date range by extrapolating from past data. Recall from section 4.3.2 that plausibility is more important than accuracy when trying to make sense of patterns in complex systems. Figure 6-7 shows that the current defined backlog will likely be complete sometime

between T₁ and T₂. Again, this is a qualitative not a quantitative analysis. This study is only concerned with analyzing these metrics from the perspective of seeing patterns in flow and identifying impediments.

The second question is some variation of “*What work will be complete by a given date?*” It is straightforward to make predictions that fall within a range of work items by extrapolating from past data. From the desired date, draw a vertical line that intersects the two diagonals. From the intersection points, draw a horizontal line from each intersection point until it reaches the backlog. This will give a range within which is the probable capacity of delivery for the given date.

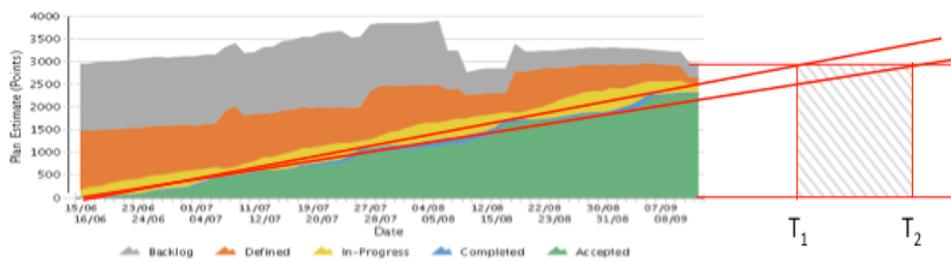


Figure 6-7 Visualizing organization predictability using CFDs

Figure 6-8 shows a representative CFD. Each flow state, or queue, is represented by a different color or shade so they can be distinguished from each other. The horizontal axis is time. The vertical axis is the size of the work represented in user story points. Section 3.3.3 showed that queue size could be measured as a count of the items in the queue, or as an estimate of the size of the work in the queue. This CFD is using the latter.

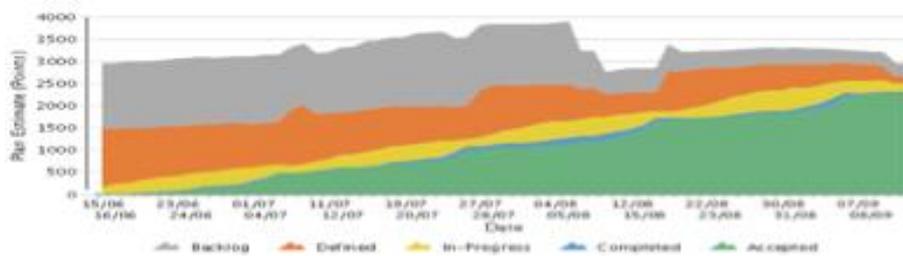


Figure 6-8 A CFD showing smooth flow of work and high organization productivity

Figure 6-9 shows another representative CFD from the case study. This one shows a large amount of WIP (yellow) and relatively little work getting

completed. The picture shows that even though the team is not getting work completed (blue), they continue to start new work. It also shows a delay between work getting to the “Completed” state, and work getting to the “Accepted” state (green).

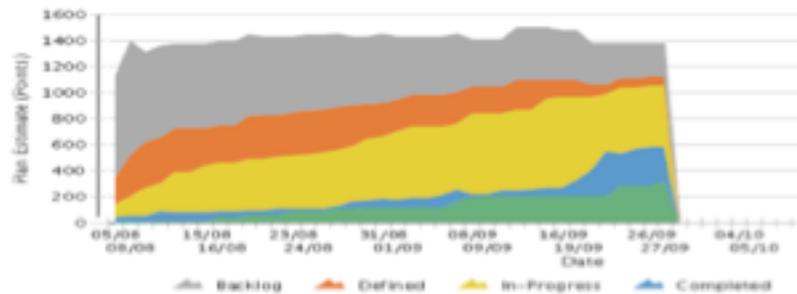


Figure 6-9 A CFD showing too much WIP and low organization productivity

Organization Culture

Flow happens within an organization context, and even across multiple organizations, as shown in the research context in section 5.5.1. This broader context influences the quality of flow. Chapter 4 reviewed the literature on software development organizations in the context of this study. Section 4.2.6 makes the point that culture is a complex adaptive system. Section 4.3 reviews the literature on sensemaking in organizations and describes how micronarrative sensemaking can provide insights into organizations and their culture. The micronarrative sensemaking approach described in section 5.6.2 results in patterns related to the culture of the organizations in this study. This section presents findings from this approach related to the culture of the software development organizations in this study.

Impediments and Flow Quality

The findings show evidence of patterns of flow quality in ORG-G, and how impediments are influencing flow quality. A majority of experiences reflect that flow of work is perceived as not smooth, and there are significant impediments impacting flow. Findings show that *delays* are a significant impediment in a majority of experiences. *Too much WIP* is a significant factor in a majority of experiences. *Extra processes* are a factor in 24% of

experiences. *Extra features* are a factor in 21% of experiences. *Defects, technical debt, and failure demand* are factors in 29% of experiences. *Unused employee creativity* is a factor in 44% of experiences.

The findings represented in Figure 6-10 shows a set of cultural patterns that provide some context about the factors influencing flow and impediments in ORG-G.

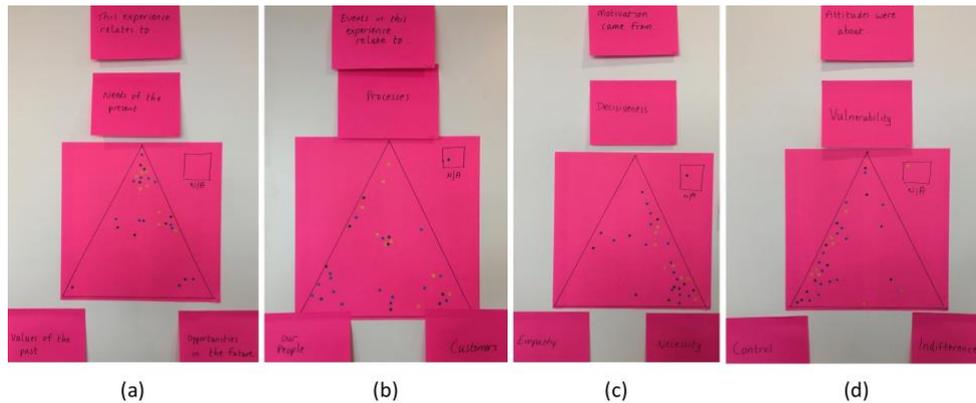


Figure 6-10 Cultural patterns identified through micronarrative sensemaking

Figure 6-10 (a) shows there is a dominant pattern in ORG-G of responding to the needs of the present over taking time to reflect on past experiences or to consider future opportunities. Figure 6-10 (b) shows a set of patterns where a significant number of experiences of people in ORG-G are dominated by customers, and another pattern where the experiences are dominated by other people in ORG-G. The findings show that there are insufficient processes in place in ORG-G, and there is a heavy reliance on people to figure out what to do in given situations. This applies to both internal activity and in dealings with customers. Figure 6-10 (c) shows a pattern where morale for action is dominated by necessity, or a combination of necessity and decisiveness. There is a notable lack of empathy when it comes to taking action, indicating a lack of consideration for the needs of others. Figure 6-10 (d) shows that people's attitude in ORG-G are typically not indifferent, are seldom vulnerable, but are dominated by a need for control.

Impediments and Emotional Sentiment

Section 4.3.5 discusses the significance of emotion in the context of sensemaking in organizations. Figure 6-11 is a summary of the emotional sentiment behind each of the experiences shared as micronarrative in BU1, BU2, and BU3.

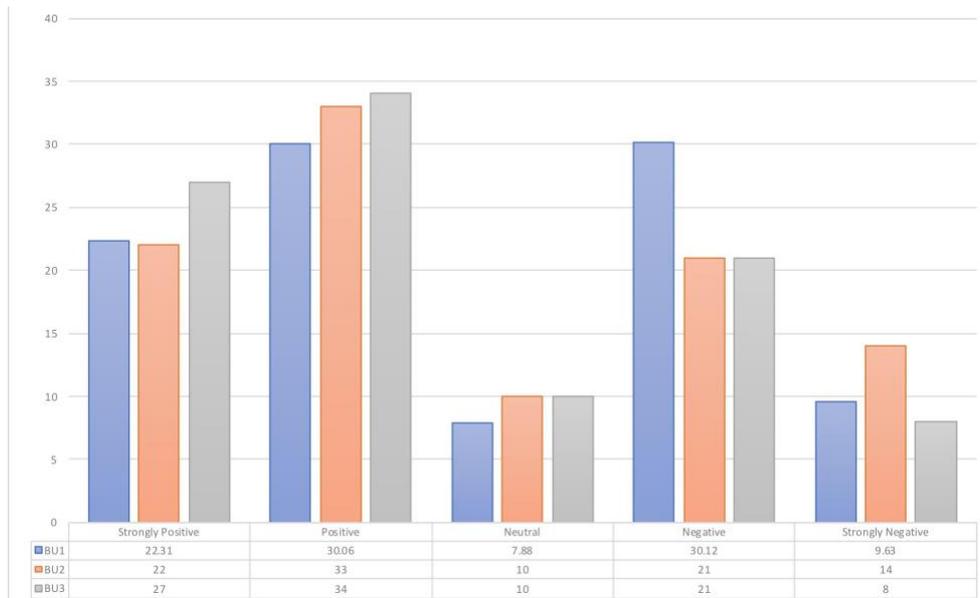


Figure 6-11 Emotional sentiment associated with people’s experiences

This section shows how the micronarratives and their emotional sentiment are mapped to impediments. Figure 6-12 is a summary of each type of impediment with the emotional sentiment of the experience for BU1.

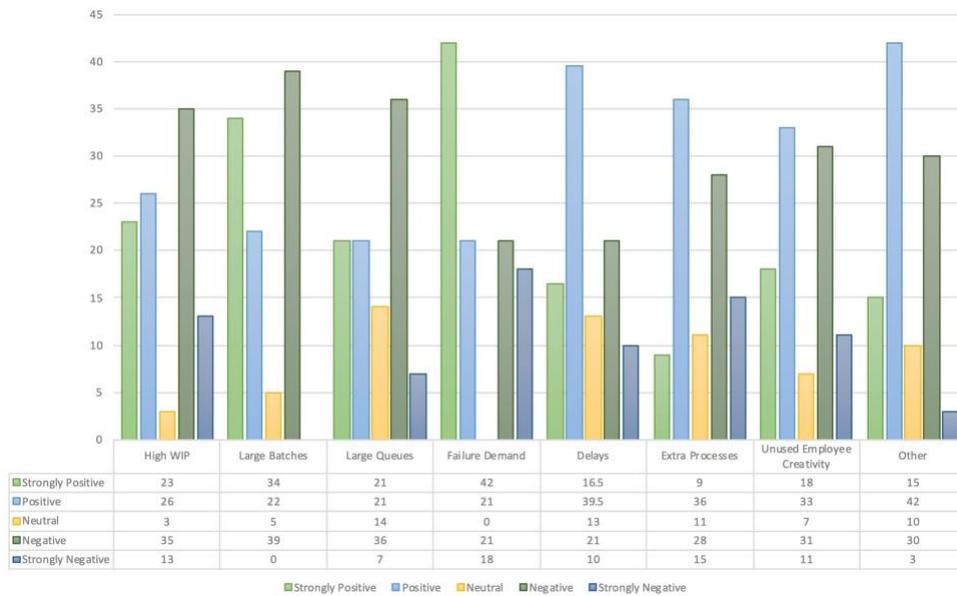


Figure 6-12 Emotion of BU1 experiences with impediments and contributing factors

Figure 6-13 is a summary of each type of impediment with the emotional sentiment of the experience for BU2. The types of impediments that feature strongly when negative experiences are observed include *extra processes*, *delays*, *partially done work*, *unused employee creativity*, and *task switching*. *High WIP* and *failure demand* also feature strongly as contributing factors.

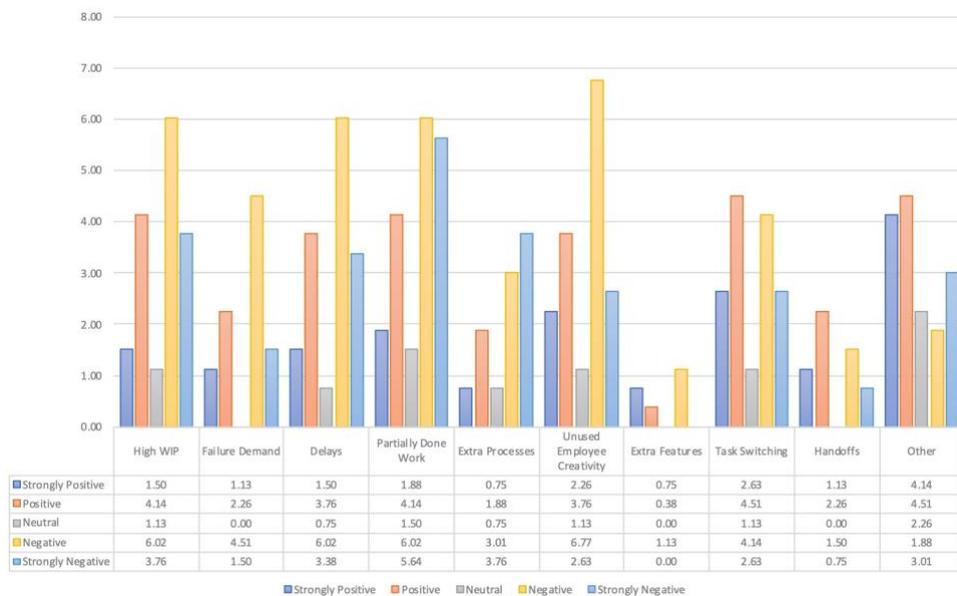


Figure 6-13 Emotion of BU2 experiences with impediments and contributing factors

Figure 6-14 is a summary of each type of impediment with the emotional sentiment of the experience for BU3.

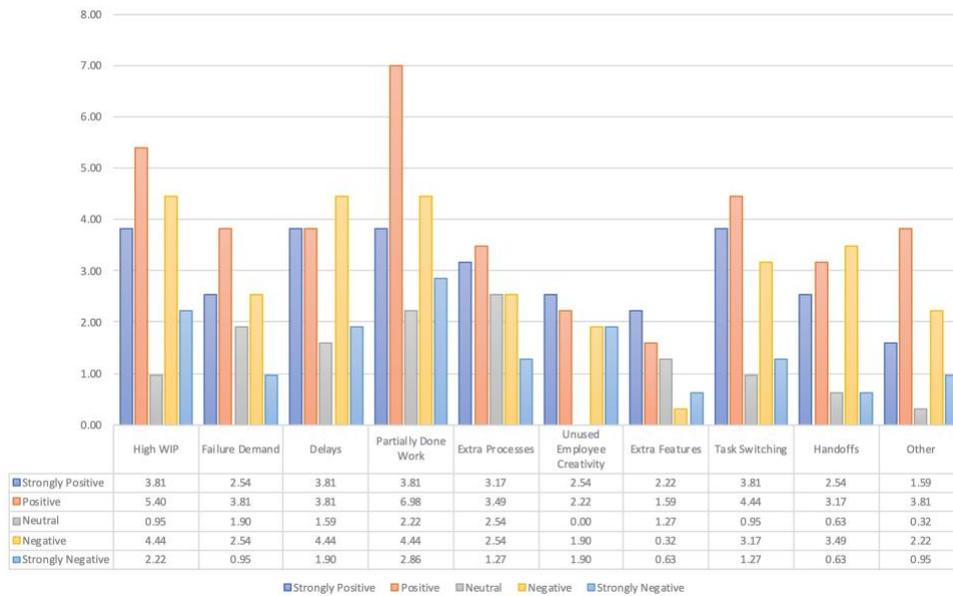


Figure 6-14 Emotion of BU3 experiences with impediments and contributing factors

As this section shows, not all experiences that involve impediments have a negative sentiment. For example, a micronarrative from BU2 reports how a product team was dealing with *productivity* issues as a result of *delays*: “We recently had a challenge of productivity on our scrum team. We have new members that were struggling with the new technologies. My scrum master and I discussed how we can rearrange the team to be more productive. We delivered this idea to the team and the team positively embraced the change. It is too early to tell if the change has been productive but the signs are definitely pointing that way. We were impressed with the can do attitude and do whatever it takes approach with our team members.” This experience involved continuous integration, and affected product *quality*. Yet, despite the challenges, the experience was *strongly positive*.

Reasons for Sharing Specific Experiences that Highlight Impediments

As discussed in section 4.3.5, it can be helpful to understand the reason that people had for sharing the particular experiences that they chose to share. Figure 6-15 shows that the two most common reasons for sharing the specific experiences in BU2 are a desire to improve things (41%) and to celebrate (16%).

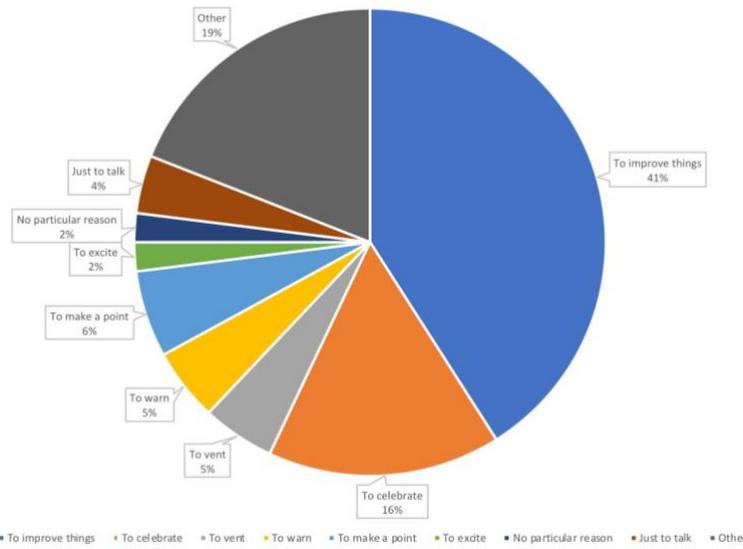


Figure 6-15 People’s reasons for sharing their specific experiences

Figure 6-16 shows impediments mapped to experiences that were shared to celebrate. The top impediments here are *WIP*, *task switching*, and *unused employee creativity*. So, even though there are impediments, people are still inclined towards celebration.

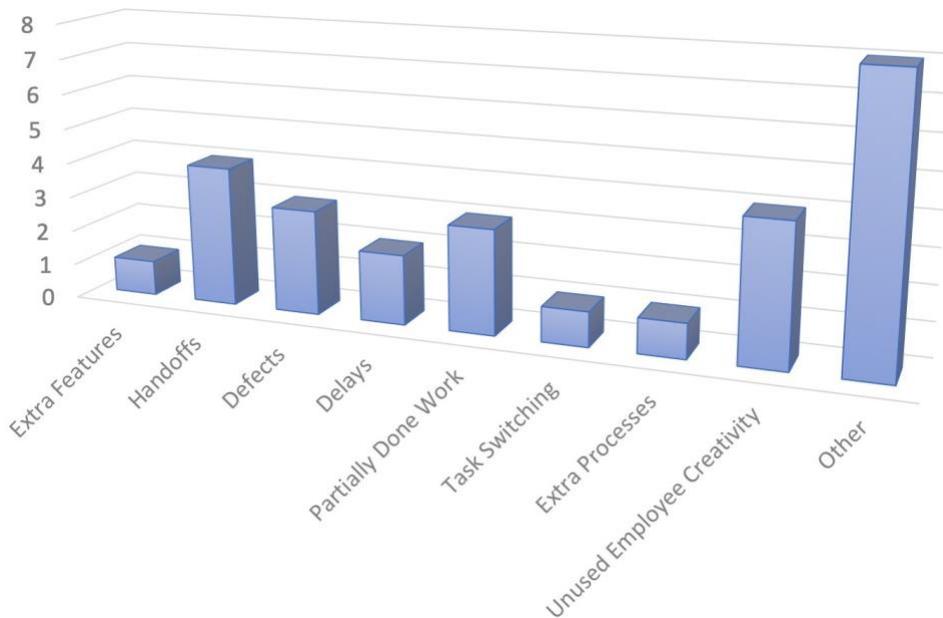


Figure 6-16 Impediments associated with experiences shared for celebration

Impediments and Morale

This section presents findings that reveal impediments that have the greatest impact on *morale* in organizations. This can provide some insight into

which impediments to improve in order to improve morale. Figure 6-17 shows impediments mapped to experiences that impact on *morale* in BU2. This shows that *unused employee creativity*, *extra processes*, *delays*, *task switching*, and *WIP* are all mapped to a negative impact on morale. In this study, experiences mapped to failure demand (including defects and technical debt), and features that customers did not want, have a much smaller impact on morale than these other factors.

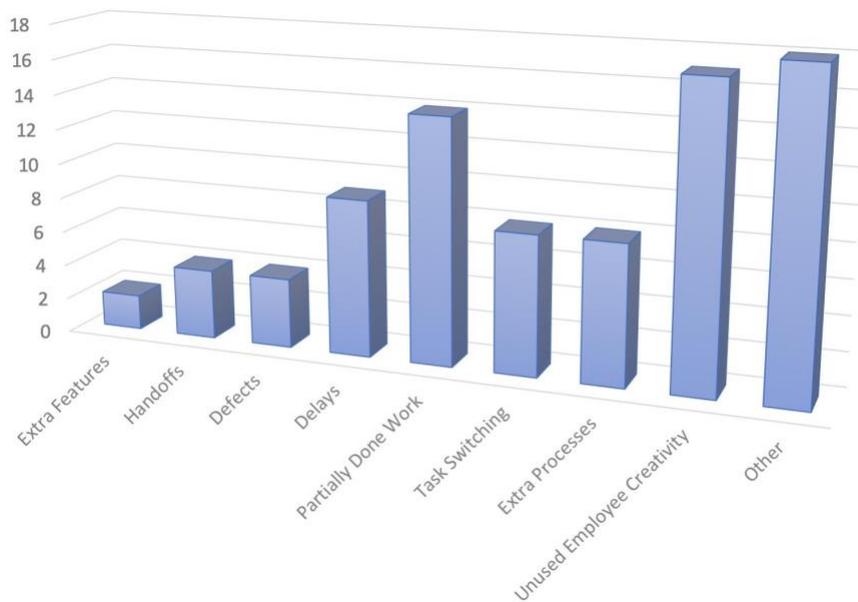


Figure 6-17 Impediments associated with experiences that impact morale

Figure 6-18 shows that the types of experiences shared have an impact on *quality*, *morale*, and *use of resources*.

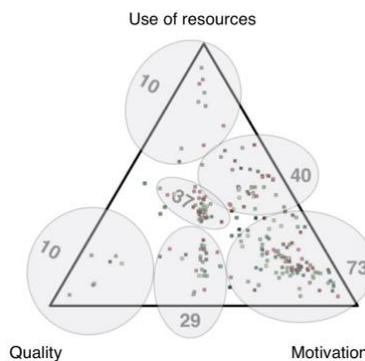


Figure 6-18 Impact on use of resources, quality, and morale

A BU2 micronarrative expressed concern about the culture: “*culture at C2 is very anti-collaborative. Even when your colleagues (on your own team)*

tell you they want to support a working agreement that you have that does not mean that they will. ... the culture is very me first and does not support shared work. My most recent experience where this was painfully proven to be true was when I left a planning meeting thinking we had alignment on how we would plan our work for the AC. We had a wonderful onsite workshop where the engineers collaborated. For those that were remote we had a [remote video] set up. The remote participation was weak and when it came to putting milestones on the [features] following the workshop the teams abandoned the process and it became a free for all despite several attempts to pull us together at a leadership level. The jockeying for assignments is too competitive and when you attempt to put a process in place that prevents behind the back moves it only seems to serve as motivation to find a better way to circumvent. This may be a result of the organization being too big and dispersed in a highly ineffective way.”

When the Experience Occurred

As discussed in section 4.3.5, it can be helpful to understand how recently the experience occurred. This, in turn, helps to inform how recent the impediment is. Figure 6-19 shows how recently the reported experiences occurred, which in turn gives an indication of how immediate the impediments are. Addressing impediments related to experiences that occurred in the most recent 3 months is likely to have more of an impact than addressing impediments related to older experiences.

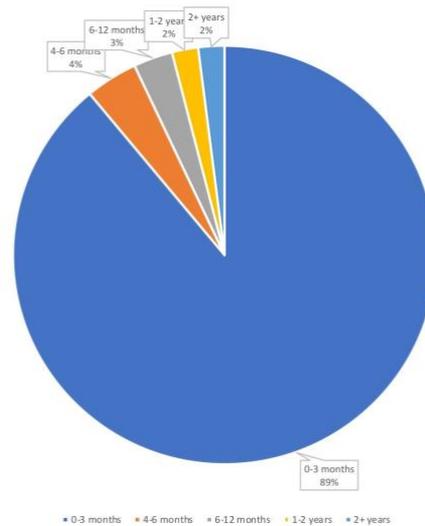


Figure 6-19 Patterns of how recently the experiences occurred

Figure 6-19 shows that the majority of experiences (89%) occurred in the preceding 3 months. This reveals that the majority of experiences leading to impediments are still quite recent for BU2.

Frequency of Occurrence of Impediments

As discussed in section 4.3.5, it can be useful to know how often a particular type of experience occurs in the organization. Figure 6-20 is a summary of how frequently each type of experience occurs in BU2. 35% of these types of experiences occur all the time or are quite common. Only 12% are one-off experiences. 53% are either very rare (15%) or happen sometimes (38%). As will be discussed later, this can serve as useful context to help inform what impediments to resolve.

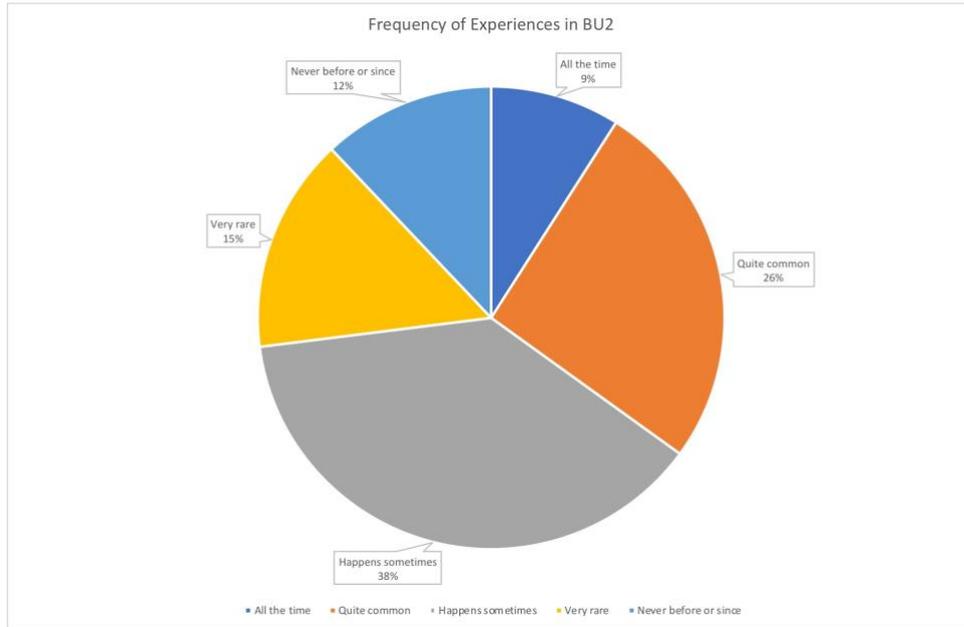


Figure 6-20 Patterns of how frequently these kinds of experiences occur

For each experience that reveals one or more impediments in the organization, Figure 6-21 shows how frequently such experiences occur. From this, it can be seen how frequently different types of impediments are occurring in BU1.

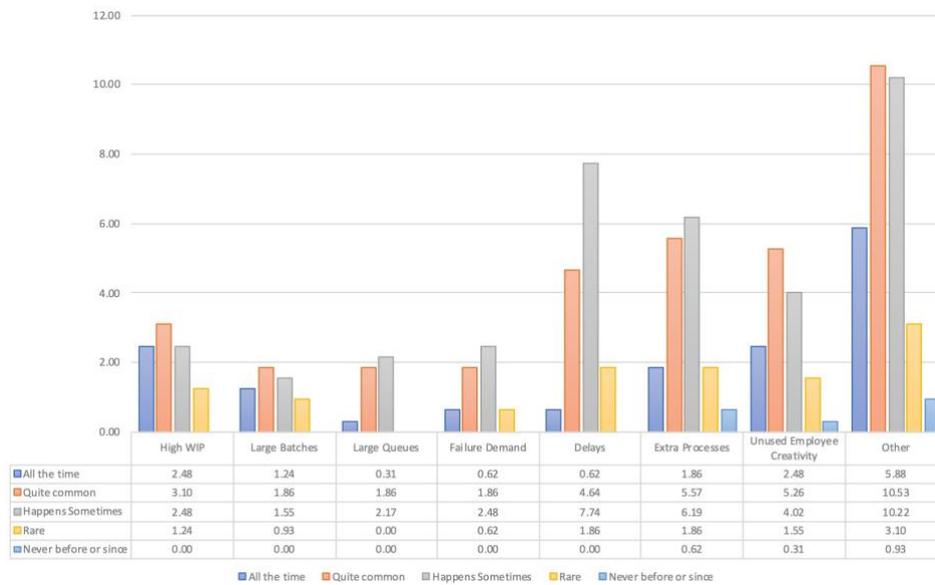


Figure 6-21 Frequency of experiences with impediments and contributing factors

Figure 6-22 shows the proportion of impediments associated with experiences that occur “all the time”, according to respondents in BU1.

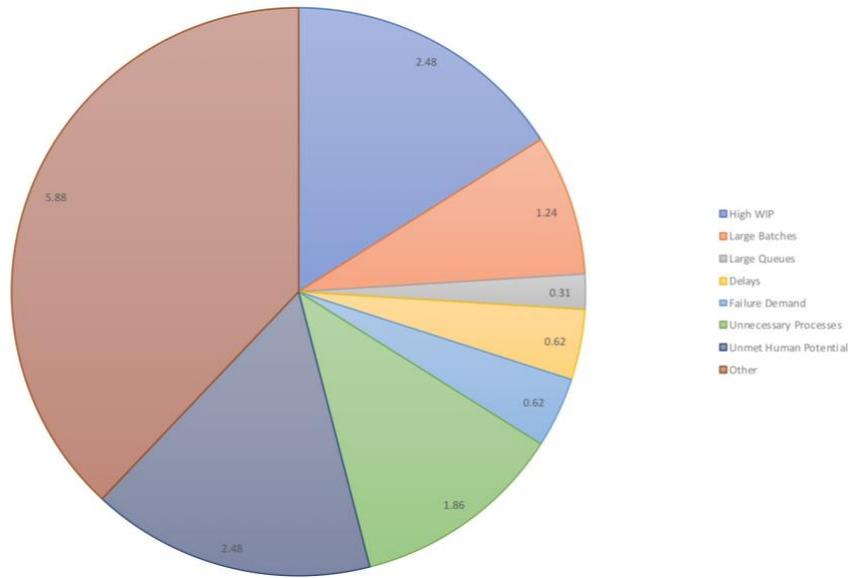


Figure 6-22 Impediments associated with experiences that occur all the time

Figure 6-23 shows how recently these experiences occurred in BU3.

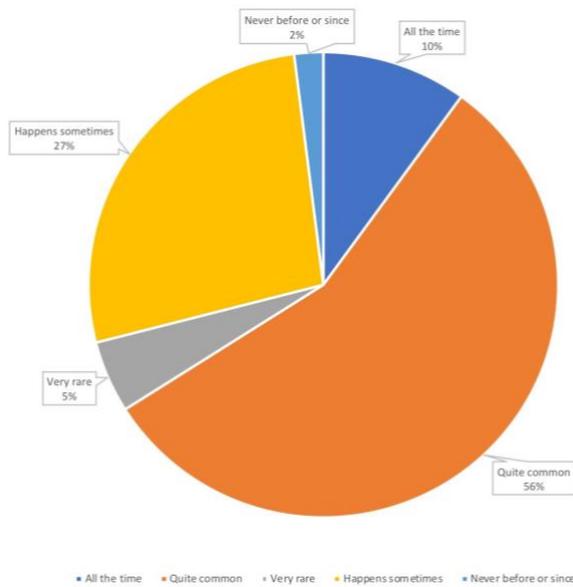


Figure 6-23 Frequency of occurrence of experiences involving impediments

The majority of experiences that occur are quite common (56%) or occur sometimes (27%).

Combined Perspective on the Context of Impediments

The preceding data can be combined to show patterns of how people in organizations feel about these experiences, why people share these

particular experiences, how often these types of experience occur, and how recently they occurred. Figure 6-24 shows an analysis of the findings from micronarrative sensemaking data in ORG-F that reveal patterns related to (a) the emotional sentiment, (b) the reason for sharing these specific experiences, (c) the frequency with which this type of experience occurs, and (d) the timeframe within which the experience happened.

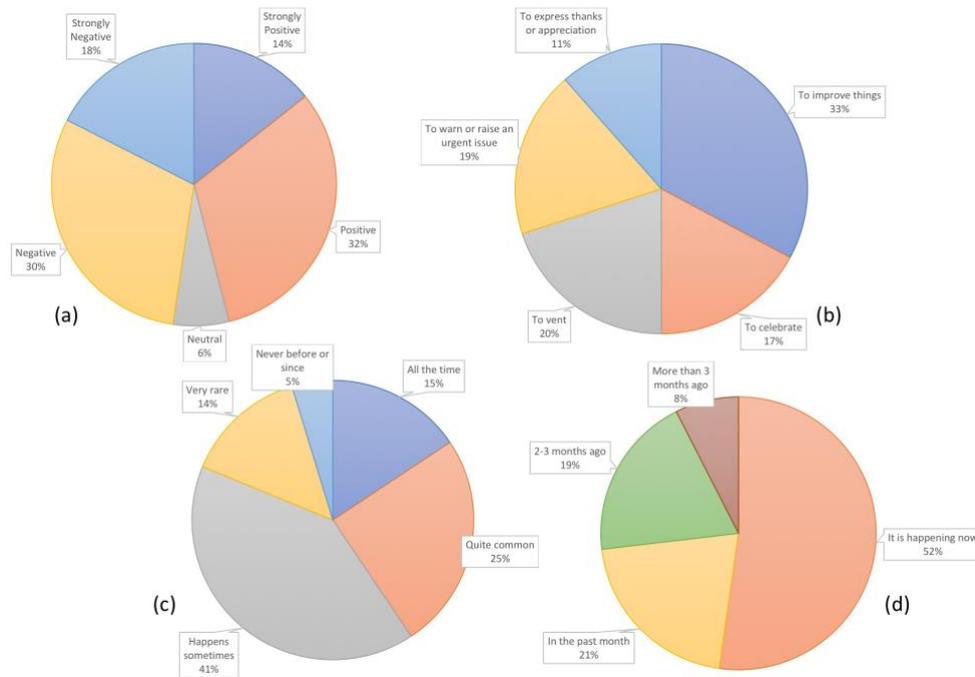


Figure 6-24 Understanding context of impediments through people's experiences

Figure 6-24 (a) shows that 46% of experiences have a positive or strongly positive sentiment, while 48% have a negative or strongly negative sentiment. Figure 6-24 (b) shows the reasons why participants opted to share the particular experiences that they chose, with the most common reason being “to improve things”. Figure 6-24 (c) shows how often these types of experiences occur in ORG-F, with the frequency of most experiences being “happens sometimes” (41%) or “quite common” (25%). Figure 6-24 (d) shows how recent these experiences are, with 52% “happening now”. Only 8% of experiences are more than 3 months old, meaning 92% of experiences occurred within the preceding 3 months.

Impediments and Psychological Safety

The topic of psychological safety (discussed in section 4.2.6 in the context of organization culture) emerged in earlier focus groups and through sensemaking in BU2 and in some of the organizations that are part of BU2 (see section 5.5.1 for research context and units of analysis). In particular, there were instances where participants expressed that they would be reluctant to raise impediments or discuss waste and other forms of impediments in their work context, outside of the focus groups and sensemaking. Fear of negative consequences from their management was cited as the primary reason. This topic also emerged during preliminary discussions with participants from ORG-F. The findings show a pattern that emerged from the sensemaking framework to specifically inquire about psychological safety, and its effects on raising impediments. The findings show that there is a significant pattern of people feeling unsafe (from a psychological safety perspective) in the organization. Findings indicate about half of peoples' experiences reflected a balanced amount of safety, neither threatened nor over-protected. There are no instances where people experience over-protection or felt "too safe".

More Experiences like These, Fewer Experiences like Those

Chapter 4 discusses the role of pattern management in complex systems to "*amplify the positive and dampen the negative*" experiences of people in organizations in order to achieve "*more stories like these, fewer stories like those*" (Van Der Merwe et al., 2019). This is a guiding heuristic for safe-to-fail intervention strategies (Van Der Merwe et al., 2019, Snowden and Boone, 2007). As discussed in section 4.3.4, micronarratives are the "*small stories*" that reflect how people make sense of the world and their experiences in it (Van Der Merwe et al., 2019). So, in the context of this study, the guiding heuristic becomes "*more experiences like these, fewer experiences like those*". In designing interventions to resolve impediments and improve flow, which is a topic of section 6.3.4 below, paying attention to the positive and negative experiences of people in the organization, and determining which experiences to amplify or dampen, provides a heuristic

that informs action, particularly in the *complex* sensemaking domain. Positive and negative experiences are referenced throughout this chapter. Table 6.2 and Table 6.3 each show some selected examples of the experiences of people in organizations in this study. The experiences have been captured as micronarratives using the SenseMaker tool. Both sets of experiences are associated with impediments and reflect challenges in their respective organization. Yet, those in Table 6.2 have positive or strongly positive sentiment, and so are candidates for experiences to encourage more of or amplify in the organization.

Table 6.2 Examples of “more experiences like these”

Experience captured as Micronarrative
As a newcomer to a project with only email communication to date, I was able to sit down with someone already involved and speak face to face in order to get a frank and clear summary of the project and how it currently stood, which helped alleviate some concerns
Analysis of [Feature X] efficiency in [Project Name] project. resulted in the customer being happy with our analysis and realised where the issues were and what needed to be done
Forming our scrum team for [Customer Name] [Capability Name], started using a full scrum framework to gather together individuals from I&T, Delivery and CustOps to form a cohesive team.
myself and a couple of colleagues noticed how the setup of [Big Room] during our [Release Name] release planning fostered collaboration as opposed to the setup of the open space we daily work. This brought us to draft a new floor plan that I'm going to suggest to the whole team in order to have a more collaborative and stimulating work environment.
Attending end of sprint demo
We worked on [feature X] for a client. The PO made a commitment on our behalf. We discovered that we cannot meet the commitment. I don't recall what exactly happened, anyway, we told the PO that we are unable to meet the commitment. Then the PO held a conversation with us to decide what can be committed and what not, and how to split the work so we can deliver value with each small commitment. ... It was uncomfortable at the beginning, and ended happy. Also we learned from the experience how to split requirements in a better way.

Experiences in Table 6.3 have negative or strongly negative sentiment. These are examples to encourage fewer of or dampen in the organization.

Table 6.3 Examples of “fewer experiences like those”

Experience captured as Micronarrative
Engineering teams are continually interrupted during roadmap sprints to work on emergency quality fixes for work that has been released. In addition, Sales will prioritize new customer features, and team is again interrupted from current sprint only to come in past the deadline with an untested product.
Customers are impacted by unscheduled quality related fire-drills. Engineering teams report that they are continually taken away from their development work in order to focus on quality related fire-drills, resulting halting previously scheduled feature work.
Strong commitments are possible only when you don't have external dependencies! If we had choice, we would often give no commitment at all because we know that we are working on things that have a lot a strong dependencies with things that are not in our end. But the way we work in this program don't allow us not to give commitments... So what should we do? ... Should we constantly be pessimistic and never commit or commit only with a few sprint of error margin? I'm concerned that engineering managers are making the engineers do extra work on the side, either out of necessity (e.g. lab work) or out of personal development goals. This can be a distraction from the committed work for the scrum team. If they factor in this extra work when they do planning, then I guess it's fine. But they should be letting the SM at the very least know about this reduction in their capacity so that the planning is done right. I don't think everyone does that.
Many times the team works hard to achieve the sprint commitments, but then after the actual coding work is completed, the CI/CD process is very long and it takes quite a while until actual changes land at the customer site. In the specific example I'm thinking on, once the changed entered the pipeline, it took almost two sprints until it was promoted. That is because continuously there were more urgent pipeline requests.
Overworked: Determine capacity of team. Set aside capacity for unexpected/unplanned plan to capacity. Get asked to take more but can't push anything out. Try to push back to take something out but usually end up having to do it anyway.

6.3.2 RO2(ii): Identify Impediments to Flow

The objective of RO2(ii) is to identify impediments to flow in organizations.

Interpreting System Patterns

This section presents findings from the study related to identifying impediments at an organization or system level identified through sensemaking. Figure 6-25 is a summary of the types of impediments identified in BU1, BU2, and BU3. For each of the three business units, Figure 6-25 shows the percentage of each impediment type identified through the sensemaking surveys.

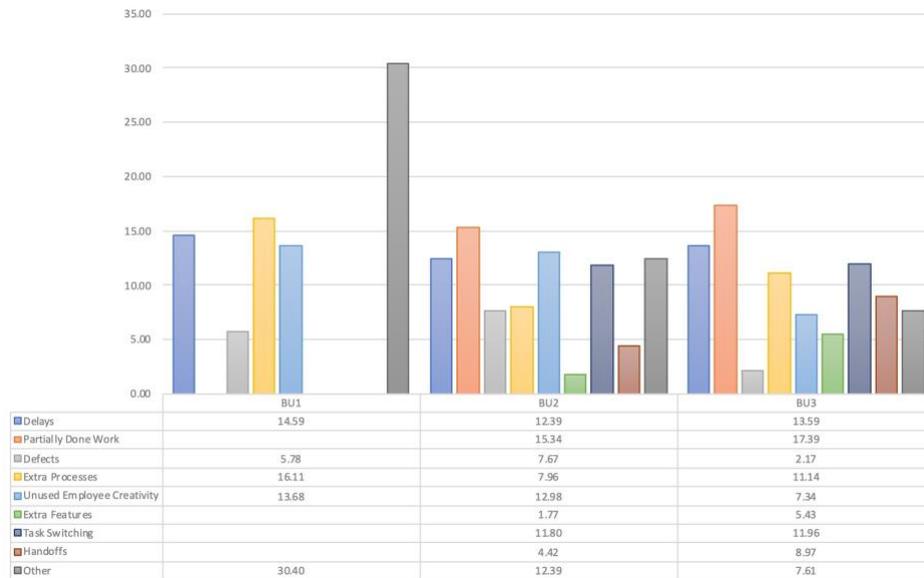


Figure 6-25 Impediments identified via micronarrative-based sensemaking analysis

Of the impediment types described in section 2.5.1, the dominant patterns of impediments in BU1 are *delays*, *extra processes*, and *unused employee creativity*. The dominant patterns of impediments in BU2 are *delays*, *partially done work*, *extra processes*, *task switching*, and *unused employee creativity*. The dominant patterns of impediments in BU3 are *delays*, *partially done work*, *extra processes*, and *task switching*. In contrast to some of the lean literature, impediments such as unneeded *extra features* are not a dominant pattern in the three business units studied. As a general pattern across all three business units, the dominant patterns of impediments are related to *delays*, *extra processes*, *task switching*, and *unused employee creativity*.

From the value stream patterns in Figure 6-2, ORG-A identified 90 impediments that were categorized under the different headings defined in section 2.5.1. A count of impediments per category is shown in Table 6.4, including the number of impediments in each category and the percentage of total impediments represented by each category.

Table 6.4 Summary of impediments identified from value stream mapping

Category	Count of Impediments	%
Delays	20	22.22 %
Defects	14	15.56 %
Partially Done Work	12	13.33 %
Task Switching	12	13.33 %
Extra Processes	11	12.22 %
Handoffs	10	11.11 %
Unused Employee Creativity	5	5.56 %
Extra Features	3	3.33 %
Other	3	3.33%

Note that the combination of *delays* and *failure demand* defects account for approximately 38% of all impediments identified in ORG-A. For each category, a further table shows all the identified impediments from a specific category. The impediments include a statement of what the perceived impediment is, who is impacted, and, where known, the extent of the impact.

Co-Occurrence of Impediments

Figure 6-26 shows for any given impediment type, other impediment types that were identified as present in the organization at the same time.

	A : Delays	B : Extra Processes	C : Unused Employee Creativity	D : Task Switching	E : Handoffs	F : Other Impediments	G : Extra Features	H : Partially Done Work	I : Defects
1 : Delays		164	94	99	70	37	35	25	9
2 : Extra Processes	164		88	73	29	27	22	21	1
3 : Unused Employee Creativity	94	88		49	20	27	6	10	2
4 : Task Switching	99	73	49		32	11	9	5	0
5 : Handoffs	70	29	20	32		11	14	4	0
6 : Other Impediments	37	27	27	11	11		9	3	0
7 : Extra Features	35	22	6	9	14	9		2	1
8 : Partially Done Work	25	21	10	5	4	3	2		0
9 : Defects	9	1	2	0	0	0	1	0	

Figure 6-26 Visualizing the co-occurrence of multiple impediments

Note that the numbers in Figure 6-26 come from an NVivo cross-coding comparison and represent the number of times that an impediment was coded in the findings simultaneously with another impediment. It is not the total number of times that this particular impediment was coded across all the findings. For example, cell A4 shows that in cases where *delays* are identified as an impediment in the findings, there are 99 cases where *task switching* is also identified as an impediment. The highest incidence of multiple impediments is between delays and extra processes, with 164 occasions of both impediments noted simultaneously. Figure 6-26 illustrates the point that multiple impediments are present in a system at any given time. For example, when there are *delays* identified as an impediment, there is also a very high presence of *extra processes*, *unused employee creativity*, *task switching*, and *handoffs*. Findings also show where there is *too much WIP* or *unused employee creativity*, there is a very high presence of *large batches*.

Monitoring the System

ORG-C, ORG-D, and ORG-F had some experience with managing impediments. They maintained impediment lists. They actively sought out impediments as part of their development approach and had started to

develop a process for managing impediments. Through this they had started to develop some organization-wide sensory capacity for managing impediments. In general, ORG-D managers met daily to review impediments. An ORG-D micronarrative describes “*focus towards obstacle removal*” as something that they wanted to see more of in the organization. Another micronarrative from ORG-D that reflects a negative experience illustrates, from one perspective, the difficulty in establishing a process for continuously monitoring and managing impediments: “*There is no belief on obstacle process. I am not generalizing 100% population. But neither my director nor my [management team] believes in obstacle process. Its only SM who has to run behind in resolving obstacles. Any obstacles raised till date [in part of ORG-D] have not been resolved to satisfaction and we still have got challenges to overcome. RCA for the obstacle is not being looked into*”. This micronarrative shows that in one case (a subset of ORG-D) only a Scrum Master is taking responsibility for impediments, and impediments largely remain uninvestigated and unresolved. ORG-F management teams met three times per week to review impediments. ORG-C and ORG-F had each established a cadence of reviewing impediments at milestones such as sprint reviews or release planning meetings.

Monitoring the system means looking for impediments, but also looking for the absence of impediments. There can be gaps when people try to identify impediments. Some managers expressed concern that teams were not surfacing impediments. As one ORG-C engineering manager noted, in a statement reminiscent of Ohno (2013), “*I don’t worry about the team with lots of impediments. I worry about the team with none.*” However, as discussed in the findings in section 6.3.1 above, there can be many reasons why teams are not identifying impediments, with organization culture and psychological safety being a significant influence. The absence of impediments can also be due to unintentional blindness to impediments, as highlighted in the previous subsection on *Co-Occurrence of Impediments*.

6.3.3 RO2(iii): Analyze Ways to Make Sense of Impediments

Impact of Impediments

Impediment Impact Diagrams help make sense of impediments through analyzing their impact. Figure 6-27 shows two Impediment Impact Diagrams from the findings that ORG-A use to visualize their impediments from different perspectives and make decisions about which ones to address. The sticky notes in Figure 6-27 are not intended to be legible here. Part of the point of presenting Figure 6-27 is to present findings that show patterns of impediments that emerged in ORG-A.

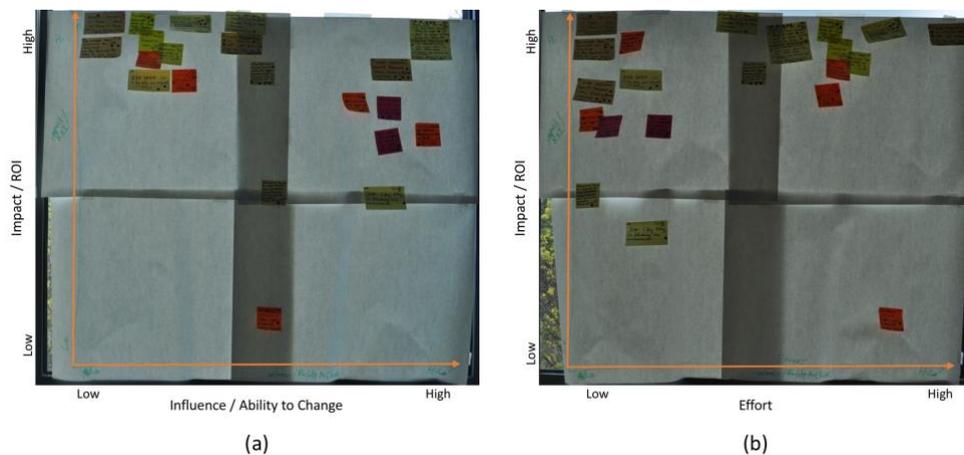


Figure 6-27 Understanding the relative impact of impediments

Figure 6-27 shows the impediments that, if addressed, would have the highest impact in terms of improving the flow of value for ORG-A. It helped them to identify which impediments were within their scope of influence in order to address them directly, and which impediments they would need help with. The findings show that, through a process of reviewing and refining their impediments, ORG-A created eleven impediments from their original set of ninety impediments. The eleven impediments they focused on in order to improve their end-to-end flow are:

1. Build Times
2. Customer Regression
3. Customer Satisfaction

4. Communication Tools
5. Delay in Setting Up System Environments
6. Jira Workflow
7. Manual Regression
8. Onboarding New Employees
9. Test Automation
10. Triaging Invalid Defects
11. Reacting to Localized Issues versus Fixing Underlying Causes

Figure 6-28 shows two Impediment Impact Diagrams from ORG-B that illustrate the impact of the impediments they identified, and also shows who they needed help from in order to resolve the impediments. The vertical axis represents the relative impact and is on a continuum from low to high. The horizontal axis is divided into a number of columns each representing a different stakeholder group from which input or help is sought in order to resolve the impediments.

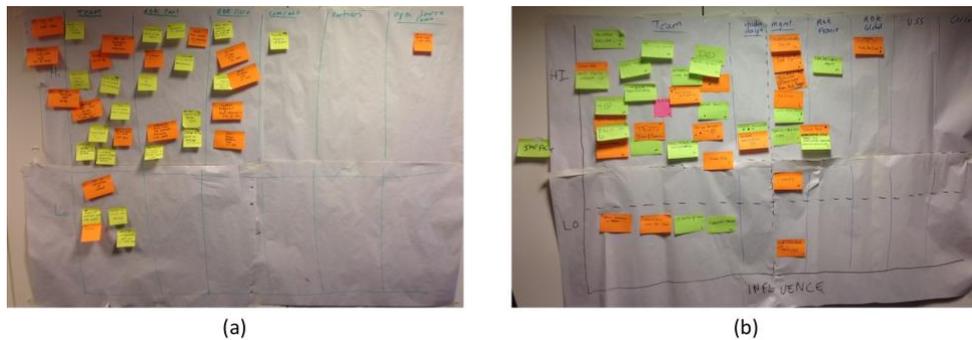


Figure 6-28 Impact of impediments and who needs to contribute to resolving them

Figure 6-28 (a) is from the perspective of the management team of ORG-B. They identified six stakeholder groups, including the management team, the local site, the global product team across their company, a specific business that is both customer and supplier, partners, and the Open Source community.

Figure 6-28 (b) is from the perspective of one of the ORG-B engineering teams. They identified their own team, a UX design team, the team's management, the local product team, the global product team, the business unit they are part of, and the company. The “management” referred to in

Figure 6-28 (b) is the same management team that created the impediment impact diagram in Figure 6-28 (a).

ORG-F described the impact of their top four impediments:

- **Agile practices.** Many items impact just productivity. Some are a balance of productivity, quality, and code/design/architecture. Of all the impediment categories, agile practices are seen to have the highest negative impact on customer *quality*.
- **People.** Impact is dominated by a balance of *productivity* and *code/design/architecture*, with some impact to *customer quality*. Some experiences reflect major *productivity* hits, with some impact to *customer quality*. Some experiences reflect an impact just to *productivity*.
- **Multi-site.** Impact is dominated by a balance of *productivity* and *code/design/architecture*, with some impact to *customer quality*. Some experiences reflect major *productivity* hits, with some impact to *customer quality*. Some experiences reflect an impact just to *productivity*.
- **External dependencies.** Dominated by impact on a balance of productivity and code/design/architecture, with some impact to customer quality. Some items are major productivity hits, with some impact to customer quality.

The findings show the top four impediment categories from ORG-F impacting the organization in terms of *profit*, *morale*, and *time*. First, when it comes to impediments categorized as *people issues*, the dominant cost is on *morale*. There is also some impact on *time*, but this is dominated by cost to *morale*. People issues are not perceived as having a high direct impact on *profit*. Second, there are two dominant patterns for impediments related to *multi-site issues*. Cost here is (a) a balance of *morale* and *time*, and (b) *time* and *money*, with (c) some citing an almost equal balance of *time*, *morale*, *money*. Third, for impediments related to *external dependencies*, there are several impediments that impact a balance of *morale* and *time*, with several others impacting an equal balance of all three. Fourth, for impediments

related to *agile practices* there is a dominant pattern that indicates that all three costs are impacted relatively equally. A small number of experiences indicate *time* as the dominant factor. Some experiences cited *morale* as the dominant cost with some impact on *time*.

The findings show that the impact of impediments can be confined to a single team, can impact multiple or all teams across a program or product, or can impact the entire organization. The impact can also be to some combination of these three. The following observations relate to the impact of the impediments in ORG-F:

- The dominant impediments impacting a balance of team and organization are *multi-site* and *external dependencies*.
- There is a significant cluster of impediments where the impact is split evenly between the program/product, and team. The dominant impediment categories here are *external dependencies*, followed by *agile practices*. There is one *people* and one *multi-site* impediment represented here.
- There is a significant cluster of impediments that indicates that there are impediments that impact more-or-less equally at all three levels of scope. These are mostly agile practices and multi-site, with some *external dependencies*, and two *people issue* impediments.
- There were only two instances where impediments were identified at the program/product level as being dominant. These were a *multi-site* impediment and an *agile practice* impediment.
- There was only one instance where impediments at the parent organization level (i.e., BU2) were dominant, and this was a *people issue* impediment.
- For impediments that are largely impacting a team, these are dominated by *people issues* and to a lesser degree, but still significant, by *agile practices* impediments.

Avoidable and Unavoidable Impediments

Some impediments are avoidable, while others are unavoidable. As discussed in section 2.5, Womack and Jones (2003) distinguish between two types of impediments. They call these “Type One” and “Type Two”. Type One are impediments that “*create no value, but that are unavoidable with current technologies and production assets*”; Type Two impediments create no value but are immediately avoidable. For Type One impediments, the findings in section 6.3.2 suggest that it is worth distinguishing between technologies and production assets that the organization does not yet possess, versus technologies and production assets that do not yet exist. Therefore, this study proposes to expand the definition by Womack and Jones (2003) to distinguish between two subtypes of Type One impediments. Type 1(a) are impediments that are unavoidable *with current technologies and production assets because alternatives do not yet exist*.

An example of a Type 1(a) impediment faced by ORG-B (section 6.3.2) was a delay due to a dependency on a third-party hardware manufacturer. The third party was developing a new hardware chipset and board that ORG-B needed for the next release of their product. At the time the impediment was identified, the hardware was still under development. Type 1(b) are impediments that are unavoidable *with current technologies and production assets in use in the organization, but alternatives exist*. There are several examples of Type 1(b) impediments in the findings in section 6.3.3, where organizations sought to address the impediments by updating their technologies and production assets. Of these, three examples serve to illustrate the point. In the first example, ORG-A faced 5-day *delays* in setting up and configuring development environments. This *delay* was partly due to the technology and development pipeline assets that they were using. The *delay* was, at least in part, unavoidable with their current technologies and production assets. They chose to address what they perceived as the underlying causes (discussed in section 6.3.4) by updating their build pipeline, creating a new automation strategy, and embracing a DevOps approach. In the second example, the ORG-A chose to address 90-

minute build *delays*. Similarly, ORG-B (see section 6.3.4) decided to improve their CI tooling and infrastructure. In the third example, findings show that customer satisfaction was very low due to “*project delays, technical debt, quality issues, firefighting, crisis management*” and also the “*fear of upgrade due to lack of confidence in product quality and stability*”. In all these examples, technology solutions to address the impediments were available in the industry, just not yet available or in use in the organization. So, while the impediments may have been unavoidable in the moment, they were addressable. This distinction between Type 1(a), Type 1(b), and Type 2 impediments is relevant in the upcoming discussion on how organizations manage impediments.

Roles Involved in Impediments

ORG-C sought to make sense of impediments along two dimensions, as shown in Figure 6-29. The vertical dimension represents the impact of the impediment. It is a continuum from low to high, with the lowest impact at the bottom of the vertical axis, and the highest at the top. The horizontal dimension represents the different stakeholder groups that ORG-C needs support from in order to resolve the impediments.



Figure 6-29 Understanding who needs to be involved to resolve impediments

“90% of our problems are within our control and caused by us! We’re doing this to ourselves!” This was the realization of one engineering manager when discussing the poster in Figure 6-29.

It can be helpful to know what roles were involved in the experiences that were captured through sensemaking. By understanding the role involved in each experience it is possible to map the experiences to roles associated with different types of impediments. Knowing who is involved in the various impediments gives some insight into who needs to be involved to resolve the impediment. Figure 6-30 shows the main roles involved in BU1 experiences where impediments and contributing factors were also present.

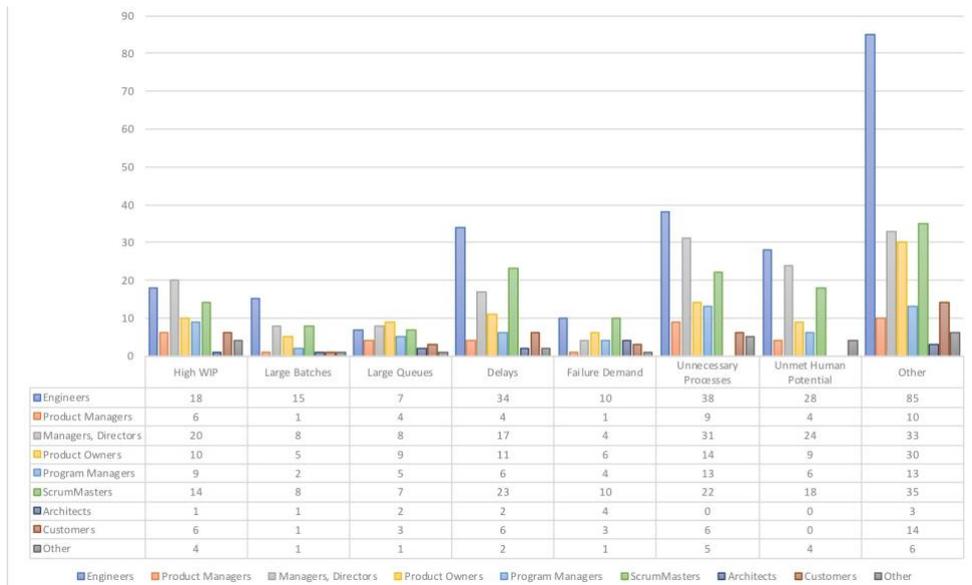


Figure 6-30 Roles in experiences involving impediments and contributing factors

Figure 6-31 shows the roles involved in positive and strongly positive experiences of people in BU2.

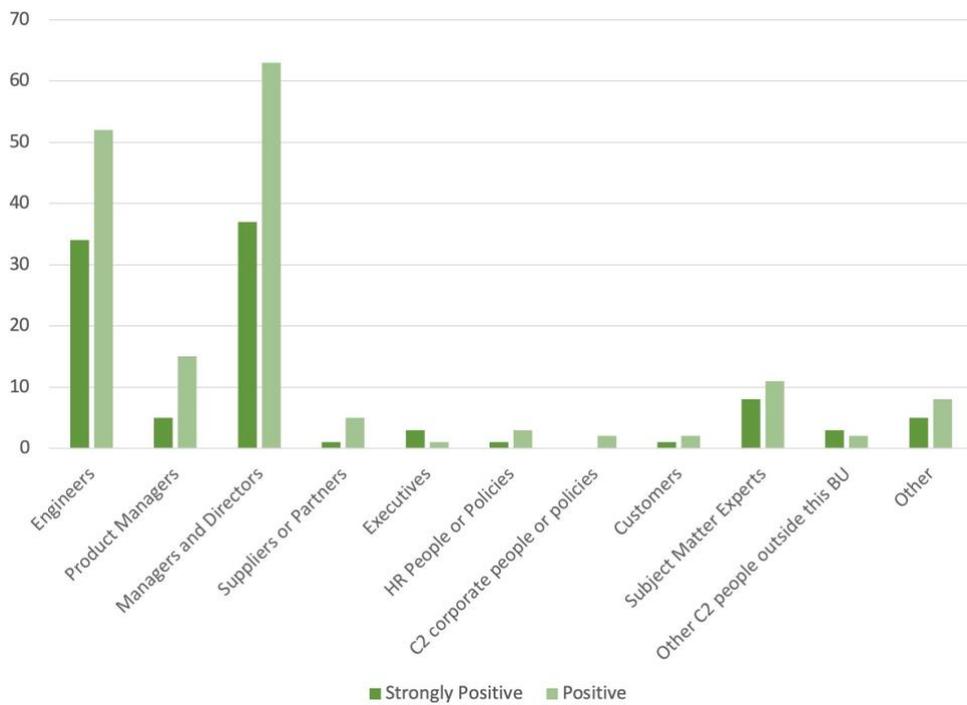


Figure 6-31 Roles involved in positive and strongly positive experiences

Figure 6-32 shows the roles involved in negative and strongly negative experiences of people in BU2.

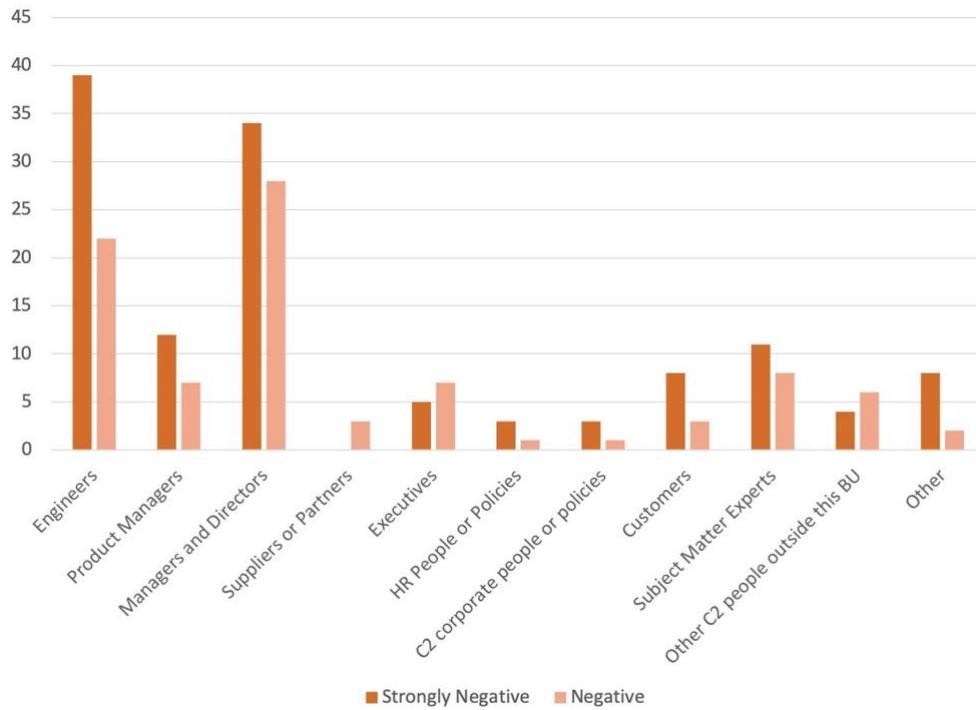


Figure 6-32 Roles involved in negative and strongly negative experiences

Figure 6-33 shows the roles involved in the experiences of people in BU3 where their experiences included impediments. For each role, Figure 6-33 shows the emotional sentiment associated with the experience. Engineers and product owners are associated with the majority of experiences where there are both negative and positive sentiment.

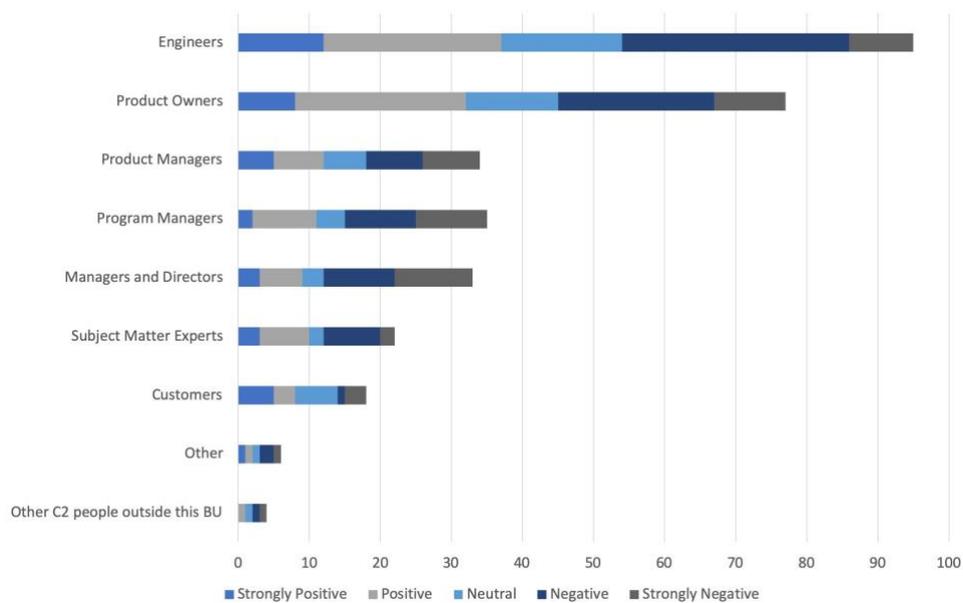


Figure 6-33 Emotion of experiences involving different roles

Context of Impediments

Section 4.3.6 describes the Cynefin sensemaking framework, and how it is used to determine the context of impediments. The findings show 19 impediments from ORG-E, identified from the end-to-end value stream map shown in Figure 6-3. These 19 impediments map to the domains of the Cynefin sensemaking framework. 8 impediments map to the *obvious* domain. For each of these impediments, ORG-E created one or more actions. These include creating shorter sprints, replacing manual system integration activities with CI, combining some meetings, having a dedicated PO for a specific team, adding a tester to a scrum team, and initiating a practice of system level demos. 5 impediments map to the *complicated* domain. Based on the value stream visualization, ORG-E decided to redefine the system integration and test activity to be simpler, and to reduce the “*handoff burden*”. They also decided to measure throughput for the reduced sprint duration. They had not been previously using throughput as a metric. Six impediments map to the *complex* domain. These included dedicated POs, reducing handoffs, an agile approach with system integration testing, definition of done, and shorter sprints. No impediments in this case map to the *chaotic* or *disorder* domains.

ORG-G had a number of impediments for three categories (*organization*, *customers*, and *process*) that were preventing them from achieving a smooth flow and impacting ORG-G business objectives. Figure 6-34 shows a Cynefin framework used to map the ORG-G impediments in order to determine an appropriate course of action for each impediment.

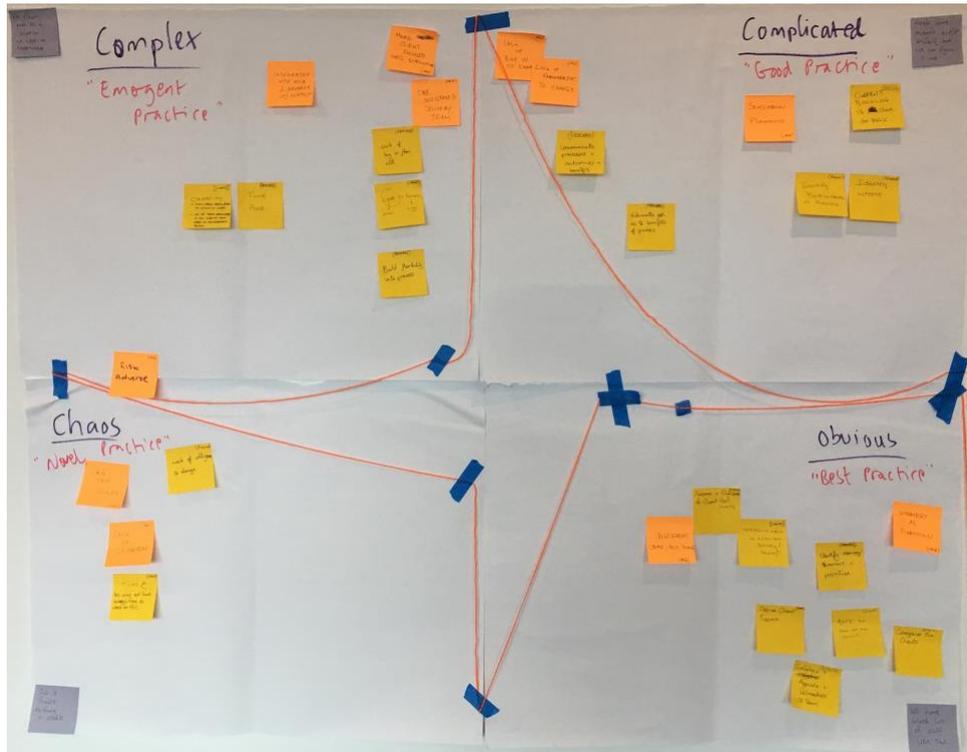


Figure 6-34 Making sense of impediments using Cynefin domain mapping

ORG-G categorized the impediments that emerged through the sensemaking activities into *organization*, *process*, and *customers*. Table 6.5 shows a summary of the impediments from ORG-G, highlighting the number of impediments in each Cynefin domain. For each Cynefin domain, Table 6.5 shows the number of impediments in each of these categories.

Table 6.5 Summary of impediment categories that map to Cynefin domains

Domain	Total Number of Impediments per Domain	Impediment Category	Number of Impediments per Category in Each Domain
Obvious	9	Organization	2
		Process	1
		Customers	6
Complicated	8	Organization	3
		Process	4
		Customers	1
Complex	9	Organization	4
		Process	3
		Customers	2
Chaotic	4	Organization	2
		Process	1
		Customers	1

6.3.4 RO2(iv): Explore Approaches to Resolving Impediments

Prioritization and Focus

ORG-F prioritized their impediments based on the priority they felt would have the biggest immediate benefits for the organization. The result of the prioritization is shown in Table 6.6, showing the number of impediments per category.

Table 6.6 Prioritization of categories of impediments

Category	# Impediments	Votes
External Dependencies	5	8
Multisite	3	7
People	7	6
Agile Practices	11	5
Impediment Removal	3	3
Management and Program	3	3
Value / Purpose	5	1
Roles	2	0

Based on this prioritization, ORG-F decided to focus on the top four impediments. These are *external dependencies*, *multisite*, *people*, and *agile practices*.

An artifact analysis of an *impediment matrix* (described in section 2.2.1) from ORG-A provides additional context for the impediments by situating the impediments in the broader context of the ORG-A organization and its stakeholders. The *impediment matrix* describes the relative importance of each impediment, the affected stakeholders, and the value of addressing the impediment. It considers timing by including a trigger date, after which addressing the impediment will be less valuable or impactful, in some cases because the opportunity will be missed, in other cases because the damage will be irreversible and there will be new problems to deal with. For each impediment, the impediment matrix articulates why the organization should invest in dealing with this impediment versus the others, and why the organization should invest in removing this impediment versus all the other work the organization could invest in. For example, ORG-A decided that tackling impediments related to build times was particularly important. Slow build times were resulting in a high loss of person hours for development and QA, directly affecting 40 developers and 27 QA. The justification for addressing this impediment versus the others is that it would result in better feedback, less time lost, more frequent commits, faster go-live, and more development throughput. The reason that ORG-A should invest in addressing this impediment versus all the other work it could do is that addressing this one would give more time to the work on other projects.

Impediment-Resolving Approach

This section presents findings related to four specific approaches used by the organizations in this study in their efforts to resolve impediments.

Continuous Improvement Goals

The findings show evidence of ORG-A using *continuous improvement goals*, as described in section 2.2, to address their impediments. An artifact analysis shows that ORG-A followed a straightforward template structure that includes a title, the current condition, the actions taken, the effect, the owner or person taking responsibility, and the date. Figure 6-35 shows examples of *continuous improvement goals* from ORG-A to record their

impediments, actions taken to remove the impediments, and the effect of removing the impediments.

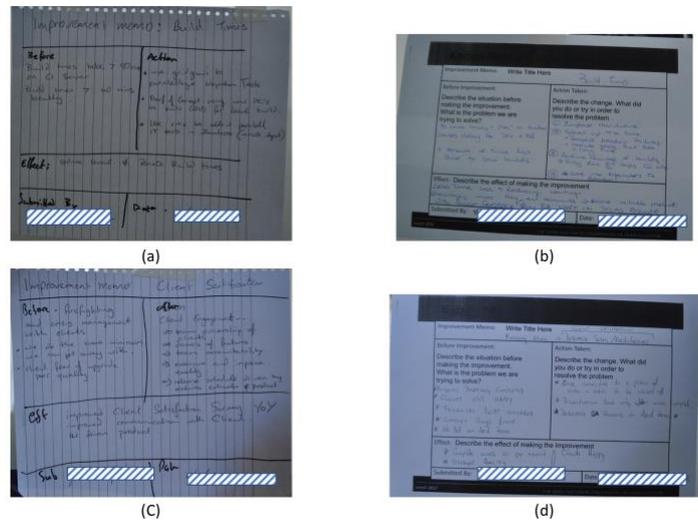


Figure 6-35 Artifacts used to record continuous improvement goals

Table 6.7 shows a summary of the impediments to flow identified in ORG-A, with the number of *continuous improvement goals* created to address those impediments.

Table 6.7 Using continuous improvement goals to resolve impediments

Impediment	Number of Continuous Improvement Goals
Build Times	3
Customer Regression Testing	1
Customer Satisfaction	3
Communication Tools	1
Delay in Setting Up System Environments	2
Jira Workflow	2
Manual Regression	1
Onboarding New Employees	1
Test Automation	1
Triaging Invalid Bugs	1
Reacting to localized issues versus fixing underlying causes	1

The details of the impediments, their contributing factors, and their effects are documented in sections 6.2.1, 6.2.2, and 6.2.3, respectively. Serving as illustrative examples, an analysis of a *continuous improvement goal* to improve the “*Manual Regression*” impediment reveals the following:

- **Current Condition.** The current condition shows that the cost of manual regression to ORG-A is in the region of 500 person-days per release.
- **Actions.** To address this impediment, ORG-A needed to build a test suite per customer, and work with each customer to take ownership of their specific test suite. They needed to decide on an automation method, create a plan with estimates, and invest in staffing the test automation effort.
- **Effect.** The effects of addressing this impediment include reducing the cost of regression from the 500 person-day days per release, improving quality of deliverables, and improving speed of delivery (better responsiveness, lower cycle time). It results in fewer branches, which leads to lower maintenance costs.

An analysis of three *continuous improvement goals* to improve the “*Customer Satisfaction*” impediments reveals the following:

- **Current Condition.** The current condition shows customers are unhappy, with an average customer satisfaction being 3.2/10. The findings note partially completed projects, technical debt, complex defects, firefighting and crisis management with customers, and customers being afraid to upgrade due to poor quality.
- **Actions.** To address this impediment, the findings show that ORG-A approached the problem in four different ways. First, they sought to improve communications and engagement with customers. Second, from a technology perspective, they invested in automating customer test suites and in speeding up the time to go live with new releases. Third, they created dedicated customer teams that would take ownership of the relationship with specific customers. Fourth, they took steps to improve their planning and management processes in order to better manage expectations and priorities, to improve quality and their release process.
- **Effect.** The effects of addressing this impediment included happier customers and an improved customer satisfaction rating, which they

sought to continually improve year-on-year. The findings show that ORG-A also cite an increased quality, better predictability, and improved customer communications.

A3 Problem Solving Reports

Section 6.3.2 lists the top 11 impediments impacting flow in ORG-A, with build times as the highest priority impediment. ORG-A used A3 problem solving (described in section 2.2.1) to address several impediments. An artifact analysis of an A3 problem solving report from ORG-A designed to resolve impediments related to build times reveals the following information:

- **Background:** The background to the build times impediments include delays of up to 90 minutes per check-in, and restricted check-in windows when engineers could check code in to version control.
- **Current condition:** The current condition they faced included delays to QA, a slow turnaround of development, and an overall delayed feedback loop.
- **Goal:** The goal or target condition they want to achieve was one where the build and test took less than 10 minutes. They sought to reach this goal within 6 months.
- **Root Cause Analysis:** They had performed a root cause analysis for these impediments that show product complexity, architecture complexity, and lack of automation. They also found that code is too tightly coupled and that it was difficult to write unit tests. They noted a propagation of heavyweight integration tests, because it was easier for them to write integration tests instead of unit tests. The ORG-A A3 report also noted that the organization had constraints around time and staffing.
- **Countermeasures:** The findings show evidence of a number of countermeasures considered to achieve the goal. These include developing a “hit parade” of priority integration tests, project decomposition (reducing batch size of projects), and training on TDD, mock objects, and unit testing. The potential countermeasures also

included the parallel execution of integration tests, the segregating long-running integration tests, the removal of a particular database switch, the use of an in-memory database, and the decomposition of unit tests.

- **Action plan:** The A3 defined a number of actions, including assigning an engineer to parallelize integration tests, allocating more time to project documentation, arranging internal training on “*TDD + How to write unit tests with Mocks*”, and assigning specific people to identify and rewrite heavyweight integration tests. They developed a plan for these actions, with owners and assigned staff. They planned a series of monthly checks to monitor if they are on track to meet the target goals.

An ORG-A A3 designed to resolve impediments related to manual regression testing includes the following information:

- **Background:** There is a “*huge resource overhead*” in manual regression testing. Manual regression testing is expensive and slow. It results in poor quality and poor customer satisfaction. ORG-A does not have a full regression testing capability with this manual process.
- **Current condition:** Manual regression testing currently takes 114 person-days of effort. In one case it is taking a customer 5 days to upgrade to a newer version of the product.
- **Goal:** ORG-A defined two related goals. First, trunk should always be releasable. Second, regression testing should take no more than 10 days.
- **Root Cause Analysis:** A root cause analysis shows defect fixing and regression testing are done in parallel. They are never able to do a full clean regression. They can never fully quantify the quality of their products. Developing features on multiple branches makes the situation more difficult.
- **Countermeasures:** The countermeasures identified include requiring just one merge to trunk and reducing the regression test cycles.
- **Action plan:** This A3 defined a number of actions, including a commitment that C1 will begin to form a team that is dedicated to this project, defining the structure of the team, and assigning dedicated management with responsibility to deliver. They will improve customer

engagement and develop a better regression testing model with their customers. They will be responsible for a detailed plan of action, developed and owned by the new dedicated team.

The findings show further evidence of A3 problem solving reports used to resolve impediments, this time in ORG-B. The impediments addressed in ORG-B using A3 problem solving were titled

- **Visibility and Transparency.** They did not have good visibility on the status of work and were getting surprised at the end of sprints and releases. Problems were not getting surfaced quickly enough. They did not have a clear picture of the overall project status. They did not have clear visibility on dependencies between teams.
- **Definition of Done.** They did not have a common DoD. Teams were not consistently using DoD.
- **Testing.** There was insufficient testing taking place.
- **Integrated Full-Stack Builds with Test Results.** They did not have automated test runs and did not have fully integrated testing of their entire product stack.
- **Dispersed Teams.** People on the same team were situated on different floors. This was resulting in communication issues, and delays in getting agreements on decisions. Compounding this, it was taking too long to get an agreement on a place where teams could sit together. Despite earlier efforts, they had not been receiving any feedback about the moving process.

The findings show evidence of A3 problem solving used to address a variety of impediments in ORG-C, with topics including *architect role in feature lifecycle, architecture, complex end-to-end system tests, dependencies, deployment support, definition of done, improving delivery, multiple teams in the same codebase, non-functional requirements, portfolio management, roles, and teams.*

Designing Experiments to Resolve Complex Impediments

Section 4.4.3 discusses the role of experiment design in dealing with complex impediments. This section presents findings that show examples of organizations using *experiment design* to resolve *complex* impediments affecting the flow in their organizations. The findings show a number of ORG-E impediments mapped to the *complex* domain. ORG-E designed experiments to address their impediments in the *complex* domain. ORG-E designed an experiment titled “#Less4More” to resolve impediments related to improving throughput. This includes the following information:

- **Actions:** Team [X] will adopt a 2-week sprint, measure throughput (in terms of user story count per sprint), publish results to the wider ORG-E, and establish baselines. The initial baselines are average defect count and sprint ceremonies, and compare with previous iterations.
- **Expected Signs of Success:** If the experiment were proving successful, ORG-E expected to see improved throughput, prioritization based on value, a faster feedback loop, quicker course correction, and happy Scrum teams.
- **Amplification Actions:** To amplify the signs of success, their plan was to celebrate, to expand the experiment to more teams, and to work on bottlenecks and overhead in order to reduce their impact.
- **Possible Signs of Failure:** If the experiment was not having the desired effect, ORG-E expected that they would see reduced throughput, unhappy teams, reduced quality, increased defect counts, increased time spent on ceremonies, and a sprint output that was not demonstratable.
- **Dampening Actions:** To counter the signs of failure, their plan was to have better User Story definition, analyze why defects increased, work on bottlenecks and overhead, reduce available capacity, and have a company-funded team outing.
- **Why this Experiment is Coherent with the Goal of improving Throughput:** Findings show that ORG-E felt that this experiment would help them understand whether their approach to agile, automation, and the knowledge base has improved, and if these were

enabling shorter sprints. This experiment should contribute to evaluating bottlenecks and overhead that can be changed, reduced, or removed, enabling smoother and faster flow. They wanted to schedule distractions into sprints, or at least allow capacity for unforeseen distractions, and allow 2 weeks to address issues. Finally, their hypothesis was that this will enable faster integration with other development teams, and with other teams in the value stream including system integration and test.

Another ORG-E experiment, this one titled “*Mission Possible*”, was also designed to resolve impediments related to improving throughput. It includes the following information:

- **Actions:** Create an adaptive, controlled, working solution reference implementation for the next release of their system using a combination of automated installation, manual testing, and automated test suites. They listed 10 specific subsystems that need to be integrated.
- **Expected Signs of Success:** Encrypted Feature Set X is working and serves as an example for other subsystems.
- **Amplification Actions:** Port to solutions automation system, solution customer system, and solution sandbox.
- **Possible Signs of Failure:** Feature V does not work; Device X does not boot; Processes will not start; No Advanced Services.
- **Dampening Actions:** Roll back component to known good state. Troubleshooting with development teams. Retrospective / post mortem. Automatic failure notifications.
- **Why this Experiment is Coherent with the Goal of improving Throughput:** Any component can be easily replaced so the solution can be tested end-to-end. Instances can be added as new components become available.

The findings show further evidence of experiment design used in ORG-E to resolve impediments. Examples, with experiment titles used, include:

- **Title: “*We are Done with CI!*”** This experiment’s actions included identifying a team to volunteer to demonstrate that their code is

“deeply” tested; create sprint backlog items with acceptance criteria; provide links to test reports; and collaborate with the solutions integration teams to improve flow. ORG-E noted that this experiment is coherent with the goals of improving productivity because, by starting small with one team, it provides a foundation to prove to themselves that ORG-E is on track to improve overall flow. It also proves that they can define and be consistent with a baseline DoD that includes getting work through their CI pipeline.

- **Title: “*Information Radiator*”.** This experiment’s actions included continuously displaying data that ORG-E already collect for all teams in this current sprint. This includes Cumulative Flow. Do this for 2 sprints and then discuss with the development teams. Try this in two separate locations. ORG-E noted that this experiment is coherent with their goals of improving productivity because it “*helps identify unsolved opportunities from sprint to sprint*”, and it focuses on the “shape” of the metrics rather than numbers.
- **Title: “*X the SITEC*”.** This experiment’s actions included providing release notes that are automatically generated by the system; eliminating the requirement to associate developer version control commits with defect tracking system IDs; and integrating user stories and defects as part of the release notes. ORG-E noted that this experiment is coherent with the goals of improving productivity because it enables shorter release cycles and increases throughput (less time manually preparing release notes), it minimizes human error, the test teams get better information with lower overhead, and ORG-E can provide custom drops of the system as needed with improved quality and a faster cycle time.
- **Title: “*Ignite CI*”.** This experiment’s actions included moving system test into the development sprints, rather than be a downstream activity by adding at least one system test engineer into the development team for one subsystem; eliminate system integration entrance criteria; the system must pass system test criteria before sprint is done. ORG-E noted that this experiment is coherent with the goals of improving productivity

because it reduces handoffs and contributes to solving the problem of not having a shippable product at the end of a release cycle.

The findings show evidence of experiment design used to address a variety of impediments in ORG-D, with experiment titles including *CI*, *Clarity on Strategy*, *Collaboration Across Sites*, *Governance for Product and Custom Offerings*, *Interpretation of Strategy*, *Job Rotation*, *Job Uncertainty*, *Lack of Specialization in CI*, *No Roadmap*, *Not Enough Variation in Work*, *Salary Pay Parity Issue*, *Synch Up Teams with Overall Strategy*, *Vision and Strategy*, *Where Do I Belong?*, and *Work Content*. ORG-D designed each of these experiments with the goal of improving overall flow and throughput through the organization. Some experiments targeted the development pipeline, some addressed strategy and business, and some addressed cultural issues of collaboration, personal identity and belonging within the organization. These findings are illustrative of tackling problems in complex systems *obliquely* (Kay, 2011, Miller and Page, 2007), as discussed in section 4.2.3.

Defining System Improvement Goals to Resolve Impediments

The value stream for ORG-A is shown in Figure 6-2. The issues raised range from technology infrastructure, to how they engage with customers. Examples from the data include:

- Need organization-wide support for investing in technology infrastructure to improve build times, test automation, and development environment setup.
- Use the output of the value stream mapping exercise to reconcile their SDLC and Jira workflows requires company-wide support and follow-up.
- Improve their onboarding for new hires at each site across the company.
- Formalize the creation of a “Design Authority” function in the organization, with responsibility for technical governance and risk assessment.

- Create a DevOps adoption plan that helps them better manage their capacity, and results in better visibility and predictability of release status.
- Engage or re-engaging with customers on a collaborative approach to test automation.

Table 6.8 shows five objectives to remove the impediments in ORG-F, and thereby improve flow.

Table 6.8 System improvement goals to resolve ORG-F impediments

System Improvement Goal	Description
Understand and visualize the flow of value to customers and users	The team does not currently understand and cannot articulate how value flows through their teams and organizations to customers, and from there to end users. The premise here is that generating that understanding will provide a better focus for teams, more engagement, better motivation, and help people identify how to improve the flow of value.
Measure Team Happiness across the program	The premise here is that happier teams will deliver better value to customers, the product will be more reliable, and then they will have happier customers. For all teams in the program, find a way to easily measure team happiness.
Communicate what capabilities are enabled at the end of each Sprint and Release	Right now, demos are a "show and tell" of what work has been done, not what value is being created. Instead, have product managers and/or product owners communicate what new capabilities the team has enabled for their customer(s) in the sprint and/or release.
Communicate cost and revenue	The premise here is that teams will benefit from understanding the revenue model associated with the products they are building. They currently do not know how the product makes money for their company, or how their customers make money from the product.
Connect teams to customers	The premise here is that bringing teams closer to customers will help with motivating teams and will help teams understand customers better. This in turn will help with delivering value to customers. From a flow perspective, the right things will flow to customers.

ORG-G created a set of shared system improvement goals to resolve their impediments, as shown in Table 6.9. For each goal, they noted which category of impediments they felt would influence achieving the objective.

Table 6.9 System improvement goals to resolve ORG-G impediments

Goal	Organization	Process	Customers
Automate customer regression test cases		✓	✓
Develop a management-approved capacity plan for the coming 12 months	✓		✓
Get a baseline on current processes and process owners	✓	✓	
Foster better cross-team alignment	✓	✓	✓
Create better shared context and understanding	✓	✓	✓
Get better insight on customers; improve overall customer satisfaction		✓	✓
Better alignment between Business Analysts, Product Management, and Engineering Teams	✓	✓	✓
Engineering teams have better understanding of customer's context	✓		✓

As can be seen from the mapping in Table 6.9, ORG-G expressed that resolving impediments under all three categories of *organization*, *process*, and *customers* would lead to achieving their goals of fostering better cross-team alignment, creating better shared context and understanding, and better alignment between the different functional groups of business analysts, product management, and engineering. Stated from another perspective, achieving these goals will help resolve the impediments and improve overall flow quality in ORG-G.

Interventions in the System

The CDE (*container, difference, exchange*) framework (Eoyang, 2001), as described in section 4.2.2, serves to frame the interventions employed by organizations in this study to resolve their impediments. An analysis of the interventions from this perspective provides insight into what changes organizations make in order to resolve impediments and improve flow. Table 6.10 shows the specific *container, difference, and exchange* interventions from the findings that map to the *complicated* domain of the Cynefin sensemaking framework (section 4.3.6). The 'ID' columns note the specific impediment that is being addressed via the intervention. E.g., 'A12' is impediment number 12 from ORG-A.

Table 6.10 shows *container* interventions from the findings that map to the *complicated* sensemaking domain. The *container* interventions from this study that address impediments in the *complicated* domain include creating

plans, commitments, test suites, development environment, and creating or improving process definitions.

Table 6.10 Container interventions in the complicated domain

ID	Container Intervention
A01	Create parallelised tests
A01	Develop a plan for reducing build times
A11	Put in place a strategy for deployment of builds
A14	Create a dedicated regression test suite per customer
A14	Decide on automation method
A14	Plan and estimate work to created dedicated regression tests per customer
A14	Reduce the number of version control branches
A16	Investigate and adopt Technical Complexity metrics
A17	Create DevOps adoption plan
A17	Manage and understand existing capacity
A17	Group environments (by model / department)
A17	Size environments appropriately
A19	Better triage process
B02	Agree a common Definition of Done
B03	Commit to delivering fewer stories but with higher quality
B04	Create integrated full-stack builds, with published test results
F03	Communication of product capabilities enabled at end of each Sprint and Release
G01	Automation of customer regression test cases
G02	Develop a management-approved capacity plan for the coming 12 months
G03	Get a baseline on current processes and process owners

Table 6.11 shows *difference* interventions from the findings that map to the *complicated* sensemaking domain. The *difference* interventions from this study that address impediments in the *complicated* domain include altering team membership, creating a buddy system to help with onboarding, providing more information to teams, and providing training and other actions to improve knowledge in the organization.

Table 6.11 Difference interventions in the complicated domain

ID	Difference Intervention
A01	Add a specific named person from the internal tools team
A02	Add dedicated management to project
A05	Look into alternatives to Bamboo for CI
A11	Add more people to the team
A14	Staff the customer test suites appropriately
A15	Create a Buddy System, with a buddy in the same office
A15	A training and education program for new hires
A16	Populate Design Authority with people from different functions
B02	Remove duplication of work between teams
F04	Provide more information to increase the team's understanding of revenue model of the products they are building

Table 6.12 shows *exchange* interventions from the findings that map to the *complicated* sensemaking domain. The *exchange* interventions from this

study that address impediments in the *complicated* domain include bringing people together to exchange information, improving communications, and improving feedback from the continuous delivery pipeline.

Table 6.12 Exchange interventions in the complicated domain

ID	Exchange Intervention
A01	Monthly checks to see if targets will be met
A02	Come together to create a plan for improving build times
A04	Use "Grid Gain" to parallelize integration tests
A04	Reduce local and remote build times
A05	Run integration tests at night only
A05	Speed up test time
A06	Add technical risks to the status report for customer engagements
A06	Report and act on technical risks
A06	Faster feedback when areas broken
A06	Use automated tools to find technical risks
A07	Automate customer test suite to increase quality
A08	Create an approval step for closing off when committed to a piece of work
A08	Prioritize work before beginning; work from the prioritized list
A10	Improve knowledge base - 1 tool, easily searchable
A12	Change Jira to reflect actual end-to-end flow process
A13	Realtime status / KPI reports
A13	Agree proposed end-to-end workflow
A16	Technical architecture performance / scrubbing refactor
A16	Design Authority provides technical governance and risk assessment
A17	Roll out config management (Puppet, Chef)
A17	Push-button provisioning
A18	Re-engage with customers on automation approach and test collateral
A18	Agree data management and ownership structure for all departments and plans
A18	Planned rollout and agreed ownership across all departments
A19	Training for customers required
A19	Better internal training and handovers
A19	On-site time to build up customer knowledge
A19	More triage analysis before coming to development
B02	Training for SMs
B04	Work with 3rd party dependency to define backlog items
B04	Define a better development environment
F02	Measure Team Happiness across the program
F03	Product Managers to communicate new capabilities at the end of a Sprint and Release
F04	Communicate cost and revenue associated with development efforts
G06	Better insight on customers

Table 6.13 shows *container* interventions from the findings that map to the *complex* sensemaking domain. The *container* interventions from this study that address impediments in the *complex* domain include formation of new containers such as new teams, and increased customer collaboration. The *container* interventions in the *complex* domain also include an emphasis on empowerment, autonomy, shared goals, and improved understanding of context.

Table 6.13 Container interventions in the complex domain

ID	Container Intervention
A02	Create a team structure
A03	Create smaller batch sizes of work going through the build pipeline
A03	Decompose projects into smaller, decoupled units of work
A06	Form a focused team working on specific tasks
A09	Give each team ownership of engagement with specific customers <ul style="list-style-type: none"> - demos of features - team accountability - measure and improve quality - release schedule driven by realistic estimate
A16	Empower an organization function called "Design Authority"
A18	Coherent vision for automation
B01	Shared goals
B05	Create a new team seating space
B05	Team has the autonomy to decide to move seats, within defined areas
F01	Understand and visualize the flow of value to customers and users
G05	Create better shared context and understanding
G08	Engineering teams have better understanding of customer's context

Table 6.14 shows *difference* interventions from the findings that map to the *complex* sensemaking domain. The *difference* interventions from this study that address impediments in the *complex* domain include some themes that are similar to those in the *complicated* domain, including adding people to teams and increasing skills. In addition, *difference* interventions in the *complex* domain include a strong focus on cultural topics such as focus, morale, engagement, and happiness. This is reflective of the findings on culture in section 6.3.1, and the fact that culture was one of the dominant themes that emerged through the system patterns.

Table 6.14 Difference interventions in the complex domain

ID	Difference Intervention
A01	Add someone with automation test skills to the team
A01	Increase skills with TDD
A08	Add QA skills to architecture team
A19	Support Team should be involved in new development cycles
F01	Provide more information to enable better focus
F01	Increase team motivation
F01	Increase team engagement
F01	Increase capability to improve flow
F02	Increase team happiness

Table 6.15 shows *exchange* interventions from the findings that map to the *complex* sensemaking domain. The *exchange* interventions from this study that address impediments in the *complex* domain include a focus on faster feedback, improved responsiveness, and improved interactions with customers. In addition, *exchange* interventions in the *complex* domain

include a focus on improving transparency, visibility, alignment, shared context, and shared or improved understanding.

Table 6.15 Exchange interventions in the complex domain

ID	Exchange Intervention
A02	Improve customer engagement
A04	Reduce build times to take less than 90 minutes on CI server
A04	Reduce build times to take less than 40 minutes locally
A05	Reduce number of builds
A05	Reduce delay in the CI pipeline
A05	Faster feedback from QA
A06	Happy customers
A06	Happy workers
A07	Customer satisfaction survey, improve customer satisfaction
A07	Communicate with customers
A07	Faster turnaround of releases
A08	Improve customer interaction
A09	Improved communication with Customer
A10	Improve knowledge exchange
A11	Increase knowledge in development environment through training
A12	Create transparency across all teams
A14	Improve speed of delivery
A15	Exchange of knowledge between buddies in buddy system
A18	Agree overall strategy across all departments
B01	Transparency and visibility
B01	Agreeing shared goals between developers and managers
B01	Understanding and managing dependencies between teams
B03	End-to-End testing of a single story, fully automated, by end of next Sprint
B03	System Integrators coding or reviewing code with developers in short cycles
B03	Integrator pair programs with developer
B03	PO and SM to work together to reduce number of stories in next Sprint
B05	Change seating so that team can sit together
F01	Multiple roles collaborating to create the value stream map
F05	Bringing teams closer to customers; motivating teams, help understand customers
F05	Closer interaction between development teams and customers
G04	Better cross-team alignment
G05	Create better shared context and understanding
G06	Improve customer satisfaction
G07	Better alignment between Business Analysts, Product Management, Engineering Teams
G08	Engineering teams have better understanding of customer's context

Some points to note from the findings in this section:

- One impediment can be addressed through multiple interventions. This is more common in the findings than organizations needing to do just one thing to address an impediment. For example, ORG-B used three distinct interventions to resolve impediment B04. These include one *container* intervention shown in Table 6.10 (integrated full-stack builds), and two *exchange* interventions shown in Table 6.12 (work with

vendor on 3rd-party dependencies, and define a better development environment).

- Those interventions can be across multiple types for the same impediment. Some combination of *container*, *difference*, and *exchange* interventions are common to address a single impediment. E.g., impediment F01 is addressed through one *container* intervention, four *difference* interventions, and one *exchange* intervention, all in the *complex* domain. This is an example of what Snowden and Boone (2007) suggest in relation to multiple, parallel, safe-to-fail experiments or probes to address problems in the *complex* domain.
- A single impediment can be addressed through multiple interventions across more than one domain. E.g., impediments A01 and A02 have interventions in both the *complex* and *complicated* domains. ORG-A used six distinct interventions to resolve impediment A01. First, there are four interventions in the complicated domain: two *container* interventions (parallelized tests, plan for reducing build times), one *difference* intervention (new team member), and one *exchange* intervention (monthly progress review). There are two interventions in the *complex* domain (new team member with specific skills, increase team TDD skills). These findings align with research from Hummelbrunner and Jones (2013), who assert that it is not necessarily helpful to view a situation as entirely in one domain or another, and that the distinctions between domains should be used “*to understand certain aspects of a situation rather than the entire intervention.*” Most situations demonstrate features of multiple domains, while subcomponents of the situation might map to a single domain (Hummelbrunner and Jones, 2013).

An analysis of these findings, summarized in Figure 6-36, shows the proportion of *container* interventions, *difference* interventions, and *exchange* interventions employed by the organizations in this study to resolve impediments, and how those intervention types map to the Cynefin sensemaking domains.

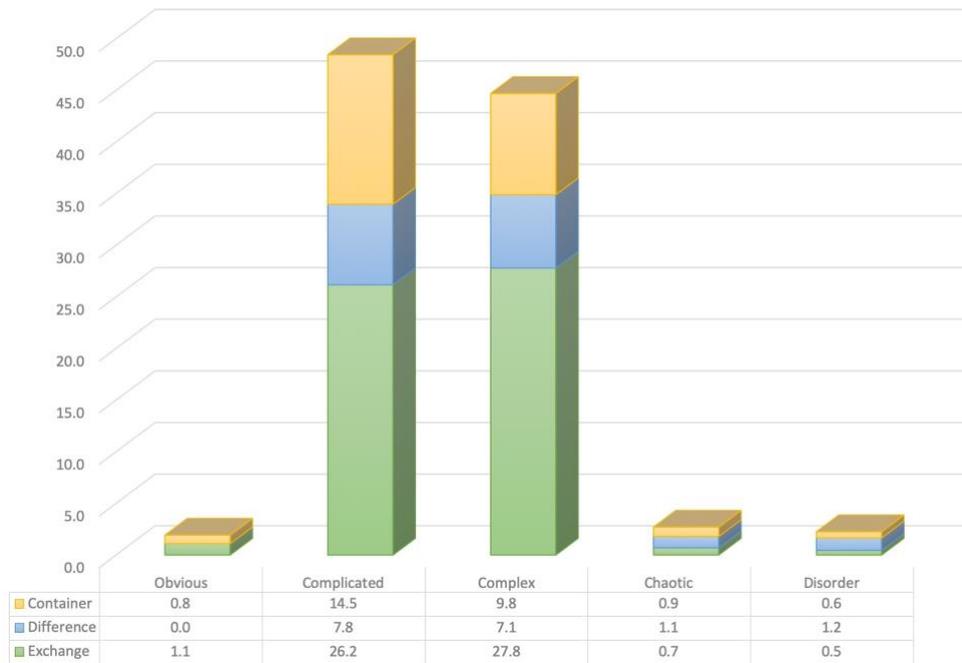


Figure 6-36 CDE interventions per sensemaking domain

Figure 6-36 shows that in this study *exchange* interventions were the most common type of intervention, accounting for approximately 56.3% of interventions. The implication here is that in the majority of cases organizations in this study chose to focus on improving interactions in some way as a means of addressing the underlying factors that lead to impediments to flow. Approximately 26.6% of interventions were *container* interventions. Approximately 17.2% of interventions were *difference* interventions. All of these interventions are designed to resolve impediments and improve flow in the organizations in this study.

Table 6.16 summarizes these relationships between intervention types and sensemaking domains, showing the focus of each intervention type to resolve impediments.

Table 6.16 Focus of CDE interventions in the *complicated* and *complex* domains

Intervention Type	Container	Difference	Exchange
Complicated domain	<ul style="list-style-type: none"> • plans • commitments • test suites • development environment • process definitions 	<ul style="list-style-type: none"> • team membership • buddy system • onboarding • providing more information to teams • training • improve knowledge 	<ul style="list-style-type: none"> • bringing people together to exchange information • introducing or improving communications • improving feedback from the CD pipeline
Complex domain	<ul style="list-style-type: none"> • forming new teams • teams with customers • Collaboration • Empowerment • Autonomy • shared goals • improved understanding of context 	<ul style="list-style-type: none"> • team membership • increasing skills • focus • motivation • engagement • happiness 	<ul style="list-style-type: none"> • faster feedback among people • improved responsiveness • improved interactions with customers • transparency • visibility • alignment • shared context • understanding

For the most part, as can be seen from Table 6.16, interventions in the *complicated* and *complex* domains each address different topics. There is some overlap in topics between both the *complicated* and *complex* domains, e.g., team membership, feedback, and adding new skills. Interventions vary in how they deal with these topics, depending on whether the impediments are in the *complex* domain or the *complicated* domain. Interventions used to resolve impediments in the *obvious* domain share similar patterns. Most include a straightforward action or set of actions to resolve the impediment directly. Examples from the findings include updating a Jira workflow, project decomposition, or creating categories for different types of customers. Interventions used to resolve impediments in the *complicated* domain share a similar pattern. Most include evaluating alternatives, research to acquire knowledge, consulting subject matter experts, trying alternative approaches, or developing expertise. Interventions used to resolve impediments in the *complex* domain share a similar pattern. Most include development of new approaches in a very context-specific way. While the interventions in the *obvious* domain choose from existing solutions, interventions in the *complex* domain develop novel approaches to resolve impediments in the context of the organization. Organizations in this study tackle these problems indirectly. There may not always be a direct surface-level relationship between the action and the original impediment. No other study, as far as the researcher is aware, has taken a complexity

perspective to analyze intervention types to resolve impediments and improve flow in software organizations.

Impediment Refinement

Many of the impediments identified in this study were refined as people worked on making sense of them and designed interventions to resolve them. Some examples from the findings illustrate this point. In the first example, ORG-A identified a set of 90 impediments. Section 6.3.3 shows that they refined these to 11 impediments they felt they needed to address. Each one of these 11 impediments encompassed several of the original 90 impediments that they identified, and which reflected a refined understanding of what they needed to do to resolve impediments. In the second example, ORG-B identified an impediment as insufficient time for testing. As they worked through the A3 problem-solving approach, they realized that the problem they need to solve is the lack of a consistent DoD. In a third example, the system patterns in ORG-G revealed that flow was of poor quality and relatively unstable. ORG-G identified several impediments related to how they work internally, and how they deliver to their customers. For example, WIP was too high, overall flow quality was low, there were significant delays impacting flow, they were dealing with the impact of a series of mergers, and they felt they had operational silos across the organization. Looking at the impact, they realized that their engineering organization did not have sufficient awareness of their business and customer priorities. They moved to talk about having a more customer-focused organization structure. Finally, when it came time to taking action, their focus had further refined, and their focus for immediate action was promoting a better understanding of their customers and improving overall customer satisfaction.

This refinement is natural and to be expected (Weick and Sutcliffe, 2011). Acting is sensemaking (Weick, 1995). Organizations continue to make sense of the impediment even as they act to resolve it. Each step provides new information, which further shapes and refines their understanding. Organizations observed and analyzed patterns about the system in which

they work. Observing the system changes the system, and also changes the probe used to observe the system, a phenomenon referred to as *The Observer Effect* (Gall, 2002). Merely seeing and considering the system patterns and contributing factors starts to influence how people think about the system. They then identified several impediments (section 6.3.2) and went through a series of sensemaking steps to make sense of those impediments (section 6.3.3). This sensemaking generally resulted in changing the description of the impediment. They refined their focus again as they worked through the planning of how they would resolve the impediments. Moreover, at times, their focus and understanding evolved as they worked to resolve the impediments.

6.4 Summary

This chapter presents the findings from this research study. Regarding RO1, this study found evidence of each of the categories of impediments identified in the literature review, and also showed that new categories of impediments specific to each organization can emerge through sensemaking. The findings also show evidence for each of the contributing factors and effects of impediments identified in the literature review.

Next, this chapter presents findings related to RO2. First, the findings for RO2 show examples of identifying and analyzing system patterns through value stream maps, flow metrics, and organization culture.

Second, the findings for RO2 show that organizations can identify impediments by interpreting these system patterns. The findings also demonstrate that impediments often occur with other impediments, i.e., there is rarely just a single impediment, though there might be one or more dominant impediments influencing flow. Some organizations show signs of developing a capability to monitor the system for impediments.

Third, the findings for RO2 show that organizations use multiple ways to make sense of impediments. One way is to understand the impact of impediments. Another is to understand which impediments are avoidable,

and which are unavoidable. Understanding what roles are involved in people's experiences in organizations when impediments are encountered, and then what emotion is associated with those experiences, provides a supplemental perspective to help make sense of impediments. This study explores the use of the Cynefin sensemaking framework to understand the context of impediments, and the type of action that might be appropriate to resolve impediments.

Fourth, the findings for RO2 show ways in which software development organizations resolve impediments. The findings show there are generally more impediments than can reasonably be addressed at one time, so this chapter explored different ways in which organizations prioritize impediments and choose which ones to focus on. There are many different approaches that are possible for resolving impediments. This chapter presents findings related to four approaches from the organizations in this study: *continuous improvement goals*, *A3 problem solving*, *experiment design*, and *system improvement goals*. The findings show that impediments are not generally resolved by a single intervention, but rather through multiple interventions of different types. This chapter used the CDE model to analyze and understand the different intervention types employed by the organizations in this study. Finally, this chapter showed that organizations go through a process of refining impediments as they work on them. This change is often reflective of organizations acquiring a better understanding of the impediments and their context as they move from analyzing system patterns, through identifying, making sense of, and resolving impediments. Next, Chapter 7 presents a discussion of these findings.

Chapter 7 Discussion and Conclusions

7.1 Introduction and Chapter Layout

The purpose of this chapter is to summarize the main aspects of this study. This chapter discusses the findings from Chapter 6 and presents conclusions, contributions, limitations, and directions for future research. As described in section 1.3, the purpose of this study is to understand impediments to flow in large technology organizations, from the perspective of understanding organizations as complex adaptive systems. Section 7.2 presents a discussion of the findings related to RO1. Section 7.3 presents a discussion of the findings related to RO2. Figure 7-1 highlights the section in this chapter that discusses each element of this study’s validated conceptual framework.

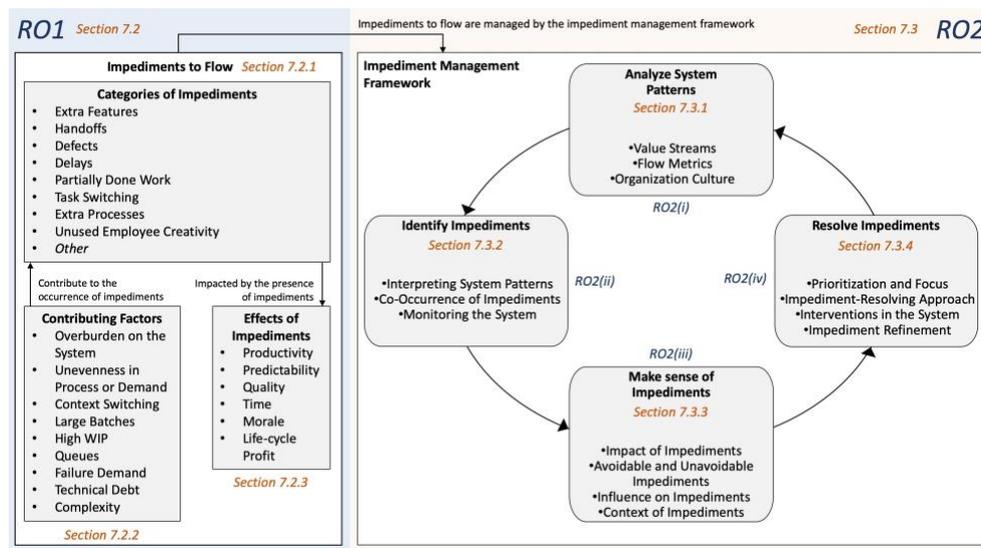


Figure 7-1 Chapter sections relating discussion to the conceptual framework

Section 7.4 summarizes the contributions from this study to research, theory, and practice. Section 7.5 presents the known limitations of this study and the research approach. Finally, section 7.6 discusses trajectories for future research based on this study’s findings.

7.2 RO1: Impediments to Flow

Section 2.5 discusses the literature related to impediments to flow. This section discusses impediments to flow in the context of the literature review and the findings from section 6.2. Discussing impediments (RO1) is a necessary foundation for the rest of the discussion on the impediment management framework (RO2) in section 7.3.

7.2.1 Categories of Impediments

The literature review in section 2.5.1 identifies 8 common categories of impediments. These are *extra features*, *handoffs*, *defects*, *delays*, *partially done work*, *task switching*, *extra processes*, and *unused employee creativity*. This study examined 10 software development organizations (section 5.5) and found evidence for each category of impediments. Figure 7-2 is a hierarchy chart from NVivo that shows a summary of findings from Chapter 6, confirming the presence of these impediments in the organizations studied. Impediments occur in varying proportions. Figure 7-2 shows that *delays*, *extra processes*, *unused employee creativity* are the impediments that are mentioned most frequently throughout the findings.

Delays	Extra Processes	Task Switching	Other	
	Unused Employee Creativity	Handoffs	Extra Features	Defects
			Partially Done Work	

Figure 7-2 Relative proportions of impediments coded in this study's findings

Impediment categories affect organizations to different extents. Some impediment categories are more prevalent than others at different times. Figure 7-3 summarizes the extent to which each category of impediment is present across the organizations studied. This shows a complimentary

perspective on the findings to that provided in Figure 7-2. Where Figure 7-2 shows the proportions of impediments coded across all findings, Figure 7-3 shows how these impediments are present in different organizations.

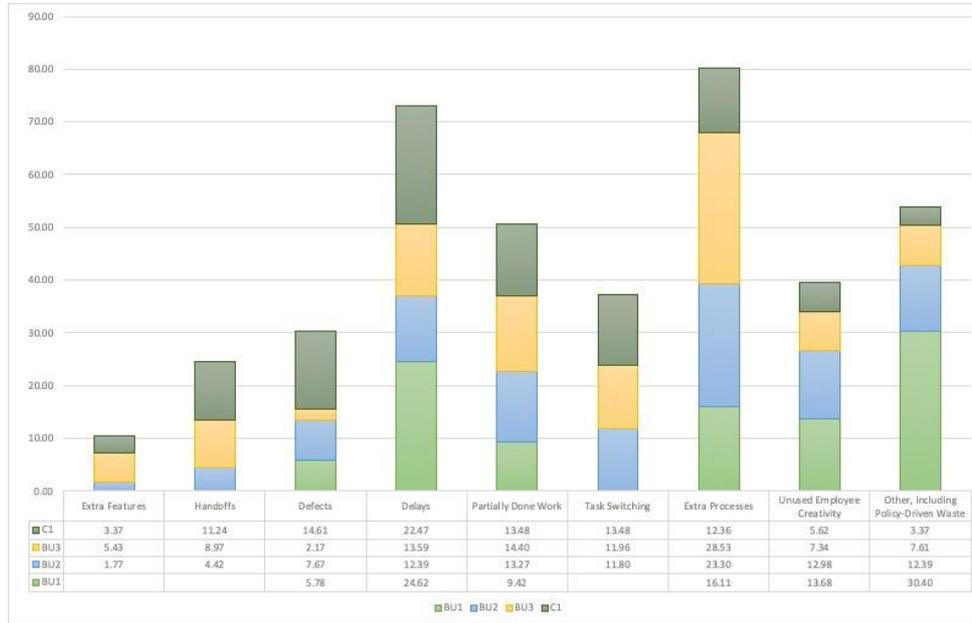


Figure 7-3 Summary of impediment categories impacting flow in 4 organizations

As described in section 2.5, much of the popular literature on lean, flow, and impediments asserts that *extra features* are the most common waste. Despite this assertion, this study found very little evidence of *extra features* being a wide-spread impediment in organizations. This could mean that the organizations studied were not developing *extra features* that their customers did not use, or it could mean that study participants were not aware of *extra features* being a problem. In any case, *extra features* did not feature strongly as an impediment to flow in this study’s findings. The most dominant impediments were *extra processes*, *delays*, and *partially done work*. *Defects*, *task switching*, *handoffs*, and *unused employee creativity* all feature prominently as impediments that impact flow.

People in organizations apply their own labels to help them see impediments. “*Other*” ranks as the third-highest impediment category in Figure 7-3. This is an indicator of at least two things. First, although people could see impediments to flow, at times they had difficulty categorizing them to one of the categories referred to above. Second, participants

identified impediments that they felt did not fit into one of the categories. The categories are intended as a starting point in learning to see impediments, not an exhaustive or definitive list (Ohno, 1988). People can name the impediment categories in a way that suits them. Some categories emerged across multiple organizations in this study. The findings in section 6.3.2 show other impediment categories that participants named. These include *agile practices*, *external dependencies*, *impediment removal*, *multisite*, *people issues*, *roles*, *value/purpose*, *management/program*, *psychological safety*, *fear*, *strategy*, *politics*, *organization*, *processes*, and *customers*. The original nine categories from the literature review are still valid; these “*self-identified*” labels are not intended as a replacement. Rather, this shows examples of people examining flow in the context of their organizations, identifying what they perceive as impediments, and giving names to those categories of impediments that have meaning for them. This is also an example of the distinction between sensemaking and categorization. In a categorization model, the categories precede the data, and data is matched to the categories of impediments. In a sensemaking model, the data precedes the categories, and impediment categories emerge from the data. The findings in Chapter 6 show examples of both.

Impediments are multi-dimensional. It can happen that an impediment fits more than one category. For example, focus group participants in ORG-A identified an impediment that was a recurring situation where they did not have a team assigned to specific work items by the time of their planning milestone, or where they were changing the assigned team afterwards. As shown in section 6.3.2, they classified this particular impediment as one of *extra processes* because of the additional, unnecessary processes involved in identifying work, assigning a team, reassigning the work, and all of the associated status and reporting updates that accompanied this. This same impediment resulted in *handoffs* - in this case the handoff from the originally assigned team to the newly assigned team. It also resulted in significant *delays* - it was taking on average 3 days to complete the handoff. The point here is that categorizing the impediment is not the objective; the objective is not to get a “right” answer about which category the

impediment fits. What is important is that people in organizations see the impediment, understand its impact, and act appropriately to address it. The initial categorization can be useful as a starting point.

7.2.2 Contributing Factors to Impediments

Section 6.2.2 presents findings that show the main contributing factors to impediments discussed by organizations in this study. These findings confirm reports in the existing literature including from Ohno (1988) (*overburden in the system (muri)*, *unevenness in process or demand (mura)*), Reinertsen (2009) (*large batches, context switching, long queues, high WIP*), Seddon (2005) (*failure demand*), and Poppendieck and Poppendieck (2007) (*complexity, technical debt*). Section 2.5.2 describes each of these contributing factors.

A thematic coding analysis of the findings, as described in section 5.7.5, reveals the proportion to which each of these contributing factors were noted in the organizations studied. Figure 7-4 shows a hierarchy chart from NVivo, represented as a tree map, that helps visualize the coding patterns from the coding analysis across all the study's findings, and shows the proportion to which each contributing factor is present in the findings.



Figure 7-4 Contributing factors to impediments to flow in the organizations studied

Recall from section 2.5.2 that *too much WIP* and *large batches* can be considered special cases of *overburden in the system (muri)*. Similarly,

context switching is a symptom of *unevenness in process or demand (mura)*. *Failure demand* can contribute to or result from *overburden* or *unevenness*. The color coding in Figure 7-4 attempts to reflect these relationships. From this, Figure 7-4 shows that *overburden in the system*, including *too much WIP*, *large batches*, and *failure demand* are the biggest contributors to impediments across all the organizations studied. This is followed by *unevenness in process or demand (mura)*, including *context switching*, and *failure demand*. *Technical debt* and *complexity* are also confirmed as significant contributors to impediments in organizations.

Figure 7-4 shows that *high WIP* is the single most noted contributing factor to impediments for organizations in this study. Taken collectively, *high WIP*, *failure demand*, and *large batches* are all forms of *overburden in the system (muri)*, accounting for more than 60% of all contributing factors found in this study. This dominance of *overburden* in the organizations in this study confirms Ohno's observations that *overburden* is the most significant contributor to impediments in organizations (Ohno, 1988). Figure 7-4 confirms that *high WIP* is the most prevalent contributing factor discussed where there are impediments. Reinertsen (2009) shows that reducing *WIP* is one of the most effective ways to improve flow.

These factors could have a different effect in different organizations. While *overburden* is the biggest contributing factor in this study, the proportions could be different in other organizations. They could even be different for these organizations if studied at another time. Understanding that these factors are present and are contributing to the presence of impediments informs how organizations approach resolving impediments (discussed in section 7.3.4).

7.2.3 Effects of Impediments

A thematic coding analysis of the findings, as described in section 5.7.5, reveals the dominant themes of *quality*, *morale*, *time*, *productivity*, *predictability*, *lifecycle profit*, and *code/design/architecture impact*. This study examined multiple teams and organizations and found evidence for

each of these impacts of impediments. The findings in section 6.2.3 present evidence of these impacts of impediments to flow in organizations. Figure 7-5 shows a hierarchy chart from NVivo, represented as a tree map, that helps visualize the patterns from the coding analysis across all the study’s findings.

Quality	Motivation	Productivity	Predictability
	Time	Lifecycle Profit	
		Code, Design, Architecture	

Figure 7-5 Impacts of impediments to flow in the organizations studied

Figure 7-5 shows that the biggest reported impact of impediments across all organizations is to *quality*, followed by *morale* and *time*. These are followed by impacts to *productivity*, *predictability*, *lifecycle profit* and *code/design/architecture*.

7.3 RO2: A Framework for Managing Impediments

This section discusses the findings for RO2, with one subsection discussing each of the four sub-objectives.

7.3.1 RO2(i): Analyze System Patterns

Addressing RO2(i), this section discusses the findings related to analyzing the system patterns that influence impediments to flow in organizations. Section 6.3.1 presents the findings from the study that identify the dominant system patterns observed in this study. These coded patterns were categorized based on the approach described in section 5.7.5 (Saldana, 2016). There are many ways to observe and analyze patterns in systems, of which this study adopted three. These are value streams (section 2.2.2), flow

metrics (section 3.3), and organization culture (chapter 4.2.6). The conceptual framework in Figure 7-1 above highlights these.

Each of these three methods provides a particular perspective on the system where impediments are occurring. First, value stream analysis shows patterns of how organizations deliver value to their customers (Womack, 2006). For example, value stream maps show what states the work goes through, and where delays are happening within or between those states. Second, flow metrics analysis shows patterns of how organizations work to get things done (Petersen and Wohlin, 2010). For example, *throughput* shows patterns in how much organizations deliver to their customers, or how responsive they are to customer needs. Third, analysis of organization culture shows patterns that influence the organization, and that permeates everything they do (Weick et al., 2005). These patterns shape flow and impediments in the organization.

In order of most commonly mentioned or discussed in the findings from Chapter 6, the themes of patterns that emerged are *technology*, *culture*, *processes*, *organization*, *strategy*, *team*, *product*, *customer*, *tools*, *people*, and *architecture & design*. Figure 7-6 is a hierarchy chart created from NVivo that visually represents the categories of patterns that emerged from the findings in Chapter 6. There are contributing factors in the organizations under each of these headings that are relevant to flow and impediments. Sections 7.3.2, 7.3.3, and 7.3.4 refer to these categories in the context of the discussion on identifying, making sense of, and resolving impediments, respectively. Note that many of the codes in Figure 7-6 contain sub-categories.

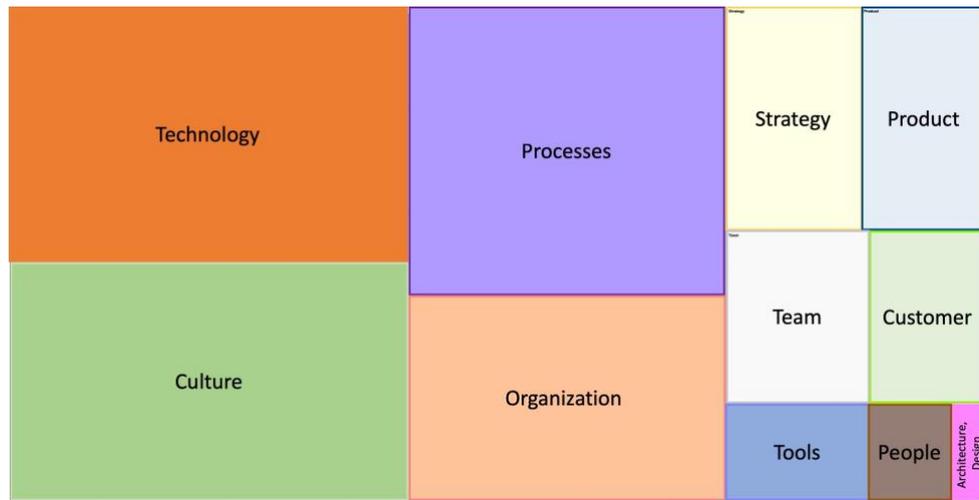


Figure 7-6 Categories of system patterns

Technology, *culture*, *processes*, and *organization* are the most frequently mentioned categories of system patterns across all the findings. *Strategy*, *product*, *team*, *customer*, *tools*, *people*, and *architecture/design* also feature significantly. These system patterns are present in the organizations studied and are considerations in understanding impediments. This ordering relates to how much the patterns show up in the findings through coding, which in turn reflects how much they are discussed or observed. It does not relate to impact, e.g., a pattern that is discussed comparatively little could have a more significant impact than one that is discussed or observed a lot. Section 7.3.3 discusses in more detail how organizations make sense of impediments in the context of these system patterns to determine factors such as impact. Next, the remainder of this section discussed each of the three methods of observing and analyzing system patterns (RO2(i)).

The findings show that several of the organizations in this study wrestle with impediments related to long-running problems that remain unresolved over a period of years. Three examples from the findings serve to illustrate the point. In the first example, a manager in ORG-A said of their user acceptance testing to product challenges that “*nothing has improved in 11 years even with process change*”. This manager spoke of challenges with *handoffs* and *quality* related to this scenario. An ORG-D micronarrative related a challenge that had been persisting in their organization for more than two years: “*Do the team members strive to have the skills necessary to*

fill in for each other when needed ? The Answer seems to be NO. This impacts deliverables and sprint commitments even after 2 years in to Agile transtion.” In the second example, an ORG-C micronarrative relates a sticky issue their organization continues to have after more than two years, this time related to how they are managing dependencies: *“In general the team delivered on time with good qauality. The team tries to minimize the dependeceis and to estimate the work according to our 2 years expirience working together.The main problem are dependencies. There are too many of them and it is very hard to anticipate them.”* The findings show this experience was negative, and experiences of this type happens sometimes in ORG-C. In the third example, an ORG-C micronarrative with the title *“We are still seeking for a long time”* illustrates a long-running and persistent challenge with a perceived lack of purposeful direction in a large organization. The text of the micronarrative captures the experience as: *“[C2] is a big company. Our engineering organization is also big. Recently our leadership team try to figure out what the problem is and to encourage change. However leadership team has been seeking different directions more inputs but the result is not great. Some people doubt the leadership team know what they are doing. The leadership team always have similar group of people. They may not be able to think out of their own thinking pattern. Many surveys and many asking for inputs.”* The findings show this experience was strongly negative, this particular experience happened more than 3 months prior, and experiences of this type are quite common in BU2. This type of context, where repeated efforts over many years have failed to improve the situation, is indicative of a complex system challenge. When an experience happened more than 3 months ago it might be a sign that it reflects a “sticky” or “wicked” problem (Rittel and Webber, 1973), where *“the formulation of a wicked problem is the problem!”*. The memory of it is not fading for people and it is part of the memory of an organization that emerges through what Boje (2008) refers to as fragmented sensemaking, reflecting aspects of an organization’s culture and institutional memory. In summary, analysis of the findings shows that some impediments are indicators of patterns to be shifted, not problems to be solved, which resonates with other work on complex systems (Eoyang and Holladay,

2013, Eoyang, 2001, Van Der Merwe et al., 2019, Snowden and Boone, 2007, Page, 2015). Approaches here include looking at the underlying patterns and the contributing factors holding the patterns in place over time.

Value Streams

Impediments occur within a value stream. Analysis of value streams helps to identify impediments that impact flow (Wang, 2011). Section 6.3.1 shows examples of value stream maps from organizations in this study. Analyzing their value streams exposed a contradiction. On the one hand, organizations in this study had a defined software development lifecycle model, and on the other hand, their defined model did not accurately reflect the process they went through to develop software systems. For ORG-A, the defined SDLC was known to be out-of-date. Three of the organizations had gone through or were going through, a transition, and the SDLC was either not updated (ORG-B), not well understood (ORG-C), or needed to be recreated (ORG-D). In the cases of ORG-E and ORG-F, the organization's formal SDLC had taken a narrower perspective than that provided by the value stream map. In all cases, analysis of the value stream map provided a different end-to-end systems perspective than the official SDLC.

A challenge with value stream mapping is that organizations are not differentiating value streams for different types of work. Womack (2006) noted that as organizations attempt to improve flow *“they will often discover that a major cause of muda and muri is the attempt to run very dissimilar products down the value stream.”* Womack's statement is consistent with this study's findings. Organizations did not have a differentiated set of value stream maps. This lack of differentiation could be a contributor to the impediments (*muda*) (discussed in section 7.2.2) and overburden (*muri*) (discussed in section 7.3.3) observed. Another challenge is that the concept of organizing work around value streams and flow remains counterintuitive. (Womack and Jones, 2003) recognized this: *“The most basic problem is that flow thinking is counterintuitive; it seems obvious to most people that work should be organized by departments in batches.”* And so, a second challenge is that while software development organizations visualize their

value streams and manage impediments there is little evidence in the findings to suggest they are taking steps to fundamentally reorganize around value streams. Organizing by department with functional responsibilities remains the norm.

Value streams aid in making the work transparent. Womack and Jones (2003) wrote that transparency is the most important stimulus to achieving perfection. When everyone involved in the value stream can see everything “*it’s easy to discover better ways to create value*” and remove impediments. Lack of transparency, then, is a significant contributing factor where there are impediments to flow. The findings in section 6.3 confirm this by citing a lack of transparency, often phrased as a lack of visibility, as a significant factor in impediments. Three perspectives on transparency emerged from this study’s findings. The first perspective is “horizontal” transparency, where the people working in and responsible for a value stream are experiencing difficulties due to lack of transparency in their value stream. The second perspective is “downward” transparency, where management don’t have sufficient visibility into the work being done by people in their organization. For example, one of the ORG-B focus groups used A3 Problem Solving to address a problem they titled “*visibility and transparency*” (section 6.3.4). Other participants noted that they do not have sufficient visibility on the status of work and are often surprised at the end of a sprint or release (section 6.3.2). The third perspective is “upward” transparency. A significant contributing factor to impediments for many software development organizations in this study is they don’t have enough upward visibility into what their management teams are doing. This was often manifested through patterns such as lack of strategy and lack of leadership and was a significant contributing factor to unevenness in flow and overburden on the system. Visualizing and discussing value streams at a system level provided a context for discussing impediments in focus groups. By analyzing their value stream maps, participants could see how work is flowing, or not flowing, and identify impediments to flow (discussed in section 7.3.2).

Flow Metrics

Impediments have an observable impact on flow metrics. The presence of impediments can be observed in the patterns exhibited by flow metrics. Petersen and Wohlin (2010) note that to improve flow, it is necessary to understand the current state through visualizing flow and impediments. This study adopted the same premise. This study examined throughput, cycle time, and cumulative flow. Emphasizing the relationship of these metrics to value streams, Mujtaba et al. (2010) also refer to flow metrics as “*value stream metrics*.” In other words, a *value stream map* as discussed above shows how work flows through organizations as they deliver value to customers; *flow metrics* provide data on how much work is flowing, how long it takes, etc. In general, flow metrics can be used to understand system patterns in an organization from a qualitative perspective. An artifact analysis of flow metrics data and visualizations reveals the presence of impediments to flow. The perspective adopted is to look for patterns in the data and the visualizations, with the goal of finding indicators to how well the work is flowing in organizations. Several examples are illustrated in the findings in section 6.3.1. Petersen and Wohlin (2010) show a CFD used to visually identify *handovers*. Similarly, in this study, the findings in section 6.3.1 show CFDs and other flow metrics used to visually identify the potential presence of impediments impacting flow. Referring to cumulative flow, a manager from ORG-C noted “*CF is a picture of how our teams behave*”. A manager from BU3 noted “*The Cumulative Flow Diagram reflects the fact that the team is currently planning for the next milestone work, hence the lack of items in the Backlog or Defined states.*” Overall, CFDs are indicators of the presence or absence of *delays*, *handoffs*, and other impediments. CFDs point to potential impediments and are a lead-in to further discussion. For example, referring to the color of the in-progress state in a CFD section 6.3.1, an engineering manager noted “*the yellow is very high, meaning we have a lot of WIP. Not sure if this is a problem or not.*” The implications of WIP are discussed further in section 7.3.3 below.

In addition to CFDs, throughput diagrams and cycle time charts (section 6.3.1) showed indicators of *mura*, or *unevenness* in how organizations are getting work done. A simple visual inspection of throughput and cycle time charts shows a visual pattern representing flow smoothness. *Mura* is discussed in the literature in section 2.2.1 as one of the major contributors to impediments (Ohno, 1988). Throughput diagrams in section 6.3.1 show additional detail, including the type of work being done. For example, throughput diagrams from ORG-C show that not all of the work the teams were doing was value-creating work that customers pay for. They used throughput charts to visually represent the different areas of work in which they were investing capacity. For example, section 6.3.1 shows they were investing approximately 60% of their capacity in new features and the remainder in CI, deployability, architecture, DevOps, build pipeline improvements, and other initiatives. In other cases, a lot of the demand placed on the system was due to rework and fixing defects. For example, section 6.3.1 shows that up 30-50% or more of the demand on a software development organization can be *failure demand* related to *defects*.

The findings reveal an apparent contradiction. On the one hand, all of the organizations in this study had a history of releasing products and features, and on the other hand, they did not have a consistent definition of what it means to be “done”. Managers viewed improvements in throughput and cycle time as desirable outcomes, e.g. one manager noted “*we want to see some improvement over time. Month-over-month progress improvement in cycle time.*” It is difficult to measure cycle time meaningfully and consistently if it is unclear what it means to be “done” (Kniberg, 2011). All of the organizations in this study struggled with defining and achieving a consistent definition of what it means to be “done” across multiple teams. It was common for teams working on the same product or system to have different definitions of done. For example, ORG-B had multiple teams working on the same system, and each had a separate definition of done. An engineering manager noted, “*we do not have a common Definition of Done*”. Even where teams have created a definition of done “*teams do not consistently use DoD.*” Behutiye et al. (2017) show that the absence of a

common Definition of Done contributes to an increase in *technical debt* (one of the contributors to impediments identified in section 2.5.2), and an increase in *defects* (one of the impediment categories identified in section 2.5.1). This lack of consistency is a contributing factor to impediments that make it difficult to achieve a smooth and continuous flow.

This study did not attempt to correlate resolving particular impediments with specific improvements in flow metrics over time. As described in section 4.2.2, this study takes the position that in a complex system there is no such direct correlation. To do so would require a different study design and is a topic for future research. This study confirms research by Birkeland (2010) that throughput and cycle time “*provide no information about the underlying cause.*” However, identifying and visualizing patterns in flow metrics at a system level provided a context for discussing impediments. Analyzing patterns in flow metrics reveals patterns in how work is flowing and provides input to identifying impediments to flow (discussed in section 7.3.2).

Organization Culture

Figure 7-6 above shows that culture is one of the two most dominant patterns influencing flow and impediments in organizations. The culture of the organization influences how effectively people identify impediments, how they report what they see, or even whether they identify impediments at all. The literature review (section 2.2.1) established that learning to see impediments is difficult. The findings (section 6.2 and 6.3) verified this, identifying four specific examples of why impediments are hard to see. First, people are unaccustomed to thinking in terms of flow and impediments. It is a mindset shift. This need for a mindset shift confirms conclusions from other research on flow. Second, participants reported fear of political ramifications in their organizations. They feared that reporting impediments could be perceived as a sign they are not able to do their jobs. Third, participants reported instances of a lack of support from their management, particularly lower-level and middle-management, in removing impediments. For example, there are cases where impediments were

“*pushed back to the team.*” Fourth, they reported being “*too busy*” or “*under pressure*” with other work taking precedence over managing impediments. This is counter to the culture envisioned by Ohno and others at Toyota, where everyone is expected to actively seek out impediments as part of their work. This aspect of Toyota’s culture is widely regarded as a determining factor in their success (Liker et al., 2008). There is growing recognition of the importance of attending to specific cultural aspects of psychological safety, fear, and transparency in contemporary software organizations (Edmondson, 1999, Kerievsky, 2016b). Culture is ultimately the responsibility of leadership to address (Schein, 2010).

Forsgren et al. (2018b) identify culture as one of five categories of capabilities that drive improvements in software delivery performance. If leaders want to improve flow and have their organizations identify impediments to flow, then they need to make it safe and desirable to do so. Of course, everyone in an organization has a responsibility to contribute to making it safe. However, leaders set the tone for their organizations (Schein, 2010). Other studies recognize the significant role of culture in lean implementations. Bortolotti et al. (2015) identify organization culture as a critical success factor in effective lean implementation, though in the context of manufacturing, not software development. In addition, as shown here and in section 4.2.6, culture is broadly mentioned as important. However, as far as the researcher is aware, there are no other studies exploring this particular relationship between culture and impediments to flow in software development organizations.

7.3.2 RO2(ii): Identify Impediments to Flow

Section 7.3.1 discussed the findings related to system patterns, and how they contribute to the conditions in organizations that allow impediments to emerge. Addressing RO2(ii), this section builds on that by now discussing the findings related to how organizations identify these emerging impediments. In particular, this section discusses the following points from the conceptual framework in Figure 7-1:

- Interpreting system patterns
- Co-occurrence of impediments
- Monitoring the system

Interpreting System Patterns

This section discusses the findings related to interpreting system patterns to identify impediments. In particular, this section discusses the following points:

- The experiences of people working along a value stream
- Insight into the organization context
- Themes related to impediments to flow
- The emotional tone of an organization
- Many impediments originate early in a value stream
- Roles and responsibilities in the occurrence of impediments
- Relationship of system patterns to impediments

The remainder of this section discusses these points in more detail.

The experiences of people working in a value stream

Impediments have an impact on the experiences of people working along a value stream in organizations. Similarly, the patterns observed by making sense of the collective experiences of people in the value stream can reveal the presence of impediments. Weick (2005) asserts that distributed sensemaking produces an emergent view of the situation that none of the individuals in the system could produce alone. This study confirms this in the specific context of understanding flow quality in organizations. Krentel et al. (2016) demonstrate how micronarrative sensemaking tools are a “*valid and effective tool to detect operational issues.*” Their research shows distributed sensemaking using micronarrative tools demonstrate benefits where a more in-depth understanding is needed to improve product and service delivery. Phase 2 of this study, described in section 5.5.2, introduced distributed micronarrative sensemaking at an organization level. The role of distributed micronarrative sensemaking in this study is (a) to provide a

better understanding of the system-wide patterns in the organization that influence flow and impediments, and (b) contribute to more in-depth insight into flow and impediments in organizations. This perspective is part of understanding a system at multiple levels, as discussed in Chapter 4.

Insight into the organization context

Section 6.3.1 presents the findings from the organization culture analysis that help make sense of what is happening regarding flow and impediments. These findings include insight into impediments and flow quality in organizations. The findings also provide richer context into patterns that allow the emergence of impediments in those organizations, and, in the same way as Maitlis and Christianson (2014), provide a basis for understanding what is happening in these organizations that lead to impediments occurring. The findings include insight into emotional sentiment, roles, experiences, morale, and how these relate to impediments and flow. Maitlis and Christianson (2014) refer to this process of distributed sensemaking as extracting cues, or patterns, from the environment, and using these as a basis for constructing a plausible account that provides order and makes sense of what has occurred.

Themes related to impediments to flow

The findings in section 6.3.1 show distributed sensemaking using micronarratives helped explore themes both expected (e.g., *impediments, technology, organization*) and unexpected (e.g., *emotion, psychological safety, strategy, what roles are involved in creating impediments, how frequently different types of impediments occur, contributing factors to impediments*). Krentel et al. (2016) note that one advantage of working with micronarrative is that respondents have a wider degree of freedom in the information they provide, versus other methods. Working with micronarratives “provides a mechanism to explore both expected and unexpected themes, using the respondent’s personal experience as the reference point for subsequent closed questions.” This study followed a similar pattern. These patterns observed through the results of distributed

sensemaking were used as input to inform the focus for some of the focus groups for identifying, making sense of (discussed in section 7.3.3), and resolving impediments (discussed in section 7.3.4).

The emotional tone of an organization

One of the themes from distributed sensemaking, as noted above, is emotional sentiment associated with experiences (section 4.3.5). Research by Vaara et al. (2016) show that the emotional tone of an organization can provide important clues about change, including stability or turbulence. In general, impediments are undesirable and inhibit flow, yet this study found a number of examples where impediments are associated with positive sentiment in organizations. Section 6.2.2 shows several examples where impediments map to experiences that have a positive sentiment. These cases relate primarily to where people worked together effectively to overcome some challenge. Although the impediment itself was not desirable, the outcome of the experience was positive. Associating micronarratives with emotional sentiment helps identify experiences people would like to see more of in their organization (micronarratives associated with positive sentiment), and those they would like to see fewer of (micronarratives associated with negative sentiment). This understanding of sentiment associated with experiences and impediments becomes a further input when deciding what action to take to address impediments (section 7.3.4 below).

Many impediments originate early in a value stream

Despite the predilection of the literature to present impediments as things that somehow surprise a team during development, many impediments originate much earlier in a value stream. Distributed sensemaking provides a way to pay attention to the “weak signals” that hint at the likely presence of impediments

This study’s findings show several cases where impediments originate much earlier in the value stream than the ‘development’ state. A significant implication of this is executives often miss the impact they have on the quality of flow, and their responsibility for improving flow. Much of the

agile literature refers to impediments as things encountered, usually by surprise, during development. For example, Greening (2015) discusses how a Scrum Master helps remove impediments during a Sprint, and how impediments contribute to decreased estimation accuracy and increased deviation in velocity. Greening (2015) recommends that executives visit the team and offer to help with impediments. This study shows that these same executives can have a role to play in creating impediments in the first place. Organizations miss an opportunity to create a smoother flow if they focus just on discovering impediments during Sprints. Table 7.1 is an aggregate report from the findings in section 6.3, and shows that executives, managers, product managers, customers, and other roles feature heavily in the experiences of people in organizations where impediments occur.

Table 7.1 Roles that impact flow and impediments in different states

State to which origins of impediment can be traced	State where impact of impediment was observed	Primary Roles Featured in Impediments
Discovering	<ul style="list-style-type: none"> • Analyzing • Developing • Deploying • Measuring 	<ul style="list-style-type: none"> • Product Managers • Executives • Managers • Customers
Analyzing	<ul style="list-style-type: none"> • Analyzing • Developing • Deploying • Measuring 	<ul style="list-style-type: none"> • Product Managers • Executives • Managers • Customers • Engineers
Developing	<ul style="list-style-type: none"> • Developing • Deploying • Measuring 	<ul style="list-style-type: none"> • Product Managers • Executives • Managers • Engineers
Deploying	<ul style="list-style-type: none"> • Developing • Deploying • Measuring 	<ul style="list-style-type: none"> • Product Managers • Executives • Managers • Customers • Engineers
Measuring	<ul style="list-style-type: none"> • Developing • Deploying • Measuring 	<ul style="list-style-type: none"> • Product Managers • Executives • Managers • Customers

The value stream maps shown in section 6.3.1 show delays and other impediments originating in the *discovering* and *analyzing* states. In several cases the delays run to weeks and months, and sometimes longer. The categories of patterns shown in Figure 7-6, particularly patterns related to *culture*, *strategy*, *customers*, and *organization*, are ultimately the responsibility of executives and managers in organizations.

Roles and Responsibilities in the Occurrence of Impediments

Executives, managers, and other roles play a role in creating the conditions that lead to impediments. They also are ultimately responsible for shifting the system patterns so that the same impediments do not recur. There are three implications of distributed sensemaking for executives and managers in organizations. First, distributed sensemaking can give them more in-depth insights into the patterns in their organizations that contribute to the occurrence of impediments. Second, they are responsible, directly or indirectly, for creating the conditions that allow impediments to occur. Third, if they want a different outcome (e.g., better flow, fewer impediments) they have a responsibility to address the system patterns that contribute to the emergence of impediments. These implications will be part of the discussion next, in sections 7.3.2, 7.3.3, and 7.3.4.

Relationship of System Patterns to Impediments

The system patterns discussed in section 7.3.1 above are factors when identifying impediments. For each of the impediment categories discussed in section 7.2 above, Figure 7-7 highlights a particular set of relationships between impediments and the system patterns from section 7.3.1 that shape the context in which these impediments occur. The percentages in Figure 7-7 indicate the percentage of impediments of that category associated with particular patterns. E.g., approximately 30% of delays identified in this study relate to *culture*, while about 6% relate to *strategy*.

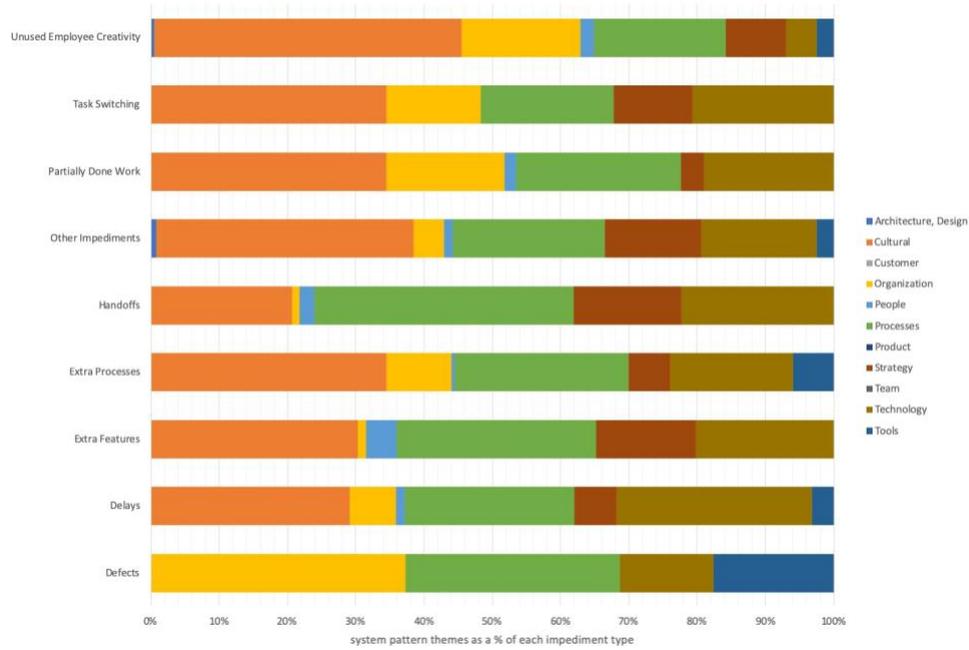


Figure 7-7 Presence of system patterns when identifying specific impediment types
 Table 7.2 articulates many of the observable relationships between impediments and system patterns from section 7.3.1.

Table 7.2 Relationship between impediments and system patterns

System Patterns Theme	Observations related to impediments
Processes	<ul style="list-style-type: none"> Mentioned as a factor in where all impediment categories occur
Culture	<ul style="list-style-type: none"> Mentioned as a significant factor in where everything except defects occur
Organization	<ul style="list-style-type: none"> Mentioned as a factor to some degree in the context of all impediment categories Mentioned as a particularly significant factor in the context of defects, partially done work, task switching, and unused employee creativity Less of a factor where handoffs and extra features occur
Architecture and Design	<ul style="list-style-type: none"> Mentioned in cases where unused employee creativity, extra processes, delays, and defects are impediments
Strategy	<ul style="list-style-type: none"> A significant factor where handoffs, extra features, task switching, and unused employee creativity occur
Cultural, processes, and technology	<ul style="list-style-type: none"> Mentioned where delays, handoffs, extra features, and unused employee creativity occur

No other study, as far as the researcher is aware, has explored this relationship between system patterns and impediments to flow in software development organizations.

Co-Occurrence of Impediments

There is never just one impediment; the organization is dynamic, and many impediments are continually co-occurring. There may not even be one dominant or obvious impediment impacting flow at a particular time. This study's findings in section 6.3.2 show that there are often many impediments impacting flow quality simultaneously in organizations. There are two additional points to note from these findings. First, they represent a snapshot in time. The findings would be different for different organizations, or even at a different time for the same organizations in this study. Second, if organizations were to focus their efforts on just one impediment category, e.g., *delays*, they may not achieve the desired improvements. This has implications for how organizations choose to act to resolve impediments, discussed in sections 7.3.3 and 7.3.4. The findings in section 6.3.2 gives some insight into the web of interconnections that can exist in complex systems. A study by Ikonen et al. (2010) talks a little about multiple impediments and their relationships, e.g., they note partially done work “*emerged from different types of inadequacies at different levels in the system*” and note that it is responsible for generating *task switching* and *extra processes*. The findings in section 6.3.2 confirm a similar, but somewhat different, picture for the organizations in this study. This study does not aim to show causation among impediments, just co-occurrence of multiple impediments. *Partially done work* is likely to be present with *delays*, *extra processes*, *unused employee creativity*, and *task switching*. As far as the researcher is aware, none of the other studies on flow and impediments examine the co-occurrence of multiple impediment types in this way.

Monitoring the System

Organizations are dynamic. Organizations and their environment are continually shifting and changing. Section 7.3.1 discussed the potential in any organization for impediments to occur at any time and any point in a value stream. Therefore, identifying impediments is something that should happen continually. It cannot be just a one-off event or an infrequent

occurrence. Identifying impediments is part of an approach to continuous improvement that has its roots in lean and TPS (Womack and Jones, 2003). As described in Chapter 2, this philosophy has made its way into contemporary approaches to software development, including agile, lean software development, and DevOps. Research by Forsgren et al. (2018b) shows that the best performing software organizations embrace continuous improvement, making it part of everybody's daily work.

Similarly, Arleroth and Kristensson (2011) note that “*worker participation in problem-solving*” is a critical aspect of continuous improvement. Similarly, managers, leaders, and development teams from all organizations in this study participated to some degree in identifying impediments. However, in contrast to Toyota and other organizations described in the lean literature, where continuous improvement is *continuous*, the organizations in this study fall somewhere on a continuum. Some organizations, including ORG-A, ORG-B, and ORG-E, were at the early stages of starting to focus on flow and to identify impediments explicitly. Other organizations, including ORG-C, ORG-D, ORG-F, and ORG-G, had been focusing on flow for some time, and have comparatively more experience with identifying impediments. In particular, ORG-C, ORG-D, and ORG-F had established practices of monitoring for impediments.

The findings suggest that developing the capability to monitor continuously for impediments can be integrated into the daily operations of organizations of an organization and has the potential to be beneficial. This capability is not widespread in the organizations studied, though all organizations indicate a desire to improve this capability. Most organizations are still at the stage of identifying impediments in response to a perceived problem. In other words, they are retroactively responding to the presence of impediments, rather than proactively identifying or monitoring for impediments. The goal should not be the complete elimination of impediments, which as Womack and Jones (2003) note, would be impossible. Instead, the goal is to develop a system of proactively monitoring for impediments so organizations can identify them early and

continuously. A more accurate description of the approach by many of the organizations at the time of this study is periodic, intermittent, or reactive improvement, rather than continuous improvement.

Poppendieck and Cusumano (2012) note that many organization impediments “*have their roots in the large batches of partially done work created in sequential development processes, in the boundaries between different functions, and in the delays and knowledge lost when work crosses these boundaries.*” This observation resonates with findings discussed in section 7.3.1 above, where many groups did not have a consistent definition of done. Data such as those in section 6.3.2 provide an opportunity to question whether there might be gaps in perception when identifying impediments.

7.3.3 RO2(iii): Analyze Ways to Make Sense of Impediments

Section 7.3.2 discussed the findings related to identifying impediments in organizations. Addressing RO2(iii), this section builds on that by now discussing the findings related to how organizations make sense of the impediments they identified. In particular, this section discusses the following points from the conceptual framework in Figure 7-1:

- Impact of impediments
- Influence on impediments
- Context of impediments
- Avoidable and unavoidable impediments

Impact of Impediments

Each impediment has a different impact on organizations. One way that organizations made sense of impediments is to articulate this impact. The findings in section 6.3.3 show examples of impact from two perspectives; the (negative) impact an impediment is having on the organization, and the (positive) impact resolving the impediment would have on the organizations. Section 6.3.3 shows that organizations in this study articulate

specific areas where they feel the impact of impediments. These include *time*, *quality*, *productivity*, *predictability*, *morale*, *lifecycle profit*, and *architecture/design/code*. In other words, these are specific aspects that are impacted by impediments.

The impact of impediments can be different in different companies and organizations. Figure 7-8 shows these impacts of impediments in two different companies. Effects on *time*, *productivity*, *lifecycle profit*, and *code/design/architecture* are the dominant impacts observed in C1. Effects on *predictability*, *morale*, *quality*, and *code/design/architecture* are the dominant impacts observed in C2. Overall, C2 perceives that impediments impact on *predictability* and *quality* more than does C1. C1 perceives that impediments impact on *time*, *productivity*, and *lifecycle profit* more than does C2. Both companies experience comparable impact in *morale* and *code/design/architecture*.

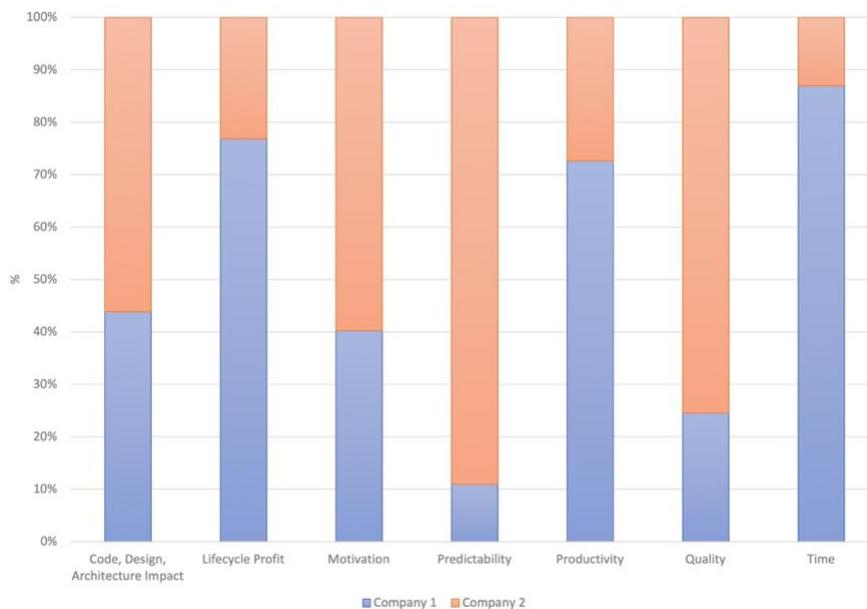


Figure 7-8 Comparing the impacts of impediments in two different companies

These findings confirm observations from Ohno (1988), Reinertsen (2009), Poppendieck and Poppendieck (2007), and others as discussed in section 2.5.3. This research provides illustrative examples of these impacts of impediments on the organizations studied.

Influence on impediments

Factors that contribute to impediments are discussed earlier in this section, with *overburden* noted as the most common contributing factor. This subsection discusses specifically how people and groups influence impediments. Two perspectives emerged as participants discussed impediment influence. First, there are people and groups whom participants mention contribute to impediments occurring. Second, there are people and groups whom they mention are needed in order to resolve impediments. This aspect of sensemaking helped inform who needs to be involved in the next step of resolving impediments, discussed in section 7.3.4.

The findings overall show that the majority of impediments are within the control of the team or organization to address. “*We’re doing this to ourselves!*” was the realization of one engineering director from ORG-C when looking at the impediment influence diagrams his focus group produced in section 6.3.3. The group had identified a wide array of impediments impacting the organization, as shown in section 6.3.2.

Several themes emerged when analyzing those impediments that are not within the sole control of the organization to address. They are both within the company and outside the company. Examples within the company include dependencies on other organizations that are part of the same company, conformance to shifting company strategies, and lack of alignment at a leadership level. Examples outside the company include third-party dependencies on vendors and partners, alignment with customer roadmaps, and changing technology standards and platforms, e.g., OpenStack or 5G. In all these cases, the organizations’ plans to address impediments needed to include engagement with stakeholders outside their organization, and sometimes outside their company.

Context of Impediments

Section 6.3.3 presented findings that show how organizations in this study took steps to understand the context of impediments. This section discusses the findings from their use of the Cynefin sensemaking framework

(Snowden and Boone, 2007), described in section 4.3.6. Regarding the Cynefin sensemaking framework, there were very few impediments that map to the *obvious* domain. If something obvious is impacting flow then it was likely to be dealt with as a matter of course during day-to-day operations, and less likely to surface in a focus group examining current issues impacting flow. Those few that did map to *obvious* were taken away as actions for participants or their colleagues. Some impediments mapped to the *chaotic* domain. Some impediments mapped to the *disorder* domain. The findings in section 6.3.4 show the majority of impediments identified by organizations in this study mapped to either the *complicated* domain or the *complex* domain. Section 7.3.4 will discuss how this domain mapping influenced the interventions that groups chose to resolve impediments.

One implication here is the idea of “best practice” for managing impediments and improving flow is often misguided. Oppenheim (2004) recognizes this, noting a variety of approaches are necessary to mitigate uncertainties in the context of lean product development flow. As discussed in section 4.3.6, the place for “best” practice in the Cynefin framework is the *obvious* domain (Snowden and Boone, 2007). With so few impediments mapping to the *obvious* domain, organizations that attempt to define their response to managing flow and impediments as a set of best practices will be ill-equipped to deal with the dynamic and emergent nature of improving flow in complex systems and could make the situation worse. Snowden and Boone (2007) note that “*best practice is, by definition, past practice.*” What worked with past challenges may not work with current impediments. This is one reason why this step of making sense of the context is so important. A reliance on the defined best practices in an organization’s management approach would be inappropriate for more than 99% of the impediments identified in this study. As discussed in section 4.3.6, good practice is appropriate in the *complicated* domain, and emergent practice in the *complex* domain. Section 7.3.4 will discuss how best practice, good practice, and emergent practice inform how organizations act to resolve impediments. Rikkilä et al. (2013) discuss and apply the Cynefin sensemaking framework in the general context of proposing a leadership

model for lean and agile project management. As far as the researcher is aware, none of the other studies on flow and impediments employ the Cynefin sensemaking framework. Hence, this is the first study to consider the impact of different domains on informing appropriate practice for acting to resolve impediments and improve flow.

Avoidable and Unavoidable Impediments

Section 2.2.1 made the case of distinguishing between avoidable and unavoidable impediments (Womack and Jones, 2003). Based on the findings in section 6.3, this study proposes to expand the definition by Womack and Jones (2003) to distinguish between two subtypes of Type One impediments. Type 1(a) are impediments that are unavoidable *with current technologies and production assets because alternatives do not yet exist*. Type 1(b) are impediments that are unavoidable *with current technologies and production assets in use in the organization, but alternatives exist*. Section 6.3 provides examples of both from the findings. The distinction between avoidable and unavoidable impediments is information that informs how organizations prioritize their focus and determine which impediments to resolve.

7.3.4 RO2(iv): Explore Approaches to Resolving Impediments

Section 7.3.3 discussed the findings related to making sense of impediments in organizations. Addressing RO2(iv), this section builds on that discussion to discuss the findings related to how organizations approached resolving the impediments they identified and made sense of. In particular, this section discusses the following points from the conceptual framework in Figure 7-1:

- Prioritization and Focus
- Approach to resolving impediments
- Interventions to resolve impediments
- Impediment Refinement

Prioritization and Focus

Every organization in this study identified more impediments than they attempted to resolve. The findings in section 6.3.4 show that organizations from this study cannot afford to resolve every impediment they identify. Given limited time and capacity, it was not feasible for them to resolve all impediments right away. Part of the discussion within every organization in this study was prioritizing the impediments on which to focus. The impediment matrix described in section 6.3.4 is one example from ORG-A that shows how organizations in this study made priority trade-offs when deciding what impediments to invest in resolving. They made priority decisions based on understanding the impact of the impediments, how important it was to address impediments relative to each other, and how important it was to address impediments relative to other work the organization needed to do, e.g., project work and feature development.

Section 7.3.3 discussed how organizations considered the impact of impediments, and who influenced impediments, as a means of making sense of them. This information also contributed to informing how they prioritized impediments for action. In these cases, organizations tended to prioritize impediments whose resolution they perceived would result in a higher level of return. They also tended to prioritize impediments that were within their control to resolve, with one participant referring to these as “*low hanging fruit*.” Impediments that were perceived to have a high impact and also required engaging stakeholders outside the group tended to be more involved. Section 6.3.4 contains more detail on the impediments that organizations chose to resolve. Table 7.3 shows a summary of impediments from each organization that they chose to prioritize for resolution.

Table 7.3 Summary of focus areas for resolving impediments in each organization

Organization	Priority Areas of Focus
ORG-A	<ul style="list-style-type: none"> • Build Times • Customer Regression Testing • Customer Satisfaction
ORG-B	<ul style="list-style-type: none"> • Visibility and transparency • Definition of Done • Testing
ORG-C	<ul style="list-style-type: none"> • Delays • CD Pipeline • Definition of Done
ORG-D	<ul style="list-style-type: none"> • Continuous Integration • Vision and Strategy • Cross-team collaboration
ORG-E	<ul style="list-style-type: none"> • Establish a synchronized 2-week Sprint cadence across all teams • Definition of Done • Continuous Integration
ORG-F	<ul style="list-style-type: none"> • Understand and visualize the flow of value to customers and users • Measure team happiness across the program • Communicate what capabilities are enabled in the product at the end of each Sprint and Release
ORG-G	<ul style="list-style-type: none"> • Automate customer regression test cases • Develop an approved capacity plan for the coming 12 months • Get a baseline on current processes and process owners

In some cases, the impediment may not be resolvable. In such cases, the organizations shifted their focus to something they can address. For example, the topic of layoffs and wider corporate reorganizations emerged as either impediments or factors influencing impediments in some organizations, including BU1, BU2, ORG-D, ORG-E, ORG-F, and ORG-G. However, none of the participants felt they could do anything about these impediments, so they focused their efforts on what they could manage. In some cases, e.g., ORG-C, ORG-D, ORG-F and ORG-G, the participants took actions to at least have discussions with their leadership teams about these issues, but their expectations for any action, let alone resolution, followed a much longer timeframe than other impediments they could address.

While other studies referenced in this research discuss identifying and removing impediments, no other study, as far as the researcher is aware, explores this aspect of explicitly choosing a focus when resolving impediments. The general model described in other studies is one of

identifying impediments and then fixing impediments. This study shows that, at least for larger organizations operating under conditions of complexity and uncertainty, they cannot address all impediments and need to choose a focus explicitly.

Approach to Resolving Impediments

Section 7.3.3 discussed how sensemaking provides insight in order to take contextually appropriate action. This section discusses how different approaches to taking action are contextually suited to impediments in the different sensemaking domains discussed in section 7.3.3. There are many approaches an organization can take to managing how it acts to resolve impediments. This study explored four such approaches used by organizations in this study. These are *A3 problem solving* (section 2.2.1), *continuous improvement goals* (section 2.2.2), *experiment design* (section 4.4.4), and *system improvement goals* (section 4.4.5). Table 7.4 shows a summary of the four different approaches together with examples of impediments they were used to resolve across the organizations in this study. Table 7.4 also shows the dominant sensemaking domain in which the approach was primarily used by organizations in this study.

Table 7.4 Impediment-resolving approaches, impediments, and Cynefin domains

Approach to resolving impediments	Example Impediments Addressed	Primary Organization(s)	Dominant Sensemaking Domain
Continuous Improvement Goals	<ul style="list-style-type: none"> • Build times • Workflow • Manual regression testing • Customer satisfaction • Test automation • Reduce delays in setting up development environments 	ORG-A	Complicated
A3 Problem Solving	<ul style="list-style-type: none"> • Visibility and transparency • Definition of Done • Testing • Integrated full stack builds • Dispersed teams • Managing dependencies • Multiple teams sharing same codebase 	ORG-A ORG-B ORG-C	Complicated
Experiment Design	<ul style="list-style-type: none"> • Information visibility and transparency • Improve throughput, productivity • Improve collaboration • Strategy • Reference implementation of complex product • Synchronize the work of multiple teams • Continuous integration • Increase customer happiness • Increase employee happiness 	ORG-D ORG-E	Complex
System Improvement Goals	<ul style="list-style-type: none"> • Understand and visualize the flow of value to customers and users • Measure team happiness across the program • Communicate new product capabilities at the end of each Sprint and Release • Automate customer regression test cases • Develop an approved capacity plan for the coming 12 months • Baseline current processes and process owners 	ORG-F ORG-G	Complex

Continuous improvement goals and *A3 problem solving* are grounded in lean, while *experiment design* and *goal setting* are grounded in complexity theory. The literature review made the point that traditional lean approaches such as “5 whys” and root cause analysis are limited when used in complex product development scenarios (Reinertsen, 2009, Davis and Daniels, 2016, Forsgren et al., 2018b). More specifically, based on how they are used by the organizations in this study these techniques seem to be more applicable in ordered systems (*obvious* and *complicated* domains), and less helpful in unordered systems (*complex* and *chaotic* domains). The distinction between ordered and unordered systems is discussed in section 4.2.2.

Continuous improvement goals and *A3 problem solving* are used primarily to resolve impediments that map to the *complicated* or *obvious* sensemaking domains. Section 6.3.4 shows examples of *continuous improvement goals* from ORG-A used to manage how they resolved improvements. Section

6.3.4 shows examples of *A3 problem solving* from ORG-A, ORG-B, and ORG-C.

Experiment design and *system improvement goals* were used primarily to resolve impediments that mapped to the *complex* sensemaking domain. Section 6.3.4 shows examples of *experiment design* from ORG-D and ORG-E to manage how they resolve impediments. Section 6.3.4 shows examples where ORG-F and ORG-G use *goal setting* to manage how they resolve impediments.

Many lean-based approaches assume identifiable root causes and a discernable relationship between cause and effect. This assumption is appropriate in some cases, e.g., an engineering manager in ORG-B noted: “*requirements will change that change everything and cause delays.*” Section 6.3.4 shows the results of root cause analyses for several impediments from ORG-A, ORG-B, and ORG-C. Impediments in the *complex* or *chaotic* domains, on the other hand, do not have discernable relationships between cause and effect (Snowden and Boone, 2007). Hence, impediments in the *complex* or *chaotic* domains require other approaches. If organizations used traditional lean approaches on all impediments in this study, it would have meant using a contextually mismatched approach to resolve almost half of impediments. No other study, as far as the researcher is aware, explores this relationship between approaches used to resolve impediments, and the sensemaking domains to which the impediments map, in order to explore contextually appropriate approaches for action to resolve impediments.

Interventions in the System

This section discusses how interventions affect the system patterns discussed in section 7.3.1. From a complexity perspective, interventions are changes to the system. Section 4.4.1 described how resolving impediments in organizations is a process of managing change in a complex adaptive system and is effectively about attempting to shift the patterns in order to get a better outcome. In this case, that better outcome is improved flow.

Section 7.3.1 discusses the patterns observed in this study, based on the findings presented in section 6.2. This section looks at the four approaches discussed in the previous section, and how they relate to shifting the system patterns described in section 7.3.1.

Section 6.3.1 showed the dominant system patterns that were present across this study’s findings. Table 7.5 shows the approaches used by organizations in this study to resolve impediments, discussed above in this section, and how the different approaches relate to the system patterns.

Table 7.5 Impediment-resolving approaches, system patterns, and Cynefin domains

Approach to resolving impediments	Continuous Improvement Goals	A3 Problem Solving	Experiment Design	System Improvement Goal
Dominant system patterns addressed using this approach	<ul style="list-style-type: none"> • Technology • Processes 	<ul style="list-style-type: none"> • Technology • Process • Culture 	<ul style="list-style-type: none"> • Culture • Processes • Organization • Technology 	<ul style="list-style-type: none"> • Organization • Culture • Customers • Processes
Other system patterns addressed to a lesser extent using this approach	<ul style="list-style-type: none"> • Tools • Customers • Organization • Architecture • Culture 	<ul style="list-style-type: none"> • Organization • Tools • People • Customers 	<ul style="list-style-type: none"> • People • Tools • Strategy • Team • Product • Customers • Architecture 	<ul style="list-style-type: none"> • People • Technology • Strategy • Architecture
Dominant sensemaking domain addressed by the approach	<ul style="list-style-type: none"> • Obvious • Complicated 	<ul style="list-style-type: none"> • Complicated 	<ul style="list-style-type: none"> • Complex 	<ul style="list-style-type: none"> • Complex • Chaotic

Note the following points about the relationship of each approach to the system patterns for organizations in this study:

- They tended to use *continuous improvement goals* primarily to effect change in system patterns based around technology and processes.
- They tended to use *A3 problem solving* primarily to effect change in system patterns based around technology, processes, and some aspects of culture.
- They tended to use *experiment design* primarily to effect change in system patterns based around culture, processes, organization issues, and technology
- They tended to use *system improvement goals* primarily to effect change in system patterns based around organization, culture, customers, and processes.

From the perspective of system patterns, the following are also worth highlighting:

- Addressing strategy belongs in the *complex* domain and is addressed mainly through *experiment design* and *goal setting*.
- Some system patterns are addressed through more than one approach, e.g., culture, processes, and organization. This highlights the point that system patterns can be influenced in multiple ways through different approaches. The sensemaking domain of the impediments is a factor here.

Figure 7-9 summarizes these relationships, how the four approaches are used by the organizations in this study, and the patterns addressed by the specific approaches. Figure 7-9 also shows visually, using NVivo hierarchy charts, the proportion of patterns addressed through each approach. For example, Figure 7-9 shows visually that it is more common for the organizations in this study to address impediments related to technology patterns using *A3 problem solving* or *continuous improvement goals*. It is more common for the organizations in this study to address impediments related to culture patterns using *experiment design* or *system improvement goals*.

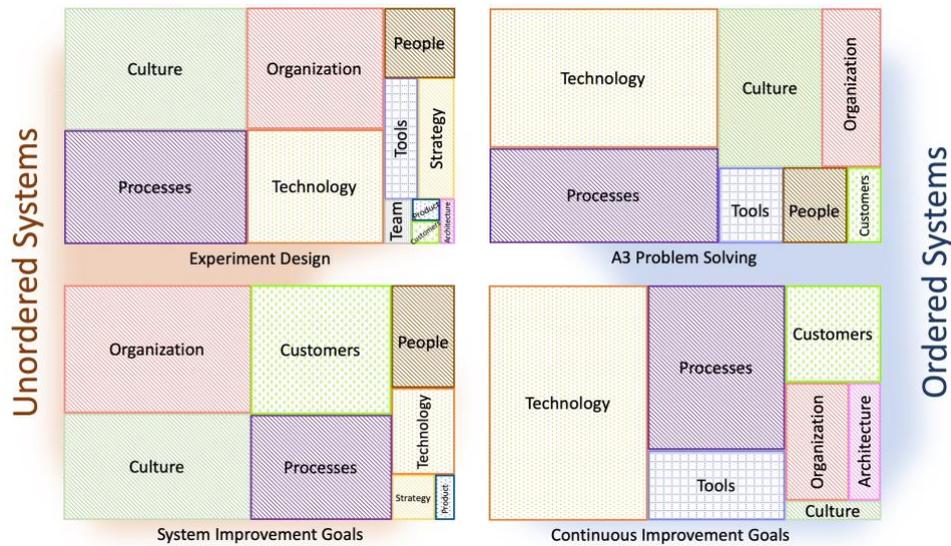


Figure 7-9 Influencing system patterns in ordered and unordered systems

No other study of flow in software organizations, as far as the researcher is aware, has explored the relationship between system patterns, and how those system patterns are related to the approaches organizations use to resolve impediments to flow.

Impediment Refinement

Impediments often got refined as organizations worked on them. The impediments they first identify do not always look like the impediments they choose to resolve. Going through a sensemaking process contributes to the groups' understanding of what the impediment is. The impediment the organization ends up tackling may not always be precisely the impediment they identified earlier. Section 7.3.3 described how the organizations refined the impediments as they worked through making sense them.

No other study of flow in software organizations, as far as the researcher is aware, has explored the process of how impediments get refined as organizations make sense of them and work to resolve them.

7.4 Contributions of this Study

This section highlights the significant contributions and implications of this study to research and practice.

7.4.1 Contributions to Research

The contributions to research from this study come from both a theoretical and methodological perspective. This section outlines the three main contributions of this study to research. These are:

- Impediments to Flow
- The Impediment Management Framework
- The Components of the Impediment Management Framework

Impediments to flow

The first contribution of this study addresses RO1. This study adds to the flow literature by explicitly focusing on identifying and managing impediments to flow. Previously, research on flow predominantly studied aspects of *waste*. This study responds to research calls for broader approaches that assess methods for achieving improved flow quality in contemporary knowledge-intensive software product development organizations (section 2.5). An analysis of the literature review and the findings resulted in four parts to this contribution that combined create a more holistic and nuanced perspective on impediments to flow than previously existed.

- First, a review of existing literature (section 2.5.1) identified eight categories of impediments to flow in organizations. Acknowledging that these categories of impediments from the literature are a starting point, this study incorporated sensemaking approaches to identify a broader set of impediments.
- Second, based on an analysis of the findings, this study proposes an extension to the taxonomy from Womack and Jones (2003) (section 7.3.3), who distinguish between unavoidable (Type 1) and avoidable (Type 2) impediments. In the case of unavoidable impediments, this study proposes to differentiate between those impediments that are unavoidable because alternative technologies and production assets do not yet exist (Type 1(a)), and those that are unavoidable because the

organization is not using technologies and production assets that do exist (Type 1(b)).

- Third, the literature review (section 2.5.2) identified nine contributing factors known to lead to impediments in organizations. These contributing factors were either explicitly stated in limited or singular research studies (such as *failure demand*) as impeding flow in organizations or were the most commonly cited factors (such as *overburden*, *unevenness*, *queues*, *WIP*, *batches*, *technical debt*, and *complexity*) affecting flow.
- Fourth, the literature review (section 2.5.3) identified six effects of impediments to flow in organizations. Again, these effects were either explicitly stated in limited or singular research studies (such as *lifecycle profit*) or were the most commonly cited effects of impediments to flow (such as *productivity*, *predictability*, *quality*, *time*, and *morale*).

As far as the researcher can tell, this study is the first to collectively assess impediments, together with their contributing factors and their impacts, and, collectively examine them in the context of large-scale product development organizations. The contribution of this study to understanding impediments to flow extends beyond software development and applies to other domains.

Impediment Management Framework

The main theoretical contribution of this study is the impediment management framework. The framework's purpose is to improve flow in software product development organizations. This framework addresses RO2. The framework is both empirically grounded and theory informed. It is a consolidation of the theoretical basis of flow-based development (itself a product of lean product development and agile software development), sensemaking, and complex adaptive systems theory. This study validated the framework through application and revision with multiple software development organizations. The framework provides a better understanding of the essential characteristics of impediments in flow-based product development. One such aspect highlighted by this study (see section 7.3.4) shows that impediments change as organizations go through a process to

first identify them, then make sense of them, then act to resolve them. In other words, the place where an organization initially observes the impediment may not be where it needs to take action. Grounded in CAS theory, the framework explicitly views organizations as complex adaptive systems. Patterns and interventions in a CAS proceed in a perpetual cycle: system-wide patterns influence perception at local levels in the organization; action taken at local levels in the organization influence system-wide patterns. This perspective is different from that employed by other studies of flow in organizations. The organizations in this study were at various stages of using agile development methods, but the application of the framework is not limited to agile software development, or even to software development organizations. It can be used to study flow in any organization.

Components of the Framework

More specific theoretical contributions of the study come from the components of the impediment management framework, each of which addresses one of the sub-objectives of RO2. The framework components enrich the understanding of flow in software development organizations.

Analyzing system patterns

This study highlights the relevance of recognizing patterns in complex systems and how those patterns influence the emergence of factors that contribute to impediments. Dysfunctional patterns in organizations lead to impediments that can bring the flow of work to a halt. The research shows how value stream maps, flow metrics, and organization culture reveal qualitative system-wide patterns. Patterns observed at a system or macro level provide rich system-wide context that helps identify and make sense of impediments. Patterns at a local level (specific product, program, or group) provide context-specific insights that help refine the understanding of impediments and how to resolve them. Section 7.3.1 details the particular patterns observed in the organizations in this study. The study highlights the

potential benefits of routinizing the analysis of pattern observation in software development organizations.

Identifying impediments to flow

This study highlights how organizations can identify impediments from system patterns. No other research, as far as the researcher is aware, has explored this relationship between system patterns and impediments to flow in software development organizations. A focus on system patterns reveals insight into an organization's culture. While related literature broadly mentions organization culture as an essential consideration, this study goes farther and shows how particular aspects of culture affects impediment identification. In contrast to other studies on flow, this study identifies the co-occurrence of multiple and varied impediments and explores how this can validate gaps in impediment identification. The study highlights the value of continuously monitoring for impediments.

Making sense of impediments to flow

Exploring the system patterns and the contributing factors helps organizations to make sense of the impediments and determine a course of action. This study of flow in software development organizations is the first, as far as the researcher is aware, to use self-signified micronarrative-based sensemaking approaches to understanding and making sense of impediments to flow. This is the first study of flow in software development organizations, as far as the researcher is aware, to use specific sensemaking frameworks including the Cynefin framework, Impediment Impact Diagrams, and the CDE model, to make sense of impediments and their context in order to determine and evaluate context-appropriate courses of action for resolving impediments.

Interventions for resolving impediments to flow

This study looks at two aspects of interventions for resolving impediments. The first aspect is the approach taken to resolve impediments. The literature review identified four common approaches to resolving impediments in

organizations. These approaches were either the most commonly cited or explicitly stated in research studies on lean-based management (section 2.2 - *A3 Problem Solving, continuous improvement goals*) or complexity-based management (section 4.4 - *experiment design, system improvement goals*). The second aspect is the type of intervention. The literature review identified the CDE framework (section 4.2.3) as a commonly cited and explicitly stated approach to understanding interventions types in complex systems.

The existing literature treats the resolution of impediments as if there were no particular distinction between them. This study shows that organizations use different approaches to resolve different types of impediments. Compared with the existing flow literature, the findings show that the nature of impediments is different based on the sensemaking domain to which they map. It shows that, at least for the organizations in this study, the different approaches to resolving impediments are more prevalent than others in different sensemaking domains.

- The findings show that resolving impediments through *experimentation* and *system improvement goals* is most likely to be used when managing impediments in the *complex* domain. These approaches are often associated with management approaches based on complexity, as discussed in section 4.2. These approaches assume an *unordered system* where there is no discernable explicit cause for the impediment.
- The findings further show that resolving impediments using *A3 Problem Solving* and *continuous improvement goals* are most likely to be used when managing impediments in the *complicated* domain. These approaches are often associated with management approaches based on lean management, as discussed in section 2.2. These approaches presume an *ordered system* where there is often an apparent connection between the impediment and likely causes.
- There are comparatively fewer findings from this study of impediments in the *obvious, chaotic, or disorder* domains. Future studies will seek to fill that gap explicitly. Of those impediments that did map to the obvious

domain, organizations generally addressed them through more traditional project management action items.

In contrast to other studies on flow and impediments, this study takes the perspective that acting to resolve impediments in software product organizations is intervening in a complex adaptive system (as discussed in the literature review in section 4.4.1). As well as exploring the intervention approaches (above), this study also explored the intervention type, using a CDE analysis. This study further demonstrated, through a CDE analysis of the findings, that interventions based on altering or improving *exchanges* are more prevalent than those based on altering or improving *containers* or *differences*. No other study, as far as the researcher is aware, has employed a CDE analysis to understand the nature of interventions undertaken in organizations to resolve impediments to flow. This study also shows how CDE interventions map to the domains of the Cynefin sensemaking framework. Again, no other study, as far as the researcher is aware, has analyzed interventions from this perspective. Finally, the findings show how individual interventions often address many impediments, as well as addressing contributing factors and system patterns identified earlier.

All of this has implications for future studies of organizations that use flow and flow-based approaches, including those organizations using lean, agile, DevOps, SRE, Continuous Deployment, and other forms of continuous development. These theoretical contributions point to new and promising trajectories for future investigation.

7.4.2 Contributions to Practice

In addition to the contributions described in section 7.4.1, this section outlines four specific contributions of this study to practice. These are:

- Improving product development flow
- Understanding the levers that influence product development flow
- Designing context-appropriate interventions to resolve impediments
- Summary of benefits and recommendations to practitioners

Improving product development flow

The study highlights the value of developing a sensemaking capability in organizations, and the corresponding value of augmenting existing management approaches by adopting tools from both lean management and management practice informed by complex adaptive systems theory. No other study, as far as the researcher is aware, takes this perspective in managing flow and resolving impediments in software development organizations. The resulting impediment management framework can facilitate efforts to improve product development flow in software development organizations. The framework is process-agnostic and is compatible with any software development process in use by an organization. The framework provides a means of understanding the connection between the patterns at work in an organization, the contributing factors that emerge from those patterns, the impediments to which those factors contribute, and the interventions that can resolve those impediments. Acting on the impediments is often a means of influencing the patterns or contributing factors in the system. The study highlights the benefits in understanding which patterns are beneficial and should be sustained or amplified, and understanding which patterns are dysfunctional or harmful and should be eliminated or dampened. The study shows the benefit in understanding that these patterns lead to the emergence of contributing factors that can lead to impediments to flow. A factor not commonly highlighted by other studies of flow is the role of organizational culture in identifying and resolving impediments. This study provides insight into how to foster a culture that makes it safe and desirable for people to identify impediments. The findings and resulting discussion in section 7.3 can serve as a guide for managers in shaping a culture that can effectively optimize flow in their organizations.

Understanding the levers that influence product development flow

A review of existing literature on contemporary software development (section 2.3) highlights that increased throughput and reduced cycle times

are outcomes about which managers of software development organizations care. The literature review shows that organizations often use these measures as proxies for organization productivity, predictability, and responsiveness. Furthermore, a review of existing flow metrics literature (section 3.3) identified that reducing *WIP*, managing *batch size*, and keeping *queue size* small is key to optimizing *throughput* and *cycle time*. They are, in effect, the levers through which organizations manage flow. This study shows that while this may sound straightforward, it remains challenging to achieve in practice. Managers in the organizations in this study show varying degrees of awareness of the connection between desired outcomes and the variables that affect those outcomes. None of the organizations in this study were successfully managing queue size in a consistent and continuous way, and several struggled with *WIP* and *batch size*. These observations confirm findings in other studies on flow and software development (see section 3.3). This study contributes some more in-depth insights and new perspectives to the literature on why organizations struggle to control *WIP* and *batch size* (see section 6.2.2 and 7.3.1). Also, this study shows that software development organization can use these metrics as proxies or indicators of the quality or smoothness of flow, and the severity of impediments impacting flow. Organizations can take these outcomes and use these metrics as a starting point to understand their flow quality and understand where there might be impediments.

Designing context-appropriate interventions to resolve impediments

Managers can influence the patterns in their organizations through designing interventions to act on impediments. Acting on impediments in this way rather than acting directly on the patterns is an example of the principle of acting obliquely in complex systems, described in the literature review in section 4.2.3. It is ultimately through acting on the impediments and contributing factors that managers can shift the patterns to ones that better serve the organization's goals. The study highlights the importance of understanding the context of impediments so that organizations can take

context-appropriate action to resolve them. Sensemaking, both distributed (see section 7.3.1) and localized (see section 7.3.3), provides this rich context. For practitioners, the study shows that impediments that map to the *complex* domain are more often handled using *experiment design* and *goal setting*, whereas impediments that map to the *complicated* domain are more often handled using *A3 problem solving* and *continuous improvement goals*. No other study, as far as the researcher is aware, has made this distinction regarding the appropriateness of different tools to address different types of impediments in software development organizations. This contribution has immediate practical value for managers and other practitioners in that it can serve as a guide for managers on selecting context-appropriate approaches to resolve impediments.

Finally, this study's findings highlight that designing *exchange* interventions is the most prevalent strategy employed by the organizations in this study for resolving impediments (section 7.3.4). This observation applies to impediments in both the *complex* and *complicated* sensemaking domains. In practical terms, this means improving (though not necessarily increasing) the collaboration and interaction among all participants in the value stream. Section 7.3.4 summarizes examples from the findings. *Container* interventions are the second-most prevalent, followed by *difference* interventions. In CAS terms, these findings provide examples for practitioners of how the organizations in this study leverage *constraints* (discussed in section 4.2.3) to manage interventions in complex systems (section 4.4). No other study, as far as the researcher is aware, has employed this perspective to use CAS, sensemaking, the Cynefin framework, and the CDE framework to provide practical examples of using complexity-informed approaches to improve flow in organizations.

Summary of benefits and recommendations to practitioners

Organizations can realize significant benefits and positive outcomes by integrating into their current product development and operational approaches a complexity-informed capability to manage flow and impediments. The findings from this study, together with the application of

the impediment management framework, provide several benefits to practitioners, foremost of which is a system of work that enables a continuous flow of value to meet the needs of customers and users. Resulting benefits include increasingly satisfied customers and users of the organization's products and services, increasingly satisfied employees working within effective value streams, and better organization insights and capabilities to influence desired outcomes. The following specific recommendations for practitioners arise from this study's findings:

1. **Analyze system patterns.**

- **Recommendation 1(a):** Start by focusing on flow from the perspective of customers. View organizations and value streams as complex adaptive systems. In order to improve desired outcomes, identify patterns that are beneficial or harmful throughout the system.
- **Recommendation 1(b):** Look for patterns across the entire system as a whole. Flow-based development encourages beginning with an understanding of who their customers and users are and what value means from their perspective. Identify the value-creating steps that, from their perspective, enable the delivery of products and services. The value-creating steps should continue beyond the cash transaction and seek to understand how the product behaves in operation and how it meets the ongoing needs of its users.
- **Recommendation 1(c):** Focus on flow and impediments over waste elimination. A focus on waste elimination brings with it a focus on efficiency and can have limiting or even damaging effects on a product development organization and its relationships with its employees and customers. Instead, a focus on flow and impediments encourages a continuous focus on effectiveness and meeting the needs of customers, employees, and everyone participating in the value stream.
- **Recommendation 1(d):** Measure the system, not individual people. Use system-level metrics such as throughput, cycle time,

and cumulative flow to provide insights into the system patterns. Using these metrics not as targets, but for generating understanding, provides insight into outcomes such as productivity, predictability, reliability, and quality produced by the organization. Working with the system patterns contributing to these outcomes is key to effecting changes in the outcomes.

- **Recommendation 1(e):** Recognize the influence of culture on flow and impediments and use approaches that provide insight into the culture of the organization. The quality of flow and the nature of the organization's response to impediments is a reflection of the organization's culture. Culture is a reflection of people's lived experiences. Micronarrative sensemaking can reveal rich insights into the culture through capturing and communicating people's lived experiences in the organization.

2. Identify impediments that prevent the value-creating steps from flowing.

- **Recommendation 2(a):** The goal is not to eliminate all possible impediments. This study shows that impediments continually emerge to impact flow. A goal for organizations should be developing the capability to sense their circumstances and adapt to these changing conditions continually.
- **Recommendation 2(b):** Emergent data can be more useful than predetermined categories of impediments. Categories are useful as a starting point to help form the habit of seeing impediments to flow. However, organizations should expect that different categories and different ways of understanding impediments will emerge as people in the organization work to identify and make sense of impediments in their particular context.
- **Recommendation 2(c):** Identifying perceived impediments is useful but not sufficient. In general, understanding the contributing factors to impediments and the effects of impediments, in the context of the system patterns, is key to making the value-creating steps flow.

3. **Make sense of impediments and their context to determine appropriate courses of action.**

- **Recommendation 3(a):** Sensemaking provides the capability for organizations to monitor complex human systems and to inquire into specific areas of concern in order to determine the context of impediments before acting to resolve them.
- **Recommendation 3(b):** It can be useful to understand the scope and impact of impediments and to understand who needs to be involved in resolving impediments.
- **Recommendation 3(c):** Recognize that impediments can originate at different levels in the system. Although people might experience the impact of impediments at one level, the impediment might originate from patterns in the level(s) above or below where people experience the impact. Generally, people should develop a coherent approach to resolving impediments at the level where they can shift the patterns.

4. **Resolve impediments to improve flow and achieve desired outcomes.**

- **Recommendation 4(a):** Consider exploring and adopting approaches that are suited to different contexts. This study explored in detail four such approaches that are beneficial for different impediments in different contexts. These can serve as illustrative examples.
- **Recommendation 4(b):** Recognize that some impediments are avoidable, while others are unavoidable in certain contexts, and use this knowledge to inform impediment resolution approaches. Impediments might be unavoidable either because technologies or approaches do not yet exist to prevent those impediments, or because they exist but are not yet in use in the organization.
- **Recommendation 4(c):** Be deliberate about choosing interventions. There are different types of interventions that organizations can employ to resolve impediments and shift system patterns, i.e., *container* interventions, *difference* interventions, and *exchange* interventions. Recognize that

resolving one impediment may require multiple interventions of different types.

- **Recommendation 4(d):** Expect that the original impediments will change over time as people work to resolve them. The particular approach selected to resolve impediments influences on how people perceive the problem, and on how the impediments change.
- **Recommendation 4(e):** Recognize that ultimately, some impediments are problems to be solved, while others are indicators of patterns that need to shift in the organization. Use appropriate approaches in either case.

7.5 Limitations and Reflections

This section presents the limitations related to both the study itself and the research approach.

7.5.1 Limitations of the Study

The researcher identified six limitations to the study. These are:

- The focus of the study
- The unit of analysis
- Influence of software development methods
- Number of flow metrics used
- Approaches to resolving impediments
- The conceptual framework

The focus of the study

The organizations in the study are all large-scale software development organizations, building correspondingly large-scale products and systems. Many of the challenges faced by such organizations may not be relatable to small organizations.

The unit of analysis

As described in section 5.4, the unit of analysis was the product development organization. An organization in this study typically is responsible for and represents a single product or system, or a set of related products and systems. Each company in this study contained multiple such organizations. This decision means the research did not study the company as a whole. Nor did it extend to studying individual teams or people within a product development organization.

Influence of software development methods

All of the organizations in the study use some form of agile development methods. This study is agnostic towards specific agile methods and software development methods in general. This study does not evaluate the success or maturity of agile method adoption in the organizations studied. Nor does this study attempt to generalize relationships between impediments and agile practices, and how struggles with a particular method or process might exacerbate impediments to flow.

Number of flow metrics used

As noted in Chapter 3, there are a large number of metrics from which to choose. This study chose to focus on just three: throughput, cycle time, and cumulative flow. It is possible that using other flow metrics would yield new perspectives on flow and impediments in organizations. Also, this study used metrics as qualitative indicators of the presence of impediments.

Approaches to resolving impediments

As outlined in the literature review, there are many approaches that organizations take to resolve impediments. This research selected four to study in detail. Two of these approaches come from lean-based management practice and two from complexity-based management. Limiting the selection to four approaches was necessary to bound the research. It is

possible that selecting other approaches could yield additional insights into how organizations resolve impediments.

The conceptual framework

The conceptual framework for this study contains a relatively new perspective for investigating impediments to flow in software development organizations. Before this study, limited research existed on sensemaking and complexity-informed management practices in the context of software product development. Consequently, this study is one of the first to assess impediments from these perspectives. Both sensemaking and complex adaptive systems are deep fields of study in their own right. It was necessary to bound the research by limiting the scope of sensemaking and complex adaptive systems within the context of this study.

7.5.2 Limitations of the Research Approach

The researcher identified five limitations to the research approach. These are:

- Generalizability of the research findings
- Data collection periods
- Data collection methods
- The dynamic setting of the study
- The volume of focus group data collected

Generalizability of the research findings

A primary concern with case study research is the generalizability of the research findings, as discussed in section 5.8. As with any exploratory case analysis, the cases in this study have limited generalizability, although thematic generalizability is possible (Creswell, 2013). As this research is exploratory, the study does not attempt to generalize the findings, but instead to present the findings from each case and identify similarities and differences across the organizations studied. This study used a multiple-case study approach by examining ten organizations in their natural setting.

While the number of cases may be considered a limitation in the context of an exploratory study, conducting three research phases with a total of ten organizations from three different companies provided sufficient insight until a point of ‘theoretical saturation’ had been reached (section 5.8).

Data collection periods

The data collection period formed a boundary for this study, which can be considered a limitation. The researcher attempted to mitigate this limitation by collecting data over three distinct phases, as described in section 5.5.2. Even so, the data collection period of each research phase sets a frame of reference for the study and also reflects the perspective of participants at a point in time (Pettigrew, 1990).

Data collection methods

Every research method has its challenges and limitations. Chapter 5 described those challenges associated with focus group research (section 5.4.2), narrative research (section 5.4.3), and case study research (section 5.4.4). Also, the concept under study was potentially sensitive. While some respondents were forthcoming about that sensitivity, it is possible that others were not forthcoming with specific information that would have been relevant to the findings. This topic is discussed further in section 7.3.1. Also, as many of the concepts studied were subtle or dynamic, it is possible that they were not apparent to the researcher. For example, a different researcher may have noted a different set of patterns.

The dynamic setting of the study

This study took place with software development organizations in an industry setting. This setting means the study was not in a controlled, reproducible environment. As is typical in large product development organizations, these organizations were dealing with a continually evolving reality that meant they needed to respond to market dynamics, customer needs, technology complexity, and other challenges. Throughout this study, these organizations dealt with people joining and leaving, changes or

evolutions in strategy, acquisitions, mergers, reorganizations, and the general ups and downs of doing business in a dynamic environment. This formed part of the context as participants engaged in the study. Studying these organizations over a more extended period may have provided additional insights that were not evident at the time of data collection.

The volume of focus group data collected

The researcher conducted a large number of focus groups for this study. In hindsight, more than was perhaps necessary. Although this led to a rich set of data to draw from, even with a purposive sampling strategy (section 5.7.6), the researcher excluded several focus groups from this study's findings. The time to conduct these focus groups was extensive. The time to analyze and interpret the findings was even more so. In hindsight, the researcher would have been more deliberate in choosing fewer focus groups.

7.6 Trajectories for Future Research

The findings from this study point to several possible trajectories for future research. This section describes six of these, including:

- Study impediments using other research methods
- Correlating flow metrics with improvements
- Explore impediments at additional units of analysis
- Study impediments in organizations of different sizes
- Explore the relationships with software development methods
- Apply the conceptual framework in other contexts

Study impediments using other research methods

This study used a qualitative methods approach. Future research could conduct a quantitative or mixed-method study that explores other aspects of impediments. In particular, this study explored flow metrics from a qualitative perspective to gain insights on impediments. A rigorous quantitative analysis may provide new perspectives on impediments to flow.

Correlating flow metrics with improvements

This study did not attempt to correlate specific improvements in flow metrics when resolving impediments. To do so would require a different study design, which could be a topic for future research.

Explore impediments at additional units of analysis

As described in section 5.5.1, this study used the product development organization as the unit of analysis. These organizations were all globally distributed and multi-cultural. Future studies could explore impediments at the level of a team or set of teams, at the level of an entire supply chain ecosystem including customers, partners, and suppliers, or explore impediments in co-located teams.

Study impediments in organizations of different sizes

This research intentionally studied impediments in large product development organizations. Future studies could explore the same research questions in small- and medium-sized organizations.

Explore the relationships with software development methods

This study is intentionally agnostic about which software development methods an organization was using or attempting to use. Future studies could elaborate on specific challenges associated with impediments, where organizations and teams are using agile, lean, DevOps, SRE, Continuous Deployment, and other forms of continuous development. Similarly, future studies could explore the relationship between the success or maturity of method adoption in organizations and their ability to manage impediments.

Apply the conceptual framework in other contexts

This study developed and validated a conceptual framework and applied it in the context of large-scale, globally distributed software product development organizations. Future studies could apply the framework in other contexts, e.g., small- or medium-sized organizations, in service

organizations, or in-house IT or enterprise development groups. Future studies could also explore the use of the framework in contexts other than software development, e.g., in manufacturing, education, healthcare, or government organizations.

References

- ABBOTT, H. P. 2008. *The Cambridge Introduction to Narrative*, Cambridge University Press.
- ABRAHAMSSON, P., SALO, O., RONKAINEN, J. & WARSTEN, J. 2002. *Agile Software Development Methods: Review and Analysis*, Espoo.
- ACKOFF, R. 1994. *The learning and legacy of Dr. W. Edwards Deming* [Online]. Available: <https://www.youtube.com/watch?v=OqFeIG8aPPk> [Accessed Sept. 11 2016].
- ALI, N. B., PETERSEN, K. & SCHNEIDER, K. 2016. FLOW-assisted value stream mapping in the early phases of large-scale software development. *The Journal of Systems & Software*, 111, 213-227.
- ANAF, S., DRUMMOND, C. & SHEPPARD, L. A. 2007. Combining Case Study Research and Systems Theory as a Heuristic Model. *Qualitative Health Research*, 17, 1309-1315.
- ANCONA, D. 2012. Sensemaking: Framing and action in the unknown. In: SNOOK, S., NOHRIA, N. & KHURANA, R. (eds.) *The Handbook for Teaching Leadership: Knowing, Doing, and Being*. SAGE Publications Inc.
- ANDERSON, D. J. 2010. *Kanban: Successful Evolutionary Change for Your Technology Business*, Sequim, Washington, Blue Hole Press.
- ANDERSON, D. J. 2013. *Lean Software Development*, Seattle, WA, Lean Kanban University (LKU).
- ANDERSON, M. H. 2006. How Can We Know What We Think Until We See What We Said?: A Citation and Citation Context Analysis of Karl Weick's The Social Psychology of Organizing. *Organization Studies*, 27, 1675-1692.
- ANDERSON, R., CRABTREE, B. F., STEELE, D. J. & JR., R. R. M. 2005. Case Study Research: the view from complexity science. *Qual Health Research*, 15, 669-685.
- ARLEROOTH, J. & KRISTENSSON, H. 2011. *Waste in Lean Construction - A case study of a PEAB construction site and the development of a Lean Construction Tool*. Master of Science Thesis in the Master of Supply Chain Management MSc Thesis, Chalmers University of Technology.
- ARROW, H., MCGRATH, J. E. & BERDAHL, J. L. 2000. *Small groups as complex systems: Formation, coordination, development, and adaptation*, Sage Publications.
- ARTINGER, F., PETERSEN, M., GIGERENZER, G. & WEIBLER, J. 2015. Heuristics as adaptive decision strategies in management. *Journal of Organizational Behavior*, 36, S33-S52.
- BAKHACHE, N., MICHAEL, S., ROUPETZ, S., GARBERN, S., BERGQUIST, H., DAVISON, C. & BARTELS, S. 2017. Implementation of a SenseMaker® research project among Syrian refugees in Lebanon. *Global Health Action*, 10.
- BALDERSTONE, S. J. & MABIN, V. J. A Review of Goldratt's Theory of Constraints (TOC)—lessons from the international literature. Annual Conference of the Operational Research Society of New Zealand, 1998. 19-30.
- BARBACCI, M., KLEIN, M. H., LONGSTAFF, T. A., WEINSTOCK, C. B., LONGSTAFF, T. H. & MILLER, T. R. 1995. Quality Attributes. *SEI Technical Reports*. Software Engineering Institute.
- BARTELS, S. A., MICHAEL, S., ROUPETZ, S., GARBERN, S., KILZAR, L., BERGQUIST, H., BAKHACHE, N., DAVISON, C. & BUNTING, A. 2018. Making sense of child, early and forced marriage among Syrian refugee girls: a mixed methods study in Lebanon. *BMJ Global Health*, 3.

- BARTHES, R. 1982. Introduction to the Structural Analysis of Narratives. *In*: SONTAG, S. (ed.) *Image-Music-Text*. New York: Hill & Wang.
- BARTSCHT, J. 2015. Why systems must explore the unknown to survive in VUCA environments. *Kybernetes*, 44, 253-270.
- BASILI, V. R. The Role of Experimentation in Software Engineering: Past, Current, and Future. Proceedings of the 18th International Conference on Software Engineering (ICSE 1996), 1996.
- BASILI, V. R., CALDIERA, G., ROMBACH, D. & MARCINIAK, J. J. 1994. Goal, Question, Metric Paradigm. *Encyclopedia of Software Engineering*. John Wiley & Sons.
- BASILI, V. R., SELBY, R. W. & HUTCHENS, D. H. 1986. Experimentation in software engineering. *IEEE Transactions on Software Engineering*, SE-12, 733-743.
- BASIRI, A., BEHNAM, N., DE ROOIJ, R., HOCHSTEIN, L., KOSEWSKI, L., REYNOLDS, J. & ROSENTHAL, C. 2016. Chaos engineering. *IEEE Software*, 33, 35-41.
- BECK, K. 2000. *Extreme Programming Explained: Embrace Change*, Reading, Massachusetts, Addison-Wesley.
- BECK, K. & ANDRES, C. 2005. *Extreme programming explained : embrace change, 2e*, Boston, MA, Addison-Wesley.
- BECK, K., BEEDLE, M., BENNEKUM, A. V., COCKBURN, A., CUNNINGHAM, W., FOWLER, M., GRENNING, J., HIGHSMITH, J., HUNT, A., JEFFRIES, R., KERN, J., MARICK, B., MARTIN, R. C., MELLOR, S., SCHWABER, K., SUTHERLAND, J. & THOMAS, D. 2001. *Manifesto for Agile Software Development* [Online]. Snowbird, UT. Available: <http://agilemanifesto.org/> [Accessed August 2011].
- BECK, K. & FOWLER, M. 2001. *Planning extreme programming*, Addison-Wesley Professional.
- BEHUTIYE, W. N., RODRÍGUEZ, P., OIVO, M. & TOSUN, A. 2017. Analyzing the concept of technical debt in the context of agile software development: A systematic literature review. *Information and Software Technology*, 82, 139-158.
- BESKER, T., MARTINI, A. & BOSCH, J. 2019. Software developer productivity loss due to technical debt—A replication and extension study examining developers' development work. *Journal of Systems and Software*, 156, 41-61.
- BEYER, B., JONES, C., PETO, J. & MURPHY, N. R. (eds.) 2016. *Site Reliability Engineering: How Google Runs Production Systems*: O'Reilly.
- BIRKELAND, J. O. 2010. From a Timebox Tangle to a More Flexible Flow. *In*: SILLITTI, A., MARTIN, A., WANG, X. & WHITWORTH, E. (eds.) *Agile Processes in Software Engineering and Extreme Programming, 11th International Confererence (XP2010)*. Trondheim, Norway: Springer.
- BOCK, L. 2015. *Work Rules! Insights from inside Google that will transform how you live and lead*, London, UK, John Murray Publishers.
- BOEHM, B. 2006a. Some future trends and implications for systems and software engineering processes. *Systems Engineering*, 9, 1-19.
- BOEHM, B. 2006b. A view of 20th and 21st century software engineering. *28th international conference on Software engineering*. ACM.
- BOESEN, N. 2010. Looking for Change Beyond the Boundaries, the Formal and the Functional. *In*: UBELS, J., ACQUAYE-BADDOO, N.-A. & FOWLER, A. (eds.) *Capacity Development in Practice*. London, Washington, DC: Earthscan.
- BOJE, D. M. 2008. *Storytelling Organizations*, Thousand Oaks, CA, SAGE Publications, Inc.

- BOJE, D. M., OSWICK, C. & FORD, J. D. 2004. Language and organization: The doing of discourse. *Academy of Management Review*, 29, 571-577.
- BORTOLOTTI, T., BOSCARI, S. & DANESE, P. 2015. Successful lean implementation: Organizational culture and soft lean practices. *International Journal of Production Economics*, 160, 182-201.
- BROOKS, F. P. 1995. *The mythical man-month : essays on software engineering*, Reading, Mass., Addison-Wesley Pub. Co.
- BROWN, A. D., COLVILLE, I. & PYE, A. 2014. Making Sense of Sensemaking in Organization Studies. *Organization Studies*.
- BURLINGHAM, B. 2015. How to build a company that will be around in 2115. *Inc. Magazine*.
- BURMAN, C. J. & APHANE, M. A. 2016. Leadership emergence: the application of the Cynefin framework during a bio-social HIV/AIDS risk-reduction pilot. *African Journal of AIDS Research*, 15, 249-260.
- BURRELL, G. & MORGAN, G. 2017. *Sociological paradigms and organisational analysis: Elements of the sociology of corporate life*, Routledge.
- BURROWS, M. 2014. *Kanban from the Inside*, Sequim, WA, Blue Hole Press.
- CA TECHNOLOGIES 2017. The Impact of Agile. Quantified.
- CAIN, B. G., COPLIEN, J. O. & HARRISON, N. B. 1996. Social patterns in productive software development organizations. *Annals of Software Engineering*, 2, 259-286.
- CAMERON, L. M. 2018. What to Know About the Scientist Who Invented the Term "Software Engineering". *ComputingEdge*. IEEE Computer Society.
- CARACCIOLO, M. 2012. Narrative, meaning, interpretation: an enactivist approach. *Phenomenology and the Cognitive Sciences*, 11, 367-384.
- CARMODY, B. 2015. Why 96 Percent of Businesses Fail Within 10 Years. *Inc.*
- CARROLL, N., O'CONNOR, M. & EDISON, H. 2018. The Identification and Classification of Impediments in Software Flow. *Twenty-fourth Americas Conference on Information Systems*. New Orleans: Association for Information Systems.
- CHANDRASEKAR, A., SUDHARAJESH, M., RAJESH, M. P., PROF, A. & COLLEGE, E. 2014. A Research Study on Software Quality Attributes. *International Journal of Scientific and Research Publications*, 4.
- CHIA, R. 2003. Organization theory as a postmodern science. In: TSOUKAS, H. & KNUDSEN, C. (eds.) *The Oxford handbook of organization theory*. New York: Oxford University Press.
- CHUI, G. 2000. Unified Theory is Getting Closer, Hawking Predicts. *San Jose Mercury News*, Jan 23, 2000.
- COCKBURN, A. 2000. Characterizing people as non-linear, first-order components in software development. *4th International Multiconference on Systemics, Cybernetics and Informatics (SCI2000)*. Orlando, Florida, USA.
- COCKBURN, A. 2005. *Crystal clear : a human-powered methodology for small teams*, Boston, Addison-Wesley.
- COCKBURN, A. & HIGHSMITH, J. 2001. Agile software development, the people factor. *Computer*, 34, 131-133.
- COHN, M. 2004. *User stories applied : for agile software development*, Boston, Addison-Wesley.
- COHN, M. 2010. *Succeeding with Agile: Software Development Using Scrum* Upper Saddle River, NJ, USA, Addison-Wesley.
- COLLABNET VERSIONONE 2018. 12th Annual State of Agile Report.

- COLLABNET VERSIONONE 2019. 13th Annual State of Agile Report.
- COLLINS, M. E., BLOCK, S. D., ARNOLD, R. M. & CHRISTAKIS, N. A. 2009. On the prospects for a blame-free medical culture. *Social science & medicine*, 69, 1287-1290.
- CONBOY, K. 2009. Agility From First Principles: Reconstructing The Concept of Agility in Information Systems Development. *Information Systems Research*, 20, 329-354.
- CONBOY, K. & CARROLL, N. 2019. Implementing Large-Scale Agile Frameworks: Challenges and Recommendations. *IEEE Software*, 36, 44-50.
- CONBOY, K., COYLE, S., WANG, X. & PIKKARAINEN, M. 2011. People over process: key people challenges in agile development. *IEEE Software*, 28, 48-57.
- CONBOY, K. & FITZGERALD, B. 2007. The views of experts on the current state of agile method tailoring. *IFIP International Working Conference on Organizational Dynamics of Technology-Based Innovation*. Springer.
- COPLIEN, J. O. Borland software craftsmanship: A new look at process, quality and productivity. Proceedings of the 5th Annual Borland International Conference, 1994.
- COPLIEN, J. O. & HARRISON, N. B. 2004. *Organizational patterns of agile software development*, Prentice-Hall, Inc.
- COYLE, S. 2013. *Group Decision Quality in Agile Software Development: The Impact of Contribution Behaviours*. PhD Thesis, National University of Ireland, Galway.
- CRESWELL, J. W. 2013. *Qualitative Inquiry & Research Design: Choosing among five approaches*. 3rd ed. Los Angeles: SAGE Publications, Inc.
- CRESWELL, J. W. 2014. *Research Design: Qualitative, Quantitative and Mixed Methods*, Thousand Oaks, CA, SAGE Publications, Inc.
- CSIKSZENTMIHALYI, M. 1997. Flow and the psychology of discovery and invention. *HarperPerennial, New York*, 39.
- CSIKSZENTMIHALYI, M. & LEFEVRE, J. 1989. Optimal experience in work and leisure. *Journal of personality and social psychology*, 56, 815.
- DAVIS, J. & DANIELS, K. 2016. *Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale*, O'Reilly.
- DEKKER, S. 2016. *Just culture: Balancing safety and accountability*, CRC Press.
- DEMARCO, T. 2001. *Slack: getting past burnout, busywork, and the myth of total efficiency*, New York, Broadway Books.
- DEMING, W. E. 1986. *Out of the Crisis*, Cambridge, MA, The MIT Press.
- DEMING, W. E. 2013. *The Essential Deming: Leadership Principles from the Father of Quality*, New York, McGraw Hill.
- DENNEHY, D. & CONBOY, K. 2017. Going with the flow: An activity theory analysis of flow techniques in software development. *Journal of Systems and Software*, 133, 160-173.
- DENNEHY, D. & CONBOY, K. 2018. Identifying Challenges and a Research Agenda for Flow in Software Project Management. *Project Management Journal*, 49, 103-118.
- DENNING, S. 2005. *The leader's guide to storytelling: Mastering the art and discipline of business narrative*, John Wiley & Sons.
- DEPREZ, S., HUYGHE, C., GOOL, C. V. & VREDESEILANDEN, M. 2012. Using SenseMaker to measure, learn and communicate about smallholder farmer inclusion. *Thematic learning programme on planning, monitoring and evaluation of complex processes of social change*. Vredeseilanden (VECO).

- DERVIN, B. 1998. Sense - making theory and practice: an overview of user interests in knowledge seeking and use. *Journal of Knowledge Management*, 2, 36-46.
- DERVIN, B., FOREMAN-WERNET, L. & LAUTERBACH, E. 2003. *Sense-Making methodology reader: Selected writings of Brenda Dervin*, Hampton Pr.
- DEVITA, M., BRAITHWAITE, R., MAHIDHARA, R., STUART, S., FORAIDA, M. & SIMMONS, R. 2004. Use of medical emergency team responses to reduce hospital cardiopulmonary arrests. *BMJ Quality & Safety*, 13, 251-254.
- DICKENS, P. M. 2012. *Facilitating Emergence: Complex, Adaptive Systems Theory and the Shape of Change*. PhD Thesis, Antioch University.
- DICKSON, E. W., ANGUELOV, Z., VETTERICK, D., ELLER, A. & SINGH, S. 2009. Use of Lean in the Emergency Department: A Case Series of 4 Hospitals. *Annals of Emergency Medicine*, 54, 504-510.
- DINGSØYR, T., FALESSI, D. & POWER, K. 2019. Agile Development at Scale: The Next Frontier. *IEEE Software*, 36, 30-38.
- DINGSØYR, T. & LASSENIUS, C. 2016. Emerging themes in agile software development: Introduction to the special section on continuous value delivery. *Information and Software Technology*, 77, 56-60.
- DITTRICH, Y., NØRBJERG, J., TELL, P. & BENDIX, L. Researching cooperation and communication in continuous software engineering. 2018 IEEE/ACM 11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2018. IEEE, 87-90.
- DOOLEY, K. & VEN, A. H. V. D. 1999. Explaining Complex Organizational Dynamics. *Organization Science*, 10, 358-372.
- DOOLEY, K. J. 1997. A Complex Adaptive Systems Model of Organization Change. *Nonlinear Dynamics, Psychology and Life Sciences*, 1, 69-97.
- DUBE, A., BARTELS, S. A., MICHAEL, S. & MICHAELSON, V. 2019. Positive worry and negative hope: paradoxical perceptions of the experiences of Syrian refugee girls in Lebanon. *Journal of International Humanitarian Action*, 4, 1-15.
- DUNSTAN, K. 2016. *Exploring anticipatory emotions and their role in self-perceived B2B salesperson effectiveness*. Curtin University.
- DUVALL, P. M., MATYAS, S. & GLOVER, A. 2007. *Continuous Integration: Improving Software Quality and Reducing Risk*, Boston, MA, Addison-Wesley.
- ECKSTEIN, J. 2015. *Retrospectives for Organizational Change: An Agile Approach*, Leanpub.
- EDMONDSON, A. 1999. Psychological Safety and Learning Behavior in Work Teams. *Administrative Science Quarterly*, 44, 350-383.
- EDMONDSON, A. C., KRAMER, R. M. & COOK, K. S. 2004. Psychological safety, trust, and learning in organizations: A group-level lens. *Trust and distrust in organizations: Dilemmas and approaches*.
- ELFORD, W. 2012. A multi-ontology view of ergonomics: applying the Cynefin Framework to improve theory and practice. *Work*, 41, 812-817.
- EOYANG, G. H. 2001. *Conditions for Self-Organizing in Human Systems*. Doctor of Philosophy in Human Systems Dynamics PhD Thesis, The Union Institute and University.
- EOYANG, G. H. 2004. The practitioner's landscape. *Emergence: Complexity & Organization*, 6, 55-60.
- EOYANG, G. H. 2013. Human Systems Dynamics Professional Certification. Cohort 32 - Roffey Park, UK: HSD Institute.
- EOYANG, G. H. & HOLLADAY, R. J. 2013. *Adaptive Action: Leveraging Uncertainty in Your Organization*, Stanford, California, Stanford University Press.

- FABIJAN, A., DMITRIEV, P., OLSSON, H. H. & BOSCH, J. The evolution of continuous experimentation in software product development: from data to a data-driven organization at scale. Proceedings of the 39th International Conference on Software Engineering, 2017. IEEE Press, 770-780.
- FERN, E. F. 1982. The Use of Focus Groups for Idea Generation: The Effects of Group Size, Acquaintanceship, and Moderator on Response Quantity and Quality. *Journal of Marketing Research*, XIX, 1-13.
- FEYH, M. & PETERSEN, K. 2013. Lean software development measures and indicators-a systematic mapping study. *4th International Conference on Lean Enterprise Software and Systems (LESS 2013)*. Galway, Ireland: Springer.
- FINOCCHIARO, M. A. 1981. Fallacies and the Evaluation of Reasoning. *American Philosophical Quarterly*, 18, 13-22.
- FITZGERALD, B. & STOL, K.-J. 2015. Continuous Software Engineering: a roadmap and agenda. *The Journal of Systems and Software*.
- FITZGERALD, B. & STOL, K.-J. 2017. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, 123, 176-189.
- FOREMAN-WERNET, L. 2003. Rethinking communication: Introducing the sense-making methodology. In: DERVIN, B., FOREMAN-WERNET, L. & LAUTERBACH, E. (eds.) *Sense-making methodology reader: Selected writings of Brenda Dervin*. Cresskill, NJ: Hampton Press, Inc.
- FORSGREN, N., HUMBLE, J., KERSTEN, N., BROWN, A. & KIM, G. 2017. State of DevOps Report. Puppet, IT Revolution.
- FORSGREN, N., HUMBLE, J. & KIM, G. 2018a. Accelerate: State of DevOps - strategies for a new economy. DORA.
- FORSGREN, N., HUMBLE, J. & KIM, G. 2018b. *Accelerate: The Science Behind DevOps : Building and Scaling High Performing Technology Organizations*, IT Revolution Press.
- FOWLER, M. 2019. *Refactoring: Improving the Design of Existing Code, Second Edition*, Addison-Wesley.
- FRANK, K. A. & FAHRBACH, K. 1999. Organization culture as a complex system: Balance and information in models of influence and selection. *Organization Science*, 10, 253-277.
- FREEMAN, P. 1992. Lean concepts in software engineering. *IPSS-Europe International Conference on Lean Software Development*. Stuttgart, Germany.
- FRENCH, S. 2013. *Cynefin, statistics and decision analysis*, Palgrave Macmillan.
- FRENCH, S. 2015. Cynefin: uncertainty, small worlds and scenarios. *Journal of the Operational Research Society*.
- GABBAY, D. & WOODS, J. 2006. Advice on Abductive Logic. *Logic Journal of the IGPL*, 14, 189-219.
- GALL, J. 2002. *The systems bible: the beginner's guide to systems large and small*, General Systemantics Press.
- GARVIN, D. A. 2013. How Google Sold Its Engineers on Management. *Harvard Business Review*.
- GATLIN, L. L. 1972. Information theory and the living system.
- GITTLESON, K. 2012. Can a company live forever? *BBC News*, January 19, 2012.
- GLASER, B. G. & STRAUSS, A. L. 1967. *The discovery of grounded theory: Strategies for qualitative research*, New York, Aldine de Gruyter.
- GOLDRATT, E. M. 1990. *Theory of constraints*, North River Croton-on-Hudson.

- GOLDRATT, E. M. & COX, J. 2004. *The goal : a process of ongoing improvement, 3rd revised edition*, Aldershot, Gower.
- GOODBURN, M. 2015. What is the life expectancy of your company? *World Economic Forum*.
- GORTON, I. 2011. Software quality attributes. *Essential Software Architecture*. Springer.
- GORZEŃ-MITKA, I. & OKRĘGLICKA, M. 2014. Improving Decision Making in Complexity Environment. *21st International Economic Conference 2014, IECS 2014*. Sibiu, Romania.
- GRAEBSCH, M., SEERING, W. P. & LINDEMANN, U. 2007. Assessing Information Waste in Lean Product Development. *16th International Conference on Engineering Design, ICED'07*. CITE DES SCIENCES ET DE L'INDUSTRIE, PARIS, FRANCE.
- GRAZIOTIN, D., WANG, X. & ABRAHAMSSON, P. 2014. Happy software developers solve problems better: psychological measurements in empirical software engineering. *PeerJ*, 2, e289.
- GREENING, D. R. 2015. Agile Enterprise Metrics. *48th Hawaii International Conference on System Sciences (HICSS), 2015*. Kauai, HI: IEEE.
- GREGOR, S. 2006. The Nature of Theory in Information Systems. *MIS Quarterly*, 30, 611-642.
- GROCER, S. 2018. A Record \$2.5 Trillion in Mergers Were Announced in the First Half of 2018. *The New York Times*, July 3, 2018.
- GROUSE, L. 2016. Post hoc ergo propter hoc. *Journal of Thoracic Disease*, 8, E511-E512.
- GUIJT, I. 2016. Innovation in Evaluation: Using SenseMaker to assess the inclusion of smallholder farmers in modern markets. In: BELL, S. & AGGLETON, P. (eds.) *Monitoring and Evaluation in Health and Social Development: Interpretive and Ethnographic Perspectives*. Taylor & Francis.
- GUO, Y., SPÍNOLA, R. O. & SEAMAN, C. 2016. Exploring the costs of technical debt management - a case study. *Empirical Software Engineering*, 21, 159-182.
- GUPTA, M. C. & BOYD, L. H. 2008. Theory of constraints: a theory for operations management. *International Journal of Operations & Production Management*, 28, 991-1012.
- HARRISON, N. B. & COPLIEN, J. O. 1996. Patterns of productive software organizations. *Bell Labs Technical Journal*, 1, 138-145.
- HIBBS, C., JEWETT, S. & SULLIVAN, M. 2009. *The Art of Lean Software Development*, O'Reilly Media, Inc.
- HIGHSMITH, J. A. 2002. *Agile software development ecosystems*, Boston, Addison-Wesley.
- HOLLAND, J. H. 2006. Studying Complex Adaptive Systems. *Journal of System Science & Complexity*, 19.
- HOWARD, E., HUBELBANK, J. & MOORE, P. 1989. Employer evaluation of graduates: use of the focus group. *Nurse Educator*, 14, 38-41.
- HOYER, R. W. & HOYER, B. B. Y. 2001. What is Quality? *Quality Progress*, 34, 52-62.
- HUMBLE, J. & FARLEY, D. 2010. *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, Boston MA, Addison-Wesley Professional.
- HUMMELBRUNNER, R. & JONES, H. 2013. A guide to managing in the face of complexity. *ODI Working Papers*. London, UK: Overseas Development Institute.
- IIVARI, J. & HUISMAN, M. 2007. The relationship between organizational culture and the deployment of systems development methodologies. *MIS Quarterly*, 35-58.

- IKONEN, M. 2011. *Lean Thinking in Software Development: Impacts of Kanban on Projects*. PhD Thesis, University of Helsinki, Finland.
- IKONEN, M., KETTUNEN, P., OZA, N. & ABRAHAMSSON, P. 2010. Exploring the sources of waste in Kanban software development projects. *36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE.
- IMAA. 2019. *M&A Statistics* [Online]. Institute of Mergers, Acquisitions and Alliances (IMAA). Available: <https://imaa-institute.org/mergers-and-acquisitions-statistics/> [Accessed July 7 2019].
- JANSSON, A. & AXELSSON, A. Knowledge Elicitation in Naturalistic Decision Making: Collegial Verbalisation with "Conspective Protocols". 13th Bi-annual Conference on Naturalistic Decision Making (NDM), 20-23 June 2017, Bath, UK., 2017. 87-93.
- JENNEX, M. E. Modeling emergency response systems. 2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07), 2007. IEEE, 22-22.
- JIMMERSON, C., WEBER, D. & SOBEK, D. K. 2005. Reducing Waste and Errors: Piloting Lean Principles at Intermountain Healthcare. *Joint Commission Journal on Quality and Patient Safety*, 31, 249-257.
- JOHANSEN, B. & EUCHNER, J. 2013. Navigating the VUCA World. *Research-Technology Management*, 56, 10-15.
- JUARRERO, A. 1999. *Dynamics in action: Intentional behavior as a complex system*, Citeseer.
- JUARRERO, A. 2009. Top-down causation and autonomy in complex systems. In: MURPHY, N., ELLIS, G. F. R. & O'CONNOR, T. (eds.) *Downward causation and the neurobiology of free will*. Berlin: Springer.
- JUARRERO-ROQUÉ, A. 1991. Fail-safe versus safe-fail: Suggestions toward an evolutionary model of justice. *Texas Law Review*, 69.
- KAY, J. 2011. *Obliquity: Why our goals are best achieved indirectly*, Profile Books.
- KELLEY, H. H., BERSCHIED, E., CHRISTENSEN, A., HARVEY, J. H., HUSTON, T. L., LEVINGER, G., MCCLINTOCK, E., PEPLAU, L. A. & PETERSON, D. R. 1983. *Close relationships*, Freeman New York.
- KEMPERMANN, G. 2017. Cynefin as Reference Framework to Facilitate Insight and Decision-Making in Complex Contexts of Biomedical Research. *Frontiers in Neuroscience*, 11.
- KERIEVSKY, J. 2016a. An Introduction to Modern Agile. *InfoQ*.
- KERIEVSKY, J. 2016b. *Modern Agile* [Online]. Available: <http://modernagile.org> [Accessed June 1 2019].
- KHONDKER, M.-E. J., ALI, H., UPRAITY, V., GURUNG, S., DHAR, G. C. & BELTON, B. 2018. Making sense of the market: Assessing the participatory market chain approach to aquaculture value chain development in Nepal and Bangladesh. *Aquaculture*, 493, 395-405.
- KIM, G., BEHR, K. & SPAFFORD, K. 2014. *The phoenix project: A novel about IT, DevOps, and helping your business win*, IT Revolution.
- KITZINGER, J. 1998. The methodology of Focus Groups: the importance of interaction between research participants. *Sociology of Health & Illness*, 16, 103-121.
- KLEIN, G., MOON, B. & HOFFMAN, R. R. 2006. Making Sense of Sensemaking 1: Alternative Perspectives. *IEEE Intelligent Systems*, 21, 70-73.
- KLEIN, H. K. & MYERS, M. D. 1999. A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems. *MIS Quarterly*, 23, 67-94.
- KNIBERG, H. 2011. Lean from the Trenches - Managing Large-Scale Projects With Kanban.

- KOCIATKIEWICZ, J. & KOSTERA, M. 2016. Grand plots of management bestsellers: Learning from narrative and thematic coherence. *Management Learning*, 47, 324-342.
- KOHN, L. T. 1997. *Methods in Case Study Analysis*. Center for Studying Health System Change.
- KOVACH, J. V., CUDNEY, E. A. & ELROD, C. C. 2011. The use of continuous improvement techniques: A survey-based study of current practices. *International Journal of Engineering, Science and Technology*, 3, 89-100.
- KRENTEL, A., DAMAYANTI, R., TITALEY, C. R., SUHARNO, N., BRADLEY, M. & LYNAM, T. 2016. Improving Coverage and Compliance in Mass Drug Administration for the Elimination of LF in Two 'Endgame' Districts in Indonesia Using Micronarrative Surveys. *PLoS Neglected Tropical Diseases*, 10, e0005027.
- KUPIAINEN, E., MÄNTYLÄ, M. V. & ITKONEN, J. 2015. Using metrics in Agile and Lean Software Development – A systematic literature review of industrial studies. *Information and Software Technology*, 62, 143-163.
- KURTZ, C. F. 2014. *Working With Stories in Your Community or Organization: Participatory Narrative Inquiry*.
- KURTZ, C. F. & SNOWDEN, D. J. 2003. The new dynamics of strategy: Sense-making in a complex and complicated world. *IBM Systems Journal*, 42, 462-483.
- KURTZ, C. F. & SNOWDEN, D. J. 2006. Bramble Bushes in a Thicket: Narrative and the intangibles of learning networks. In: GIBBERT, M. & DURAND, T. (eds.) *Strategic Networks: Learning to Compete*. Blackwell.
- LABOV, W. 2008. Oral narratives of personal experience. *Cambridge encyclopedia of the language sciences*, 546-548.
- LABOV, W. & WALETZKY, J. 1997. Narrative analysis: oral versions of personal experience.
- LADAS, C. 2009. *Scrumban: Essays on Kanban Systems for Lean Software Development*, Modus Cooperandi Press.
- LANGER, L. L. 1991. *Holocaust testimonies: The ruins of memory*, New Haven & London, Yale University Press.
- LAPINSKI, A. R., HORMAN, M. J. & RILEY, D. R. 2006. Lean Processes for Sustainable Project Delivery. *Journal of Construction Engineering and Management*, 132, 1083-1091.
- LARMAN, C. & VODDE, B. 2009. *Scaling lean & agile development : thinking and organizational tools for large- scale Scrum*, Boston, Mass. ; London, Addison-Wesley.
- LENCIONI, P. 2006. *Overcoming the five dysfunctions of a team*, John Wiley & Sons.
- LEWIN, R. 1992. *Life at the Edge of Chaos*.
- LIKER, J. K. 2004. *The Toyota way : 14 management principles from the world's greatest manufacturer*, New York, McGraw-Hill.
- LIKER, J. K. & CONVIS, G. L. 2012. *The Toyota way to lean leadership : achieving and sustaining excellence through leadership development*, Maidenhead, McGraw-Hill Professional.
- LIKER, J. K., HOSEUS, M. & CENTER FOR QUALITY PEOPLE AND ORGANIZATIONS. 2008. *Toyota culture : the heart and soul of the Toyota way*, New York, McGraw-Hill.
- LIKER, J. K. & MORGAN, J. M. 2006. The Toyota Way in Services: The Case of Lean Product Development *Academy of Management Perspectives*, 20, 5-20.
- LITTLE, D. 2012. Explanatory autonomy and Coleman's boat. *THEORIA. Revista de Teoría, Historia y Fundamentos de la Ciencia*, 27, 137-151.

- LITTLE, J. D. 1961. A proof for the queuing formula: $L = \lambda W$. *Operations research*, 9, 383-387.
- LITTLE, J. D. C. & GRAVES, S. C. 2008. Little's Law. In: CHHAJED, D. & LOWE, T. J. (eds.) *Building Intuition: Insights from basic operations management models and principles*. Springer Science + Business Media, LLC.
- LITTLEJOHN, A., LUKIC, D. & MARGARYAN, A. 2010. How organisations learn from safety incidents: a multifaceted problem. *Journal of Workplace Learning*, 22, 428-450.
- LYNAM, T. & FLETCHER, C. 2015. Sensemaking: a complexity perspective. *Ecology and Society*, 20.
- MACINTYRE, A. 2007. *After Virtue: A Study in Moral Theory, 3rd Edition*, Indiana, Notre Dame Press.
- MAGER, F., SMITH, B. & GUIJT, I. 2018. How Decent is Decent Work? Using SenseMaker to understand workers' experiences. *Oxfam Research Report*. Oxfam.
- MAITLIS, S. & CHRISTIANSON, M. 2014. Sensemaking in organizations: Taking stock and moving forward. *The Academy of Management Annals*, 8, 57-125.
- MANDIĆ, V., OIVO, M., RODRÍGUEZ, P., KUVAJA, P., KAIKKONEN, H. & TURHAN, B. 2010. What Is Flowing in Lean Software Development? In: ABRAHAMSSON, P. & OZA, N. (eds.) *Lean Enterprise Software and Systems First International Conference (LESS 2010)*. Helsinki, Finland: Springer.
- MANDLER, G. 1984. *Mind and body: Psychology of emotion and stress*, WW Norton New York.
- MANN, A., BROWN, A., STAHNKE, M. & KERSTEN, N. 2018. State of DevOps Report. Puppet + splunk.
- MANN, D. 2014. *Creating a lean culture: tools to sustain lean conversions*, Productivity Press.
- MANNAVA, S. 2014. Micro-narratives Compensating the Omissions of Grand Historical Narratives. *Procedia - Social and Behavioral Sciences*, 158, 320-325.
- MARTIN, E. 2017. The Evolution of Analysis; Changing Expectations and attitudes.... *Naturalistic Decision Making and Uncertainty.*, 12.
- MARTIN, K. & OSTERLING, M. 2014. *Value stream mapping : how to visualize work and align leadership for organizational transformation*, New York, McGraw-Hill.
- MARX, D. 2008. Patient safety and the "just culture": a primer for health care executives. New York (NY): Columbia University; 2001.
- MAURYA, A. 2012. *Running Lean, Second Edition: Iterate from Plan A to a Plan That Works*, O'Reilly Media, Inc.
- MAXWELL, J. A. 2013. *Qualitative Research Design: An interactive approach*, Thousand Oaks, CA, SAGE Publications, Inc.
- MCCORD, P. 2017. *Powerful: building a culture of freedom and responsibility*, Silicon Guild.
- MCLAFFERTY, I. 2004. Focus group interviews as a data collecting strategy. *Journal of Advanced Nursing*, 48, 187-194.
- MCLEOD, J. & CHILDS, S. 2013. The Cynefin framework: A tool for analyzing qualitative data in information science? *Library & Information Science Research*, 35, 299-309.
- MEADOWS, D. H. 2008. *Thinking in systems: a primer*, White River Junction, Vermont, Chelsea Green Publishing.
- MEHEUS, J. & BATENS, D. 2006. A Formal Logic for Abductive Reasoning. *Logic Journal of the IGPL*, 14, 221-236.

- MELEGATI, J., WANG, X. & ABRAHAMSSON, P. Hypotheses engineering: first essential steps of experiment-driven software development. Proceedings of the Joint 4th International Workshop on Rapid Continuous Software Engineering and 1st International Workshop on Data-Driven Decisions, Experimentation and Evolution, 2019. IEEE Press, 16-19.
- MENEELY, A., SMITH, B. & WILLIAMS, L. 2012. Validating software metrics: A spectrum of philosophies. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 21, 24.
- MILES, M. B., HUBERMAN, A. M. & SALDANA, J. 2014. *Qualitative data analysis: A methods sourcebook*, Thousand Oaks, Sage.
- MILLER, J. H. & PAGE, S. E. 2007. *Complex Adaptive Systems : An Introduction to Computational Models of Social Life*, Princeton, N.J. ; Oxford, Princeton University Press.
- MILNE, K. M. G. 2015. Can sense-making tools inform adaptation policy? A practitioner's perspective. *Ecology and Society*, 20.
- MITCHELL, M. 2009. *Complexity: A Guided Tour*, Oxford University Press.
- MODIG, N. & AHLSTROM, P. 2013. *This is Lean: Resolving the Efficiency Paradox*, Rheologica Publishing.
- MORGAN, D. L. 1997. *Focus Groups as Qualitative Research*, Thousand Oaks, California, Sage Publications.
- MORGAN, D. L. 2019. *Basic and Advanced Focus Groups*, Thousand Oaks, CA, SAGE Publications.
- MORGAN, D. L. & BOTTORFF, J. L. 2010. Advancing our craft: Focus group methods and practice. *Qualitative health research*, 20, 579-581.
- MORGAN, D. L., FELLOWS, C. & GUEVARA, H. 2008. Emergent Approaches to Focus Group Research. In: HESSE-BIBEER, S. (ed.) *Handbook of Emergent Methods*. Guildford Press.
- MORGAN, L. 2011. *Open Innovation Networks: An Exploration of Value Creation and Capture with Open Source Software*. Doctor of Philosophy, National University of Ireland, Cork.
- MOSES, J. 2009. Should we try to measure software quality attributes directly? *Software Quality Journal*, 17, 203-213.
- MUJTABA, S., FELDT, R. & PETERSEN, K. 2010. Waste and Lead Time Reduction in a Software Product Customization Process with Value Stream Maps *21st Australian Software Engineering Conference (ASWEC), 2010*.
- MURAUŠKAITE, A. & ADOMAUSKAS, V. 2008. *Bottlenecks in Agile Software Development Identified Using Theory of Constraints (TOC) Principles*. Master thesis in Applied Information Technology Master Thesis, IT University of Gothenburg, Chalmers University of Technology, and University of Gothenburg.
- NAKAMURA, J. & CSIKSZENTMIHALYI, M. 2014. The concept of flow. *Flow and the foundations of positive psychology*. Springer.
- NAUR, P. & RANDELL, B. 1968. Software Engineering: Report on a conference sponsored by the NATO SCIENCE COMMITTEE. *NATO Software Engineering Conference*. Garmisch, Germany: NATO.
- NICOLETTE, D. 2015. *Software Development Metrics*, New York, Manning Publications Co.
- O'HEOCHA, C., WANG, X. & CONBOY, K. 2012. Evaluate an applied focus group approach using Klein & Myers principles for interpretive research. *Information Systems Journal*, 22, 235-256.

- OHNO, T. 1988. *Toyota production system : beyond large-scale production*, Cambridge, Mass., Productivity Press.
- OHNO, T. 2013. *Taiichi Ohno's Workplace Management, Special 100th Birthday Edition*, New York, McGraw-Hill.
- OLSON, E. E. & EOYANG, G. H. 2001. *Facilitating Organization Change: Lessons from Complexity Science*, San Francisco, Jossey-Bass/Pfeiffer, A Wiley Company.
- OPPENHEIM, B. W. 2004. Lean Product Development Flow. *Systems Engineering*, 7, 352-376.
- ORLIKOWSKI, W. J. & BAROUDI, J. J. 1991. Studying Information Technology in Organizations: Research Approaches and Assumptions. *Information Systems Research*, 2, 1-28.
- OZKAYA, I., KRUCHTEN, P. & NORD, R. L. 2019. *Managing Technical Debt*.
- PAGE, S. E. 2007. Making the Difference: Applying a Logic of Diversity. *Academy of Management Perspectives*, 21, 6-20.
- PAGE, S. E. 2015. What Sociologists Should Know About Complexity. *Annual Review of Sociology*, 41, 21-41.
- PALINKAS, L. A., HORWITZ, S. M., GREEN, C. A., WISDOM, J. P., DUAN, N. & HOAGWOOD, K. 2015. Purposeful sampling for qualitative data collection and analysis in mixed method implementation research. *Administration and policy in mental health*, 42, 533-544.
- PERNSTÅL, J., FELDT, R. & GORSCHKE, T. 2013. The lean gap: A review of lean approaches to large-scale software systems development. *Journal of Systems and Software*, 86, 2797-2821.
- PETERSEN, K. 2010. An Empirical Study of Lead-Times in Incremental and Agile Software Development. In: MUNCH, J., YANG, Y. & SCHAEFER, W. (eds.) *International Conference on Software Process (ICSP 2010)*. Paderborn, Germany: Springer-Verlag.
- PETERSEN, K. 2012. A Palette of Lean Indicators to Detect Waste in Software Maintenance: A Case Study. *Agile Processes in Software Engineering and Extreme Programming (XP2012)*. Malmo, Sweden: Springer.
- PETERSEN, K., ROOS, P., NYSTRÖM, S. & RUNESON, P. 2014. Early identification of bottlenecks in very large scale system of systems software development. *Journal of Software: Evolution and Process*, 26, 1150-1171.
- PETERSEN, K. & WOHLIN, C. 2010. Measuring the flow in lean software development. *Software: Practice and Experience Special Issue: Focus on Agile Software Development*, 41, 975-996.
- PETTIGREW, A. M. 1990. Longitudinal Field Research on Change: Theory and Practice. *Organization Science*, 1, 267-292.
- PFEFFER, J. & SALANCIK, G. R. 1977. Organization design: The case for a coalitional model of organizations. *Organizational Dynamics*, 6, 15-29.
- PINCIROLI, F. 2016. Improving Software Applications Quality by Considering the Contribution Relationship Among Quality Attributes. *Procedia Computer Science*, 83, 970-975.
- PINK, D. H. 2010. *Drive: The Surprising Truth about what Motivates Us*.
- PLATT, E. 2018. Global M&A activity hits new high. *Financial Times*, September 28, 2018.
- PLOWMAN, D. A., SOLANSKY, S., BECK, T. E., BAKER, L., KULKARNI, M. & TRAVIS, D. V. 2007. The role of leadership in emergent, self-organization. *The Leadership Quarterly*, 18, 341-356.

- POPPENDIECK, M. & CUSUMANO, M. A. 2012. Lean Software Development: A Tutorial. *IEEE Software (Special Issue on Lean Software Development)*, 29, 26-32.
- POPPENDIECK, M. & POPPENDIECK, T. 2003. *Lean Software Development: An Agile Toolkit*, Boston, Addison-Wesley.
- POPPENDIECK, M. & POPPENDIECK, T. 2007. *Implementing lean software development : from concept to cash*, London, Addison-Wesley.
- POPPENDIECK, M. & POPPENDIECK, T. 2010. *Leading lean software development : results are not the point*, Upper Saddle River, NJ, Addison-Wesley.
- POPPENDIECK, M. & POPPENDIECK, T. 2013. *The Lean Mindset: Ask the Right Questions*, Upper Saddle River, NJ, Addison-Wesley.
- POWELL, R. A. & SINGLE, H. M. 1996. Focus Groups. *International Journal for Quality in Health Care*, 8, 499-504.
- PUIK, E. & CEGLAREK, D. 2015. The Quality of a Design will not Exceed the Knowledge of its Designer; an Analysis Based on Axiomatic Information and the Cynefin Framework. *9th International Conference on Axiomatic Design (ICAD 2015)*.
- PUTNAM, L. H. & MYERS, W. 2003. *Five core metrics : the intelligence behind successful software management*, New York, NY, Dorset House Publishing.
- RAJLICH, V. 2016. *Software Engineering: The Current Practice*, CRC Press.
- REINERTSEN, D. G. 1997. *Managing the design factory : a product developer's toolkit*, New York ; London, Free Press.
- REINERTSEN, D. G. 2009. *The principles of product development flow: second generation lean product development*, Redondo Beach, Calif., Celeritas.
- RESNICK, L. B., LEVINE, J. M. & TEASLEY, S. D. (eds.) 1991. *Perspectives on socially shared cognition*: American Psychological Association.
- RIES, E. 2011. *The Lean Startup: How Constant Innovation Creates Radically Successful Businesses*, Penguin.
- RIES, E. 2017. *The startup way: how modern companies use entrepreneurial management to transform culture and drive long-term growth*, Currency.
- RIESSMAN, C. K. 2008. *Narrative Methods for the Human Sciences*, Thousand Oaks, CA, Sage Publications, Inc.
- RIKKILÄ, J., WANG, X. & ABRAHAMSSON, P. 2013. Agile Project—An Oxymoron? Proposing an Unproject Leadership Model for Complex Space. *Lean Enterprise Software and Systems*. Springer.
- RIKOLTO. 2017. *Rikolto is born! Vredeseilanden/VECO changes brand name* [Online]. Rikolto. Available: <https://www.rikolto.org/nl/node/1520> [Accessed July 21 2019].
- RITTEL, H. & WEBBER, M. 1973. Dilemmas in a general theory of planning. *Policy Sciences*, 4, 155-169.
- RODRÍGUEZ, P., PARTANEN, J., KUVAJA, P. & OIVO, M. Combining lean thinking and agile methods for software development: A case study of a finnish provider of wireless embedded systems detailed. 2014 47th Hawaii International Conference on System Sciences, 2014. IEEE, 4770-4779.
- ROSILE, G. A., BOJE, D. M., CARLON, D. M., DOWNS, A. & SAYLORS, R. 2013. Storytelling Diamond: An Antenarrative Integration of the Six Facets of Storytelling in Organization Research Design. *Organizational Research Methods*, 16, 557-580.
- ROTHER, M. & SHOOK, J. 2003. *Learning to see : value stream mapping to add value and eliminate muda*, Cambridge, Mass., Lean Enterprise Institute.

- ROTHMAN, J. 2009. *Manage your project portfolio : increase your capacity and finish more projects*, Raleigh, N.C., Pragmatic Bookshelf.
- RUNESON, P. & HÖST, M. 2008. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14, 131-164.
- RUNESON, P., HÖST, M., RAINER, A. & REGNELL, B. 2012. *Case Study Research in Software Engineering: Guidelines and Examples*, Hoboken, NJ, John Wiley & Sons, Inc.
- SALDANA, J. 2016. *The Coding Manual for Qualitative Researchers*, London, SAGE Publications, Inc.
- SALDANA, J. & OMASTA, M. 2017. *Qualitative Research: Analyzing Life*, Los Angeles, SAGE Publications, Inc.
- SANDELOWSKI, M. 2012. The weakness of the strong/weak comparison of modes of inquiry. *Research in Nursing & Health*, 35, 325-327.
- SANDELOWSKI, M. 2014. Unmixing Mixed-Methods Research. *Research in Nursing & Health*, 37, 3-8.
- SCHEIN, E. H. 1996. Three cultures of management: The key to organizational learning. *Sloan management review*, 38, 9-20.
- SCHEIN, E. H. 2010. *Organizational Culture and Leadership*, San Francisco, CA, Jossey-Bass.
- SCHRAAGEN, J. M. C. Beyond macrocognition: The transaction level. NDM13 Naturalistic Decision Making and Uncertainty, Proceedings of the 13th bi-annual international conference on Naturalistic Decision Making, 2017 Bath, UK. 182-188.
- SCHWABER, K. & BEEDLE, M. 2002. *Agile software development with Scrum*, Prentice Hall Upper Saddle River.
- SEDANO, T., RALPH, P. & PERAIRE, C. 2017. Software development waste. *Proceedings of the 39th International Conference on Software Engineering*. Buenos Aires, Argentina: IEEE Press.
- SEDDON, J. 2005. *Freedom from command & control : rethinking management for lean service*, New York, Productivity Press.
- SHALLOWAY, A., BEAVER, G. & TROTT, J. 2010. *Lean-agile software development : achieving enterprise agility*, Upper Saddle River, NJ, Addison-Wesley.
- SHEWART, W. A. 1931. *Economic Control of Quality of Manufactured Product*, New York, American Society for Quality.
- SHINGO, S. 1989. *A Study of the Toyota Production System*, New York, Productivity Press.
- SHINGO, S. 2007. *Kaizen and the art of creative thinking : the scientific thinking mechanism*, Bellingham, WA, Enna Products Corporation and PCS Inc.
- SHOOK, J. 2009. Toyota's Secret: The A3 Report. *MIT Sloan Management Review*, 50, 30-33.
- SHORE, J. & WARDEN, S. 2007. *The art of agile development*, Farnham, O'Reilly.
- SMITH, D. Forming an incident response team. Proceedings of the FIRST Annual Conference, 1994. University of Queensland Brisbane, Australia, 1-37.
- SMITH, P. 2012. *Lead with a story: A guide to crafting business narratives that captivate, convince, and inspire*, AMACOM Div American Mgmt Assn.
- SMITH, P. G. & REINERTSEN, D. G. 1998. *Developing products in half the time : new rules, new tools*, New York ; London, Van Nostrand Reinhold.
- SNOWDEN, D. 2002. Complex acts of knowing - paradox and descriptive self-awareness. *Journal of Knowledge Management*, 6.

- SNOWDEN, D. 2005a. Strategy in the context of uncertainty. *Handbook of Business Strategy*. Emerald Group Publishing Limited.
- SNOWDEN, D. 2010. Naturalizing sensemaking. In: MOSIER, K. L. & FISCHER, U. M. (eds.) *Informed by knowledge: Expert performance in complex situations*. New York: Psychology Press.
- SNOWDEN, D. 2015. *Cynefin dynamics* [Online]. Available: <http://cognitive-edge.com/blog/cynefin-dynamics/> [Accessed December 29 2016].
- SNOWDEN, D. J. 2005b. Multi-ontology sense making: a new simplicity in decision making. *Informatics in Primary Care*, 13, 45-53.
- SNOWDEN, D. J. & BOONE, M. E. 2007. A Leader's Framework for Decision Making. *Harvard Business Review*.
- SNYDER, M. & WHITE, P. 1982. Moods and memories: Elation, depression, and the remembering of the events of one's life. *Journal of personality*, 50, 149-167.
- SOBEK, D. K. & JIMMERSON, C. 2006. A3 Reports: Tool for Organizational Transformation. *IIE Annual Conference*.
- SOBEK, D. K. & SMALLEY, A. 2008. *Understanding A3 thinking : a critical component of Toyota's PDCA management system*, Boca Raton, CRC Press.
- SPEAR, S. & BOWEN, H. K. 1999. Decoding the DNA of the Toyota Production System. *Harvard Business Review*, 77, 96-106.
- SPEAR, S. J. 2004. Learning to lead at Toyota. *Harvard Business Review*, 82, 78-86.
- SPEAR, S. J. 2005. Fixing Healthcare from the Inside, Today. *Harvard Business Review*, September.
- SPEAR, S. J. 2006. Fixing healthcare from the inside: Teaching residents to heal broken delivery processes as they heal sick patients. *Academic medicine*, 81, S144 -S149.
- SQUIRE, C., ANDREWS, M. & TAMBOUKOU, M. 2012. What is Narrative Research? In: ANDREWS, M., SQUIRE, C. & TAMBOUKOU, M. (eds.) *Doing Narrative Research*. Second ed. Thousand Oaks, CA: SAGE Publications Ltd.
- SQUIRE, C., DAVIS, M., ESIN, C., ANDREWS, M., HARRISON, B., HYDÉN, L.-C. & HYDÉN, M. 2014. *What is Narrative Research?*, Bloomsbury Academic.
- STACEY, R. 1996. Emerging Strategies for a Chaotic Environment. *Long Range Planning*, 29, 182-189.
- STARBUCK, W. H. & MILLIKEN, F. J. 1988. Executives' perceptual filters: What they notice and how they make sense.
- STARON, M. & MEDING, W. 2011. Monitoring bottlenecks in agile and lean software development projects—a method and its industrial use. *PROFES'11 Proceedings of the 12th international conference on Product-focused software process improvement*. Torre Canne, Italy: Springer.
- STEERS, R. M. & SHIM, W. S. 2013. Strong leaders, strong cultures: Global management lessons from Toyota and Hyundai. *Organizational Dynamics*, 42, 217-227.
- STRIPE 2018. The Developer Coefficient: Software engineering efficiency and its \$3 trillion impact on global GDP.
- SUTHERLAND, J. & SCHWABER, K. 2013. *The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game*. Scrum.org.
- SWAMINATHAN, B. & JAIN, K. Implementing the lean concepts of continuous improvement and flow on an agile software development project: An industrial case study. 2012 Agile India, 2012. IEEE, 10-19.
- TAKEUCHI, H. & NONAKA, I. 1986. New New Product Development Game. *Harvard Business Review*.

- TAVORY, I. & TIMMERMANS, S. 2014. *Abductive Analysis: Theorizing Qualitative Research*, University of Chicago Press.
- TAYLOR, J. R. & VAN EVERY, E. J. 2000. *The Emergent Organization: Communication as its site and surface*, Mahwah, NJ, Lawrence Erlbaum Associates, Inc.
- THIETART, R. A. & FORGUES, B. 1995. Chaos Theory and Organization. *Organization Science*, 6, 19-31.
- THOMKE, S. & REINERTSEN, D. 2012. Six Myths of Product Development. *Harvard Business Review*, 90, 84-94.
- TRIPP, J., SALTZ, J. & TURK, D. Thoughts on current and future research on agile and lean: ensuring relevance and rigor. Proceedings of the 51st Hawaii International Conference on System Sciences, 2018.
- UNITED NATIONS ECONOMIC AND SOCIAL COUNCIL 2019. Special edition: progress towards the Sustainable Development Goals - Report of the Secretary-General. *High-level political forum on sustainable development, convened under the auspices of the Economic and Social Council*. United Nations.
- VAARA, E., SONENSHEIN, S. & BOJE, D. 2016. Narratives as Sources of Stability and Change in Organizations: Approaches and Directions for Future Research. *The Academy of Management Annals*, 10, 495-560.
- VAN BEURDEN, E. K., KIA, A. M., ZASK, A., DIETRICH, U. & ROSE, L. 2011. Making sense in a complex landscape: how the Cynefin Framework from Complex Adaptive Systems Theory can inform health promotion practice. *Health Promotion International*, 28, 73-83.
- VAN DER MERWE, S. E., BIGGS, R., PREISER, R., CUNNINGHAM, C., SNOWDEN, D. J., O'BRIEN, K., JENAL, M., VOSLOO, M., BLIGNAUT, S. & GOH, Z. 2019. Making Sense of Complexity: Using SenseMaker as a Research Tool. *Systems*, 7, 25.
- VANMAANEN, J. & KUNDA, G. 1989. Real feelings: Emotional expression and organizational culture. *Research in organizational behavior*, 11, 43-103.
- VASCONCELOS, F. C. & RAMIREZ, R. 2011. Complexity in business environments. *Journal of Business Research*, 64, 236-241.
- VERSIONONE 2015. 9th Annual State of Agile Report.
- VERSIONONE 2016. 10th Annual State of Agile Report.
- VERSIONONE 2017. 11th Annual State of Agile Report.
- VIDGEN, R. & WANG, X. 2006. Organizing For Agility: A Complex Adaptive Systems Perspective On Agile Software Development Process. In: LJUNBERG, J. & ANDERSSON, M. (eds.) *14th European Conference on Information Systems*. Gothenburg, Sweden.
- WACHTER, R. M. & PRONOVOST, P. J. 2009. Balancing “no blame” with accountability in patient safety. *N Engl J Med*, 361, 1401-1406.
- WAGNER, E. T. 2013. Five Reasons 8 Out Of 10 Businesses Fail. *Forbes*.
- WALTON, M. 1994. *The Deming Management Method: The Complete Guide to Quality Management*, UK, Mercury Books.
- WANG, X. 2007. *Organizing to be Adaptive: a Complex Adaptive Systems based Framework for Software Development Processes*. Doctor of Philosophy, University of Bath.
- WANG, X. The combination of agile and lean in software development: An experience report analysis. Agile Conference (AGILE), 2011, 2011. IEEE, 1-9.
- WARD, A. C. 2007. *Lean Product and Process Development*, Cambridge, MA, The Lean Enterprise Institute Inc.

- WEICK, K. E. 1995. *Sensemaking in Organizations*, Thousand Oaks, CA, SAGE Publications, Inc.
- WEICK, K. E. 2005. Managing the unexpected: Complexity as distributed sensemaking. In: R.R.MCDANIEL & D.J.DRIEBE (eds.) *Uncertainty and Surprise in Complex Systems*. Berlin Heidelberg: Springer-Verlag.
- WEICK, K. E. 2009. *Making sense of the organization. Vol. 2, The impermanent organization*, Chichester, U.K., Wiley.
- WEICK, K. E. & SUTCLIFFE, K. M. 2011. *Managing the Unexpected: Resilient Performance in an Age of Uncertainty*, Wiley.
- WEICK, K. E., SUTCLIFFE, K. M. & OBSTFELD, D. 2005. Organizing and the process of sensemaking. *Organization science*, 16, 409-421.
- WEINBERG, G. M. 2014. Why software gets in trouble. Leanpub.
- WIKLUND, K., SUNDMARK, D., ELDH, S. & LUNDQVIST, K. 2013. Impediments in Agile Software Development: An Empirical Investigation. In: HEIDRICH, J., OIVO, M., JEDLITSCHKA, A. & BALDASSARRE, M. T. (eds.) *Product-Focused Software Process Improvement: 14th International Conference, PROFES 2013, Paphos, Cyprus, June 12-14, 2013. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg.
- WOMACK, J. P. 2006. Value Stream Mapping. *Manufacturing Engineering*, 136, 145-156.
- WOMACK, J. P. 2011. *Gemba Walks*, Cambridge, MA, Lean Enterprise Institute, Inc.
- WOMACK, J. P. & JONES, D. T. 1996. Beyond Toyota: How to Root Out Waste and Pursue Perfection. *Harvard Business Review*.
- WOMACK, J. P. & JONES, D. T. 2003. *Lean Thinking: Banish Waste and Create Wealth in Your Corporation*, Simon and Schuster.
- WOMACK, J. P., JONES, D. T. & ROOS, D. 2007. *The machine that changed the world : the story of lean production - Toyota's secret weapon in the global car wars that is revolutionizing world industry*, London, Simon & Schuster.
- YIN, R. K. 2012. *Applications of case study research*, Thousand Oaks, CA, Sage Publications.
- YIN, R. K. 2014. *Case study research : design and methods, 5th Edition*, London, SAGE.
- YIN, R. K. 2016. *Qualitative Research from Start to Finish*, New York, NY, The Guildford Press.