Intellectual Property and Architecture: How architecture influences intellectual property lock-in

by

Christopher W. Berardi

B.S. Management, United States Air Force Academy, 2005
S.M. Engineering and Management, Massachusetts Institute of Technology, 2013

Submitted to the Engineering Systems Division in
Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in Engineering Systems at the
Massachusetts Institute of Technology
June 2017

Signature of Author .....
Engineering Systems Division
May 7, 2017

Certified by .....
Edward F. Crawley
Professor of Aeronautics and Astronautics
Chair, Thesis Committee

Certified by .....
Dr. Bruce G. Cameron
Director, System Architecture Lab
Member, Thesis Committee

Certified by .....
Carliss Y. Baldwin
Professor of Business Administration, Harvard Business School
Member, Thesis Committee

Certified by .....
Steven D. Eppinger
Professor of Management Science and Innovation
Member, Thesis Committee

Accepted by .....
John N. Tsitsiklis
Professor of Electrical Engineering
IDSS Graduate Officer

Intellectual Property and Architecture: How architecture influences intellectual property lock-in

by

Christopher W. Berardi

Submitted to the Engineering Systems Division
on May 26, 2017, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Intellectual property lock-in is a wicked problem particularly pervasive under monopsony market structures, such as the Department of Defense (DoD). However, little research exists on the mechanisms of action that induce intellectual property lock-in. This work postulates the conjuncture of architecture and intellectual property is one such mechanism of action and erects a research methodology to investigate this link.

This dissertation began with a review of literature, which revealed more research is needed into basic trends or estimates of magnitude for intellectual property lock-in. To quantitatively frame the magnitude of the problem an investigation was conducted into all DoD contracts for the last eight fiscal years to establish bounds. These results were used to formulate a conceptual model of the problem and suggest the concept of intellectual property architecture, which is the conjuncture of architecture and intellectual property.

To investigate links between intellectual property architecture and lock-in, an intermediate-N fuzzy-set Qualitative Comparative Analysis research approach was formulated and executed using 14 DoD software cases representing over 34 million lines of code. The model used three input conditions: high quality technical architecture, accessible intellectual property architecture, and unlimited rights to study the avoidance of lock-in.

The fuzzy-set Qualitative Comparative Analysis concluded intellectual property architecture or unlimited rights were quasi-necessary conditions for the avoidance of lock-in. Additionally the model yielded both a five condition conservative expression and two condition parsimonious expression for sufficient conditions.

From those expressions, this research concludes three direct findings. First, intellectual property architecture is an empirically supported mechanism of action for the avoidance of lock-in. Implying, intellectual property architecture, absent any other explanatory conditions, is sufficient to avoid lock-in. Second, the research herein finds evidence to support a novel taxonomy of intellectual property architectures. Allowing practitioners to understand potential trade-offs between architecture and intellectual property lock-in. Third, intellectual property architecture or unlimited rights is a

theoretically supported expression for the avoidance of lock-in. This finding implies that as few as two conditions are required to understand whether a case may, or may not, avoid lock-in.

Thesis Supervisor: Edward F. Crawley
Title: Professor of Aeronautics and Astronautics

Thesis Supervisor: Dr. Bruce G. Cameron
Title: Director, Systems Architecture Lab

Thesis Supervisor: Carliss Y. Baldwin
Title: Professor of Business Administration, Harvard Business School

Thesis Supervisor: Steven D. Eppinger
Title: Professor of Management Science and Innovation

Acknowledgments

The research which follows is not a reflection of my abilities, but rather the innumerable meaningful contributions from mentors, peers, friends, and family. For these contributions, I am forever grateful.

Specifically, Professor Carliss Baldwin who graciously agreed to take time away from her research and teaching at Harvard Business School to help mentor my research. Carliss often saw the merit in even my most poorly formulated concepts. Her encouragement to pursue these ideas and advice on how to refine the concepts was instrumental in formulating the basis for this research.

Professor Steve Eppinger served as an invaluable voice in my research, often taking the role of devil's advocate. Much of what I held as irrefutable tenants of my research Steve questioned and thereby forced a deeper contemplation. Although receiving criticism is never comfortable, my research was significantly bettered by Steve's thoughtful criticisms.

I owe credit to Professor Ed Crawley for continuously forcing abstraction beyond what I would have considered on my own and for holding my research accountable to a rigorous standard.

No one provided more support and guidance for my research than Dr. Bruce Cameron. This research is a direct byproduct of Bruce's application of the Socratic Method over three years. His efforts to constantly re-frame the question while driving towards a result with practical impacts is reflected in almost every Chapter.

I also owe an immense gratitude to the Silverthread Team: Dr. Dan Sturtevant, Natalia Mocan, and Sean Gilliland. Particularly Dan, for taking the time to help code the first iterations of the extraction software and his guidance as a software expert. Said plainly, this research would not have been possible without Dan and the Silverthread Team's relentless support.

To the Air Force Acquisition organizations that provided data for my research, but which desired to remain anonymous, I owe an immense debt of gratitude. Early in

my research it was unclear whether it would yield useful results, but the professionals in the Air Force acquisition community remained steadfast in their support.

Lastly to my Parents, who have provided unwavering love and support throughout all my aspirations. This work and all my accomplishments, however big or small, are a direct reflection of the work ethic I learned from both of my parents.

Chris Berardi

MIT

June 2017

Contents

1 Introduction **19**

2 Literature Review **27**

2.1 Definitions 27

2.2 Policy Review 29

2.2.1 Post World War II 30

2.2.2 Congressional Interventions 32

2.2.3 Architecture and Intellectual Property 37

2.3 Literature 39

2.3.1 Legal Literature 39

2.3.2 Research Literature 45

2.3.3 Literature Gaps 47

2.4 Conclusion 48

3 Quantitative Framing **51**

3.1 Guiding Quantitative Questions 51

3.2 Background: Policy and Statute 52

3.3 Independent Research and Development 55

3.4 Investigation Method 57

3.4.1 Phase One: Business Understanding 58

3.4.2 Phase Two: Data Understanding 59

3.4.3 Phase Three: Data Preparation 60

3.4.4 Phase Four: Modeling 63

3.5 Findings/Discussion 72
3.6 Limitations/Recommendations 74
3.7 Conclusion 75

4 Conceptual Model **77**
4.1 Bottleneck Model 77
4.1.1 Technical Architecture and Contract Structure 78
4.1.2 Bottleneck Defined 79
4.1.3 Technical Bottlenecks 80
4.1.4 Strategic Bottlenecks 80
4.2 Intellectual Property Lock-in 81
4.3 Bottleneck Model Applied to DoD 83
4.3.1 Assertion of Rights 84
4.3.2 Determine Government Rights 84
4.3.3 Determine Contractor Rights 86
4.3.4 Determine Functional Relevance 89
4.3.5 Determine Formal Relevance 91
4.4 Criteria for Intellectual Property Lock-in 95
4.5 Discussion 96
4.5.1 Lost Competition 97
4.6 Conclusion 98

5 Scientific Methods **99**
5.1 Extracting Architecture from Software 99
5.1.1 Networks and DSM Architecture Visualizations 100
5.1.2 Extraction Process 101
5.1.3 Abstraction of Architecture into Categories 105
5.1.4 Visualize the Architecture 107
5.2 Measuring Substitutability from Architecture 108
5.2.1 Hardness-to-Kill 109
5.3 Extracting Intellectual Property 109

5.3.1 Type of Intellectual Property . . . 110
5.3.2 Extraction Process . . . 112
5.4 Intellectual Property Architecture . . . 112
5.5 Architecture Patterns . . . 116
5.5.1 Technical Architecture Sub-Patterns . . . 117
5.5.2 Intellectual Property Sub-Patterns . . . 118
5.5.3 Novel Theoretical Patterns . . . 119
5.6 Conclusion . . . 119

6 Research Methods **121**
6.1 Research Design . . . 121
6.1.1 Qualitative Comparative Analysis . . . 123
6.1.2 Model Specification . . . 125
6.2 Case Selection . . . 128
6.2.1 Sample Size . . . 128
6.2.2 Sampling Methodology . . . 129
6.3 Qualitative Methods . . . 131
6.3.1 Qualitative Data Collection . . . 133
6.4 Quantitative Methods . . . 135
6.4.1 Quantitative Data Collection . . . 135
6.5 Threats to Validity . . . 137
6.6 Conclusion . . . 139

7 Contextual Case Summary **141**
7.1 General Context . . . 142
7.2 Hazard Patterns . . . 143
7.2.1 Case A . . . 143
7.2.2 Case K . . . 146
7.2.3 Case J . . . 148
7.2.4 Case I . . . 150
7.2.5 Case G . . . 153

7.2.6 Case B 155
7.2.7 Case M 157
7.3 Boundary Patterns 159
7.3.1 Case N 159
7.3.2 Case E 162
7.3.3 Case F 164
7.3.4 Case H 166
7.4 Ideal Patterns 168
7.4.1 Case L 168
7.4.2 Case C 170
7.4.3 Case D 172
7.5 Conclusion 174

8 Cross-Case Analysis **175**
8.1 Model Calibration 175
8.1.1 Technical Architecture Quality 176
8.1.2 Accessible Intellectual Property Architecture 178
8.2 Conditions 180
8.2.1 Parameters of Fit 181
8.3 Necessary Conditions 181
8.4 Sufficient Conditions 183
8.4.1 Conservative Expression 183
8.4.2 Parsimonious Expression 187
8.5 Conclusion 190

9 Discussion **191**
9.1 Necessary Conditions 191
9.1.1 Analysis and Accuracy 192
9.1.2 Sensitivity of Necessary Conditions 193
9.1.3 Practical Relevance of Necessary Conditions 197
9.2 Sufficient Conditions 199

9.2.1 Sensitivity Analysis and Accuracy . . . 201
9.2.2 Findings . . . 203
9.2.3 Practical Relevance of Sufficient Conditions . . . 206
9.2.4 Impact Outside DoD . . . 210
9.3 Potential Criticisms . . . 211
9.4 Limitations . . . 214
9.5 Future Research . . . 215
9.6 Conclusion . . . 218

10 Conclusion **219**
10.1 Contributions . . . 220
10.1.1 Theoretical . . . 220
10.1.2 Methodological . . . 221
10.1.3 Empirical . . . 222

A OLS Regression Results **225**

B Interview Question Template **229**

C Copyright Regular Expressions **233**

D Case Summary **235**

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

2-1 Timeline of 10 U.S.C. §§ 2320 and 2321 Amendments. 33

3-1 Visualizations of DoD R&D as well as IRAD Funding 56
3-2 Lagging Pearson product-moment correlation coefficients for IRAD vs PDR JnA 57
3-3 Longitudinal Trend Analyses 68
3-4 Inter-Year PDR Obligation Trends 69
3-5 Product, Service, and R&D PDR JnA by Fiscal Year and PSC Taxonomy Level 1 72

4-1 Theoretical Formation of Intellectual Property Lock-in 82
4-2 Theoretical Relationship between *De Jure* Strategic Bottlenecks and Likelihood of IP lock-in 83
4-3 Conceptual Model for Intellectual Property Lock-in 84
4-4 Open Source License Compatibility 88
4-5 Hypothesized Model for Intellectual Property Lock-in 92
4-6 Example of Intellectual Property Modularity 94
4-7 Equilibrium Curve for Number of IP Lock-in Cases 97

5-1 Network Representations of Simple Networks 101
5-2 Network Representations of Visibility Networks 104
5-3 Examples of Architectural Abstractions which Assist in Categorization 106
5-4 Core-Periphery View of Architecture 108
5-5 Intellectual Property Architecture Open Source View 113

5-6 Intellectual Property Architecture License Type View 115
5-7 Intellectual Property Architecture Owner View 116
5-8 Taxonomy of Architectural Patterns (assuming no Unlimited Rights) 119

6-1 Theoretical differences between research approaches and case selection 124
6-2 Case Sample Scatter Plot 132

7-1 Case A Owner View 145
7-2 Case K Owner View 147
7-3 Case J Open Source View 149
7-4 Case I Open Source View 152
7-5 Case G Owner View 154
7-6 Case B Open Source View 156
7-7 Case M Owner View 159
7-8 Case N Open Source View 161
7-9 Case E Open Source View 163
7-10 Case F Open Source View 165
7-11 Case H Open Source View 167
7-12 Case L Owner View 169
7-13 Case C Open Source View 171
7-14 Case D Open Source View 173

8-1 Plot of Membership in Set of Quality Technical Architecture 177
8-2 Plot of Fuzzy Membership in Set of Accessible IP Architecture 179
8-3 Scatter Plot of Set TA+IP+UR against Set 1-LI 185
8-4 Example Boolean minimization of Equation (8.8) to Equation (8.10) . 188
8-5 Boolean minimization resulting in Equation (8.13) 188
8-6 Parameters of Fit for Parsimonious Expression 189

9-1 Venn Diagram of Candidate Necessary Conditions 193
9-2 Venn Diagrams for Sufficient Conditions 201
9-3 Comparison of Taxonomy against Conservative Expression 205

9-4 Plot of “Condition Space” Coverage 216

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

2.1 Rights in *Noncommercial* Computer Software (CS) and Technical Data (TD) Covering Noncommercial Items Rights 36
2.2 Rights in *Commercial* Computer Software (CS) and Technical Data (TD) Covering Commercial Items Rights 37

3.1 Defense R&D Expenditures by Performer and Funding source for FY84-FY15 55
3.2 Brief Description of USASpending Data 60
3.3 Description of Data Fields 61
3.4 Count, Sum, Median, and Mean of *dollarsobligated* grouped by *reasonnotcompeted*, groups sorted by descending sum of dollars obligated. . 64
3.5 Summary of PDR JnA Purchasing Categories 71

5.1 Description of Four Canonical Groups 106
5.2 Description of Technical Architecture Sub-Patterns 117
5.3 Description of Intellectual Property Architecture Sub-Patterns 118

6.1 Design of fuzzy-set Qualitative Comparative Analysis Model 126
6.2 Description of Selected Cases 131
6.3 Summary of Interviews Conducted 133
6.4 File-Level Variable Descriptions 137

7.1 Summary of Architectural Patterns by Case 142

8.1 Fuzzy Membership in Set of Technical Architecture Quality 178

8.2 Fuzzy Membership in Set of Accessible IP Architecture 179
8.3 Calibrated Summary of Cases 180
8.4 Parameters of Fit for Necessary Conditions 185
8.5 Fuzzy membership in three-dimensional vector space 185
8.6 Truth Table for outcome 1-LI 188
8.7 Sufficient Conditions Parameters of Fit 188

9.1 Fuzzy-set membership scores in candidate Necessary Conditions . . . 194
9.2 Outcomes of Equation (9.1) for TA+IP+UR 196
9.3 Necessary conditions under relaxed criteria ($Con_N > 0.89$) 196
9.4 Prime Implicant Chart 200

A.1 OLS Regression Results for Ratio of JnA to Total Awards ($) 226
A.2 OLS Regression Results for Ratio of JnA to Total Awards (Contract Actions) 226
A.3 OLS Regression Results for Ratio of PDR to Total Awards ($) 227
A.4 OLS Regression Results for Ratio of PDR to Total Awards (Contract Actions) 227
A.5 OLS Regression Results for Ratio of PDR to JnA ($) 228
A.6 OLS Regression Results for Ratio of PDR to JnA (Contract Actions) 228

D.1 Case Summary (Sorted by System File Size) 236

Chapter 1

Introduction

In 1973, Rittel and Webber, professors of design and urban planning, defined *wicked problems* as a class of problems which are not easy to define, have innumerable causes, are tough to describe, and have no single correct answer [1]. Although there is no single characteristic that defines a problem as wicked, a large contributor is policy related. Rittel and Webber explain that problems in the natural sciences are definable and separable and may have solutions that are "findable"; however, problems involving policy are ill-defined and they rely upon political judgment for resolution. Problems of this nature are not easily addressed with conventional methodologies.

This work argues that one such wicked problem is intellectual property lock-in. Although "intellectual property lock-in" is used often in practice to describe a situation where some manner of intellectual property is used to reduce or preclude competition, the phrase is rarely defined in literature. In attempting to create a definition for intellectual property lock-in, some of the characteristics of a wicked problem become evident. What is the cause of intellectual property lock-in? Since intellectual property is largely a legal issue, the problem of lock-in is surely due to inefficient statute? This is rationale first step, but statute is further implemented in regulations. Perhaps then it is not statute that is correct, but the formulation of regulations. *Regardless of abstraction level, the legal domain seems to be a correct formulation of lock-in.* While potentially true, this problem formulation allows no consideration of economics. Instead of a legal approach, perhaps intellectual prop-

erty lock-in is the byproduct of certain types of market structures. This approach allows for the inclusion of arguments concerning competition dynamics in a monopsony, catalyzing innovation, or maintaining health of an industrial base. *Although there are many angles of investigation, economics seems to be a correct formulation of lock-in.* Economics is an attractive formulation, but fails to capture the nature of technology on lock-in. Perhaps intellectual property lock-in occurs more extensively in certain classes of technology; software versus hardware. Rationally this makes sense as intellectual property is created at different speeds in software and hardware. *Technology seems to be a correct formulation of lock-in.* However, the different speeds of intellectual property creation are potentially due to the different intellectual property statute sachems which protect software versus hardware. *Perhaps, the legal domain is the correct formulation.*

This circular argument is meant to exemplify the wickedness of the intellectual property lock-in problem. Note wickedness in this context does not imply evil, instead Rittel and Webber are clear that the term wicked in this context is, "akin to that of 'malignant' (in contrast to 'benign') or 'vicious' (like a circle) or 'tricky' (like a leprechaun) or 'aggressive' (like a lion, in contrast to the docility of a lamb)" [1, p. 161]. They define ten characteristics of a wicked problem. However, it is important to note that these characteristics are not a checklist or set of tests to mechanically determine if a problem is wicked. A problem may demonstrate one or all of the ten characteristics. The characteristics are designed to provide insights that help judge whether a problem is wicked. Each of the ten characteristics are listed below in bold accompanied with a brief general explanation, as well as a specific explanation in the context of intellectual property lock-in.

There is no definitive formulation of a wicked problem. It is not possible to write a well-defined statement of the problem as can be done with ordinary problems. Is the problem of intellectual property lock-in confined strictly within the boundaries of statute and policy? Or are there exogenous factors such as innovation, industrial base, and politics which confound the problem? Are the complete list of factors which influence intellectual property lock-in known?

Wicked problems have no stopping rule. There are no criteria that tell when, "*the* or *a* solution has been found"[1, p. 162]. How does one know when enough steps have been taken to avoid intellectual property lock-in? Is there a prescriptive approach for the avoidance of lock-in?

Solutions to wicked problems are not true-or-false, but good-or-bad. Objective evaluation of the solution to a wicked problem is difficult; consequently, choosing a solution is largely a matter of judgment. If the intellectual property lock-in problem does not have a definitive formulation, then it is difficult, if not impossible, to judge a solution as true-or-false.

There is no immediate and no ultimate test of a solution to a wicked problem. The solutions to wicked problems generate an unbounded number of unexpected consequences (1 to n order effects); therefore, evaluation of a solution is impossible until all consequences have run out. If the complete list of exogenous factors which confound intellectual property lock-in are not known or are infinite, then understanding and forecasting the impact of a solution is not possible or is beyond human comprehension.

Every solution to a wicked problem is a "one-shot operation"; because there is no opportunity to learn by trial-and-error; every attempt counts significantly. In ordinary problems, the problem solver can try various solutions without penalty. However, in wicked problems every implemented solution is consequential. Rittel and Webber use the example, "One cannot build a freeway to see how it works, and then easily correct it after unsatisfactory performance"[1, p. 163]. There is no simulator to test potential solutions to intellectual property lock-in problems as each contract is a binding agreement with millions to billions of dollars at stake.

Wicked problems do not have an enumerable (or and exhaustively describable) set of potential solutions, nor is there a well-described set of permissible operations that may be incorporated into the plan. Problems of an ordinary nature come with a bounded set of potential solutions. Even if the boundaries encompass a large sample space, the sample space is still defined. Wicked problems offer a potential infinite solution space.

Every wicked problem is essentially unique. Ordinary problems may belong to a larger class of similar problems, which are all solved in a specified way. "A wicked problem is substantially without precedent; experiences do not help you address it" [2, p. 101]. No two intellectual property lock-in problems are the same, because no two contracts or contextual details behind a contract are uniform. Therefore, what may work in one contract does not imply it will work across all contracts.

Every wicked problem can be considered to be a symptom of another problem. Ordinary problems may be self-contained, a wicked problem is directly or indirectly connected with many other problems. Is the frequency of intellectual property lock-in in the DoD a byproduct of statute or a downstream symptom of the monopsony-oligopoly market structure?

The existence of a discrepancy representing a wicked problem can be explained in numerous ways. The choice of explanation determines the nature of the problem's resolution. "Wicked problems contain many stakeholders, who all have different ideas about what the problem really is and how to solve it" [2, p. 101]. The stakeholder space for intellectual property lock-in is dominated by lawyers and policy makers, who believe the problem can be resolved with either more statute or further policy, respectively.

The planner (read: problem solver) has no right to be wrong. The problem solver is held responsible for actions taken to resolve a wicked problem, because those actions, by the nature of wicked problems, will have a large impact and are difficult to justify. One possible solution to the intellectual property lock-in problem is purchasing a right to the underlying intellectual property. This potential multi-million dollar decision may be right, but it may also be a waste of money; regardless of the outcome, the program manager who made the decision will have to account for the success or failure.

Rittel and Webber provide this descriptive classification of wicked problems, but offer no prescriptions for addressing these problems. The research herein also makes no attempt at reaching a prescriptive solution for wicked problems, but does endeavor to add meaningful understanding to the problem of intellectual property lock-

in; specifically in the domain of software in the Department of Defense (DoD). For reasons already enumerated in the discussion of wicked problems, the scoping of the domain is necessary to define a quasi-tractable subordinate problem of the overall wicked problem. Consequently, the findings herein will address only the specific formulation of the problem studied, but will hopefully guide future research efforts on other subordinate problems or the wicked problem as a whole. To guide this inquiry two overarching research questions are used:

Research Questions:

1. What mechanisms link formal software architecture and intellectual property licenses to induce intellectual property lock-in?
2. How can software be objectively evaluated to identify potential incidences of intellectual property lock-in?

This research draws no explicit hypothesis for deductive investigation; instead it uses two postulates to guide an inductive inquiry. Given the nature of wicked problems, these postulates cannot be validated simply through the research herein. Instead the goal is to offer empirical support for the existence of these postulates as one possible explanation for intellectual property lock-in in DoD software acquisitions. These postulates are informed from the authors experience in practice.

Postulates:

1. Intellectual property lock-in occurs at the conjuncture of technical architecture and contract structure
2. Intellectual property architecture is a mechanism of action for intellectual property lock-in

Although these postulates are listed at the outset for the reader's convenience, terms such as "intellectual property architecture," "technical architecture," "contract structure," and the theoretical foundations for each postulate remain undefined until subsequent Chapters. These Chapters are organized to investigate the research questions and outline the theoretical basis for the postulates:

Chapter 2 Literature Review The first part of this Chapter is a chronological history of intellectual property regulations and statute in the Department of Defense. This section is necessary because it is illustrative of shortcomings in policy and lessons learned over the past 60 years of policy changes. The second part of the Chapter is a review of the literature on DoD intellectual property policy, which is further decomposed into legal literature and research based literature. The Chapter concludes with a discussion of gaps in the current body of literature.

Chapter 3 Quantitative Problem Framing This section addresses one of the largest gaps in literature outlined in the previous Chapter, a quantitative framing of the problem. The Chapter constructs a quantitative investigation to address the magnitude of the intellectual property lock-in problem within the DoD and identify temporal trends in intellectual property lock-in.

Chapter 4 Conceptual Model Drawing on the concept of bottlenecks, this Chapter defines the incidence of lock-in as a theoretical conjuncture of architectures. In doing so the Chapter explains the current DoD model for analyzing lock-in and explains the shortcomings of such a model.

Chapter 5 Scientific Methods Said simply, this Chapter explains the "shoulders of giants" on which the research herein is based and extends these theoretical foundations into a new methodology to address the research questions.

Chapter 6 Research Methods Investigation of any wicked problem requires detailed attention to any applied research methodology due to the many causal

interactions. This Chapter argues for a fuzzy-set Qualitative Comparative Analysis research approach and outlines the design of the specific research method as well as collection of qualitative and quantitative data.

Chapter 7 Contextual Case Details The discussion of wicked problems revealed that no two problems are identical; therefore, this Chapter outlines the contextual details of each case used in the research approach and explains the differences as well as similarities across cases.

Chapter 8 Cross Case Analysis This Chapter is focused on the execution of the fuzzy-set Qualitative Comparative Analysis outlined in Chapter 6. The reader can expect a detailed discussion of model calibration as well as an analysis of necessary and sufficient conditions for avoidance of lock-in.

Chapter 9 Discussion Whereas the previous Chapter was concerned primarily with execution of the model, this Chapter is focused on discussing the results and identifying theoretical as well as empirical findings. This Chapter spends time discussing not only impacts on practice, but also limitations, threats to generalizability, and future research.

Chapter 10 Conclusion The final Chapter is used to discuss the contributions of the overall work organized into three categories: theoretical, methodological, and empirical.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Literature Review

This chapter is segregated into two primary sections. The first is a chronological history of intellectual property regulations and statute in the Department of Defense (DoD). This section is necessary because it is illustrative of shortcomings in policy and lessons learned over the past 60 years of policy changes. The second section is a review of the literature on DoD intellectual property policy, which is further decomposed into legal literature and research based literature. The chapter concludes with a discussion of gaps in the current body of literature.

2.1 Definitions

Within the parlance of the DoD acquisition community "proprietary rights," "data rights," "technical data rights," and "proprietary data" are terms often used synonymously to refer to a contractor's intellectual property. Although these catchall phrases are semantically dissimilar and not interchangeable, they are nonetheless terms-of-art within the DoD. These catchall terms may refer to any of the conventional definitions of a patent, copyright, trademark, trade secret, or technical data. However, to avoid confusion "technical data" and "proprietary data" are used synonymously herein to refer to the specific sub-class of intellectual property and "data rights" is used to refer to the Government's right to a piece of contractor's intellectual property. Furthermore, since the focus of this paper is primarily on computer software, which started

as a subset of technical data within DoD regulation, the definitions of software and technical data are included below to both remind the reader of the meaning and frame the subsequent discussion using the formal definitions published in the Defense Federal Acquisition Regulation Supplement (DFARS):

Technical data "means recorded information, regardless of the form or method of the recording, of a scientific or technical nature (including computer software documentation). The term does not include computer software or data incidental to contract administration, such as financial and/or management information" DFARS 252.227-7013

Computer software documentation "means owner's manuals, user's manuals, installation instructions, operating instructions, and other similar items; regardless of storage medium, that explain the capabilities of the computer software or provide instructions for using the software" DFARS 252.227-7013

Computer software "means computer programs, source code, source code listings, object code listings, design details, algorithms, processes, flow charts, formulae, and related material that would enable the software to be reproduced, recreated, or recompiled. Computer software does not include computer databases or computer software documentation" DFARS 252.227-7014

The above definitions may vary slightly in other legal dictionaries, however, there is little argument over the definitions themselves. Instead, most issues arise from determining whether the Government has rights to a piece of intellectual property and subsequently what type of rights the Government is entitled. This is an important departure from conventional intellectual property law and bears discussing in further detail. Department of Defense data rights regulations define the rights to which the Government is entitled under a government contract, not the protections to which an intellectual property holder is entitled. This is in contrast to traditional patent or copyright regulatory schemes where protections are focused on the patent holder or copyright holder. Said in the words of legal scholar Lionel Lavenue, "Notably, a data right does not provide any rights to the contractor, instead, these rights tend to limit

rights that a contractor may have in data by requiring the license of the technology to the government - effectively, a compulsory license" [3, p. 29]. Lastly, it is important to note that the rights provided under the DFARS are not always the same as those provided under the FAR. The discussion herein solely focuses on the rights provided under the DFARS[1].

2.2 Policy Review

William Anderson, in an article for the Public Contract Law Journal [4], recounts the contracting actions of Captain Charles Wallace, who entered into a procurement contract with Wilbur and Orville Wright for delivery of the first "heavier-than-air flying machine" [4, p. 38] in 1908. Within this seminal contract was terse, yet effective, intellectual property language; which required no patent rights be transferred to the Government as a result of the sale and required delivery of drawings under the promise that they would not be shown to competitors. In total, the entire contract was printed on two legal-sized pages. [4]

Unfortunately, the Wilbur and Orville Wright days of contracting are a long-forgotten memory. Today terms such as "unduly confusing," "daunting," "onerous," and "counter-intuitive" [5, p. 466] are all common, and arguably accurate adjectives used to describe the DoD's regulations on intellectual property. A simple review of articles in public contract law journals reveals a chorus of both practitioners [4]–[9] and academics [3], [10], [11] who almost universally agree that the state of intellectual property regulation is either too difficult to understand or too protective of government interests. Furthermore, most argue that the confusing labyrinth of statute and policy deters innovation from large and small companies; companies would rather "forgo contracting with the Federal Government rather than risk the misuse or loss of valuable intellectual property" [12, p. 290]. Extending this argument further, one could argue that this vast uncertainty over intellectual property right ownership in

[1]See generally [3] for a thorough review of the differences between FAR and DFARS intellectual property rights

Government contracts is indicative of a near anti-commons scenario, which is defined as "a property regime in which multiple owners hold effective rights of exclusion in a scarce resource" [13, p. 668].

2.2.1 Post World War II

Although the current intellectual property regulatory system within the DoD may appear ineffectual at first glance, it is the culmination of many decades of positive developments. Most scholars agree that today there is a "finer balance between allowing contractors to retain intellectual property rights and granting the Government access to the technical data necessary" [12, p. 296]. However, it requires a discussion of the DoD's intellectual property political history to understand the benefits. For practical purposes this background begins in the post-World War II era and discusses only major statutory milestones in the policy evolution. Prior to World War II, few intellectual property rights issues are documented because the Government asserted no rights to the data delivered by contractors [3]. In fact, the topic did not appear in DoD procurement regulations until the 1949 revision of the Armed Services Procurement Regulation (ASPR), which was the first DoD policy to govern intellectual property. The original piece of regulation, section IX, titled "Patents and Copyrights" covered government acquisition rights in patents and copyrights, but failed to mention rights in proprietary data or trade secrets. It was not until the ASPR was amended in 1955 with the "Technical Data in Research and Development Contracts" clause, that the subject of proprietary data was specifically recognized. This new clause was to be included in all new research and development (R&D) contracts. Of note, the clause did not contain any provisions for protecting a contractor's proprietary information and no consideration was given to whether the proprietary data may have originated before contract award. The only Government limitation on the contractor's proprietary data was that the "Government's reproduction, use or disclosure of the contractor's submitted data must be for Governmental Purposes" [14, p. 306].

In 1957, the ASPR was amended to amplify Government rights to contractors' intellectual property. This regulation recognized a contractor's proprietary data

through the introduction of two new clauses: unlimited clause and limited clause. In the new "Rights in Data – Unlimited" clause, which applied to R&D contracts, the Government acquired the right to "duplicate, use, and disclose in any manner and for any purpose whatsoever, and have other so do, all Subject Data[2] delivered under this contract" [14, p. 306]. Under this clause, contractors were obligated to deliver all forms of intellectual property including trade secrets, which the Government could then furnish to competitors. Conversely, in the "Rights in Data – Limited" clause, which applied to supply contracts, the Government mandated delivery of only those rights which it required for a limited purpose; "proprietary data need not be furnished unless suitably identified in the delivery schedule of the contract" [15, p. 72].

As probably anticipated by the reader, the 1957 revision to the ASPR, which entitled the Government to unlimited rights for R&D contracts, resulted in many complaints from contractors. Consequently, in 1958, the Government revoked the previous ASPR language in favor of new language that permitted for the withholding of proprietary data "developed at private expense." Although a theoretical step forward for contractors, this new regulation caused problems because the term "developed at private expense" was not defined. As a result, the policy had a negative impact on military readiness in that contractors began to omit and withhold excessive information under the auspices of the undefined phrase, resulting in expurgated, or more commonly referred to as "swiss cheese" technical data [3].

As a response to the excessive omission of data by contractors, the DoD, in 1964, moved away from the concept of withholding pieces of proprietary data developed at private expense and transitioned to the concept of requiring delivery of contractor proprietary data with limited rights[3]. This concept of limited rights in proprietary data is part of the basic framework which is still in use today. To determine the appropriate set of Government rights, a contracting officer used the funding test: if the data were developed solely at private expense, then the Government was entitled to limited rights; conversely, if the data were developed solely at public expense then

[2]Subject Data is defined elsewhere in the amendment to include, "writings, sound recordings, pictorial reproductions, drawings, or other graphical representation and works of any similar nature (whether or not copyrighted); which are specified to be delivered under this contract"

the Government was entitled to unlimited rights. However, if the funding was a mix of public and private, the regulation required unlimited rights in these mixed-funding situations [12]. Again, key terms such as "developed" and "private expense" remained undefined by 1964 regulations, which resulted in ambiguities in mixed-funding situations. As a result, contractors remained unsatisfied with the conceptual framework for governing proprietary data rights.

During the 1970's little progress was made to improve the scheme for governing proprietary data with two minor exceptions. In 1974 regulations were revised to include a new category of data, computer software, which followed the same rules as other forms of proprietary data, but would become a more prominent issue in the 1980s and 1990s as the Government began to procure more software-based weapon systems [5]. Additionally, the term "developed" was formally defined in 1974 to assuage some of the legacy ambiguity; however, the term "private expense" remained undefined [3].

2.2.2 Congressional Interventions

In contrast to the relatively quiet 1970s, the 1980s were marked with multiple Congressional interventions, which enacted statutory provisions requiring more protection of intellectual property in DoD regulations. Prior to this point, intellectual property was governed solely using the ASPR; however, the beginning of the 1980s marks a shift from DoD self-regulating intellectual property policy to Congressionally directed regulations in statute. The first major piece of legislation, although not specifically targeted as a DoD reform, was the Bayh-Dole Act, in 1980, and the ensuing 1983 President Reagan Executive Order that broadened the scope of the Bayh-Dole Act to extend the Act's provisions to any Government contractor regardless of nonprofit or for-profit status [6]. "By doing this, President Regan was hoping to attract more for-profit entries into the government R&D arena with the carrot that they would be

[3]Limited Rights means "rights to use, duplicate, or disclose technical data in whole or in part by or for the Government, with the express limitation that such technical data may not be released outside the Government, or used, duplicated, or disclosed in whole or in part for manufacture or procurement"[15, p. 78]

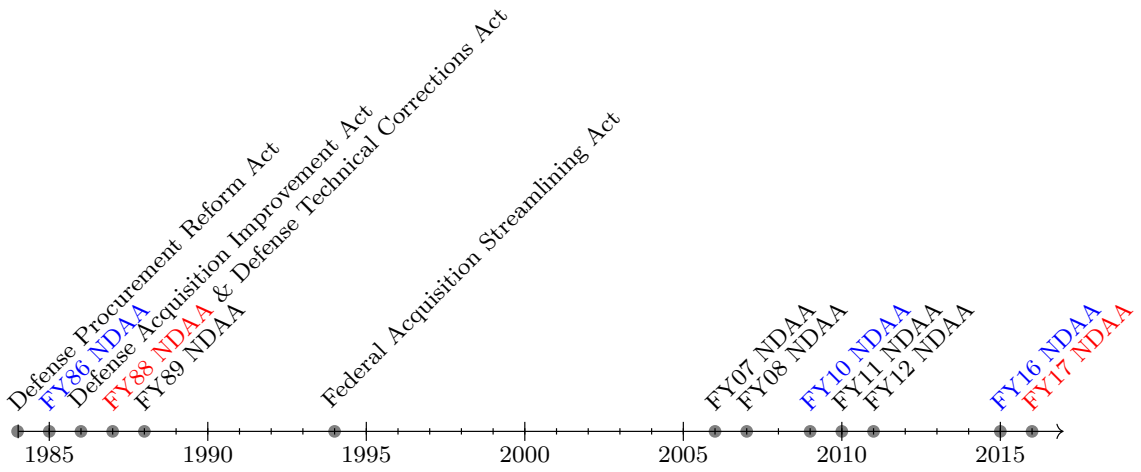


Figure 2-1: Timeline of 10 U.S.C. §§ 2320 and 2321 Amendments. Simultaneous amendments to both are denoted in black, and amendments to either 2320 and 2321 are denoted in red or blue respectively

able to retain some intellectual property rights" [6, p. 68]. For all legislation subsequent to Bayh-Dole Act, there is some argument in literature regarding the genesis of these statutes; some argue they are a direct result of Congress' "disgust with the lack of [DoD] progress on data rights" [3, p. 15] and others argue they are a result of spare-parts breakout scandals[4] of the early 1980's [12], [17]. Regardless of the reason, Congress enacted further statutory requirements for DoD's acquisition of technical data under the Defense Procurement Reform Act of 1984[5] which amended Title 10, United States Code (U.S.C.), to include sections 2320, titled *Rights in Technical Data*, and 2321, titled *Validation of Proprietary Data Restrictions*. These two new sections of U.S. Code, although amended many times (see Figure 2-1), still serve as the primary governing statutes in contemporary DoD acquisitions.

[4]The DoD is encouraged, and sometimes required, to compete the acquisitions of spare parts (i.e. reparable and consumable parts of a weapons system) to obtain the lowest possible price. However, in the early 1980's contractors are alleged to have obscured the competition of spare parts by using restrictive legends on drawings and engineering data. Consequently, because technical data on the spare part was proprietary, competition was precluded. For example, a 1982 GAO report cites a C-5 aircraft door seal purchased from Lockheed Martin for $188 per unit as an instance of spare parts abuse. When questioned about the high cost, Lockheed Martin told the Government it was an "extrusion" product that had to have holes drilled in it. Because all the engineering data and drawings were proprietary the Government was unable to verify the contractor's claim and had to accept the price at face value. However, after delivery, Air Force technical engineers examined the part and concluded it could be fabricated for approximately $4.62 per unit [16]

[5]Pub. L. No. 98-525 §§ 1201-1252, 98 Stat. 2588

This act marked the first time requirements were specified for acquisition of technical data rights by emphasizing the need for better planning when procuring major weapon systems by setting forth procedures to validate the rights in proprietary data claimed by contractors [3], [12], [18]. Additionally, the Defense Procurement Reform Act tasked the DoD to define the terms "developed" and "private expense." Later Congress enacted another reform, the Defense Acquisition Improvement Act of 1986[6], which reclassified the scheme used by the Government to determine ownership rights in mixed-funding situations. Instead of choosing between unlimited or limited rights, the new statutory guidance and subsequent regulations gave way to a new type of ownership rights; "Government Purpose License Rights[7]" [3]. Under this new form of license, the Government maintained the right to use the data, where necessary, for government purposes, but the ownership of the data remained with the contractor. The 1980s concluded with an interim regulation, in 1988, dealing specifically with computer software. Prior to this regulation, computer software was treated as proprietary data, but the interim regulation split the regulation into specific sets of rules for computer software and for proprietary data [3].

Although the goal of the Congressional interventions, in the 1980s, was to bring equity to a cumbersome DoD intellectual property regulation schema, that harmony between industry and DoD was never reached. Consequently, in the National Defense Authorization Act of 1991[8], Congress directed the formation of a working group to review acquisition laws applicable to the DoD with a view toward streamlining the defense acquisition process. This committee was known as the "Section 800 Committee" after the section of the 1991 Act that empowered it. Although not specifically directed by Congress to review intellectual property laws, the Committee was given sweeping powers to review all areas of DoD acquisition. The Section 800 Committee

[6]Pub. L. No. 99-661 § 953, 100 Stat. 3910, 3949 (1986)

[7]Government Purpose License Rights, later renamed Government Purpose Rights (GPR), "is a license level that falls between unlimited rights and limited rights. GPR allows the Government to [u]se, modify, reproduce, release, perform, display, or disclose technical data within the Government without restriction; and [r]elease or disclose technical data outside the Government, as well as authorize persons to whom release or disclosure has been made to use, modify, reproduce, release, perform, display, or disclose that data for United States government purposes" [6, p. 75]

[8]Pub. L. No. 101-510 § 800, 104 Stat. 1587 (1990)

created a working group to review intellectual property laws and eventually published a chapter of findings and recommendations, which included the treatment of technical data in DoD acquisitions [19]. The findings of the Section 800 committee recommended a "new alternative approach for dealing with technical data that, instead of focusing on rights, focuses on the Government's need to ensure reasonable life-cycle costs, ordinarily through competition, for spare parts and other follow-on purchases" [19, p. 5-5]. This approach focuses not on the distribution of rights, but rather on ways to ensure that the Government has the means to ensure procurement prices are reasonable. However, the subsequent recommendations were met with criticisms from both Government as well as Industry and were not ultimately adopted [3].

Shortly after, Congress established the formation of another Committee in the National Defense Authorization Act of 1992[9]. Although the Section 800 Committee's work was still ongoing, the second committee, titled the "Section 807 Committee", was chartered solely to review DoD data rights statues. The Section 807 Committee was comprised of federal acquisition experts from academia, industry, and DoD. The overall goal of this committee was to reach a compromise on the proper balance of rights between the DoD's necessity to maintain a system over its life-cycle and industry's necessity to protect costly investments. Unfortunately, the committee never reached a unanimous decision and eventually released a majority [20] and minority [21] position. The recommendations of the majority position were ultimately adopted by the DoD in April of 1994.

The next major piece of legislation to impact procurement of proprietary data and computer software was the Federal Acquisition Streamlining Act (FASA) of 1994. It made several changes to the DoD regulations; the most important of which were the separate treatment of commercial and noncommercial items and a requirement to grant a minimum "Government Purpose Right" license in all mixed-funding situations [3], [12]. Under this new system, the technical data and computer software from items procured as commercial items (e.g., commercial-off-the-shelf software), were

[9]Pub. L. No. 102-190, § 807, 105 Stat. 1421 (1991)

Rights Category	Applicable to TD or CS?	Criteria for Applying Rights Category	Permitted Uses within Government	Permitted Uses Outside Government
Unlimited Rights	TD and CS	Development exclusively at Government expense; also any deliverable of certain types regardless of funding	Unlimited - no restrictions	
Government Purpose Rights	TD and CS	Development with mixed funding	Unlimited; no restrictions	Only for "Government purposes"; no commercial use
Limited Rights	TD only	Development exclusively at private expense	Unlimited, except may not be used for manufacture	Emergency repair/overhaul; evaluation by foreign government
Restricted Rights	CS only	Development exclusively at private expense	Only one computer at a time; minimum backup copies; modification	Emergency repair; certain service and/or maintenance contracts

Table 2.1: Rights in *Noncommercial* Computer Software (CS) and Technical Data (TD) Covering Noncommercial Items Rights (Reproduced in part from [18])

treated differently from noncommercial items. Tables 2.1 and 2.2 outline the major differences between data rights in noncommercial (Table 2.1) and commercial (Table 2.2) items. Although over twenty years old, this method for determining licensing rights in technical data and computer software is still in use today. However, it is important to note that there are many special cases that warrant exceptions to the simplified explanations in Tables 2.1 and 2.2. These Tables are meant solely for the purpose of explaining the difference between the two categories and should not be treated as an exhaustive resource for evaluating data right ownership (for a thorough explanation see [18]).

The next major piece of legislation to impact DoD procurement of technical data was the John Warner National Defense Authorization Act (NDAA) for Fiscal Year 2007[10]. This piece of legislation required the DoD to, "require program managers for

[10]Pub. L. No. 109-364 § 802, 120 Stat. 2312

Rights Category	Applicable to TD or CS?	Criteria for Applying Rights Category	Permitted Uses within Government	Permitted Uses Outside Government
Unlimited Rights	TD only	Any TD of certain specified types or classes, regardless of commercial status	Unlimited - no restrictions	
Standard DFARS Right	TD only	Default rights category for all TD covering commercial items	Unlimited, except may not be used for manufacture	Only for emergency repair overhaul
Standard Commercial License	CS only	Default rights category for all commercial CS	As specified in the license customarily offered to the public, DoD must negotiate for any specialized needs	

Table 2.2: Rights in *Commercial* Computer Software (CS) and Technical Data (TD) Covering Commercial Items Rights (Reproduced in part from [18])

major weapon systems and subsystems of major weapon systems to assess the long-term technical data needs of such systems and subsystems and establish corresponding acquisition strategies that provide for technical data rights needed to sustain such systems and subsystems over their life cycle" ([22] citing 10 U.S.C. § 2320(e)). This is a difficult statute to comply with, as it asks members of the DoD acquisition community to predict what data rights are needed in the future. Being that there is no readily agreed upon method for accomplishing data rights forecasting, Eli Mazour, in his article for Public Contract Law Journal, correctly points out that the easiest way to comply with this new law, "is to acquire as many rights as possible in as much technical data possible" [22, p. 681]. The logic being, if one acquires all possible data rights then one is certainly prepared to sustain a weapon system over its life cycle.

2.2.3 Architecture and Intellectual Property

After 2007, there considerably less scholarly work that either aggregates or offers analyses of changes to DoD regulations or statute. However, what follows is a summary of updates which accrued since the last previously published scholarly work of material importance. After minor updates to technical data statutes occurred in the National

Defense Authorization Acts of 2010[11] and 2011[12], a substantive change to statue was enacted in the National Defense Authorization Act of 2012[13]. This law authorized the DoD to release proprietary data that is commonly referred to as "interface data", which was defined in the Act of 2012 as data "necessary for the segregation of an item or process from, or the reintegration of that item or process (or a physically or functionally equivalent item or process) with, other items or processes." This amendment to 10 U.S.C. § 2320 is significant because it marks the emergence of theoretical principles of architecture in DoD intellectual property statute. The goal was to permit the government rights to the interfaces, which would theoretically allow for competition of any modular system. However, in practice it is difficult to determine what data is necessary for "segregation." This uncertainty over data necessary for segregation and questionable applicability of the amended statute to systems, which do not utilize a modular architecture, muted the impact of the amendment.

This changed with the National Defense Authorization Act of 2015[14], which required the DoD to develop standards and define architectures necessary to enable open systems approaches and mandated DoD acquisition programs employ Open Systems Architectures to the maximum extent practicable. This legislation reinforced the importance of the 2014 amendment to 10 U.S.C. § 2320 and the ability to release interface data.

The last material change to DoD intellectual property statute occurred in the National Defense Authorization Act of 2016[15]. This legislation took two important steps. First, it amended 10 U.S.C § 2321 to rescope the definition of *developed solely at private expense* for commercial items. The amended U.S.C. permitted industry to claim private expense status for components or subsystems of major systems if the major system was acquired as a commercial item. Although, logical, these changes

[11]Pub. L. No. 111-84 § 821, 123 Stat. 2411 (2009), which authorized "a covered Government support contractor access to and use of any technical data delivered under a contract for the sole purpose of furnishing independent and impartial advice or technical assistance directly to the Government"

[12]Pub. L. No. 111-383 § 824, 124 Stat. 4269 (2010), which amended both U.S.C. § 2320 and § 2321; the changes enacted in this law were either later reversed or not a substantive change to statute.

[13]Pub. L. No. 112-81 § 815, 125 Stat. 1491 (2011)

[14]Pub. L. No. 113-291 § 801, 128 Stat. 3425

[15]Pub. L. No. 114-92 § 813, 129 Stat. 891

again introduce theoretical principles of architecture (e.g. reducibility of components) into DoD intellectual property statute. Additionally, the Act of 2016 directed the formation of another Government-Industry Advisory Panel, referred to as the "Section 813 Panel," to review 10 U.S.C. §§ 2320 and 2321, "ensuring that such statutory and regulatory requirements are best structured to serve the interests of the taxpayers and the national defense." At the time of this writing the results from the Section 813 Panel were not released.

The historical description above is a simplified summary of the actual regulations and how they are applied in practice. Within actual regulations there are countless exceptions to policy and various exceptions to the exceptions in policy. Consequently, the outline above is meant to be an initial explanation of the plain language in government regulations, but makes no attempt explore or explain the relating case law that might have a substantive impact on the actual application of regulations in practice or all the exceptions to policy.

2.3 Literature

The literature review herein departs slightly from a traditional research literature review in that legal literature is also included. Legal literature differs from research literature in that it does not employ a research based methodology. There are few quantitative results and none of the arguments are backed by statistical significance. Nonetheless, the inclusion of legal literature is paramount because that is the primary domain of DoD intellectual property regulation. The authors below are the foremost thinkers on the topic and make arguments advancing practice using case law and statute as the foundation.

2.3.1 Legal Literature

All legal literature reviewed, with the exception of Lavenue [3], unanimously argues that intellectual property regulations are overly complex[16]. The arguments against

overly complex regulations will be addressed shortly, but prior to beginning it is important to investigate Lavenue's counterargument.

> "With an infancy in the patent clauses of the 1940s, and following various incantations over the last fifty years, the 1995 data rights regulations represent not only the most considered and precise regulations for the protection of computer software and technical data, but also the most fair." [3, p. 68]

Lavenue argues that the regulations are not overly complex, but rather "precise." He argues this precision is necessary for the type of work contracted under the DFARS and also drives equity amongst interested parties. Most literature that criticizes the complex nature of DoD intellectual property regulations are doing so relative to commercial intellectual property statutes. Lavenue makes the point that the needs of the military are drastically different from the commercial industry and that the regulations are a manifestation of this difference. Lavenue supports this claim by citing an example where the military must frequently make repairs in austere locations far away from the contractor who built the system; in such cases technical data is often a necessity to accomplish the military objective. The civilian intellectual property schema offers no consideration of such exigent circumstances, which is why it may be less complex.

Lavenue's arguments are rational, but pose the question what is the correct balance between the specificity needed to handle situations unique to the military, but not overly burdensome to those contractors who wish to do business with the DoD? Identifying this optimal balance is the topic of much of the remaining legal literature. The format for each of these legal analyses is similar, the author reviews the history of intellectual property regulations in the DoD, explains the current regulations and makes one or more recommendations to improve the balance. Given the extensive discussion of intellectual property regulation history in the first part of this chapter, what follows is a discussion of only the arguments of each author(s).

[16]See also Petrillo [23] for an editorialized argument against reducing complexity in DoD Acquisition Regulations

Lazure et al.[24], Trend [12], Sidebottom [6], Murray [17] and Mazour [22] argue that the government is, at times, limited by statute. The premise of this argument revolves around the language in the FY07 NDAA, which created a statutory requirement for program managers to foresee long-term data rights needs. Although well intentioned, compliance is difficult because there is no methodology to forecast data rights needs. Therefore, to comply with this statute most program managers acquire more data rights than are truly needed. All six authors cite this as the prime reason why tension exists between industry and government; arguing that if this statutory requirement were relaxed it would allow the DoD to acquire a smaller set of rights at later dates in the contract. Theoretically this argument is valid, deferring the acquisition of data rights to the latest point possible will allow for the purchases of the minimum possible rights, but in doing so it also forfeits most of the DoD's negotiating power. Interestingly, all of the authors addressed this criticism under the lens of timing the purchase of data rights to acquire the least amount of rights necessary, but none examined it from the perspective of improving the methodology to determine which rights are needed.

Anderson [4], Trend [12], Sidebottom [6], and Mazour [22] argue that the regulations need to either be simplified or unified across all Government agencies. This criticism is rooted in the hierarchical nature of federal acquisition regulations. Depending on which agency of the government a contractor is doing business with, the intellectual property regulations may vary. For example, a defense contractor doing business with the Coast Guard, Air Force, National Aeronautics and Space Administration (NASA), and Department of Homeland Security will face four different intellectual property schemes. Even more exemplary of the problem, a contractor selling identical satellite technology to the Air Force and NASA is subject to the regulations in the FAR, DFARS, and Air Force Federal Acquisition Regulation Supplement (AFFARS) for one contract and subject to FAR and NASA FAR Supplement (NFS) in another. Although the product and intended use may be identical, the acquisition regulations are different. These differences make doing business with the government burdensome, because of the different rules for each potential market. The authors

make the point that if the system were unified it would reduce complexity as well as tension between industry and government. The argument is rational, but it does not consider Lavenue's counterargument that precision in regulation is necessary for specific agency objectives [3]. It is possible that there are not compelling agency objectives that drive multiple sets of regulations, but without subsequent analyses it makes determining the validity of this argument difficult.

Dungan [5], Lill [11] and Mazour [22] argue that the DoD's current system of marking intellectual property is overly onerous and simplifying the requirement would attract more potential innovators. Under current regulations there are two ways for a contractor to protect data in transactions with the government: mark the data using restrictive legends or withhold the data entirely [11]. A restrictive legend is designed to notify the government of its rights to any information contained in a deliverable. In regulation, there are three ways a contractor can fail to comply: by not marking data, by using unauthorized marking, or failing to correctly mark data. If a contractor fails to mark data, the government assumes the data is furnished with an unlimited right. If a contractor marks the data, but is unauthorized to do so under the DFARS or FAR, the government may remove or ignore the markings. Lastly, if a contractor marks the data incorrectly or uses a nonconforming marking, the Government may ignore the marking. In practice, the execution of such rules is typically not as definitive and contractors are often given an opportunity to correct markings even in cases where the regulations do not require such consideration. However, the nature of these regulations make companies wary of doing business with the DoD because of the potential that they will make an unintentional "gift" of intellectual property simply by failing to mark data correctly. The authors argue this marking requirement is a significant disincentive to promoting innovation and a large issue driving tension.

Dungan [5], Trend [12], and Mazour [22] argue that the DoD should eliminate the unlimited right. The authors argue that the differences between the government purpose right and unlimited right are subtle and that the government purpose right is adequate for all purposes while still protecting a contractor's data. Mazour goes even further in arguing against the unlimited right, "the only justification for the Govern-

ment to have such broad rights is to provide it with an excuse if it mishandles the technical data."[22]. These arguments are valid, but similar to other arguments fail to address Lazour's military necessity counterargument. At the outset of a contract the DoD is not capable of foreseeing all possible use cases for a system or military objectives. In such cases where the unlimited right is purchased at the outset of a contract, the right provides an avenue to hedge against uncertainty. While it is possible that a Government Purpose Right may suffice in such situations, none of the authors address the counterargument.

Murray [17] and Sharp [8] take a different approach by arguing that regulations are not necessarily overly complex, but that education in the regulations is lacking. This is a salient point. There are entire legal institutions and firms whose sole purpose is to either instruct or assist in arbitrating commercial intellectual property matters, but the same institutional instruction and arbitration capacity does not exist for DoD intellectual property matters. By virtue of this, there are fewer legal experts on DoD intellectual property. Outside of legal expertise, there are also few educational opportunities for practitioners[17]. This point is also observed in the literature reviewed. Over half of the scholarly legal articles reviewed were andrological in nature [3], [4], [8]–[10], [15], [24]–[27]. Their sole purpose was to help advance the understanding of DoD intellectual property policy versus advance policy with an argument. Some made the occasional closing argument for simplifying regulation at the conclusion, but these arguments were not the objective of the article. This is further evidenced by many of the article titles, which are similar to Sharp's title, *A Layman's Guide To Intellectual Property In Defense Contracts* [8]. There is an additional set of literature, which serves as a "how-to" guide for contractors to protect their data [7], [28]–[31]. These are generally characterized by the title of Baker et al., *"GOVERNMENT DATA RIGHTS CHALLENGES: Tips on How to Avoid & Respond"* [30]. DoD intellectual property experts in both academia and practice are few. The complex nature of regulations *as well as* the lack of expertise is part of the problem. However, it is

impossible to argue causality between these two variables, as fixing one would likely not fix the other.

Larson [10] and Mazour [22] argue for the creation of a R&D funding "watermark." This would create a threshold in mixed funding situations where the government would have to fund a certain percentage before a government purpose right would vest. The premise of this argument is that a private contractor could fund 99% of the development costs and still have to relinquish a government purpose right. Although seemingly valid, it is important to note that the inverse of this situation is also possible; where the government could fund 99% and still only receive a government purpose right instead of an unlimited right. The difficulty with establishing a funding threshold is that it incentivizes contractors and the Government to game the system; staying below or above the watermark. Larson acknowledges public criticism of "watermark" efforts because, without some form of license, the public could pay twice for inventions; "once when taxes are used to fund R&D and again via high prices due to a monopoly on the invention" [10, p. 199]. Any solution would have to balance these concerns.

The arguments presented, thus far, are those where there was consensus among legal experts; that is more than a single author argued the point. However, there are other arrangements set forth by authors that lack consensus, but still merit consideration. Sidebottom [6] argues regulations should not allow for Government march-in rights, which permits the Government to force a contractor to license a piece of intellectual property to another contractor under conditions set forth in 35 U.S.C § 203. The premise of this argument is that this unique Government right is a disincentive to potential innovators. Although Dungan also argues for the removal of march-in rights, he softens the argument, "the value of the march-in provision lies more in the threat of its use than in its actual use" [5, p. 183]. This assertion is supported by Rabitschek et al., who note the government has not exercised march-in rights since 1980 [32].

[17]The Defense Acquisition University offers courses on intellectual property, but completion of these courses is not mandatory and the material is not of sufficient detail to cover the entirety of regulation and statute

2.3.2 Research Literature

The above summary of legal literature is now juxtaposed against a much smaller body of research based literature. Research based literature are those contributions which are based on a formal research based methodology.

Muzzelo and Arndt hypothesized that the government ownership of data rights makes no difference on the transition of technology projects from lab to field [33]. To test this hypothesis, they drew data from the 71 Army Science & Technology (S&T) Advanced Technology Development (ATD) programs. Of the initial 71 programs, 57 were selected for final analysis. These 57 programs were then scored if a project successfully transitioned from the S&T community to a Program of Record (PoR) according to the category of government rights (i.e., limited, government purpose, or unlimited). Those that transitioned were given a value of 1 and those that did not transition were given a value of 0. Using an ANOVA statistic test Muzzelo and Arndt were able to reject the null hypothesis ($p = 0.024$) and accept the alternate hypothesis being that government ownership of technical data rights makes a difference in technology transition.

Although an interesting finding the generalizability of Muzzelo and Arndt's conclusions is questionable. First, the sample collected only represents programs from the Army. This is problematic because intellectual property regulations are executed with minor differences between the services. No action was taken to control for these differences. Furthermore, Muzzelo and Arndt do not explain the methodology used to reduce the sample from 71 to 57. They simply state, "excluding the data determined to be meaningless" [33, p. 640]. Without understanding this process, it is difficult to determine what impact, if any, this sampling had on the end result. Lastly, assuming the research were entirely valid, the results only speak to the transition of technology from the S&T community to PoR. Consequently, these results are likely not generalizable to technologies that do not start in the S&T community; which is the majority of DoD R&D efforts.

Liedke and Simonis [34] analyzed contract data from Army Tank Automotive and Armaments Command (TACOM) FY12 Annual Competition Report to determine if the government's lack of data rights impacted the Army's ability to competitively acquire non-commercial hardware. In FY12, TACOM executed $3.7 billion on a competitive basis (38% of total budget) and $9.7 billion on a noncompetitive basis (62% of total budget). Seven out of the top ten dollar value non-competed contract actions cited lack of technical data rights as a primary factor in preventing competition. Liedke and Simonis research is the only one which reviewed the Justification and Authorizations[18] (J&A) for each noncompetitive contract. Their research led to interesting findings, albeit with a small sample:

> "Most of the J&As reviewed discussed how the government requested that the contractor provide a cost estimate for selling unlimited rights to the government for their data. In all these cases, the estimate the government received was either so high that procuring the data rights was far from being a cost-effective approach, or the contractor refused to even provide an estimate whatsoever." [34, p. 46]

The small domain specific sample collected may reduce the generalizability of Liedke and Simonis results, but the research efforts are salient because they represent the first attempt at quantifying the impact of lacking data rights on DoD acquisitions. Although it is not possible to extrapolate trends for the entire DoD based on this research, it introduces a methodology for understanding and analyzing trends in DoD noncompetitive acquisitions due to lacking data rights. This work was later built upon by Berardi et al, [35] and is discussed in great detail in Chapter 3.

Lastly, there is a tangential field which deals with researching methodologies to equitably price technical data. Hasik [36] argues the benefits of a heuristic economic model, which is adapted to the DoD from a model originally presented by Dixit [37]. Hasik does not conclude any findings directly, but presents a model for use by Program Managers in evaluating the economic outcomes from the purchase of data

[18] A Justification and Authorization is a contracting artifact used to justify why a contract was not competitively procured. This artifact is discussed in detail in Chapter 3.

rights. Ross [38] compares payoffs in a game involving a duopoly of defense firms and the government to analyze potential cost savings to the government by purchasing technical data rights. He concludes the price largely depends on a rival firms R&D as well as production costs. Importantly, Ross notes that the DoD "should tend not to purchase [technical data] while blindly assuming it will minimize costs through competition" [38, p. 468]. Head and Nelson [39] use a two-stage bargaining problem to identify the factors that determine the value, to the government, of licenses in software acquisitions. Finding that two factors must occur for the procurement of expanded license[19] to yield a cost savings. First, the license must be transferable to alternative suppliers. Second, the information procured under the license must be useful to alternative suppliers in a way that materially lowers production costs. Head and Nelson also generally conclude, "the separate acquisition of additional data rights will not always lead to future cost savings" [39, p. 31].

These valuation models are important because they theoretically demonstrate that the procurement of data rights does not always correlate to a cost savings. This is not a conventional understanding in practice; research efforts to dispel this misconception are imperative. However, the use of these models in practice is limited by both their complexity and, in some cases, the simplifying assumptions used in the models.

2.3.3 Literature Gaps

The literature presented in this review was organized into legal literature and research literature, with a preponderance falling into the legal literature category. This general organization demonstrates the largest gap in DoD intellectual property literature.

Addressing the gaps between categories first, the legal category and research category are essentially silos of academic thought with little to no connection between the two fields. The result is rather limited discussions that only consider arguments that fall within each domain. For example, in the entirety of the legal authorship

[19]Head and Nelson use the term "expanded license" to refer to any class of license that offers greater rights than a restricted right or the rights that are conveyed by commercial licenses available to the public [39].

examined for this review, not a single author argued for a non-legal solution (i.e., a solution that does not involve changing policy or statute). The best example of this phenomena is the argument concerning the government's limitation by statute. Six authors all agreed on the salience of the argument and six authors all made recommendations within the legal silo. The premise of this argument is that the DoD does not have the methodology to determine what rights are needed in the future. However, each author ignores advancing the methodology and seeks to remedy the argument through adding additional complexities to statute and regulation. The situation is akin to the old adage *"everything looks like a nail when all you have is a hammer."* Consequently, the gap is a lack of cross domain approaches, which combine legal expertise, but also engineering and business practices, in an attempt to advance the methodology of data rights forecasting and making effective business cases around such decisions. Some of this cross-domain effort is evident in the recent NDAAs which are beginning to target system architecture as a viable methodology.

Within each category, the gap is evidenced by the disproportionate amount of legal arguments relative to the amount of research that employs a formal academic based research methodology. Given the extensive, and oftentimes contentious, DoD intellectual property history, one may expect that there has been much study around the impacts of these policies on competition levels in defense spending. However, there are less than five articles of research which address even these basic questions. The absence of such analyses and quantitative analysis is concerning because it raises the question, what data were used to inform policy decisions?

This literature review concludes that gaps are twofold: cross-domain approaches towards identifying methodological approaches to analyzing data rights and quantitative analyses on the impacts of intellectual property policy on DoD acquisition.

2.4 Conclusion

This chapter began with a discussion of the history of intellectual property regulations in the DoD. Beginning with the Wright Brothers contract for a heavier-than-air flying

machine and transitioning through the early years of the ASPR to the early 1980s and the inclusion of the Bayh-Dole Act. The history continued with discussion of congressional interventions in DoD regulations and the transition from self-governance in DoD regulation to external governance in statute. The first section ultimately concluded with a discussion of the current changes in statue. This historical background is important in the context of DoD intellectual property because it is necessary to understand why many of the regulations take the shape they do today. For each piece of regulation there is likely an event in the history of DoD acquisition that crystallized its need. Without understanding the history of the regulations, it is possible to overlook previous lessons learned and undo advances in policy.

The second half of this chapter examined the literature surrounding DoD intellectual property regulations and statute. Although this literature spans over half of a century, the preponderance of it originates from law review journals. After examining both legal and research based literature, this review concluded there were two major gaps in literature. First, a lack of cross-domain solutions and second, too few quantitative analyses of data to inform policy decisions.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Quantitative Framing

This Chapter builds upon one of the literature gaps identified in Chapter 2 and endeavors to analyze Department of Defense (DoD) acquisitions data to capture metrics on either the increase or decrease in use of contracts awarded using FAR 6.302-1(b)(2) exceptions to other than full and open competition; which are used herein as an indicator variable for intellectual property lock-in. The choice of using FAR 6.302-1(b)(2) as an indicator variable comes with limitations that are discussed at length. However, it is the best available variable given the data publicly available. The methods utilized herein look for broad changes in DoD competition patterns with a specific focus on contracts awarded using noncompetitive procedures due to FAR 6.302-1(b)(2) exceptions. These metrics and resulting data are instrumental to inform future legislative and policy decisions.

3.1 Guiding Quantitative Questions

Little of the previous literature, with exception of Berardi et al., [35] and Liedke and Simonis [34], endeavored to either quantify the magnitude of intellectual property impact on DoD acquisitions or assert quantitatively whether there were any distinguishable trends. These avenues of investigation are imperative because policy makers must understand the magnitude of the problem before making any cost benefit decision about adopting or directing changes. To address these fundamental questions,

this analysis turns to the Federal Procurement Management System (FPDS), which has been used by previous authors to successfully analyze trends in DoD acquisitions [40]–[42]. Although there are limitations in this database, FPDS is a sought-after data source because it contains up to 225 variables for each contract. This analysis uses two research questions to guide the investigation:

1. What is the magnitude of FAR 6.302-1(b)(2) exceptions on DoD acquisitions?
2. What trends, if any, are there in FAR 6.302-1(b)(2) exceptions on DoD acquisitions?
   (a) What trends exist in the frequency of FAR 6.302-1(b)(2) JnAs?
   (b) What trends exist in the products or services acquired using FAR 6.302-1(b)(2) JnAs?

3.2 Background: Policy and Statute

As discussed in Chapter 2, intellectual property regulations and statute within the DoD are among the most central issues in current policy discussions. This is further evidenced by the appearance of intellectual property initiatives in all three versions of the Defense Acquisition Reform policy initiative titled Better Buying Power (BBP) [43]–[45]. The most recent iteration BBP 3.0, outlines three strategies which confront the issue of intellectual property in DoD procurement. The first, *Remove Barriers to Commercial Technology Utilization*, argues that the DoD should capture private sector innovation by using commercially available technologies and products, but directs further analysis of the implications on intellectual property. The second strategy, *Increase The Productivity of Corporate Independent Research and Development (IRAD)*, targets the misuse of IRAD funds by defense contractors on "de minimis investments primarily intended to create intellectual property" [43] to secure a competitive advantage in future DoD contracts. The final strategy, *Use Modular Open Systems Architecture to Stimulate Innovation*, argues that the DoD must control relevant interfaces to ensure competitors with superior products are not occluded from competition due to intellectual property restricted interfaces.

Additionally, in five[1] of the last nine National Defense Authorization Acts, Congress altered statute governing rights to intellectual property in DoD acquisitions. Most recently, in the National Defense Authorization Act of 2016, Congress went as far as directing the establishment of an Government-Industry Advisory Panel on rights in technical data for "the purpose of ensuring that such statutory and regulatory requirements are best structured to serve the interests of the taxpayers and the national defense"; this suggests further changes are imminent. However, with all the undulation in statute and policy, there is no research which analyzes trends in DoD acquisitions as a result of the aforementioned changes to intellectual property policy or to inform future policy decisions. Basic questions, for example, "how much does lacking rights in intellectual property cost the DoD each year in lost competition?" remain unanswered. The closest proxy are the annual measurements of competition levels in DoD acquisitions (e.g., [40], [46]).

The 2015 version of the *Performance of the Defense Acquisition System*, released annually by the office of the Under Secretary of Defense for Acquisition Technology and Logistics (USD AT&L), argues that competition is starting to increase. To substantiate this statement, the report uses a fractional measure of contracts competitively awarded by dollar amount. The most recent measures show that 58.3% of FY14 (often times abbreviated herein as FY plus the two-digit year 00, fiscal year 2000 would be FY00)[2] contracts, by dollar amount, were competitively awarded; which is up from 57% in FY13 [46]. However, this methodology is sensitive to an outlier bias, where a few large contracts awarded competitively (i.e., contracts on the order of magnitude in the $100s of millions) overshadow the many smaller contracts awarded using other than full and open competition. Independent analyses have also taken exception to the claim in the USD AT&L report by concluding, "the rate of effective competition (that is, competed contracts receiving at least two offers) has

[1] FY07 NDAA, Pub. L. No. 109-364 § 802, 120 Stat. 2083 (2007); FY10 NDAA, Pub. L. No. 111-84 § 821, 123 Stat. 2190 (2010); FY11 NDAA, Pub. L. No. 111-383 § 824, 124 Stat. 4137 (2011), FY12 NDAA, Pub. L. No. 112-81 § 815, 125 Stat. 1298 (2012); and FY16 NDAA, Pub. L. No. 114-92 § 813, 129 Stat. 726 (2016)

[2] The federal government operates on a fiscal year that begins on October 1st and ends the following September 30th.

been largely unchanged in recent years" [40] and "overall DoD competition rates . . . have been largely steady in recent years" [47]. Additionally, the USD AT&L analyses makes no effort to breakdown the type of noncompetitive contracts, which makes it difficult to determine the root causes for trends in competition or to investigate trends in specific competition types.

However, independent analysis is possible. Contracts awarded noncompetitively are documented using a contracting artifact referred to as a Justification and Approval[3] (JnA), which is a document released to the public when the DoD uses a procurement strategy other than full and open competition[4]. The Federal Acquisition Regulation (FAR) does not enumerate all possible uses of JnAs, but it does provide guidance on application of the regulation. In doing so, it provides situations in which the authority in FAR 6.302-1 may be appropriate. Of particular interest for this research is FAR 6.302-1(b)(2):

> "The existence of limited rights in data, patent rights, copyrights, or secret processes; the control of basic raw material; or similar circumstances, make the supplies and services available from only one source (however, the mere existence of such rights or circumstances does not in and of itself justify the use of these authorities)"

The research herein relies upon the example at FAR 6.302-1(b)(2) as an exception to full and open competition as the prime indicator of licensing issues with technical data, computer software, or intellectual property. This may include, but is not limited to: limited rights in technical data, or other associated factors such as failure to have purchased a technical data package, or failure to have taken delivery of a validated and verified technical data package. Although this FAR exception is an indicator of technical data licensing or intellectual property issues, it is a noisy indicator because the "similar circumstances" part of this FAR exception may apply

[3] "Justification and Approval is a document required to justify and obtain appropriate level approvals to contracts without providing for full and open competition as required by the Federal Acquisition Regulation"[48]

[4] Other than full and open competition is defined as any sole source or limited competition contract action that does not provide an opportunity for all responsible sources to submit proposals.

Table 3.1: Defense R&D Expenditures by Performer and Funding source for FY84-FY15 (all FY deflated to 2015 dollars prior to aggregation)

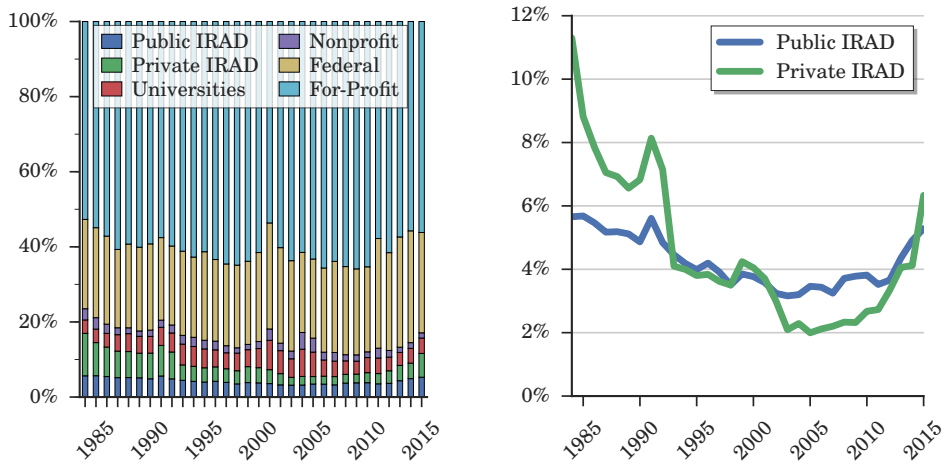
Performer	Funding Source	Sum	% of Total
DoD	DoD	$542.5B	23.7%
Universities	DoD (Contract R&D)	$105.3B	4.6%
Nonprofit Firms	DoD (Contract R&D)	$48.8B	2.1%
For-Profit Firms	DoD (Contract R&D)	$1,389.1B	60.7%
For-Profit Firms	DoD (IRAD)	$97.3B	4.2%
For-Profit Firms	For-Profit Firms (IRAD)	$104.9B	4.6%

Source: All values except for the two IRAD values are from National Science Foundation's WebCASPAR database [50] and are obligations. The two IRAD numbers are from the Defense Contracting Audit Agency [51] and are incurred costs.

even though the Government has acquired unlimited rights to use the technical data or computer software. This is an acknowledged limitation of using FAR 6.302-1(b)(2) JnA as an indicator variable. However, to demonstrate that a FAR 6.302-1(b)(2) JnA is a reasonable indicator for technical data licensing rights or intellectual property issues, this paper draws on Rogerson's IRAD analysis technique [49] to demonstrate a correlation between IRAD and FAR 6.302-1(b)(2) exceptions.

3.3 Independent Research and Development

The rights to technical data or computer software are generally determined by which party funded the development of a work or invention. These criteria are outlined in 10 U.S.C. §§ 2320, as well as 2321, and are promulgated within the FAR and the Defense Supplement to it. As a general example, the contracting officer uses a funding test to determine licensing rights. If a contract is entirely publicly funded, then the government should have unlimited rights to use technical data or computer software. Conversely, if a project is entirely privately funded, then the government should be entitled to limited rights or restricted rights. If the contract is funded using a mix of public and private funding, then the government should be entitled to government purpose rights. This is an example to demonstrate the general process, but does not cover all product types or exceptions to policy.



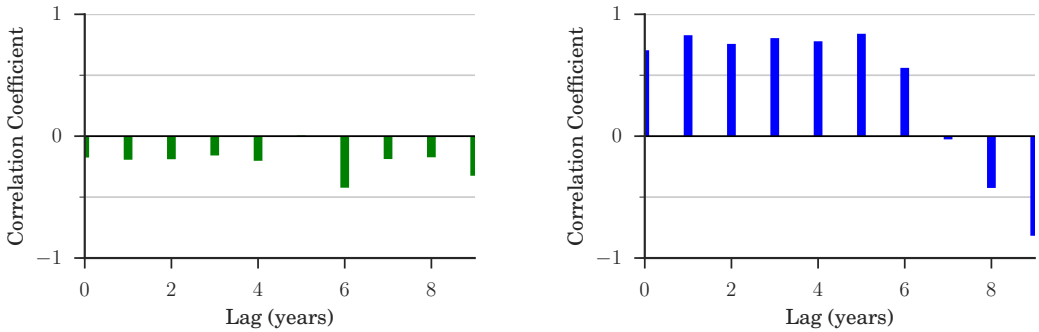
(a) Percent of DoD R&D Spending by performer and/or funding source

(b) IRAD Trends from FY84-FY14 (constant FY15 dollars)

Figure 3-1: DoD R&D Funding. *Public IRAD* referrs to "DoD (IRAD)" in Table 3.1 and *Private IRAD* referrs to "For-Profit Firms (IRAD)" in Table 3.1. (Source: [51])

Assuming program offices and contractors correctly follow the funding test for determination of rights, it is reasonable to expect that the level of private IRAD funding should correlate to the level of FAR 6.302-1(b)(2) JnAs. That is, if FAR 6.302-1(b)(2) JnA are a reasonable indicator of intellectual property issues, an increase in private funding should correlate to an increase in FAR 6.302-1(b)(2) JnA. To explore this potential correlation, Table 3.1 contains the sum of R&D funding within the DoD by funding source and performer from FY84-FY15. Most of the combinations are self-explanatory, with the exception of "DoD (IRAD)." This is the proportion of For-Profit IRAD that is an allowable cost to the DoD; typically bundled in a firm's overhead rate. Whereas the For-Profit Firm IRAD is the proportion of IRAD that was incurred, but was not allowable as an indirect cost to the DoD. To complement Table 3.1, Figure 3-1 shows the changes in IRAD spending from FY84-FY15.

It is likely that IRAD funds invested in a fiscal year would not impact FAR 6.302-1(b)(2) exceptions until, at the very least, the following fiscal year. To explore this potential relationship a lagging correlation analysis is used. These results are outlined in Figure 3-2, which show lagging Pearson product-moment correlations between both sum FAR 6.302-1(b)(2) annual JnA obligations, Figure 3-2a, and mean annual FAR



(a) Sum PDR JnA Obligations

(b) Mean PDR JnA Obligations

Figure 3-2: Lagging Pearson product-moment correlation coefficients (3-2a) sum PDR JnA obligations correlation with Private IRAD at lags $0 \leq n \leq 10$ and (3-2b) mean PDR JnA obligations correlation with Private IRAD at lags $0 \leq n \leq 10$

6.302-1(b)(2) JnA obligations, Figure 3-2b, to Private IRAD[5]. These two graphs illustrate which eight-year period of Private IRAD is the highest correlated with the PDR JnA obligations from FY08-FY15. From this analysis, it is clear that there is a weak negative correlation between the sum FAR 6.302-1(b)(2) JnA obligations and Private IRAD; however, there is relatively strong positive correlation between mean annual FAR 6.302-1(b)(2) JnA obligations and Private IRAD. Although, this analysis does substantiate a correlation between private funding and FAR 6.302-1(b)(2) JnAs, it is critical to point out that this does not suggest causality between the two variables; merely that there is a connection between these two variables. While FAR 6.302-1(b)(2) contains provisions other than intellectual property, it is still a reasonable, but albeit noisy, indicator of intellectual property issues.

3.4 Investigation Method

This paper uses the Cross-Industry Standard Model for Data Mining (CRISP-DM) model to investigate the impact of FAR 6.302-1(b)(2) exceptions in DoD acquisitions [52]. Although designed primarily for data mining and machine learning applications,

[5]The "sum" annual obligations is simply a sum of all contracts authorized under a FAR 6.302-1(b)(2) JnA for each year in the range FY08-FY15. Conversely, the "mean" annual obligations is an average of the annual obligations authorized by a FAR 6.302-1(b)(2) JnA for each year in the range FY08-FY15.

this model offers a structured approach for the analysis of large data sets. The CRISP-DM model is broken down into six phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment. However, in the interest of parsimony only the first four sections are discussed herein.

3.4.1 Phase One: Business Understanding

Regarded as the most important phase of the process, the first phase focuses on building an understanding around what questions or objectives are to be answered by data. This methodological approach puts the question before the data. In other words, first the question(s) is defined then the data which has the highest potential of addressing the question(s) is selected. The questions used, herein, are outlined at the beginning of this Chapter in Section 3.1.

The data available in FPDS represents all prime unclassified federal government contract awards above the micro purchase threshold. Consequently, the data in FPDS comes with the following limitations:

1. FPDS includes only data on prime contract awards. A separate database (i.e., Federal Sub award Reporting System) tracks subcontract awards, but has historically been incomplete [53].
2. FPDS reporting regulations require the disclosure of only unclassified contracts. It is unknown the magnitude of spending this regulation omits from the overall FPDS numbers; some estimate it at as high as 10% [54].
3. FPDS reporting regulations require disclosure of contracts above the micro purchase threshold, which is defined in FAR subpart 2.1 as $3,500 (although at the time of writing, FPDS still uses the past threshold of $3,000). Contracts below this amount are not captured in FPDS.

As with any publicly available database, any analysis is wholly dependent on the quality of the underlying data. Previous Government Accountability Office studies commented on the inaccuracies and problems within FPDS (see GAO-10–833), but those reports are over ten years old and reflect in-part, problems with the migration of government acquisitions data into the FPDS system [54], [55]. However, despite the

limitations noted above, FPDS is one of the few publicly available exhaustive sources of data on government acquisitions and is suitable for analyses aimed at identifying trends or making order-of-magnitude comparisons.

3.4.2 Phase Two: Data Understanding

This phase of the CRISP-DM process begins with the collection of data and then transitions to identifying data quality problems. The goal of this phase is to build an understanding of the limitations in the data as well as to build an understanding of quality issues that may need to be remedied before the application of analytic models. The process concludes with a verification of data quality.

There are two primary avenues to access data from FPDS. The first is to download XML archives directly from the FPDS website by fiscal year. The second method is USASpending.gov, which is designed to aggregate federal government spending data and display it to tax payers. USASpending was ultimately selected and used for all the subsequent analyses herein[6]. The data in USASpending is available from FY00 through the current fiscal year. However, an inspection of fiscal years prior to 2008 revealed many missing data points and subsequently a lack of the necessary fidelity for inclusion in these analyses. In addition, the most recent fiscal year (FY16 at the time of writing) is excluded from this analysis because the DoD data in USASpending is intentionally delayed by 90 days for national security reasons and there is an additional 30 to 60-day delay to update the data in FPDS after the initial 90 days; amounting to a 150-day lag in current fiscal year data. Subsequently, the analyses that follow will evaluate only fiscal years 2008-2015.

Across eight fiscal years there are over 12 million contract actions within the DoD. For the purposes of this analysis, all zero-dollar contract obligations (i.e., a contract action that neither adds nor subtracts funds) were removed from the data

[6]To cross validate the quality of data, the FPDS totals in each fiscal year were compared to the totals reported on the USASpending.gov website for the DoD. The total obligations in each FPDS fiscal year are within a tolerance of 0.02% of the totals listed on the USASpending.gov website. The relatively trivial disparities between the data reported on USASpending.gov and those in FPDS are most likely explained by updates to FPDS between USASpending versions (the data used herein is USASpending version: 15 July 2016).

Service	Sum	Count
DEPT OF THE ARMY	$908,943,882,349	2,128,614
DEPT OF THE NAVY	$741,787,602,050	1,797,791
DEPT OF THE AIR FORCE	$496,212,601,573	836,929
OTHER DOD	$311,293,701,289	562,164
DEFENSE LOGISTICS AGENCY	$284,921,110,343	6,036,863
TOTAL	**$2,743,158,897,605**	**11,362,361**

Table 3.2: Brief Description of USASpending Data

set. Zero-dollar obligations are used oftentimes to make administrative changes or no-cost changes to the contract; leaving them in the data set would skew later analyses that look at ratios of contract awards. This removal has no material impact on the overall dollars obligated, but does adjust the count of contracts across all fiscal years under study. After removing these contract actions, the remaining data set contains over 11 million contract actions totaling approximately $2.7 trillion; see Table 3.2.

3.4.3 Phase Three: Data Preparation

Phase three involves selecting, cleaning, constructing, integrating, and formatting data into the final data set before application of selected modeling tools. All figures and tables displayed before this point are representative of the organic data with no data preparation applied.

Data Selection

From the 225 available variables, only a small subsample is required to address the research questions[7]. In total eleven variables were selected; these are outlined in Table 3.3 to include data descriptions quoted directly from the USASpending data dictionary [56]. One could make an argument for including additional fields of data in the analyses, but computational memory limitations require a parsimonious set of variables for a data set in the tens of millions order of magnitude.

[7]For a full list of variables see the USASpending data dictionary [56]

Field Name	Description
descriptionofcontractrequirement	A brief description of the goods or services bought (for an award) or that are available (for an IDV).
dollarsobligated	The net dollar amount that is obligated or de-obligated by this transaction. If the net is a de-obligation, the amount will be negative.
fiscal_year	The federal government fiscal year (starting October 1), determined by the "Signed Date" provided by FPDS.
fundedbyforeignentity	Indicates that a foreign government, international organization, or foreign military organization bears some of the cost
maj_agency_cat	The combination of two leftmost characters of the contracting agency code representing major federal organizations and departments description
modnumber	An identifier that uniquely identifies a modification for a contract, agreement, or order.
principalnaicscode	The principal NAICS code (used to designate major sectors of the economies of Mexico, Canada, and the United States) that indicates the industry in which the contractor does business.
productorservicecode	The code that best identifies the product or service procured. If more than one code applies, then the code that represents most of the ultimate contract value is reported.
psc_cat	The major category that the Federal Procurement Data System Product or Service Code for the record falls within.
reasonnotcompeted	A code for the reason the contract was not competed (i.e., solicitation procedures other than full and open competition pursuant to FAR 6.3).
signeddate	The date that a mutually binding agreement was reached. The date signed by the Contracting Officer or the Contractor, whichever is later.

Table 3.3: Description of Data Fields

Data Cleaning

In 2015, Defense Logistics Agency (DLA) contract actions more than quadrupled over the previous fiscal year, while the total dollars obligated slightly decreased. This aberration is explained by a change in the way DLA accounted for contracts under the micro purchase threshold beginning in FY15. To control for this change, the data for all DLA obligations in FY15 were cleaned to only those contract obligations that were over or under the micro purchase threshold (i.e. $x < -\$3,000$ or $x > \$3,000$). This cleaning step brought the number of DLA obligations in FY15 from 2,396,568 to 589,474, and are comparable to previous fiscal years. Although this precipitous drop in obligations, a -75.4% change, may seem like a large impact it ultimately reduced the total DLA obligations in 2015 by $967,630,808 or only -3.15%.

Finally, to enable comparisons across multiple fiscal years the *dollarsobligated* field was converted from nominal dollars to constant FY15 dollars using the annual average All-items Consumer Price Index for the respective fiscal years under study. All expressions of dollars obligated or total obligations from this point forward are constant FY15 dollars unless otherwise noted in the analyses.

Data Construction

From the selected eleven variables, an additional two were constructed. *Psc_group* and *psc_simple* data fields were constructed to mirror the CSIS taxonomy of Product Service Codes[8] (PSC), which aggregate hundreds of PSC codes into ten product categories and five service categories; plus a category for research and development (R&D)[9] [57].

1. **Product Categories:**
   (a) Aircraft
   (b) Clothing & Subsistence (C&S)

[8]Product Service Codes (PSC) that represent products are four digit codes (e.g., 7110: OFFICE FURNITURE), whereas PSC that represent services are alpha numeric four digit codes: (e.g., R425: ENGINEERING AND TECHNICAL SERVICES) codes

[9]The full CSIS PSC classification tables (for products, services, and research & development) are available on the CSIS GitHub data repository: `https://github.com/CSISdefense/Lookup-tables/blob/master/ProductOrServiceCodes.csv`

(c) Electronics & Communications (E&C)

(d) Engines & Power Plants (E&PP)

(e) Fuels

(f) Ground Vehicles

(g) Launchers & Munitions (L&M)

(h) Missiles & Space

(i) Ships

(j) "Other Products"

2. **Service Categories:**

(a) Equipment-related Services (ERS)

(b) Facilities-related Services & Construction (FRS&C)

(c) Information and Communications Technology (ICT) services

(d) Medical (MED) services

(e) Professional, Administrative, and Management Support (PAMS)

3. **Research & Development (R&D)**

The *psc_group* was created by mapping the *productorservicecode* field to the second level of the CSIS PSC taxonomy (i.e., Aircraft, Ships, Fuels, PAMS, etc.) and the *psc_simple* field was created by mapping the *productorservicecode* field to the first level of the taxonomy (i.e., Product, Service, or R&D). These additional fields enable categorical analyses that would not be possible using only the data available in *productorservicecode*.

3.4.4 Phase Four: Modeling

As discussed in earlier phases, there are characteristics in FPDS data and subsequently USASpending data which limit the useful analyses to identification of trends and order of magnitude comparisons. Consequently, Phase Four consists primarily of complex Boolean logic queries, visual modeling techniques, and regression models to answer the research questions set out at the beginning of this analysis.

Analysis of Question 1

This inquiry is most easily addressed by analyzing the *reasonnotcompeted* data field. In the most recent version of the FPDS data dictionary, the field "Other Than Full and

Reason Not Competed	count	sum	median	mean
ONLY ONE SOURCE - OTHER	807,717	$600,738,669,601	$11,446	$743,749
UNIQUE SOURCE	200,842	$209,486,641,145	$16,154	$1,043,042
AUTHORIZED BY STATUTE	397,234	$99,108,891,481	$24,894	$249,498
FOLLOW-ON CONTRACT	98,742	$74,653,183,136	$10,628	$756,043
MOBILIZATION, ESSENTIAL R&D	34,166	$62,704,044,715	$126,988	$1,835,276
INTERNATIONAL AGREEMENT	17,478	$61,851,393,409	$128,663	$3,538,814
NATIONAL SECURITY	18,904	$31,207,522,647	$161,249	$1,650,842
URGENCY	56,299	$30,980,330,505	$10,580	$550,282
AUTHORIZED RESALE	39,693	$26,421,588,758	$45,446	$665,649
SAP NON-COMPETITION	270,884	$8,763,887,633	$6,848	$32,353
PATENT/DATA RIGHTS	13,150	$6,338,305,329	$21,039	$482,000
UTILITIES	11,645	$4,830,634,587	$19,800	$414,825
BRAND NAME DESCRIPTION	8,254	$1,824,756,424	$17,580	$221,075
PUBLIC INTEREST	3,910	$1,821,008,661	$15,485	$465,731
STANDARDIZATION	3691	$197,856,965	$7,299	$53,605
UNSOLICITED RESEARCH PROP.	294	$144,568,046	$108,482	$491,728
8AN	79	$49,346,244	$-48	$624,636

Table 3.4: Count, Sum, Median, and Mean of *dollarsobligated* grouped by *reasonnotcompeted*, groups sorted by descending sum of dollars obligated.

Open Competition" is used as the designator for solicitation procedures other than full and open competition pursuant to FAR 6.3 or FAR 13 (USASpending retitled this field as "Reason Not Competed"). This field provides 17 different codes which correspond to FAR 6.301 or FAR 13. FPDS and subsequently USASpending use the code "PDR" or the short description "Patent or Data Rights" to report a contract action justified pursuant to FAR 6.302-1(b)(2). Unfortunately, no information is provided on the specific justification, only the reference to FAR 6.302-1(b)(2). This makes segregating out the different exceptions in FAR 6.302-1(b)(2), for other than full and open competition, impossible. Although *reasonnotcompeted* denotes the FAR exception used in the JnA, it is important to note that the unit of analysis here is the contract action authorized by the JnA, not the JnA itself. To analyze the data, each contract action was organized into a group based on its *reasonnotcompeted* category, with null values omitted. Within the resulting groups the *dollarsobligated* field is used to quantify the magnitude of the effect. The results are captured in Table 3.4, in which the Patent/Data Rights group is highlighted. This category denotes FAR 6.302-1(b)(2) exceptions.

This analysis suggests that FAR 6.302-1(b)(2) exceptions occluded over $6 billion in DoD contracts[10], from full and open competition, between FY08-FY15. However, this number may be misleading because of the way FPDS collects data. For each contract an administrator is only permitted to select one field for *reasonnotcompeted* from the list of values outlined in Table 3.4. Although, in practice a JnA may contain more than one exclusion category; forcing the administrator to make a judgment call on which category is most accurate. This nuance may explain the large number of "ONLY ONE SOURCE – OTHER" contract actions. Typically, a Justification and Approval for a contract would be "ONLY ONE SOURCE-OTHER" because of "UTILITIES" or "PATENTS/DATA RIGHTS." This is because "PATENTS/DATA RIGHTS", outlined in FAR 6.302-1(b)(2), is a subsection of "ONLY ONE SOURCE-OTHER"; outlined in FAR 6.302-1. Therefore, when an administrator is faced with selecting a single category that most accurately represents why an action was not subject to full and open competition, they may select "ONLY ONE SOURCE-OTHER." As a result, the aggregation of FAR 6.302-1(b)(2) exceptions in Table 3.4 are most likely only a lower bound of the true impact of on DoD acquisitions.

Analysis of Question 2(a)

The second question sought to identify trends, if any, in the use of FAR 6.302-1(b)(2) exceptions in DoD acquisitions. To address this question, the data set was converted into a time series using the *signeddate* as a date-time index. With the data set in this format, parallel lines of inquiry were used to identify two trends: ratios of dollars obligated and ratios of contracts awarded. For example, to determine ratio of dollars obligated the data set was down sampled into the 96 months that span fiscal year 2008 to 2015; summing the dollars obligated for each month from both the subset (α) and full set (β), see Equation 3.1.

[10]Of the $6 billion in contract actions, $126,630,720 are foreign military sales

$$\alpha_{month} : \beta_{month} = \frac{\sum_1^n \alpha_n}{\sum_1^n \beta_n}$$
$$\text{where: } \alpha \subset \beta, \quad \beta \neq 0, \quad n = \text{days in a given month} \tag{3.1}$$

The result is two time series of data. The first is of dollars obligated by month. The second is a count of contract awards, using only initial contract awards and not subsequent modifications, by month. Leaving contract modification data in the dataset would overstate the number of sole source contract awards supported by a FAR 6.302-1(b)(2) exception, since a JnA is typically not issued for each modification. Using this time series transformation process, six different ratios were calculated:

1. The ratio of all contracts using a JnA to total number of contracts. Measured in terms of dollars obligated by contract and count of contracts. (referred to henceforth as *JnA to Total Awards*)
2. The ratio of only contracts using FAR 6.302-1(b)(2) JnA to total number of awards. Measured in terms of dollars obligated by contract and count of contracts. (referred to using the FPDS designator of "PDR" or Patent or Data Rights, which is henceforth *PDR to Total Awards*)
3. The ratio of contracts using a FAR 6.302-1(b)(2) JnA to contracts using any JnA. Measured in terms of dollars obligated by contract and count of contracts. (referred to using the FPDS designator of "PDR" or Patent or Data Rights, which is henceforth as *PDR to JnA*)

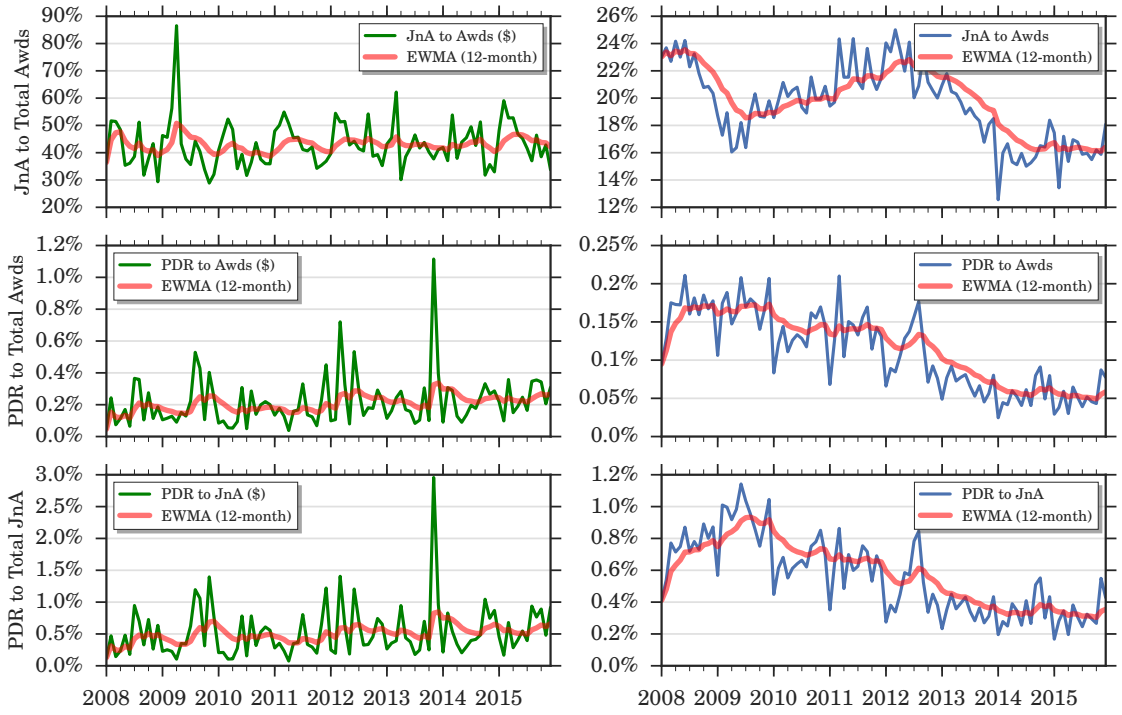
These two lines of inquiry illustrate whether the frequency of contracts awarded using a FAR 6.302-1(b)(2) exception are increasing or decreasing over the fiscal years under study. The results of this analysis are found in Figure 3-3a and Figure 3-3b; organizing each of the ratios into a row with a column for ratio of obligation amounts (left column, denoted with a "$" in the legend) and ratio of contract actions (right column). The red line on each graph in Figure 3-3a shows a twelve month

exponentially weighted moving average (EWMA) for each time series ratio, which removes some of the irregular variations.

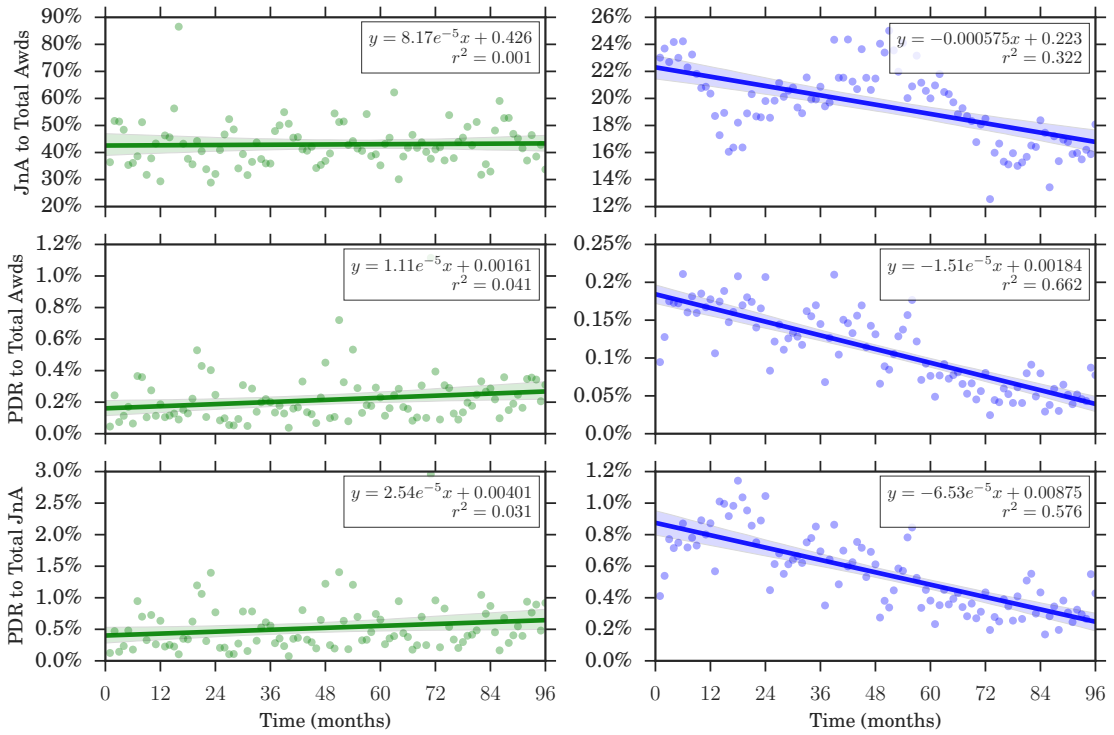
Additionally, to assist in quantifying the trend of data displayed in Figure 3-3a, Figure 3-3b illustrates an ordinary least squares (OLS) linear regression of the data. To compute this trend line the dates on the x-axis were converted into numerical months ($\{Jan08, Feb08 \ldots Dec15\} = \{1, 2, \ldots 96\}$). Consequently, the x-axis in Figure 3-3b now corresponds to the number of months elapsed since the start of FY08. All other formatting and labeling factors remain constant between Figure 3-3a and Figure 3-3b.

Beginning with an analysis of Figure 3-3a, nearly all the ratios calculated from obligations indicate almost no trend in the data, with a possible exception of *PDR to JnA*. This suggests that, although the data are noisy, there is no meaningful change in the number of dollars obligated in any of the ratios over the past eight fiscal years. Conversely, the ratios calculated from contract awards show a relative decline through the start of FY15, with a slight uptick in FY15. In summary, although the ratio of contract actions using JnAs and FAR 6.302-1(b)(2) JnAs declined over the past eight fiscal years, there has not been a meaningful increase or decrease in the ratio of dollars obligated using FAR 6.302-1(b)(2) JnA.

Transitioning to Figure 3-3b, all the OLS regressions for the ratios of dollars obligated are slightly positive. However, *PDR to JnA* and *JnA to Total Awards* have a confidence interval that spans zero (95% confidence interval illustrated in Figure 3-3b by the shaded area on either side of the regression line); implying it is possible, although not necessarily probable, that the trend could be positive or negative. Conversely, the regression results computed using the ratio of contract actions all have negative coefficients with confidence intervals that do not span zero. As a result, it is reasonable to conclude that the frequency of both JnA relative to total awards (*JnA to Total Awards*), and PDR relative to both total JnAs (*PDR to JnA*) and total awards (*PDR to Total Awards*), declined from FY08 to FY15. This analysis suggests that some measures of trends in FAR 6.302-1(b)(2) used in DoD acquisitions indicate a downward trend, or improvement; however, the results are

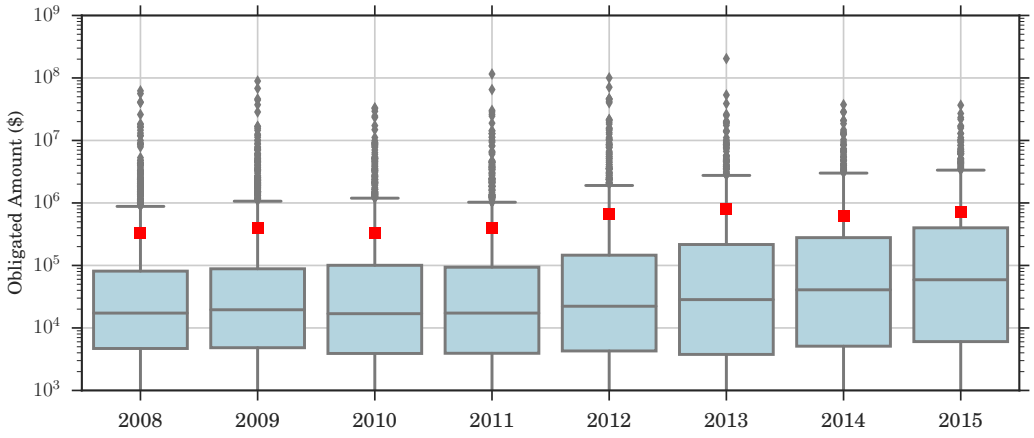


(a) Ratios of Obligations (left) and Ratio of Contract Awards (right)

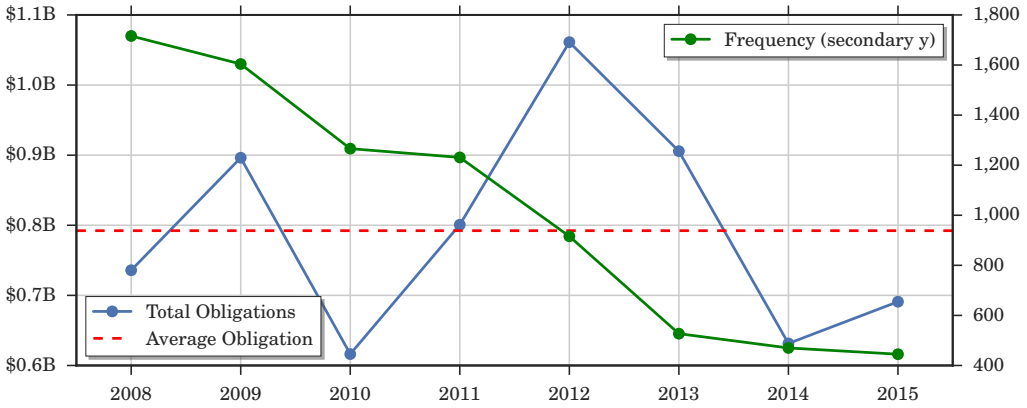


(b) Linear Trend Lines for each ratio in Figure 3-3a

Figure 3-3: Longitudinal Trend Analyses



(a) Boxplot of PDR obligations by Fiscal Year (red square denotes mean)



(b) Line plot of PDR obligations by Fiscal Year

Figure 3-4: Inter-Year PDR Obligation Trends

confounded by other measures which show no meaningful trend. (A full output of the OLS regression models used herein are located in Appendix A.)

To investigate further, Figure 3-4a and Figure 3-4b look at the inter-year patterns in FAR 6.302-1(b)(2) JnA use. Specifically, Figure 3-4a shows the change in distribution of obligations by contracts using FAR 6.302-1(b)(2) JnA between FY08-FY15, with few exceptions, both the median and mean annual obligation amount increased year-over-year. Note the log scale in Figure 3-4a and the number of outliers in the boxplot with high order of magnitude obligations. Compare these results with those in Figure 3-4b, where the number of contracts is decreasing, but the annual obligation amounts trend erratically. These findings are important because they show that,

although trends in FAR 6.302-1(b)(2) JnA use may be decreasing, FAR 6.302-1(b)(2) JnA contract obligations measured by either median or mean value, are increasing.

Analysis of Question 2(b)

In addition to time series trends, it is important to look at "what" FAR 6.302-1(b)(2) JnA are buying to identify purchasing trends. To address this question, analyses focus on four of the data fields; *psc_simple*, *psc_groups*, *principalnaicscode*, and *productorservicecode*. Using these fields, it is possible to categorize each contract action as either a product or a service and get a relative sense of the domain. This is done by employing a taxonomy which aggregates individual PSCs into broader baskets of goods. Table 3.5 outlines the sum obligations for the CSIS taxonomy at level 1 and level 2, as well as top ten PSC and NAICS codes for FAR 6.302-1(b)(2) JnAs.

Analysis of the PSC codes reveals that FAR 6.302-1(b)(2) JnA are just as commonly used for services as they are for products. This is evidenced not only by the PSC taxonomy level 1 results, but also by all the PSC codes in Table 3.5 that begin with a letter. This same phenomenon is evident as well in the top ten NAICS codes, where the 54 series indicate a service. The two largest categories from level 2 of the PSC taxonomy are *Professional, Administrative, and Management Support* (PAMS), which is a service category, and *Aircraft*, which is a product category. Some of the service codes, (e.g., J016), are logical given it denotes the maintenance and repair of aircraft components. In this case, one could speculate the repair and manufacturing process of an aircraft component is likely protected by a manufacturer's intellectual property. However, it is less clear what intellectual property is the limiting factor in the codes which makeup PAMS;(e.g., R425), which is a generic code for engineering support services.

The data in USASpending does not offer any insight into why a FAR 6.302-1(b)(2) exception was used for a given contract. However, it does offer insight into product, service and R&D trends: see Figure 3-5a and Figure 3-5b. In all, except FY08 and FY09, there were more service than product contracts authorized under a FAR 6.302-1(b)(2) JnA. Since FY09, the number of product FAR 6.302-1(b)(2) authorized

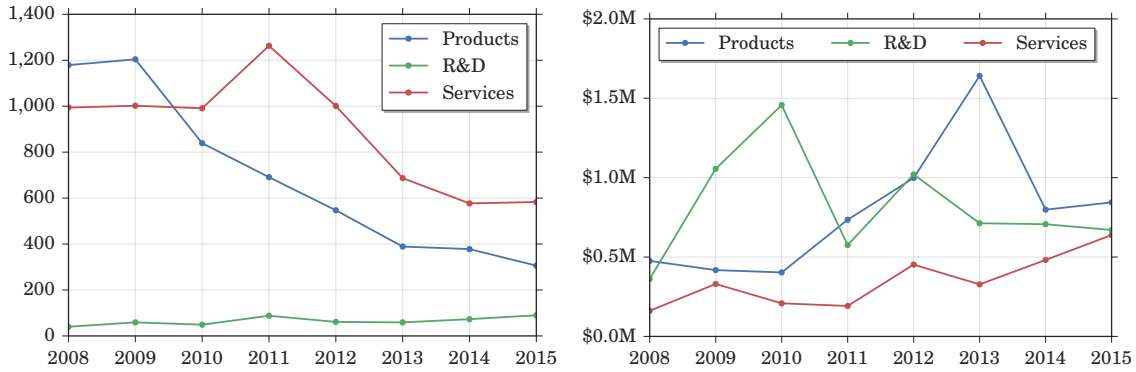
PSC Taxonomy Level 1	count	sum	mean
Products	5,533	$3,654,163,121	$660,431
Services	7,098	$2,269,110,084	$319,683
R&D	519	$415,032,124	$799,677

PSC Taxonomy Level 2	count	sum	mean
PAMS	2,159	$1,593,098,494	$737,887
Aircraft	1,449	$1,389,506,928	$958,942
Missiles & Space	135	$693,343,385	$5,135,877
Electronics & Communications	1,592	$632,919,208	$397,562
ERS	4,359	$590,987,877	$135,579
R&D	519	$415,032,124	$799,677
Engines & Power Plants	452	$272,723,325	$603,370
Launchers & Munitions	194	$269,714,770	$1,390,282
Other	1,333	$167,492,024	$125,650
Ground Vehicles	104	$125,907,354	$1,210,648
Ships	126	$87,284,428	$692,734
ICT	488	$81,447,823	$166,901
Clothing & Subsistence	148	$15,271,699	$103,187
FRS&C	68	$2,450,382	$36,035
MED	24	$1,125,509	$46,896

PSC Code	count	sum	mean
R414: SYSTEMS ENGINEERING SERVICES	672	$934,139,957	$1,390,089
1410: GUIDED MISSILES	62	$651,325,023	$10,505,242
1510: AIRCRAFT, FIXED WING	220	$509,709,584	$2,316,862
R425: SUPPORT- PROFESSIONAL: ENGINEER...	557	$339,143,920	$608,876
J016: MAINT/REPAIR/REBUILD OF EQUIPME...	476	$215,031,480	$451,747
1560: AIRFRAME STRUCTURAL COMPONENTS	96	$199,453,727	$2,077,643
1680: MISCELLANEOUS AIRCRAFT ACCESSOR...	86	$161,457,768	$1,877,416
1005: GUNS, THROUGH 30 MM	47	$145,990,079	$3,106,172
J016: MAINT-REP OF AIRCRAFT COMPONENTS	356	$117,614,327	$330,377
2510: VEH CAB BODY FRAME STRUCTURAL...	9	$117,345,238	$13,038,360

NAICS Code	count	sum	mean
336413: OTHER AIRCRAFT PARTS AND AUXI...	3,399	$1,311,566,685	$385,868
336411: AIRCRAFT MANUFACTURING	429	$1,084,203,683	$2,527,281
541330: ENGINEERING SERVICES	1,044	$1,082,087,325	$1,036,482
336414: GUIDED MISSILE AND SPACE VEHI...	62	$684,330,795	$11,037,593
336412: AIRCRAFT ENGINE AND ENGINE PA...	496	$226,071,643	$455,790
334511: SEARCH, DETECTION, NAVIGATION...	222	$208,272,040	$938,162
541712: RESEARCH AND DEVELOPMENT IN T...	256	$203,694,628	$795,682
332994: SMALL ARMS, ORDNANCE, AND ORD...	57	$140,785,462	$2,469,920
541512: COMPUTER SYSTEMS DESIGN SERVICES	221	$139,646,028	$631,882
541710: RESEARCH AND DEVELOPMENT IN T...	204	$133,528,895	$654,553

Table 3.5: PSC Taxonomy Level 1 and PSC Taxonomy Level 2 as well as Top 10 PSC and NAICS codes sorted by sum obligated dollar amount for PDR JnA (descriptions truncated to first 40 characters)



(a) Count of FAR 6.302-1(b)(2)

(b) Mean obligation of FAR 6.302-1(b)(2)

Figure 3-5: Product, Service, and R&D PDR JnA by Fiscal Year and PSC Taxonomy Level 1

contracts fell from a high of 1,204 per year to a low of 306 in FY15; a 75% drop. Although the drop seems precipitous, Figure 3-5b shows that the average product FAR 6.302-1(b)(2) JnA contract obligation amount increased since FY08. Consequently, it is reasonable to conclude the DoD is using less product FAR 6.302-1(b)(2) JnAs in contracts, but the average cost of each is rising. A similar conclusion can be reached for service FAR 6.302-1(b)(2) JnAs, with the exception of the percentage decrease in frequency and increase in mean obligated dollars are of a lower magnitude. The exception is FAR 6.302-1(b)(2) JnA used for R&D, in which both the frequency and mean obligation amount increased over the past eight fiscal years.

3.5 Findings/Discussion

Beginning with Question 1, magnitude of impact, the analyses outlined above established a lower bound for the impact of FAR 6.302-1(b)(2) exceptions at approximately $6 billion for the fiscal years under study. This number does set an order of magnitude for the impact, but as discussed earlier it falls short due to the way FPDS ingests data and because of the "similar circumstances" provision included in FAR 6.302-1(b)(2) exceptions. The only way to reach a potentially more accurate estimate of the magnitude would be to take a random sample of JnAs, approximately 100-200, from all exceptions to full and open categories, and retrieve the soft copy JnAs from

the Governmentwide Point of Entry[11]. After retrieval of all soft copy JnAs in the random sample, each would need to be hand coded to identify reason(s) not competed. Once all of the random samples were coded, the total number of JnAs could be resampled according to the distribution of coded JnAs. Although this method may generate a more accurate bound, there would be no way to validate the new bound and the time required to hand code each JnA in the random sample is prohibitive. Future research is needed to identify a more accurate way to categorize exceptions to full and open competition at the contract level in FPDS.

This analysis was equally concerned with what was being purchased using FAR 6.302-1(b)(2) JnAs. One of the key findings was the higher proportion of service contracts to product contracts in six of the eight fiscal years examined. This finding is significant because FAR 6.302-1(b)(2) JnAs are conventionally considered a byproduct of intellectual property protections on physical design process artifacts (i.e., drawings, blueprints, technical data, etc.). Given the results, which counter this conventional understanding, it suggests more policy focus should be given to the numerous service contracts that are excluded from full and open competition due to FAR 6.302-1(b)(2) exceptions. Additional research into the justification for these artifacts of FAR 6.302-1(b)(2) is needed to inform future policy decisions.

Lastly, analyses identified an increasing trend, both in terms of frequency and mean obligations, in the use of FAR 6.302-1(b)(2) exceptions for R&D. This is also an unexpected finding since the very nature of R&D suggests investigation of a novel product. Given the nature of R&D, it is difficult to understand how a contract could be authorized by FAR 6.302-1(b)(2) exceptions. These R&D contracts could be evidence of the "similar circumstances" section in FAR 6.302-1(b)(2); evidence of an improper use of FAR 6.302-1(b)(2), evidence of a novel use of FAR 6.302-1(b)(2), or evidence of an error in the PSC Taxonomy. Regardless, more investigation and research is needed into this finding.

[11]Governmentwide point of entry (GPE) means the single point where Government business opportunities greater than $25,000, including synopses of proposed contract actions, solicitations, and associated information, can be accessed electronically by the public. The GPE is located at `http://www.fedbizopps.gov`. (48 CFR 2.101 Definitions)

3.6 Limitations/Recommendations

The limitations previously outlined above are not only confined to magnitude of impact estimates, but also to trend analyses. It is impossible to say that the above trends are, with any certainty, representative of the entire effect. This analyses must acknowledge the possibility that the trends previously discussed may not be a truly accurate representation of the actual trends in DoD acquisitions. *This difficulty in examining, even the simplest questions in DoD acquisitions, is the most salient finding of this quantitative investigation.* Trends are necessary to inform multi-billion dollar DoD policy decisions, but gathering even the simplest trends results in a task mired by a complex web of policy and data interactions; confounded by limitations in publicly available databases and unnecessarily broad FAR language. Making a few changes in the way DoD contract data is collected should bring the community a step closer to much richer analyses and more generalizable results:

1. **Allow for more than one FAR exception to full and open competition in FPDS per contract**. This change to the data structure in FPDS should help more accurately define which exceptions are used in each JnA and permit more accurate analyses of trends in non-competitive contracting types especially in situations where multiple non-competitive reasons are justified.
2. **Define or remove "similar circumstances" from FAR 6.302-1(b)(2)**. This phrasing in the FAR only adds ambiguity to a section that predominantly concerns intellectual property and technical data. Furthermore, it is unnecessarily redundant given the qualifying language already in FAR 6.301-1(b), "Use of this authority may be appropriate in situations such as the following (these examples are *not intended to be all inclusive* and do not constitute authority in and of themselves)" (emphasis added). If FAR 6.301-1(b) is not meant to be all inclusive, what is the necessity for a catchall statement in FAR 6.301-1(b)(2)? Removal of this language still maintains the purpose of FAR 6.302-1(b)(2) given the language in FAR 6.302-1(b).

3.7 Conclusion

This Chapter set out to capture metrics which would yield a better understanding of limitations in DoD acquisitions due to intellectual property lock-in. In doing so, the analyses established a lower bound on the magnitude of impact at $6 billion and noted trends of decreasing FAR 6.301-1(b)(2) use, but no material trend in the amount of money obligated in contracts using a FAR 6.301-1(b)(2) exception. Furthermore, the prevalent use of FAR 6.301-1(b)(2) exceptions for service contracts was a key finding. Service contracts using FAR 6.301-1(b)(2) exceptions, measured in terms of frequency, were more common than product contracts; however, product contracts using FAR 6.301-1(b)(2) exceptions, in terms of dollars obligated, outpaced service contracts. This finding is salient because it countermands the conventional understanding of what types of contracts authorized using a FAR 6.301-1(b)(2) exception to full and open competition.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 4

Conceptual Model

Up to this point intellectual property lock-in has been discussed in detail, but not explicitly defined. This choice in delaying discussion of the definition is due to the wickedness of the problem and a lack of prior research, as discussed in Chapter 2. There is a vast set of literature which seeks to explain DoD procurement using incentive based models [37], [49], [58]–[61] and an additional set which seeks to explain it based on characteristics of either the market structure [62] or the industrial base [53], [63], [64]. One could attempt to extend any of these models to intellectual property lock-in, but such an effort would be forced and likely not capture the eccentricities and nuances of intellectual property lock-in within the DoD. Consequently, this Chapter endeavors to first conceptually define intellectual property lock-in and then discuss the conceptual models used to identify intellectual property.

4.1 Bottleneck Model

One of the fundamental concepts in any large technical system is the creation of value and the capture of value. Many in academic literature argue the key to value capture is controlling bottlenecks [65]–[70]. In this context, a bottleneck begins as a nontrivial technical problem. Once an organization identifies a bottleneck and creates an effective solution (i.e., value creation), it may then combine that solution with property rights to extract a stream of rents (i.e., value capture). Although this

example represents a bottleneck as a technical problem, bottlenecks take many forms. To understand the spectrum of bottlenecks in the context of software it is necessary to understand both technical and contract architectures.

4.1.1 Technical Architecture and Contract Structure

A technical system is made up of many parts or components, each designed to accomplish a specific task, where the sum of these parts creates more value than the disassembled individual [71]. Architecture is the term given to describe the specific abstraction of components or parts into a whole; or said differently, the assignment of functional elements to constituent components [72]. The formal concept of system architecture originates primarily in the seminal work of Herbert Simon [73]; wherein, he proposed the theories of hierarchy and "near decomposability." At the most basic form, architecture formalizes abstract patterns of interfaces between components[1]. However, Ulrich offers a more precise three-part definition of architecture: "1) the arrangement of functional elements, 2) the mapping from functional elements to physical components, and 3) the specification of interfaces among physical components" [74, p. 2]. For any given product, there could be many different arrangements of functional elements and subsequently even more mappings from functional elements to physical components which satisfy the functional requirements [75]. That is, a product may be designed for ease of manufacturing or efficient development as opposed to maximized performance. Each of these many potential architectures likely differ along the spectrum of performance and cost dimensions. Understanding the range of choices across many dimensions, and making trade-offs, is the domain of systems architecture.

Baldwin and Clark argue a complete architectural description includes a *design structure*, *task structure*, and *contract structure* [76], [77]. Their argument is based on

[1] Ulrich and Eppinger define architecture as "the scheme by which functional elements of a product are arranged into physical chunks and by which the chunks interact" [72, p. 165], where physical chunks are defined as a collection of parts, components, subassemblies. Elsewhere, Ulrich also defines architecture as, "the scheme by which the function of the product is allocated to physical components" [74, p. 420]

the premise that the *design structure* and the *task structure* correspond respectively to what Simon called the "state description" and the "process description" of a complex system [73]. The design structure of an artifact comprises a set of design parameters and the dependencies among them. A design structure is often captured in physical design artifacts, such as engineering schematics, blueprints, or diagrams. Whereas a task structure is a list of tasks that must be done to manufacture or create a specific design. Engineers or product designers are primarily concerned with the physical structure of a system and processes necessary to create that system. The combination of a design structure and a task structure is the *technical architecture* of a system. However, for the technical architecture to materialize into a product, a set of business practices and agreements must be overlaid. This business infrastructure is responsible for the sourcing of materials, negotiating of contracts, recruiting workers, paying taxes, selling products, and collecting payment.

Baldwin and Clark define an overlaid business infrastructure as the *contract structure*, which is "the set of contracts used to form a collective enterprise, and to organize its dealing with the greater economic environment" [77, p. 107]. Their definition could include, agreements to manufacture, departmental organization, employee compensation, intellectual property rights, and customer transactions. If a technical architecture is designed with the end goal of *both* value creation and value capture, a contract structure is integral. It is important to note that the contract structure is usually the domain of business managers and lawyers. Whereas, the technical architecture is the domain of engineers and product designers.

4.1.2 Bottleneck Defined

The question still remains, "What is a bottleneck?" To address this Baldwin erects an illustrative example of a bridge over a river [76]. This bridge serves to allow traffic to pass over an obstruction, which could be either another flow of traffic, or perhaps water. Continuing with this analogy, if all traffic routes must pass over this bridge, and this bridge is a source of congestion, then it is a bottleneck. Applied more generally to a system, a bottleneck is any part, component, or subassembly whose

performance limits the performance of the entire system. Baldwin more formally defines a bottleneck using a two-part definition, "(1) it is necessary to the functioning of the whole; and (2) there is no good way around it" [76, p. 7]. Consequently, it is rational to conclude that a system bottleneck, relative to the whole, must be both inseparable and non-substitutable. Within this definition of bottlenecks there are two primary categories: *technical bottlenecks* and *strategic bottlenecks*.

4.1.3 Technical Bottlenecks

A technical bottleneck is a bottleneck in which the properties of the system hinder performance. For example, in a traffic system, if there is no bridge over a river, goods would need to be off loaded from trucks onto ferries and reloaded onto rail cars or different trucks on the other side of the river. In this example the river is a technical bottleneck as it hinders the performance of the entire system and there is not an easily substitutable solution.

In a system, technical bottlenecks can be mapped onto one or more components or parts of a system's technical architecture. A technical bottleneck that maps to just one component is referred to as an *intensive* bottleneck and a technical bottleneck that maps to more than one component is referred to as an *extensive* bottleneck. Although this mapping conforms to terminology used in some definitions of architecture and modularity, it is important to point out that bottlenecks are theoretically distinct from modules of a technical system. Modules are components which are purposefully designed into a system; whereas, technical bottlenecks typically exist regardless of designer intent.

4.1.4 Strategic Bottlenecks

A strategic bottleneck is a point of value capture in a technical system and consequently a potential source of rents. Baldwin defines a two-part definition for strategic bottlenecks "(1) a unique solution to an underlying technical bottleneck; plus (2) control over access to the solution" [76, p. 11]. Using the analogy of the bridge over

a river (technical bottleneck), for an organization to create a strategic bottleneck it must first build a bridge, then exclude others from using it unless they pay a toll. It is important to point out that the bridge in this example, must be also be non-substitutable (i.e., it cannot be build next to another bridge).

Baldwin points out this ability to control access to a resource is the basic definition of a property right [76]. Although there are multiple types of property rights, the research herein is primarily concerned with *de jure* property rights, or those based on the law (e.g., copyright, patent, etc.). Exclusive property rights to a technical bottleneck is the basic form of a strategic bottleneck. Furthermore, since property rights create boundaries, they are considered to be part of contract structure. For this research, strategic bottlenecks which emanate from *de jure* property rights are referred to as *de jure strategic bottlenecks*.

4.2 Intellectual Property Lock-in

In terms of architecture, intellectual property lock-in occurs at the intersection of the design structure, task structure, and contract structure, where a *de jure* property right creates a strategic bottleneck, see Figure 4-1. It is important to note that without an accompanying contract structure that secures a *de jure* property, intellectual property lock-in is not possible. However, the mere presence of a contract structure does not guarantee a strategic bottleneck nor intellectual property lock-in. Recall from Baldwin's definition that it is both a unique solution to a technical bottleneck (design structure or task structure) and the exclusive right over the solution (contract structure) that are required to form a strategic bottleneck.

Similarly, it is important to note that the presence of a *de jure* strategic bottleneck does not guarantee intellectual property lock-in. For example, in software, an intensive technical bottleneck may exist in a component that limits the performance of the overall system. A firm may have a property right to this component, but unless this component is unique and sits at a non-substitutable juncture in the technical architecture it is unlikely to create intellectual property lock-in. However, if the intensive

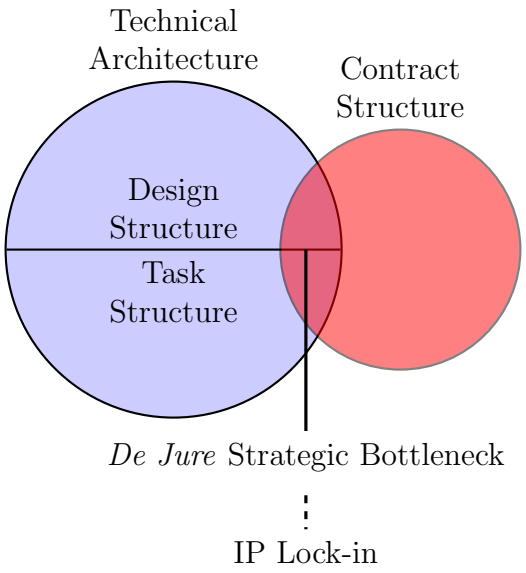


Figure 4-1: Theoretical Formation of Intellectual Property Lock-in

technical bottleneck exists at a non-substitutable juncture then intellectual property lock-in is possible. Additionally, consider the existence of an extensive technical bottleneck with a *de jure* property right, such a configuration would create a series of strategic bottlenecks. Continuing with the software example, an extensive technical bottleneck exists in multiple components, where one or more of these instances is hindering the overall performance of the system. Each instance of this technical bottleneck, accompanied with a property right, is a distinct strategic bottleneck. Again, a single *de jure* strategic bottleneck does not guarantee lock-in, but as the number of *de jure* strategic bottlenecks increase so too does the likelihood of lock-in.

The relationship between instances of *de jure* strategic bottlenecks and likelihood of intellectual property lock-in is theorized in Figure 4-2. The relationship is depicted as a s-curve closely approximating a logistic function. For example, at point S_1 the likelihood of lock-in is described as value P_1. The value P_L describes the threshold at which the expectation of likelihoods above this point result in lock-in. In other words, this is considered the probabilistic tipping point between a low-likelihood of lock-in (values where $P_n < P_L$) and a high-likelihood of lock-in (values where $P_n > P_L$). This non-linear relationship between *de jure* strategic bottlenecks and intellectual property lock-in likelihood outlines the important difference between a strategic bottleneck

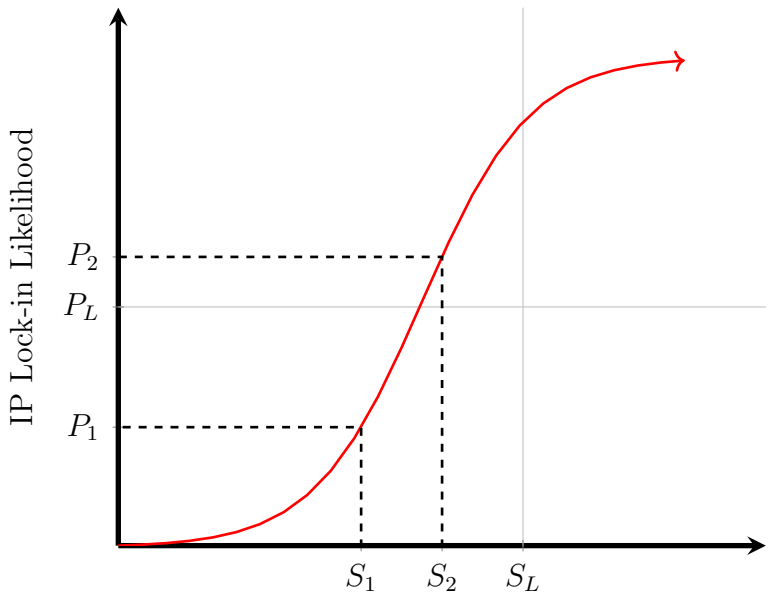


Figure 4-2: Theoretical Relationship between *De Jure* Strategic Bottlenecks and Likelihood of IP lock-in

based on an *intensive* versus *extensive* technical bottlenecks. In cases where a single strategic bottleneck exists the likelihood of intellectual property lock-in is said to be relatively low; however, small increases in the incidences of strategic bottlenecks, ΔS, yield super-linear increases in the likelihood of intellectual property lock-in, ΔP; up until a point where additional strategic bottlenecks produce a diminishing incremental increase in probability of lock-in, point S_L.

Although theoretical, the relationship visualized in Figure 4-2 is significant because it illustrates how small changes in strategic bottlenecks can impact the likelihood of IP lock-in. In practice, this relationship implies that relatively small reductions in the number of *de jure* strategic bottlenecks could drive the likelihood of IP lock-in below P_L.

4.3 Bottleneck Model Applied to DoD

The conceptual bottleneck model is the theoretical basis to determine if a product or program is locked-in. The bottleneck model, coupled with the statute required

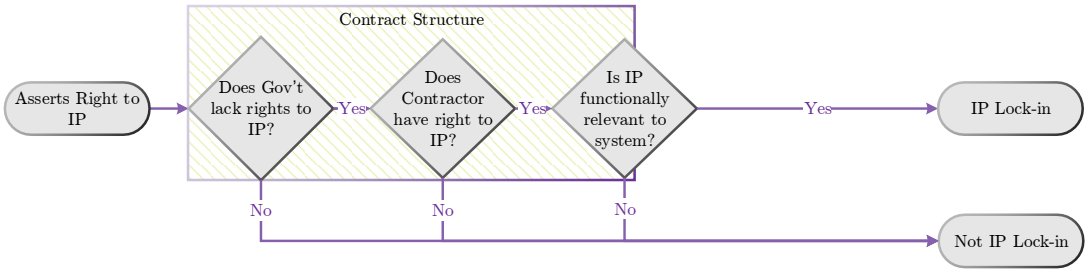


Figure 4-3: Conceptual Model for Intellectual Property Lock-in

contract structure, forms a conceptual model for identifying lock-in in DoD systems. The model in Figure 4-3 illustrates the basic steps initially involved in the contract structure. In essence, this initial portion of the model remains almost exclusively in the legal domain.

4.3.1 Assertion of Rights

The prerequisite for intellectual property lock-in is an assertion of rights. A contractor must assert that it possesses an exclusionary property right to some piece of intellectual property for lock-in to exist. Note this is not a contractor asserting ownership of a piece of intellectual property, but instead the assertion of a right to use, or exclude others from using, that intellectual property. This is an important distinction, because a contractor could have exclusionary rights to a piece of intellectual property which it does not own (e.g., exclusive license to a piece of third-party intellectual property).

4.3.2 Determine Government Rights

In a non-DoD case, the next step would be to verify that the contractor owns the intellectual property or has rights to the intellectual property. However, if the intellectual property was created during the course of a DoD contract, then ownership of the intellectual property belongs to the contractor, but determination of Government entitlements to specific rights in that intellectual property follows the guidelines set forth in the FAR and Defense Supplement to it. The regulations that govern intel-

lectual property rights in the DoD differ from the commercial sector because they do not provide any explicit rights to the creator of the intellectual property. Instead, they outline the rights which are entitled to the government and thereby outline the rights also entitled to the contractor by omission. For example, if the Government is entitled to an unlimited right to a piece of intellectual property, the contractor who created the intellectual property still maintains ownership, but relinquishes an unlimited right to the Government. The possession of an unlimited right implies that the Government could give the intellectual property to the contractor's competitors. This is why the first step in the model is determining if the Government has rights to the intellectual property asserted by a contractor.

After asserting rights, the burden becomes verifying what class of rights the Government has, if any, for the specific intellectual property. This sub-process is not explicitly modeled in Figure 4-3, but is nonetheless integral to the overall model. The determination of rights is based on the "funding test", which afford rights to intellectual property created based on the source of funding. Depending on the type of product and combination of public and private funding, there are many different types of rights that could be entitled to the Government (see Tables 2.1 and 2.2 for a detailed explanation). However, as a general rule, as long as the Government used some public funding, the DFARS does not set forth a threshold, then the Government is assured of a right to that specific intellectual property. However, if the Government did not use any public funding, then it is likely entitled a limited right. In situations where the intellectual property was developed using a mix of public and private funding, the Government is entitled to a Government Purpose Right. However, in addition to the rights allocated using the funding test, 10 U.S.C. § 2320(a)(2)(C) outlines special cases in which the government is always entitled an unlimited right. The funding test does not apply to technical data that:

1. Constitutes a correction or change to data furnished by the United States
2. Relates to form, fit, or function

3. Is necessary for operation, maintenance, installation, or training (commonly referred using the acronym OMIT); excluding detailed manufacturing or process data
4. Is otherwise publicly available or has been released or disclosed by the contractor or subcontractor without restriction on further release or disclosure.

Additionally, the United States may release technical data to parties outside the Government, notwithstanding the funding test, under the following conditions, quoted directly from 10 U.S.C. § 2320(a)(2)(D):

5. Is necessary for emergency repair and overhaul
6. Is a release, disclosure or use of technical data pertaining to an interface between an item or process and other items or process necessary for the segregation of an item or process from, or the reintegration of that item or process with, other items or processes
7. Is a release or disclosure of technical data to, or use of, such data by, a foreign government that is in the interest of the United States and is required for evaluation or informational purposes

If the Government has the appropriate rights to the piece of intellectual property asserted by a contractor or the piece of intellectual property asserted by a contractor falls into one of these seven special cases outlined in 10 U.S.C. § 2320, then intellectual property lock-in is likely unachievable.

4.3.3 Determine Contractor Rights

Assuming the Government lacks rights to the piece of intellectual property in question, the next step becomes verifying that the contractor has rights to the piece of intellectual property asserted. In the case of intellectual property which protects a physical artifact (e.g., drawings, technical data), the process is relatively straightforward as there are likely only two parties to consider: Government and contractor. However, the case involving software is further complicated by the presence of open source software. Open source software is quickly becoming an integral part of software development, both inside and outside the DoD [78]. However, with the benefits

of open source software come specific copyright licensing requirements. There are a number of different open source licensing schemes[2], but two of the most important classes of licenses are *reciprocal licenses* and *permissive licenses*. Reciprocal licenses, also sometimes referred to as "copyleft" licenses, transfer license obligations to derivative software systems. This implies that if a contractor built a weapon system using open source code with a copyleft license, then that copyleft license would transfer to all, or part, of the derivative software; making it open source as well. Therefore, a contractor would only be able to make exclusionary intellectual property claims to the parts of the software where the copyleft license did not transfer.

In contrast, the other common form of open source licenses is the permissive, or sometimes referred to as the "unrestrictive" license. This license allows for the creation of derivative works without a transfer of license. The only requirement of this license is that any derivative works both acknowledge the original author and do not remove the original license markings. This allows for open source software with a permissive license to be incorporated into a proprietary closed source product.

However, much like DoD intellectual property regulations for each of the 100 plus open source licenses, there is a multitude of exceptions and exceptions to the exceptions for even the very basic license explanations outlined above. Unfortunately, merging the fields of DoD intellectual property regulations and open source license regulations creates a mess of, sometimes conflicting, guidance and policy. Most of which will require establishment of legal precedence to decide which policy supersedes others.

To outline a basic explanation, Figure 4-4[3] illustrates generalized outcomes from combining various open source licenses with privately developed code or "own" code. Within this Figure there are two important concepts: derivative work and combined work. Although definitions vary across different open source licenses, the Apache Li-

[2]The Free Software Foundation (FSF) lists 125 different open source copyright licenses. However, only 93 of the 125 are considered "free software" by the FSF definition, which is roughly "the users have the freedom to run, copy, distribute, study, change and improve the software"[79]. However, the standards of what qualifies as "free software" or open source software differ between organizations.

[3]Adapted from: `https://en.wikipedia.org/wiki/File:Software-license-compatiblity-graph.svg`

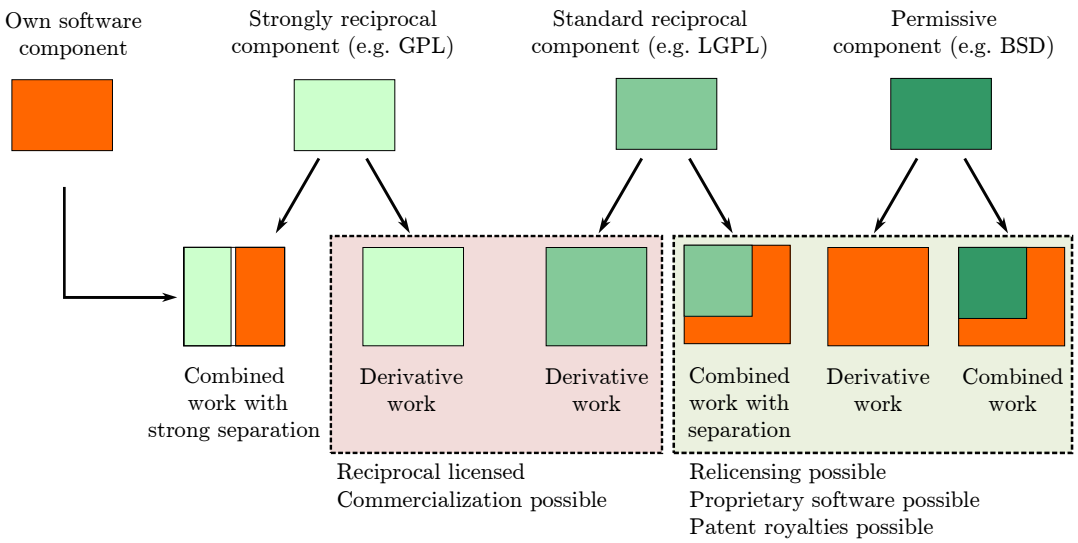


Figure 4-4: Open Source License Compatibility

cense 2.0 provides a definition which adequately demonstrates the difference between derivative and combined works:

> "Derivative Works shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of the Work and Derivative Works thereof."[80]

Although this definition relies on many other formally defined concepts (i.e., difference between big-W "Work" and little-w "work"), the intent of the definition is clear. Derivative work is that which is based upon previously licensed work. Whereas combined work remains separable from the original work or connects to an interface in the original work. There are three general outcomes which can occur from combining own code with reciprocal and permissive licenses. First, combined work with strong separation is an instance where own code is combined with a strong reciprocal license, but was developed in a way to maintain separation and isolation between the two licenses. The intellectual property outcome of these situations is case-by-case

dependent, as it largely depends upon the strength of separation. The next category is derivative works which contain a reciprocal license. The outcome of any derivative work based, in whole or in-part, on a reciprocal licensed work, is a new work with the same reciprocal license. Commercialization of this new work is possible, but in such cases the source code must be made available, upon request. Furthermore, this new work cannot "go closed" or be licensed under a subsequent proprietary license. The third and final case is when own code is combined with a work that is permissively licensed. In these situations, whether the final work is combined or derivative, the ultimate outcome is a new work which is licensable under a proprietary license and fully commercializable.

Consequently, determination of contractor rights is based not only on the funding test and regulations, but also upon the amount, type, and location of open source code used; if any. At this juncture in the model, if the government lacks rights and the contractor maintains rights, this establishes an exclusionary property right. However, recall the two-part definition for a strategic bottleneck, which requires both an exclusionary right and a unique solution. The model up to this point establishes an exclusionary right, but not a unique solution.

4.3.4 Determine Functional Relevance

The next step is to determine whether the solution protected by the property right is unique. This evaluation of "uniqueness" is referred to herein as the *functional relevance*. Functional relevance is defined as the necessity of component features or subassembly features (referred by Ulrich as "functional elements" [74]) to the system-level function delivered and how easily such characteristics can be substituted in the functional structure of the product [74]. Items or components with low functional relevance provide commodity features that are easily substitutable and common (e.g., geographic coordinate system conversions); whereas, items with high functional relevance provide features which are not easily substitutable and unique (e.g., agile thrust vector control). For instance, a weapon system may contain a component which uses a state-of-the-art software package for mensurating target coordinates within an error

of one meter, which is protected by intellectual property. If there are no other software packages capable of providing similar features and similar error tolerance, then this package is considered to have high functional relevance. This concept of substitutability and uniqueness is critical to intellectual property lock-in, because items which are easily functionally substitutable will not form a strategic bottleneck.

However, even if the Government has no rights to a piece of intellectual property, the contractor has legitimate rights and the item has high functional relevance; there are "competitive techniques" available to circumvent a lock-in [81]. The majority of these techniques are generally targeted at creating a substitute product or enabling a new vendor to recreate the original product. Well known competitive techniques for obtaining substitute products include [82]:

Competitive Copying A common practice to incentivize competition is to solicit bids without furnishing the necessary intellectual property. This method encourages competitors to copy an original manufacturers part; thereby, avoiding the piece of intellectual property which may initially preclude full and open competition.

Form-Fit-Function The Government always has an unlimited right to technical data relating to the form-fit-function of a product. It is possible using this data to solicit bids for a substitute good using only the form-fit-function data.

Direct Licensing This strategy relies on the Government licensing the intellectual property for manufacture elsewhere. This strategy is useful when a contractor refuses to sell the intellectual property and finds the continued manufacture of the product unprofitable.

Leader Follower This tactic formalizes a direct relationship between the original manufacturer and a competitor. Under this scheme, the original manufacturer is contracted to provide manufacturing assistance to enable to follower to become a competitive source for the product. This model differs from direct licensing because no royalty payment is made.

Specific Acquisition This technique involves purchasing rights to the intellectual property and then uses the rights as the basis for a follow-on competitive pro-

curement. This is the most often strategy used in practice [81], however, owning rights to intellectual property (i.e., design structure) and having the "know how" to actually manufacture a product (i.e., task structure) are not synonymous.

Reverse Engineering Finally, the Government has the ability to contract a manufacturer to reverse engineer another manufacturers product. Although, potentially expensive, this option has the advantage of possibly improving the product during the reverse engineering process.

Some of the aforementioned options rely on original manufacture participation, but others (i.e., reverse engineering and competitive copying) do not. Consequently, functional relevance is only maintained if the features provided are of enough singularity and complexity to preclude any of the competitive techniques outlined above.

It is important to note in Figure 4-5 that the assessment of formal relevance lies in the domain of both Contract Structure and Technical Architecture. This is because the assessment of functional relevance is an iterative assessment made by members of both domains. Engineers need to answer questions such as, "Are there viable technological substitutes?" Program managers need to answer questions such as, "Are the available technological substitutes economically viable?" Consequently, the assessment of functional relevance overlaps the Contract Structure and Technical Architecture.

To have functional relevance the piece of intellectual property must govern a component whose functionality is not easily substitutable. Furthermore, the intellectual property must govern a product which is technologically advanced enough and protected in a way that prohibits the aforementioned competitive techniques. Only at the confluence of these two functional factors does a *de jure* strategic bottleneck occur.

4.3.5 Determine Formal Relevance

Up until this point the DoD model of intellectual property lock-in was concerned primarily with criteria rooted in the contract structure. However, recall that a bottleneck must be both necessary to the functioning of the whole and non-substitutable. This second test of substitutability refers equally to the formal relevance of a product as

well as the already discussed functional relevance. Formal relevance is defined as the degree of difficulty to remove the physical abstraction on which a functional element is mapped. The model located at Figure 4-5 argues that the architectural choices in mapping function to form have implications on the flow of intellectual property licenses within a system.

In determining the flow of intellectual property licenses from open source code to own code the aforementioned analysis considered, among other factors, location of the code. The consideration of formal location or technical architecture location is just as important when attempting to establish a *de jure* strategic bottleneck. Consider an example in which a software program is comprised of 100,000 files. Manufacturer "XYZ" owns the intellectual property in exactly 11,000 of those files. Is this significant? At face value a property right to 11% of the files in a given software program may not seem like a tremendous amount, but the significance of those 11,000 files largely depends on where they are located within the formal architecture. Software and inherent architectures are complex as they are "made up of a large number of parts that interact in a nonsimple way"[73, p. 468]. This complexity is driven by two primary factors: the number of elements that compromise a system or piece of software and the coupling among those elements [83]. It is assumed that each element has at least a single connection, but many elements have significantly more connections within a system[4]. As a general rule, the more connections between an element or group of elements and the whole, the greater potential impact on the rest of the system and; subsequently, the greater cost to change that element. To deal with this complexity, good software uses abstractions and groupings to reduce complexity and

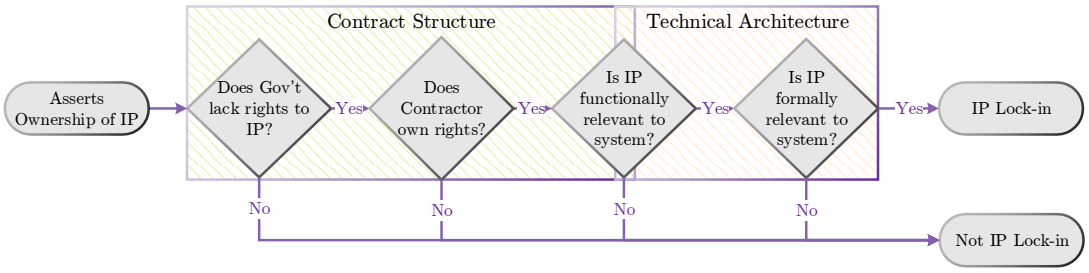


Figure 4-5: Hypothesized Model for Intellectual Property Lock-in

increase comprehensibility. Once such grouping technique is modularity, which at the most basic form involves breaking up a system into discrete chunks that communicate using defined interfaces [84].

Modularity

Modularization in software was pioneered in the 1970s as a way to shorten development times, create more flexible products, and improve comprehensibility [86]–[88]. Product agnostic modularity is defined in literature using the module as the unit of analysis:

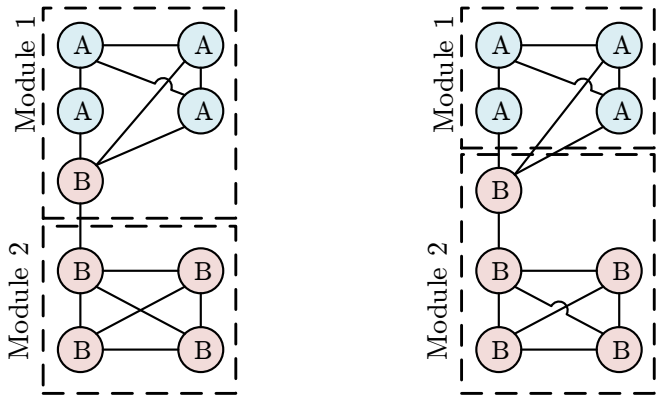
> "A module is a unit whose structural elements are powerfully connected among themselves and relatively weakly connected to elements in other units" [77, p. 63]

Within this definition are two critical concepts of modularity:[5] interdependence within and independence across. Given these properties of a module, Baldwin and Clark define six different modular operators [77]. Of particular interest to this model is the *substitution operator*. Baldwin and Clark argue that substitution forms the economic basis for competition because competition is only economically justified if two competitors have different products that serve the same purpose, but not equally well [77]. If both competitors offered different products that served the same purpose equally well, they would be substitutable and neither would have a competitive advantage over the other[6] [90]. This notion mirrors the definition of a strategic bottleneck.

[4] In many types of systems the number of connections follow a power-law distribution [85]

[5] Ulrich defines and views modularity differently; modularity is the one-to-one mapping of each formal component and functional element and views modularity as depending on two characteristics: 1) "similarity between the physical and functional architecture of the design", and 2) "minimization of incidental interactions among physical components."[89, p. 2]

[6] Barney argues that for a firm to have a potential competitive advantage a firm resource (e.g., knowledge or intellectual property) must have four attributes: 1) it must be of value, meaning that it either exploits market opportunities or marginalizes threats from competitors, 2) it must be rare, 3) it must be "imperfectly imitable", and 4) "there cannot be strategically equivalent substitutes for this resource that are valuable, but neither rare or imperfectly imitable." [90, p. 106]



(a) (b)

Figure 4-6: With "A" and "B" representing distinct pieces of intellectual property, let (a) represent systems with modular architecture and (b) intellectual property modular architecture. (Reproduced from: [93])

Intellectual Property Modularity

This concept is integral to the determination of lock-in because if the *de jure* strategic bottleneck is encapsulated into a module, it becomes easily substitutable; assuming a substitute function exists. This extension of academic work on modularity is known as *intellectual property modularity*, which is a form of modularity that takes the value of intellectual property into consideration when forming module boundaries [91]–[93]. Figure 4-6 illustrates the difference between an architecture that is modular, but not intellectual property modular (Figure 4-6a), and a system that is both modular and intellectual property modular (Figure 4-6b). This distinction is important because both modules in Figure 4-6a are easily substitutable; but, in Figure 4-6a it is relatively more difficult to remove "B" from the entire system because it crosses module boundaries. In terms of bottlenecks, Figure 4-6a represents a system with an extensive *de jure* strategic bottleneck and Figure 4-6b represents a system with an intensive *de jure* strategic bottleneck.

However, the theoretical concept of intellectual property modularity relies on an architecture that obeys the two fundamental concepts in modularity: interdependence

within and independence across. Consequently, it is equally important to the formal relevance as to how those modules are connected to the remainder of the software. This is critical because modules with tight coupling are difficult to substitute [94], which is due in-part to the way the modules are organized in an architecture. The organization of modules within an architecture is theorized [73] and observed empirically [95] as a hierarchy. The implications of a module hierarchy for the model are simple. A module at the top of a hierarchy cannot be substituted without requiring changes in many other modules in an architecture; whereas, substituting modules at the bottom of an architecture does not trigger cascading or widespread changes [96]. Consequently, the hierarchical properties of a piece of intellectual property are just as significant as both modularity and coupling.

Given these architectural foundations, formal relevance is defined, in some sense, as the antithesis of modularity. That is, a file or group of files is said to be highly formally relevant if it demonstrates tight coupling, loose interdependence, and a relative high position in the hierarchy. Such files or groups are difficult to substitute, which makes them formally relevant. Conversely, files or groups of files with loose coupling, tight interdependence, and low position in a hierarchy are easy to substitute and are said to have low formal relevance because their modular nature makes them easily substitutable.

4.4 Criteria for Intellectual Property Lock-in

From the model and the above discussion, the criteria for intellectual property lock-in the DoD domain can be distilled down to four criteria. For intellectual property lock-in to occur a situation would have to satisfy all four of the organizational and system level criteria:

1. The Government lacks sufficient rights to asserted intellectual property
2. The Contractor maintains sufficient rights to asserted intellectual property
3. The functional elements protected by the intellectual property are not substitutable

4. The formal location of the intellectual property prevents substitution

Providing these four criteria are met, the end result is arguably a *de jure* strategic bottleneck and potentially intellectual property lock-in. However, this is not to say that all cases will follow these four generalized criteria. The conceptual model and resulting criteria acknowledge that there are certainly boundary cases that will not fit neatly into these four criteria. Nonetheless, the majority of potential intellectual property lock-in cases in the DoD can be evaluated and concluded by addressing these four criteria.

4.5 Discussion

The inclusion of both formal and functional relevance in the conceptual model is salient because it permits a more equilibrium level of competition. This relationship is visualized in Figure 4-7. Each step in the model is designed to ensure the specific instance of intellectual property lock-in satisfies the regulatory and statute criteria. The natural evolution of such instances is that the initial number of lock-in cases is reduced, with each subsequent step, until an equilibrium point is reached. This equilibrium point in the contract structure portion of the conceptual model in Figure 4-5 is labeled as "A" in Figure 4-7. Equilibrium in this context refers to the conventional dictionary definition, "a state of rest or balance due to the equal action of opposing forces" [97]. In practice, particularly in physical systems, the contract structure of the model is the only portion evaluated. However, physical systems different significantly from software systems. Sturtevant lays out four principal differences [98, p. 15]:

1. "Software is an artifact that embodies pure function unencumbered by the burden of physical form"
2. "Software development firms engage in design but have no need for manufacturing or assembly"
3. "Software development is unencumbered by many economic costs, such as the large fixed costs found in an aerospace plant"
4. "Software is encumbered by few serious constraints imposed by physics"

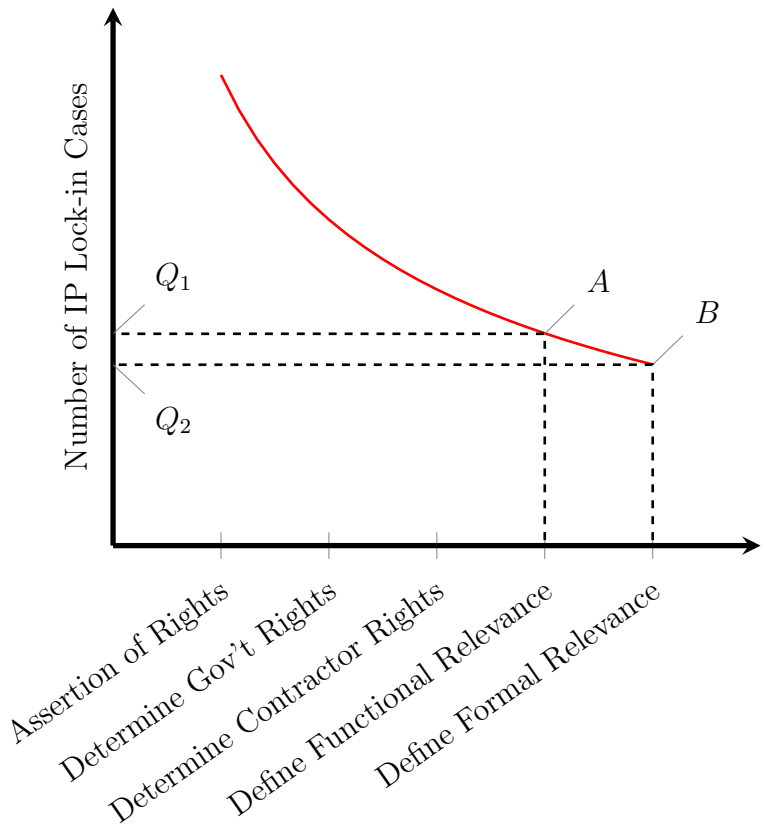


Figure 4-7: Equilibrium Curve for Number of Intellectual Property Lock-in Cases

Given these differences from physical systems, it is imperative that the conceptual model consider both the contract structure and technical architecture of a software system. Although, the contract structure serves as the foundation for definitions and criteria used to determine intellectual property rights in software, used by itself it occludes a significant aspect needed to determine the flow of rights in large complex software systems. For these reasons, the conceptual model reaches an imperfect equilibrium, point "A", which subsequently settles on a higher level of intellectual property lock-in than is necessary.

4.5.1 Lost Competition

This difference between the equilibrium point reached using the contract structure portion of the model and in using the complete model, points "A" and "B" respectively in Figure 4-7 is defined as *lost competition*. More formally, lost competition

is the theoretical distance between points Q_1, which is the number of intellectual property lock-in cases at the conclusion of the partial conceptual model (Figure 4-3) and Q_2; which is the number of cases at the conclusion of the full model (Figure 4-5). This difference, or ΔQ, represents a quantity of cases that are incorrectly deemed as intellectual property lock-in. The inclusion of a fourth step, which uses the architectural properties of a system to make judgments about the flow of intellectual property, allows for a more optimal equilibrium point and theoretically recaptures some lost competition.

4.6 Conclusion

This Chapter examined the various conceptual models that both influence and define intellectual property lock-in in the DoD. It began with a discussion of bottlenecks and the theoretical connection between bottlenecks and intellectual property lock-in. This sets the foundation for the conceptual model of intellectual property lock-in applied to DoD system. Finally, this chapter argued that using only a portion of the model was not well adapted for use in software because it was unable to capture the eccentricities and complexity only found in software. This chapter concluded with a discussion on the implication of adopting a new model and argued that the newer model would recapture some of the *lost competition* which resulted from the model. Given the analysis in Chapter 3, this new model could recapture some portion of the $6 billion and return those funds to tax payers.

Chapter 5

Scientific Methods

Chapter 4 outlined a conceptual model for defining intellectual property lock-in as well as identifying intellectual property lock-in in the DoD. To remind the reader, the proposed intellectual property conceptual model added a formal relevance evaluation function. This Chapter endeavors to answer "how" formal relevance can be evaluated in software. To accomplish this end, this Chapter is broken down into two main sections; beginning with an explanation of the formal architecture extraction process and followed by an explanation of the intellectual property extraction process. Recall from Chapter 4 that a *de jure* strategic bottleneck can only occur at the intersection of these two architectures.

Before beginning, it is important to note that the formal architecture extraction, used herein, is based on the set of works by Baldwin [76], [91], [92], [96], [99], [100], MacCormack [75], [94], [101], Rusnak [102], and Sturtevant [98]. Where possible, this section attempts to give credit to the specific work from which the ideas originated, but at times this is difficult because the works were both collaborative and compounding.

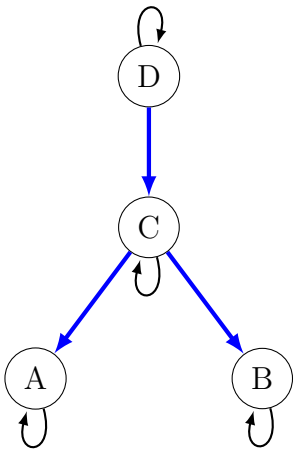
5.1 Extracting Architecture from Software

Architecture, at the most basic form, is a set of complex choices made by a system designer on the mapping of functional elements to a functional form. For any given

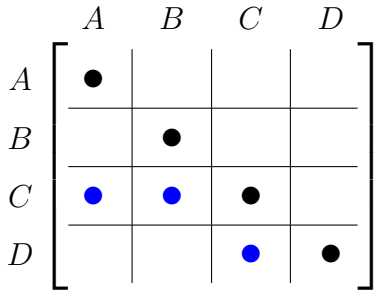
set of functional elements there are numerous ways they could be mapped to a functional form; each mapping carries a set of complexity and performance trade-offs. Particularly in software, architecture has the ability to scale into a complex set of connections and patterns; some conscious choices of the designer and some unforeseen connections that will later drive up complexity. In addition to the purposeful mapping of function to form, MacCormack et al., argue that architecture is equally influenced by designers reacting to incentives and organizational structures [75]. Capturing this complex set of interactions requires a process specifically tailored to the nuances in software, but generalizable enough so someone with only minimal software knowledge can understand the architecture [102].

5.1.1 Networks and DSM Architecture Visualizations

Before discussing the architecture extraction process in great detail, it is important to understand the basic structures on which the process is based. A common way to view and analyze a system architecture is using a network representation; see Figure 5-1a. Networks are useful in architectural analysis because they provide a natural expression for many complex architectural phenomena: hierarchical relationships, coupling, cohesion, modularity, and cyclicality. Such networks are comprised of nodes, which designate an element within a network, and arcs or edges, which designate a connection or link between two nodes. Importantly, arcs can also designate the directionality of connections between two nodes. In an undirected network the links or relations between two nodes are nondirectional; that is, nothing can be inferred from the link about which node depends on the other (e.g., in an undirected connection between two nodes n_i and n_j, it is impossible to distinguish if the connection is $n_i \rightarrow n_j$, $n_i \leftarrow n_j$, or $n_i \leftrightarrow n_j$) [103]. In a directed network, the links between nodes maintains the directionality of the relationship (e.g., a relationship where node n_i is friends with node n_j is noted as $n_i \rightarrow n_j$). It is important to note that directional relationships are not reciprocal; that is, $n_i \rightarrow n_j$ does not imply n_j is directionally connected to n_i (i.e. $n_j \not\rightarrow n_i$) [103].



(a) Simple Network



(b) Design Structure Matrix

Figure 5-1: Network Representations of Simple Networks

The distinction between directed and undirected is important because relationships in software are directional. Returning to Figure 5-1a, supposing this is the network for a small piece of software the nodes in this network, $\mathcal{N} = \{A, B, C, D\}$, could be files and the arcs between nodes, $\mathcal{L} = \{D \rightarrow C, C \rightarrow A, C \rightarrow B\}$, are a link between those files. However, the example in Figure 5-1a is just one possible expression of the architecture. Figure 5-1b is another expression of the same architecture in Figure 5-1a. This method of network visualization is a design structure matrix (DSM). In a simple network visualization, Figure 5-1a, a node is a single point; whereas, in a DSM, Figure 5-1b, each link is a single point. Although identical information is expressed, each visualization can highlight different network features.

5.1.2 Extraction Process

This method applies a DSM approach to analyze the relationships between different entities in a software system. The basic approach relies on the following 4 steps [75], [94]:

1. Capture a network representation of software source-code using dependency extraction tools
2. Find all the paths (direct and indirect) between files in the network by computing transitive closure

3. Calculate visibility scores to each file, which represent a file's reachability from other files or ability to reach other files in the network
4. Organize files into one of four canonical groups based on visibility scores

This section enumerates each step beginning with network extraction. There are two basic choices regarding the application of DSMs to software: 1) the unit of analysis, and 2) the type of dependency between units. In regard to the unit analyzed, it is possible to analyze software at the directory, source file, and function levels. Herein, the source file is used as the unit of analysis, which is supported by prior work on software design [75], [98], [102], [104], [105]. There are also choices in the dependency type between these source files. Keeping with previous literature [75], [106] the function call is used herein. A function call is an instruction in the software code that requests a specific task be executed; sometimes making the request internally or of another source file. When one source file executes a function call that requests a task to be completed by another source file, this is characterized as a directional dependency between the two source files (e.g., using Figure 5-1a, if a function call in file D calls a function call in file C, we would say that file D depends on file C). It is important to note that this is a directional dependency and just because file D depends on file C, does not imply file C depends on file D. This connection is referred to as a first order dependency because it is an explicit call from file D to C. The extraction of both file names and function calls within software is accomplished using a commercially available call extractor, specifically Scientific Toolworks *Understand*.

$$M_{n,n} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \tag{5.1}$$

To illustrate the output of step 1 and the process of step 2, see Figure 5-1a. In this toy example dependencies were extracted from a piece of software that contains four source files (i.e., $\{A, B, C, D\}$). Upon completing the extraction, we see that D calls

C and C calls both A as well as B. These are direct, or level 1, connections between the files. The resulting DSM is an illustration of the same connections in the network graph, except arrayed as a matrix. The DSM is read from right to left. For example, starting at row C, from left to right, we see that two blue dots denote that file C is connected to both A and B. It is important to note that the DSM in Figure 5-1a, and all DSMs henceforth, are square matrices. A square matrix is a matrix with the same number of rows and columns. An n-by-n matrix is known as a square matrix of order n, see Equation (5.1).

$$M_{vis} = \sum_{i=0}^{n-1} M^i \tag{5.2}$$

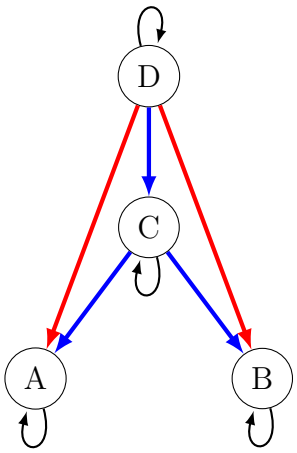
$$M_{vis} = M^0 + M^1 + M^2 + M^3 \tag{5.3}$$

$$M_{vis} = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{array} + \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array} + \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \end{array} \tag{5.4}$$

$$M_{vis} = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array} \tag{5.5}$$

The next step in the methodology calls for identifying all direct and indirect paths[1] through the software. This full set of paths is known as visibility or transitive closure. Matrix multiplication is used to identify the visibility of each source file [107], [108]. Specifically, by raising the DSM to successive powers from 0 to $n-1$, it is possible to obtain the direct and indirect dependencies for each successive path length, where n is the length of a square matrix, Equation (5.2). Summing these matrices yields the visibility matrix, which shows both direct and indirect dependencies between source

[1]The terms "path" and "walk." are important in network analysis. "A *path* is a walk in which all nodes and all lines are distinct" [103, p. 107], and "a *walk* is a sequence of nodes and lines, starting and ending with nodes, in which each node is incident with the lines following and preceding it in the sequence" [103, p. 105].



(a) Simple Network

	A	B	C	D
A	●			
B		●		
C	●	●	●	
D	●	●	●	●

(b) Design Structure Matrix

Figure 5-2: Network Representations of Visibility Networks

files for all possible paths up to the maximum path length. See Equations (5.3), (5.4), and (5.5) for an example of these calculations using the network in Figure 5-1a[2]. The result from these calculations are visible in Figure 5-2, wherein two new links for source file D are now visible. This is because file D originally connected to file C, but file C also connected to files A and B. Therefore, in the visibility matrix we have captured this second order connection between $D \rightarrow A$ and $D \rightarrow B$.

The penultimate step is to calculate the visibility scores for each file in the software. There are four metrics of concern [75], [98]:

Fan-in (FI) is a measure of how many other nodes depend upon it *directly*. FI is computed by counting the number of links down the column (including the diagonal link) of the first-order dependency DSM.

Fan-out (FO) is a measure of how many other nodes it *directly* depends upon. FO is computed by counting the number of links across the row (including the diagonal link) of the first-order dependency DSM.

Visibility Fan-in (VFI) Is a measure of how many other nodes depend upon it *directly or indirectly*. VFI is calculated in the same fashion as FI, except links

[2]It is always assumed a file connects to itself; this calculation is expressed in Equation (5.4) where M^0 yields a matrix of connections along the diagonal.

are summed down the column (including the diagonal link) of the transitive closure DSM.

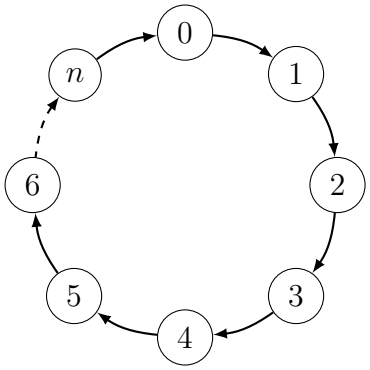
Visibility Fan-out (VFO) is a measure of how many other nodes it *directly or indirectly* depends upon. VFO is calculated in the same fashion as FO, except links are summed across the rows (including the diagonal link) of the transitive closure DSM.

The key distinction between the measures above are the matrix which they are derived from (i.e., first order DSM or transitive closure DSM). To illustrate an example, in Figure 5-2b, file C has a VFO of 3, which means that it depends on 75% of the files in the software either directly or indirectly. Visibility fan-in (VFI) is calculated similarly, instead by summing down the columns of the visibility matrix. Continuing the example, file C is seen by both itself and file D; 50% of the files in the system.

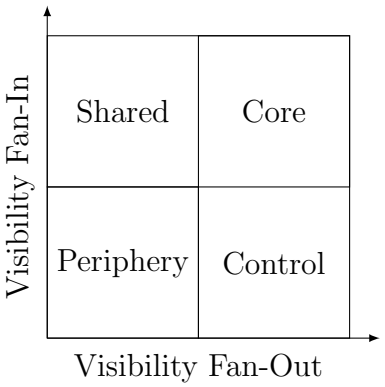
5.1.3 Abstraction of Architecture into Categories

To categorize all the files into distinct groups it is first necessary to identify all the cyclic groups in the network [96]. "A cycle [or cyclic group] is a closed walk of at least three nodes in which all lines are distinct, and all nodes except the beginning and ending node are distinct" [103, p. 108], see example in Figure 5-3a. By this definition, all nodes in a cycle are connected directly or indirectly to all other nodes in the cycle. Architectures can vary from systems that are purely hierarchical (i.e., acyclic networks) to those which are comprised of many large cycles. In the methodology outlined by Baldwin et al., the main focus is on the largest cyclic group, measured in terms of total nodes [96].

Once cycles are computed, the individual files are categorized into one of four canonical groups. This grouping is determined by creating scatter plot with VFI on the y-axis and VFO on the x-axis, see Figure 5-3b. The result is four clear groups of files which populate the upper extremities of each of the four noted regions (i.e., core, shared, control, periphery). From this grouping it is possible to classify each file as "low" VFI and/or VFO as well as "high" VFI and/or VFO. These relative distinctions are what place a file into the final classification as either a Core, Shared,



(a) Cyclic Group of Length n



(b) Notional Scatter Plot of Visibility Measures

Figure 5-3: Examples of Architectural Abstractions which Assist in Categorization

VFI	VFO	Group	Description	Equalities
High	High	Core	Core components rely upon many other files and are relied upon by many other files. This component is by definition the largest cyclic group in a system	VFI_c & VFO_c
Low	Low	Peripheral	Peripheral components do not influence and are not influenced by many nodes in a system	$VFI < VFI_c \cap VFO < VFO_c$
High	Low	Shared	Shared components are relied upon by a large proportion of the system, but do not rely upon many other components	$VFI \geq VFI_c \cap VFO < VFO_c$
Low	High	Control	Control components rely upon many other files, but are relied upon little by other components	$VFI < VFI_c \cap VFO \geq VFO_c$

Table 5.1: Description of Four Canonical Groups

Control, or Periphery component. A tabular version of the classification taxonomy, as well as descriptions of each group, are listed in Table 5.1. It is interesting to note that other researchers define the "high" VFI and "low" VFO group as "utility" [98], explaining that these files which populate this group provide utility-like functions to the rest of the software.

From this four group approach, architectures are classified into three different patterns [96]. The first being a "core-periphery architecture," which is defined as an

architecture that has a single cyclic group that is both large relative to the whole and substantially larger than the next largest cyclic group. A "multi-core architecture" is defined as a system which contains two, or more, cyclic groups of similar sizes that are large relative to the overall size of the whole. Finally, a "hierarchical architecture" is defined as a system in which all cyclic groups are small relative to the system as a whole. Neither in theory nor practice has a rigorous quantitative threshold been established for what constitutes a "relatively large" or "relatively small" cyclic group, but by convention Baldwin et al., use a threshold value of 5% [96]. That is, if a cyclic group contains more than 6% of the whole, it is said to be a core-periphery architecture and if the largest cyclic group contains 4% or less then the architecture is said to be hierarchical [96]. Although these definitions seem loose, a survey of 1,286 software releases finds that 80% of the releases contain a core-periphery structure [101], which provides some empirical support for the thresholds.

5.1.4 Visualize the Architecture

The final step in the process is to visualize the architecture. Given the data extracted thus far, there are many visualization options. The one selected for use, herein, is the "core-periphery" architecture view, which is defined as a view where the largest cyclic group is the basis. Baldwin et al., outlines two steps to visualize the core-periphery view [96].

1. Organize the four groups of files along the DSM diagonal in the following order Shared, Core, Peripheral, and Control.
2. Sort within each group by VFO descending and VFI ascending.

The result is an architecture view in which all cyclic groups are clustered along the DSM diagonal, there are no between group dependencies above the diagonal, and except for cyclic groups each component has no dependencies above the DSM diagonal. Figure 5-4 is an example of the "core-periphery" view of a software system. Note the cascading groups listed along the diagonal with one additional group "singleton." This group contains "singleton" networks or networks with a node size of $n = 1$. There are many plausible explanations for singletons in software, such as text files,

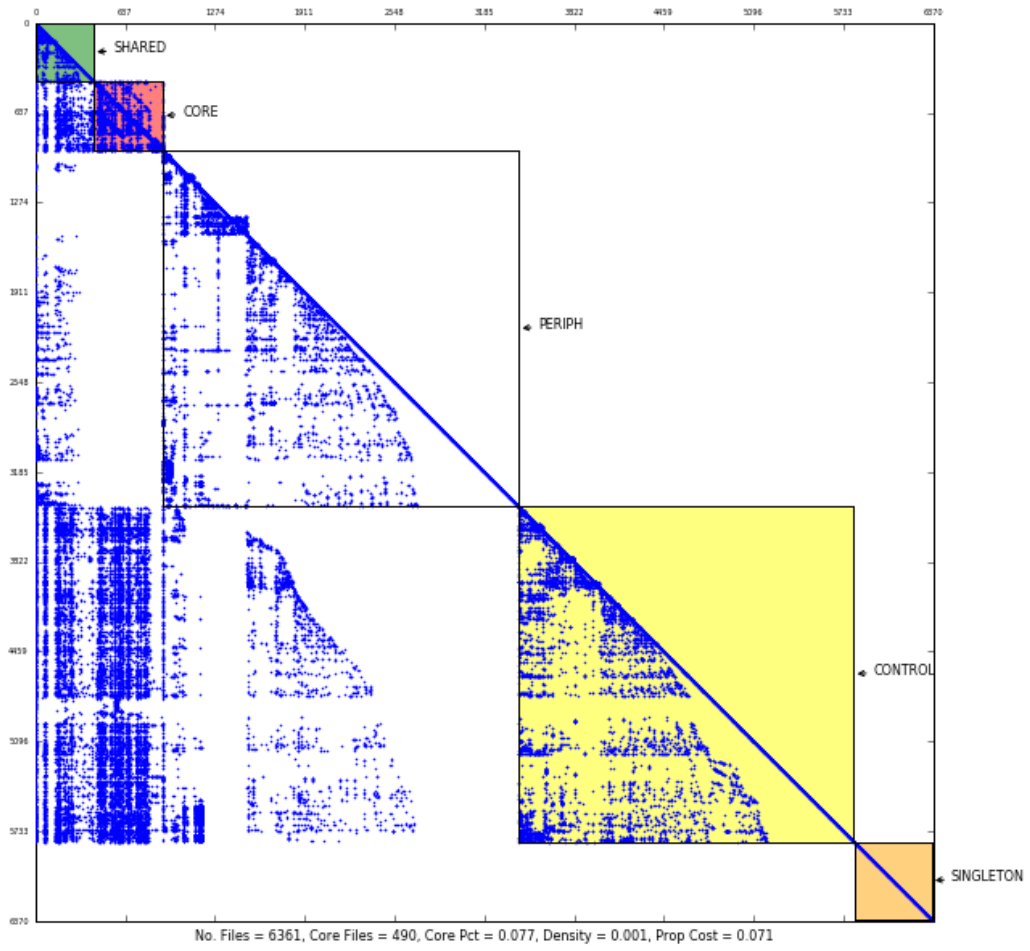


Figure 5-4: Core-Periphery View of Architecture

files that are dynamically linked versus statically linked, or artifacts in the extraction process.

5.2 Measuring Substitutability from Architecture

In earlier Chapters lock-in was explained as a function of switching costs or substitutability. Taking it one step further, the concept of switching costs is based on the substitutability of a new technology or component. If a new piece of technology is easily substitutable, in terms of time and money, into a system, then one could argue there is a relatively low switching cost. Conversely, if a piece of technology is not easily substitutable, one could argue that with this lack of substitutability comes a high

switching cost and subsequently a high potential for lock-in. What defines intellectual property lock-in, as opposed to technological lock-in, are switching costs determined by property rights due to intellectual property (i.e., *de jure* strategic bottleneck).

5.2.1 Hardness-to-Kill

Previous work on the evolution of software design and modularity provide a means to assess the substitutability of files in software with little *a priori* knowledge of the functionality of the software itself [94]. This work builds from previous theory on the architectural design [77] and research on quantifying modularity in software [75]. Specifically, previous work studied the evolution of software, over a series of sequential versions, to identify the most important factors in component survival, "which is an indicator of the degree to which components can be removed or substituted" [94, p. 4], and shows that tightly-coupled components have a higher probability of survival as software evolves, making them "harder-to-kill." Since this measure of hardness-to-kill is a proximal measure for substitutability, it should also serve to identify those files which have high switching costs and; ergo, a large potential for lock-in.

Previous work suggests files with high VFI and high VFO are statistically significant indicators of hardness-to-kill [94]. However, high VFO, by itself, was not uniformly significant across all samples in previous research; suggesting that high VFI is more dominant in explaining survival. Intuitively this makes sense; files with a high VFI score imply they are relied upon extensively by other files in the software. Substituting a file which is relied upon extensively by other files is difficult, whereas substituting a file which relies upon others (i.e., high VFO) is relatively easier. Consequently, the two types of files, which are both more survivable and least substitutable, fall into the groupings with the highest VFI: Shared and Core groups.

5.3 Extracting Intellectual Property

The processes up to this point were focused on extracting the technical architecture of software, specifically the design structure. However, it is also necessary to extract

vestiges of the contract structure. Although the extraction processes differ, the general method of extraction, cleaning, and visualizing is applicable to both extraction of technical architecture and contract structure. Before discussing the specifics of the process in detail, it is important to illustrate, for the reader, exactly how intellectual property artifacts appear in software.

5.3.1 Type of Intellectual Property

Software is largely protected using a copyright. As a general rule, a copyright is automatically attached to any novel expression of an idea, whether it is sound, imagery, or text. In the case of software, the code in each file is considered a literary work (17 U.S.C. § 101) which is an expression of an idea. That expression could be for something as simple as a calculator function. A copyright protects that specific expression of the idea, not the idea itself[3]. Equally important is that a copyright does not need to be registered to be legally effective, nor does a published work need to bear an explicit notice of a copyright protection; assuming it was published after March 1, 1989 [109]. At face value, this makes determining copyright ownership problematic in large software systems. However, in practice, very seldom does a software designer write all the code in a program from scratch; code is often reused. As discussed in Chapter 4, when code is reused the product is a derivative work. Depending on the license agreement for code reused, it is common for derivative works to retain the original copyright notice. This is the case for nearly all open source licenses; however, closed licenses must be evaluated on a case-by-case based on the specific license provisions carried.

Where notice of copyrights is required 17 U.S.C. § 401 outlines general provisions for the form of the notice. Specifically, every copyright notice must consist of three elements. First, the symbol © (usually written in software as `(C)` or `(c)`), the word "Copyright", or the abbreviation "Copr" must appear in the notice. Second, the

[3]This distinction is the general difference between a copyright and a patent. A copyright protects the novel expression of an idea, but no protection of the idea. Whereas, a patent does not protect the expression of the idea, but the underlying substance of it [109].

notice must contain the year of first publication of the copyrighted work. Finally, the notice must contain the name of the copyright owner or a "generally known alternative designation of the owner" (17 U.S.C § 401). These three elements must be affixed "in such a manner and location as to give a reasonable person notice of the claim of copyright" (17 U.S.C. § 401). In practice, copyright notices in software are often disclosed at the source file level; typically, as a commented-out section in the beginning of each source file. See example in Listing 5.1.

```
1  /* -*-c++-*-
2   * MapDump - main (.h & .cpp) - Using 'The MIT License'
3   * Copyright (C) 2010, MOVES Institute
4   *
5   * Permission is hereby granted, free of charge, to any person obtaining a copy
6   * of this software and associated documentation files (the "Software"), to deal
7   * in the Software without restriction, including without limitation the rights
8   * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
9   * copies of the Software, and to permit persons to whom the Software is
10  * furnished to do so, subject to the following conditions:
11  *
12  * The above copyright notice and this permission notice shall be included in
13  * all copies or substantial portions of the Software.
14  *
15  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
16  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
17  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
18  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
19  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
20  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
21  * THE SOFTWARE.
22  */
```

Listing 5.1: Example Copyright Notice

This example clearly demonstrates the requirements outlined in 17 U.S.C § 401. Note that the "*" at the beginning of each line is a delimiter in the C++ programming language to denote a comment versus executable code. Inspection of lines two and three in Listing 5.1, show that these two lines contain all three required elements for a copyright notice. Line two also contains a reference to a specific license, "`The MIT License`", which is a type of open source license. This discloses, to any potential re-user, that the code is subject to the terms outlined in the MIT License or any named license type (i.e., GPL 3.0, BSD, Creative Commons, etc.). Additionally, lines twelve and thirteen provide an empirical example of a copyright that requires the notice be included in any subsequent copies.

5.3.2 Extraction Process

At the most basic form a file of source code is file full of text. Consequently, it is possible to search for phrases, characters, or words quickly and efficiently using regular expressions. Three pieces of information from the copyright notice are of interest; presence of a copyright, license type, and owner. Although each piece of information is searched for using different regular expressions, the process is identical. Using the presence of a copyright notice as an example, the process begins by building a directory of all files in a software program. Then each file in the directory is opened and evaluated line by line for any regular expressions matches. In this case, the regular expressions are looking for case insensitive matches to ©, "`copyright`", "`copr`", and "`(C)`". Effort was taken to make the regular expressions robust enough to detect imperfect matches that may represent a positive match (e.g., "`( c)`" instead of "`(c)`" or "`copyr`" instead of "`copr`"). If a match exists, the name of the file and the specific line where the match is contained are recorded. For example, if this same process were run on the file which contains Listing 5.1, it would return "`Copyright (C) 2010, MOVES Institute`". In this example, the regular expression returned a line that contains both the copyright notice and the owner. This line is later parsed to break out the owner, the `MOVES Institute`, which is then added as a variable to each file. This same process is run for every line in every file in a software program.

5.4 Intellectual Property Architecture

Once all three intellectual property characteristics are extracted from software and parsed, the intellectual property information is then overlaid onto a core-periphery view of the software. Herein, the combination of intellectual property information and the core-periphery view is referred to as the *intellectual property architecture*. The benefits of this view are that it easily allows for analysis of the intellectual property in the two groups which have the largest potential for lock-in; Shared and Core. From this new visualization, three intellectual property architecture views are created: open source view, license type view, and owner view.

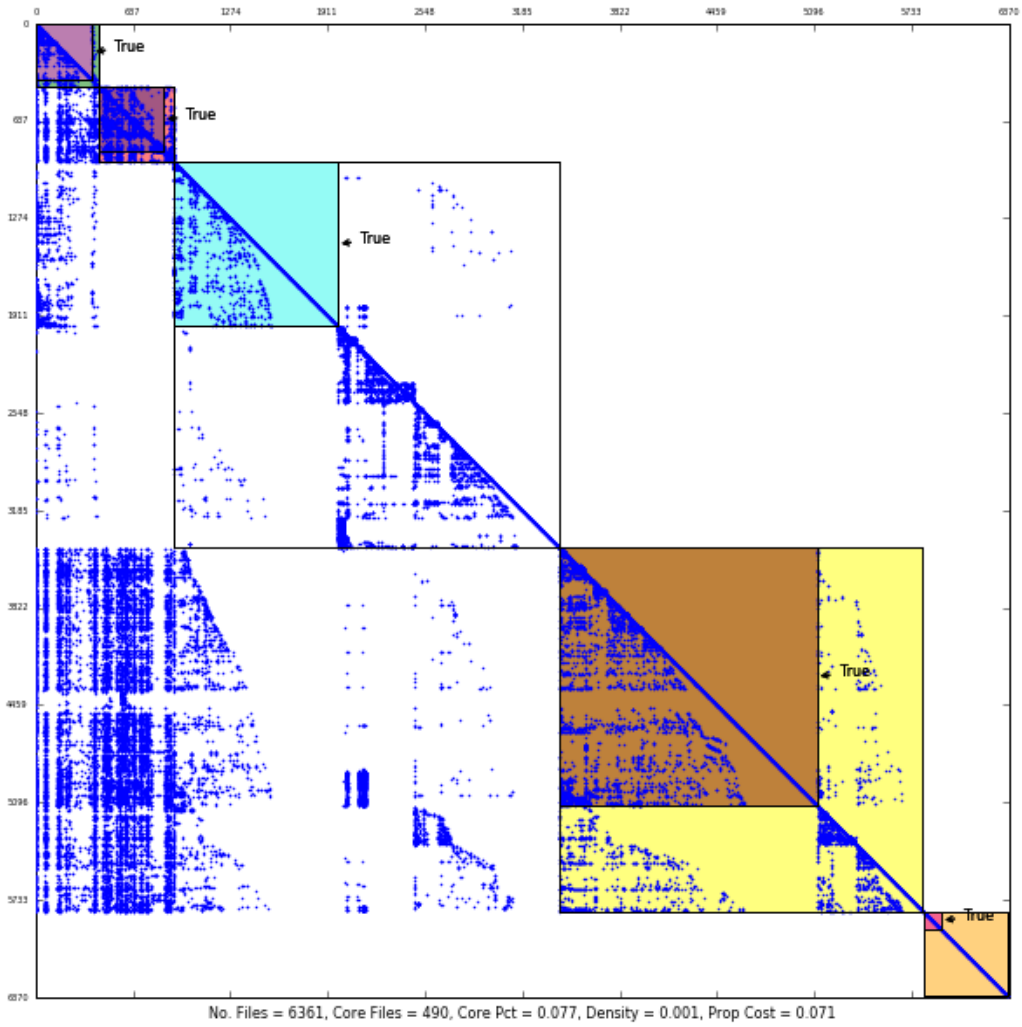


Figure 5-5: Intellectual Property Architecture Open Source View

An important note, in each of the new architectural views more connections above the main diagonal will appear than in the traditional core-periphery view. This is because the sort order of the matrix must be changed to group files with like licenses, owners, or open source properties before sorting on the core-periphery criteria. If this first grouping step were removed, the resulting visualizations would contain meaningless patterns of license or owners.

Figure 5-5 is an example of the open source view. In this view, all the files are labeled as either open source (denoted with a "TRUE" label) or not open source (not labeled). This binary classification allows for inspection of quantity of open source in both the Shared and Core groups. Those files that are labeled open source contained

a clear copyright notice and those that are not labeled open source contained either a "closed" copyright notice or no copyright notice at all. In the latter situation, where no copyright notice is provided, it must be assumed that the file is "closed" for reasons outlined in the previous section on copyrights. From this view, it is possible to understand what quantity of both the Shared and Core are built using open source code. In theory, this should allow for more informed judgment about the prospect of being locked-in. For example, if 75% of the Core is open source, then it is reasonable to conclude that the likelihood of being locked into this software is relatively less than a system where the level of open source in the Core is 0%. The reasonableness of this conclusion is predicated on the ability to use one of the many competitive techniques outlined in Chapter 4. If the core of a system is 75% open source, regardless of open source type, the software should be easier to reverse engineer than a software where the Core is 0% open source.

However, earlier discussions of open source license types noted that there are certain open source licenses which can "go closed." Consequently, it is equally important to look at the type of open source license used. Figure 5-6 is an example of the license type view. This view allows for inspection of specific license types, at key points, in the architecture. Again supposing 75% of the Core is open source, the benefit of the license type view over the open source view is that it is now possible to see what type of licenses comprise that 75%. Suppose that the majority of those 75% open source licenses are protected by a reciprocal license. Meaning that by creating a derivative work the license from the open source code transfers to the derivative work unless strong separation exists. By virtue of this reciprocal license located in the Core of the system, it is reasonable to argue that no separation exists between the reciprocal licensed code and the remainder of the system. Ergo, the reciprocal license should transfer. Making the preponderance of the derivative work reciprocally licensed and unable to "go closed."

Finally, Figure 5-7 is an example of the owner view. This architectural view visualizes the copyright owner for each file. If more than one copyright is present in a file the copyright owner names are joined with "AND"(e.g., in Figure 5-7 some files

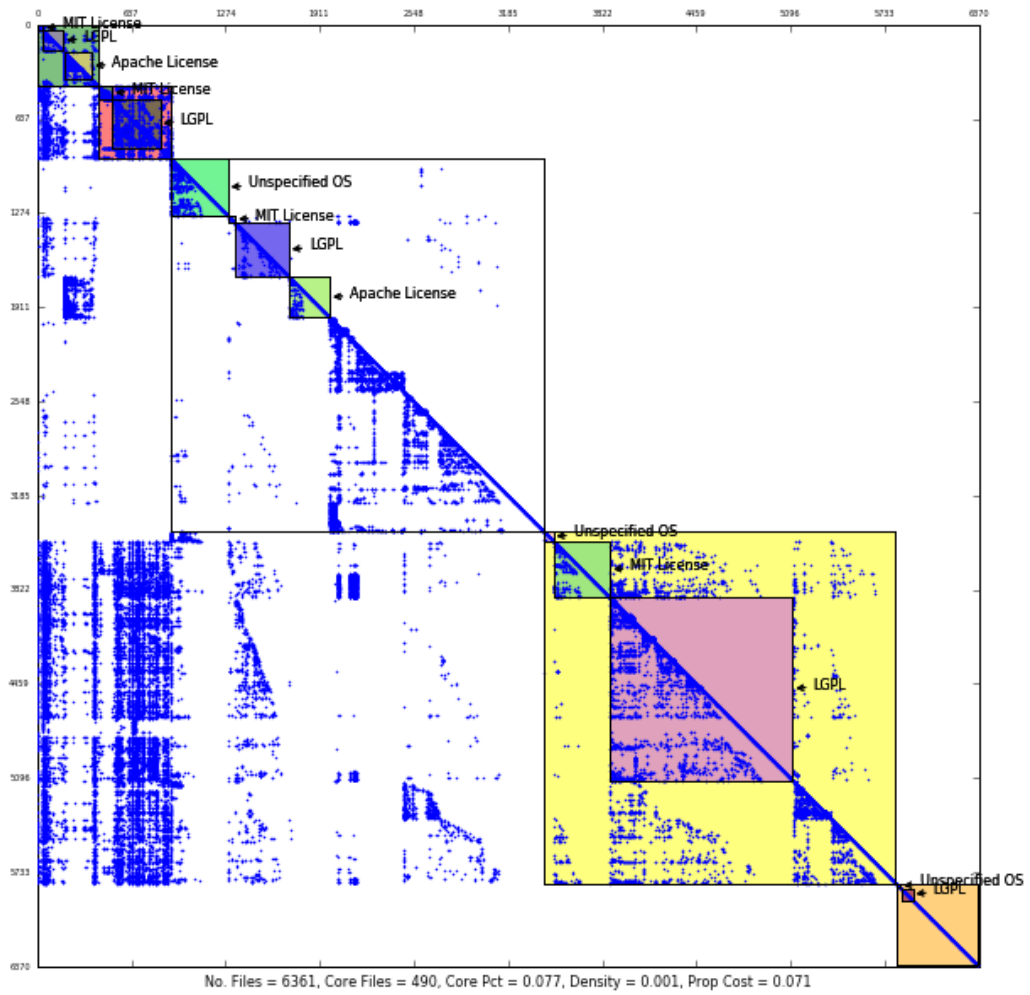


Figure 5-6: Intellectual Property Architecture License Type View

are copyright "Raytheon AND Hughes Aircraft", indicating that there are copyrights for both Raytheon and Hughes in that file). This view differs from the license view in that the copyright owner is visualized, not the license owner. To illustrate the difference, consider a file created by a private individual, John Doe. John Doe is the copyright owner for the file he created, but he may release the file under an open source license maintained by an external organization (e.g., "copyright John Doe - using the GPL license". The license type view would depict "GPL" and the owner view would depict "John Doe").

These three novel architectural views provide a way to analyze the intellectual property architecture of software with little to no *a priori* knowledge of the software

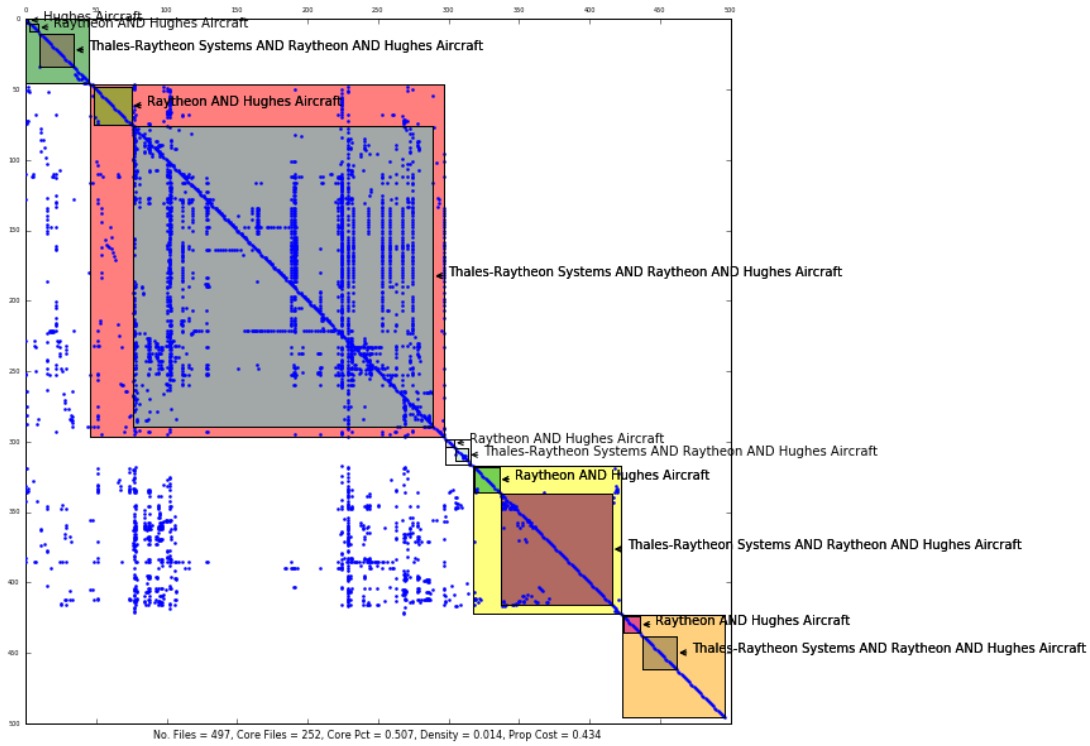


Figure 5-7: Intellectual Property Architecture Owner View

itself. Furthermore, they focus the attention to a sub-sample, Core and Share groups, of the entire architecture. This allows for a more deliberate and focused analysis than was previous possible.

5.5 Architecture Patterns

The ability to overlap architectural views allows for combinatorial classification of systems into architecture patterns. An architecture pattern is defined, herein, as a class of architectures which adheres to specified properties. Using the two extraction methods described above results in two distinct architectures: Intellectual Property Architecture and Technical Architecture. Each of these distinct architectures are broken down into sub-patterns.

Table 5.2: Description of Technical Architecture Sub-Patterns

Architecture	Sub-Pattern	Equalities
Technical Architecture	Hierarchical	$Core_{pct} < 5\%$
	Core-Periphery	$5\% < Core_{pct} < 50\%$
	Degraded	$Core_{pct} > 50\%$

5.5.1 Technical Architecture Sub-Patterns

In the case of technical architecture, Section 5.1.3 outlined three distinct architectural patterns: core-periphery, multi-core, and hierarchical. However, in practice multi-core architectures are relatively uncommon and the goal of this research is to measure the magnitude of high VFI files. Therefore, the sub-patterns of technical architecture are redefined as: hierarchical architecture, core-periphery architecture, and degraded architecture. The only new category is *degraded architecture*. A technical architecture is deemed degraded when the core size represents a preponderance of the system. That is, there is a single cyclic group which accounts for more than 50% of the software. In cases with such large cores, the ability to make meaningful impacts to the quality of the software is hampered [98]. The properties for each of these sub-patterns are outlined in Table 5.2. The theoretical cut points between hierarchical and core-periphery were adopted from Baldwin et al. [92]. However, there is not an academically supported cut line between core-periphery and degraded; 50% is used, herein, because it represents an obvious degraded architecture. One could make the argument to decrease this number to anything above 30%; however, often times the complexity of architecture is a trade-off for performance. There are some practical applications where a high core size is required to obtain performance; the threshold of 50% allows ample tolerance to account for those cases as core-periphery instead of degraded.

Table 5.3: Description of Intellectual Property Architecture Sub-Patterns

Architecture	Sub-Pattern	Equalities[1,2,3,4,5]
IP Architecture	Accessible	$X_{OS} > 50\% \cap Y_{OS} > 50\%$, OR... $X_{Gov't} > 50\% \cap Y_{Gov't} > 50\%$
	Closed	$X_{OS} < 50\% \cup Y_{OS} < 50\%$

[1] For systems with a core-periphery or degraded architecture $X = Core$ and $Y = Shared$. For systems with a hierarchical architecture $X = D_9$ (ninth decile) and $Y = D_8$ (eight decile) of files sorted by VFI.
[2] X_{OS} - Percentage of the core/decile that has an Open Source License
[3] $X_{Gov't}$ - Percentage of the core/decile that is copyrighted to U.S. Government
[4] Y_{OS} - Percentage of the shared/decile that has an Open Source License
[5] $Y_{Gov't}$ - Percentage of the shared/decile that is copyrighted to U.S. Government

5.5.2 Intellectual Property Sub-Patterns

The intellectual property architecture is divided into two sub-patterns: Accessible and Closed. However, the defining properties for intellectual property architectures are based on the type of technical architecture. This is a necessary step to ensure the taxonomy, described later, is extensible across the spectrum of technical architectures. The unit of analysis for pattern identification is the percentage of files with the highest VFI score or hardness-to-kill that are open source or where the copyright holder is the U.S. Government. In core-periphery and degraded architecture these groups of files are defined as the core and shared. However, in the case of hierarchical systems, with very small cores, this measurement is meaningless. Instead groups of files with high VFI are identified by the deciles of files sorted by VFI. The ninth decile (D_9) represents the group of files with the top 10% VFI score and the eighth decile (D_8) represents the group of files with the top 20% of VFI scores.

The properties which define accessible and closed architectures are outlined in Table 5.3. The properties were chosen for an *accessible intellectual property architecture* because in these patterns enough code is either owned by the DoD or in the public domain to potentially utilize one of the many competitive techniques outlined in Chapter 4. Thus, these patterns are termed *accessible* as opposed to *open*. The other intellectual property pattern is closed architecture, this pattern of architecture is referred to as *closed* because it is unlikely that there is enough of the original code available to utilize one of the competitive techniques outlined in Chapter 4.

5.5.3 Novel Theoretical Patterns

With three sub-patterns defined for both technical architecture and intellectual property architecture, novel architectural patterns are theorized at the multiple junctures, see Figure 5-8. These patterns are defined as ideal, boundary, and hazard. Each of these new patterns represents a likelihood of intellectual property lock-in, with the ideal pattern representing the most desirable configuration and the hazard pattern representing the least desirable configuration to avoid intellectual property lock-in. Each of these patterns will be discussed in greater detail in Chapters 7 and 9. This section was designed to lay the theoretical underpinnings before discussing empirical results.

5.6 Conclusion

This chapter explained the process by which architecture is extracted from software and the process by which intellectual property information is extracted from each file in a software system. Additionally, this chapter discussed which components

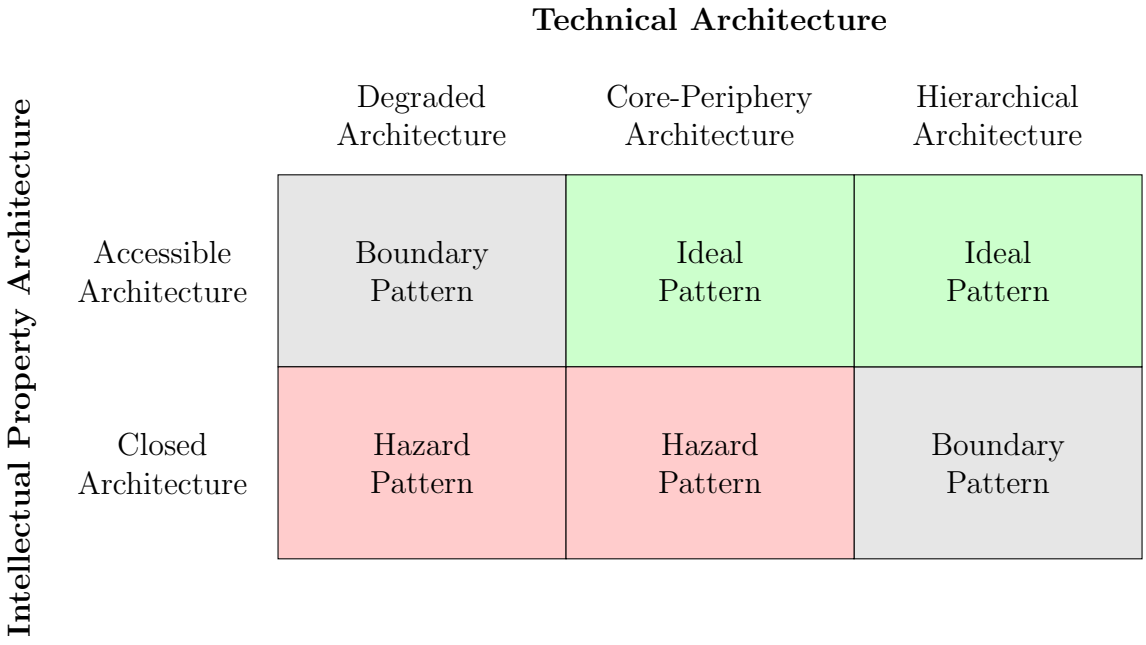


Figure 5-8: Taxonomy of Architectural Patterns (assuming no Unlimited Rights)

in an architecture are "hardest-to-kill" and theorized that those components have the highest likelihood for lock-in because of the difficulty to substitute those files. Finally, the two extraction processes were combined to form a novel view of software architecture, the *intellectual property architecture*. The result is a view of architecture that permits easy inspection of the "hardest-to-kill" files and the ability to make more informed judgments about the prospect of getting locked-in to a software based on copyright ownership of those "hardest-to-kill" files.

Chapter 6

Research Methods

The first attempt to address the research questions employed a deductive approach to construct tests establishing a causal connection between formal software architecture and intellectual property alignment as the independent variable and the presence or absence of intellectual property lock-in as the dependent variable. However, deductive reasoning is not appropriate here given wickedness of the problem (e.g., many unknown confounding factors and potential causal mechanisms). Consequently, to effectively investigate these research questions, a multiple case study design across a number of DoD software programs was used. Although, DoD software programs vary widely in terms of requirements, domain, and design choices, this research seeks to answer the research questions using the same basic framework for each case study. This common methodology, coupled with semi-structured interviews to develop context around each case, allowed for deeper cross-case comparison and ultimately led to more meaningful conclusions on the mechanisms that induce intellectual property lock-in.

6.1 Research Design

Maxwell argues, "[design] selection depends not only on your research questions but on the actual research situation and what will work most effectively in that situation to give you the data you need" [110, p. 74]. There are two factors defining avail-

able research methods herein: multiple conjunctural causation (or equifinality) and characteristics which violate basic assumptions of statistical approaches. Conjunctural causation is the condition where conjunctions of causally relevant conditions form to yield a specific outcome ($AB \rightarrow Y$, read as A *AND* B yield outcome Y). A conjunctural causation is referred to as a single "path" to causality. Multiple conjunctural causation refers to the number of "paths," which yield a specified causal outcome ($AB + CD \rightarrow Y$, "+" indicates a Boolean "OR"). Depending on the context, a "path" may be dependent on the presence or absence of a given condition ($AB + aC \rightarrow Y$, lowercase conditions are Boolean notation for absence). These characteristics are evidenced in both Chapter 4, where intellectual property lock-in is argued as the conjunction of technical architecture and contract structure, and Chapter 5 where it is argued that there are multiple combinatorial options for each architectural pattern. Thus any causation herein, is likely both conjunctural and multiple.

The phenomenon of multiple conjunctural causality violates many of the necessary assumptions for mainstream statistical analysis [111]. First, each condition does not have an independent impact on the causal outcome, instead the combination of conditions or absence of conditions yields a causal outcome; ergo, "additivity" cannot be assumed. Second, permanent causality is not assumed because there may not be any commonality across causal "paths." Third, uniformity of causal effects cannot be assumed because a condition in any given conjuncture may act in favor of the causal outcome or, combined in a different conjuncture, act against it. Finally, causal symmetry cannot be assumed; instead, asymmetry is assumed, which means the presence or absence of a specific condition may require different explanations (i.e., $ABc \rightarrow Y$ does not necessarily imply $abC \not\rightarrow Y$).

The violation of so many mainstream statistical assumptions makes a variable-oriented approach to the research design infeasible. Another common research design available is a case-oriented approach. Many, if not all, case-oriented approaches draw their logical foundations from John Stuart Mill's Method of Agreement or Method of Differences [112]. These represent *comparativist* approaches towards identifying

patterns of invariance or variance across multiple cases. Theoretically, if one is able to identify two or more cases with the same outcome, that have conditions in common (Method of Agreement: $ABCD \rightarrow Y$ and $AEFG \rightarrow Y$, A is the cause of Y), then the circumstances on which the conditions agree are the phenomenon of interest. Or if one is able to identify two or more cases with different outcomes, but only one condition not in common (Method of Difference: $ABCD \rightarrow Y$ and $BCD \not\rightarrow Y$, A is the cause, or partial cause, of Y), then the circumstances in which the conditions disagree are the phenomenon of interest [113]. However, both methods are confounded by multiple causation (or in Mill's words "plural causation"[112]). Consider the situation where lock-in results from either a poor technical architecture or poor intellectual property architecture, the Method of Agreement will incorrectly conclude that neither condition yields the causal outcome because causation varies. In a similar example, the Method of Difference is confounded in situations where poor intellectual property architecture and/or poor technical architecture both cause intellectual property lock-in, as there may be instances where lock-in is caused by intellectual property architecture absent of technical architecture and vice versa. In such cases the Method of Difference would likely conclude that intellectual property architecture *or* technical architecture cause lock-in (not an entirely incorrect conclusion), but would not identify the combinatorial variations of intellectual property architecture *and* technical architecture that yield lock-in.

6.1.1 Qualitative Comparative Analysis

Either approach in isolation is confounded by one or more aspects of multiple conjunctural causation. In such situations, social scientist Charles Ragin argues for a synthetic approach that combines aspects from case-oriented and variable-oriented approaches, which led to Ragin's development of a synthetic class of inquiry broadly termed Configuration Comparative Methods or more specifically Quantitative Comparative Analysis (QCA) [113]. The relationship between methodologies and trade-offs is illustrated in Figure 6-1

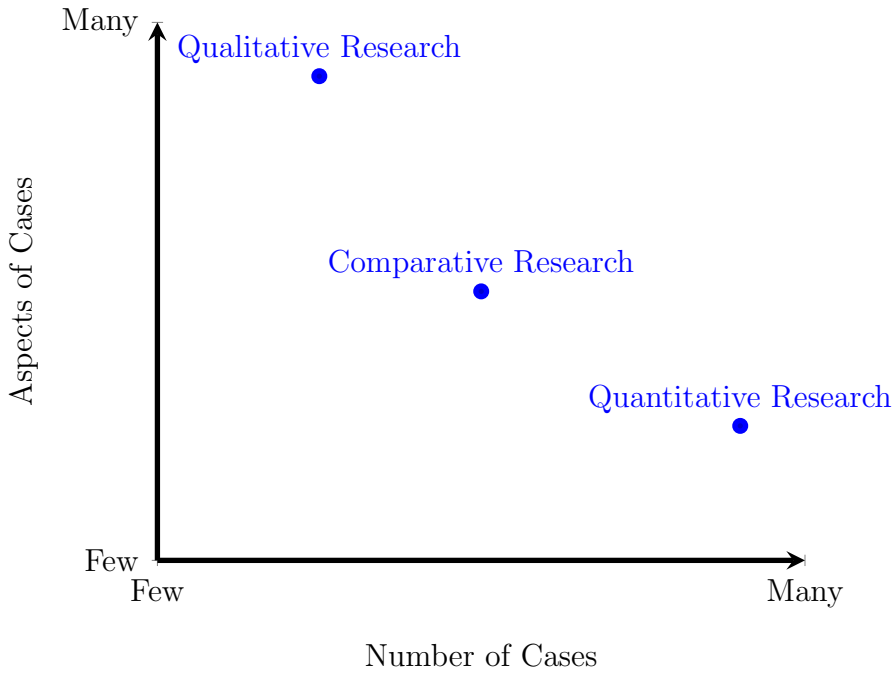


Figure 6-1: Theoretical differences between research approaches and case selection. (Reproduced from [114, p. 49])

Over the subsequent decades, the development of QCA, as a method, was furthered by Ragin [114]–[116] and Rihoux [117]–[120]. QCA adopts some of the key strengths of the qualitative approach, in that each case is considered as a complex entity (i.e., a whole), not just a collection of variables. Additionally, it develops a conception of causality which allows for complexity [121]. When using QCA a researcher does not need to specify a single causal model which fits the data best (as is done in many statistical techniques), but rather to "determine the number and character of the different causal models that exist among comparable cases" [113, p. 167]. Additionally, QCA strives to meet some of the key strengths of the quantitative approach. First, it allows for the analysis of more than one or two cases and ultimately a conclusion which is, under certain conditions, generalizable. Second, it relies on a rigorous set of quantitative tools (Boolean mathematics) from which to draw logic-based conclusions and reduce cases into a series of explanatory conditions. Finally, the use of various Boolean techniques aimed at reducing complexity in logical inquiries allows for conclusions which are parsimonious [121]. Although a relatively

recent method of inquiry, reviews of QCA literature have noted a "signifigant growth" in both the quantity of QCA research (313 peer reviwed articles between 1984 and 2011) and domain diversity (top three diciplines: political science: 51%, Sociology: 34%, and Economics: 26%) [122].

Within QCA there are three methodological approaches: crisp-set QCA (csQCA), multi-value QCA (mvQCA), and fuzzy-set QCA (fsQCA). Each of these approaches uses the same theoretical foundations, but execute the approach of that foundation in different ways. In csQCA each explanatory condition is dichotomous representaion of set membership ("1" signifies membership, "0" signifies nonmembership). Cases are combined across these dichotomous variables and reduced using Boolean algebra to reach a parsimonious conclusion [123]. However, problems arise in csQCA due to the information loss associated with dichotomization. Conditions that may be only somewhat similar are given the same dichotomous value, which leads to cases with identical conditions, but conflicting outcomes. Multi-value QCA deals with this information loss by removing the dichotomy requirements and allowing for a multi-value category ($X\{0, 1, 2\}$ instead of $X\{0, 1\}$). Although a small adjustment, this extension of csQCA into mvQCA reduces possible conflicts and keeps the sample space small [124]. However, researchers argue the concept of mvQCA violates the assumptions of set theory and pose long-term questions on the validity of mvQCA [125]. The final approach is fsQCA, which draws upon fuzzy-set theory, a well-developed mathematical system of addressing partial membership in sets, to analyze conditions along a continuous interval scale (between 0 and 1) [126]. Although this method is more challenging computationally, it addresses many of the limitations of csQCA and mvQCA. Further, fsQCA is particularly well suited for conditions which are not easily organized into a dichotomy or multichotomy, as it permits researchers to preserve natural patterns in raw data by allowing partial set membership.

6.1.2 Model Specification

Among the most central aspects of QCA research is selection of conditions [127]. These are the conditions, which in conjunction with another, are theorized as the

cause of the outcome under study. Amenta and Poulsen outline five methods for selection of conditions, one of which is the *conjunctural theory* approach [128]. This approach exploits QCA's ability to "produce conjunctural results that are causally heterogeneous ... [and] allows the researcher to take advantage of QCA's hypothesis-testing function"[128, p. 29]. Specifically, this feature of QCA isolates individual theories of causality for appraisal by hypothesizing what combinations are expected, then evaluating which were found, which combinations are unexpected by theories, and which were found or not found [113]. This method of condition selection was chosen because it allows for the most explicit testing of the postulated relationship between technical architecture and contract structure (operationalized as intellectual property architecture). Therefore, at a minimum, two conditions (technical architecture and intellectual property architecture) are required, but are not enough for an accurate hypothesis test. As Chapter 2 recounted, much of intellectual property statute within the DoD revolves around the concept of rights. To exclude this condition would yield an insufficient explanatory model; thus, unlimited rights is a required third condition.

The research design chosen to address the research questions, herein, is a fsQCA model, see Table 6.1. The exact calibration of each condition is covered in full detail in Chapter 8; what follows herein is a discussion of the operationalizaiton of each condition. The first condition is fuzzy representation of quality technical architecture. Fuzzy-set membership values close to "1" indicate membership in the set Quality Technical Architecture. The fuzzy membership scores are extrapolated from the core

Table 6.1: Design of fsQCA Methodology

	Inputs			Outcome
	Technical Architecture	IP Architecture	Unlimited Rights	Lock-in
$Case_1$				
$\vdots$	$TA\{0 \leq x_i \leq 1\}^1$	$IP\{0 \leq x_i \leq 1\}^2$	$UR\{0,1\}^3$	$LI\{0,1\}^4$
$Case_n$				

[1] Indicates membership in set of Quality Technical Architecture
[2] Indicates membership in Set of Accessible IP Architecture
[3] $UR\{1\}$: Unlimited Rights and $UR\{0\}$: Not Unlimited Rights
[4] $LI\{1\}$: Locked-in and $LI\{0\}$: Not Locked-in

size of each case's architecture. For reasons discussed in Chapter 5, a lower core size is associated with higher quality technical architecture. Conversely, fuzzy membership scores close to "0" represent nonmembership in the set of Quality Technical Architecture. The second condition, intellectual property architecture, is operationalized in an almost identical manner to quality technical architecture, where values close to "1" represent membership in Accessible Intellectual Property Architecture and values close to "0" represent nonmembership. The fuzzy-set membership scores of this set are extrapolated from the intellectual property architecture measurements outlined in Chapter 5. The final input condition is unlimited rights; this condition captures the dichotomous representation of whether the DoD acquired an unlimited right at the outset of the contract ($UR\{1\}$) to the software or did not acquire an unlimited right ($UR\{0\}$). It is important to note, that this is a binary representation of membership in the set of cases where the DoD acquired an unlimited right to the software code. It does not indicate anything about other classes of rights (i.e., $UR\{0\} \not\rightarrow$ Restricted Rights, $UR\{0\} \not\rightarrow$ Government Purpose Rights). Finally, the outcome condition is a dichotomous representation of intellectual property lock-in, where $LI\{1\}$ indicates a case is locked-in and $LI\{0\}$ indicates a case is not locked-in. Although this research will study the cases where lock-in was avoided (expressed as: 1-LI or verbally as "avoidance of lock-in"), the condition LI is maintained because it is conceptually easier to understand than "not locked-in," which runs the risk of being interpreted as a double negative.

The choice to use fsQCA over csQCA and mvQCA was primary to maintain a parsimonious set of conditions. This same research approach could be mirrored with a csQCA approach using dicotimous indicators for each state of technical architecture (i.e., $Hierarchical\{1,0\}$, $Core\ Periphery\{1,0\}$, $Degraded\{1,0\}$). However, this five condition dichotomous configuration yields 32 combinatorial outcomes ($2^5 = 32$). Marx and Dusa demonstrated that, given a particular sample size, there is an upper limit on the possible number of conditions [129], [130]. For example, the five condition csQCA model specification risks generating an explanatory outcome even on random data, if the ratio between conditions and cases is out of proportion[1](this is similar

to the concept of degrees of freedom in conventional statistics). Similarly, the same research approach could be mirrored with mvQCA using a multichotomy for technical architecture (i.e., $TechnicalArchitecture\{2,1,0\}$, $IntellectualProperty\{1,0\}$, $UnlimitedRights\{1,0\}$). However, this 3 condition configuration yields 12 combinatorial outcomes ($3 * 2^2 = 12$). Fuzzy-set QCA models control condition space by using a partial set membership to reduce the number of combinatorial outcomes. The fsQCA model specification in Table 6.1 yields 8 combinatorial outcomes ($2^3 = 8$) and covers the same "conditions space". In reducing the number of potential condition states, it subsequently minimizes the likelihood of an ill-fitting explanatory outcome [114], [127], [130].

6.2 Case Selection

To explore the research questions, an intermediate-N ($10 < n < 20$) fsQCA case study design was chosen with a mix of qualitative and quantitative methods. In the most basic form, a case study is a detailed examination of a historic episode used to develop or test explanations [131]. Using case studies to investigate the relationship between architecture and intellectual property, allowed for inductive theory development and identification of generalizable conclusions.

6.2.1 Sample Size

The first task in case study research is selecting the number of cases. Although, the research, herein, will use an intermediate-N fsQCA design, it is still important to justify why a single or small-N case study structure was not appropriate. Selection of case size is largely a theoretical argument between breadth and depth. A large number of cases allow for more sample space coverage and potential stronger generalizations, but does so at the cost of contextual details. Whereas, a small number of cases allow for a deep contextual understanding of each case, but it lacks enough

[1]Marx and Dusa find a 5 condition csQCA model would need 17 cases to protect against fitting an explanatory outcome to random data [130].

sample space coverage to generate conclusions that are generalizable. Prominent case study researchers argue the merits of both the single case approach, which is particularly useful for testing an existing theory or if the case has singular revelatory characteristics [132], and also a multi-case approach; arguing that multiple cases allows for replication and a subsequent increase in generalizability [132]. Others argue case selection is a trade-off between the quantity a researcher *can study*, *how* the research is studied, and the *relevance* of the information to the research question(s) [114].

An intermediate-N was chosen due to the breadth of available DoD software cases. The single or small-N case approach was ruled out because there is not a single type of architecture or intellectual property common across the entire DoD portfolio. It would be difficult, if not impossible, to identify a single critical case or small number of cases that covered all possibilities. The analysis of such an extreme case would most likely not yield generalizable results. Additionally, and most compelling, given the QCA research design proposed above, Marx and Dusa found that at least 12 cases were needed to secure a safe threshold against fitting an explanatory outcome to random data [130]. Therefore, this research employed an intermediate-N holistic research design [132]. Eisenhardt and Graebner argue case study research should not try to be representative of a sample population [133]. It is important to note the number of cases selected for this research is not a direct attempt at representing the sample population, but instead representing the possible causal mechanisms that link architecture and intellectual property lock-in.

6.2.2 Sampling Methodology

With case sample size determined, the task becomes case selection. For this task Eisenhardt and Graebner argue for a theoretical sampling methodology, which selects cases because they are particularly illuminating [133]. Additionally, Yin argues theoretical sampling for multiple case research designs should include or exclude cases for reasons such as replication, extension of theory, contrary replication, and elimination of alternate explanations [132]. Comparative researchers take a different approach

and argue for a classical theoretical sampling methodology; Most Different, Similar Outcome (MDSO) [113], [114], [127], asserting this design allows for testing of hypotheses on a "somewhat larger universe showing ... 'external validity' " [127, p. 23]. Ultimately the MDSO approach was adopted because it offered the greatest potential to yield generalizable results. However, these methods proved infeasible for direct application to the research herein. The primary issue is that objective software architectures are not known *a priori*. Therefore, it is impossible to select cases because of a particularly unique architecture or intellectual property characteristics. Each case needed quantitative analysis before a theoretical sampling decision could be made. Therefore, the sampling methodology used herein was a two-step process. The first step used a convenience sample to collect as many DoD software code bases as possible. The only criteria used in this convenience sample was that the code base had to be part of a DoD contract. In total, 37 cases were collected in the convenience sample. After collecting a sufficiently large convenience sample, the architectures and intellectual property architectures were analyzed using the methods outlined in Chapter 5. The second step theoretically sampled the collected cases to select those that were unique and/or illustrative of the postulated relationship between architecture and intellectual property. Prior to theoretically sampling for illustrative phenomena, cases with less than 100 files, or classified, were removed from the sample. Additionally, up to three different software versions were sampled for each case (reflected in the number of cases in the convenience sample); however, only the most recent version was used in the ultimate theoretical sample. Table 6.2 contains a list of the final cases and summary statistics for each case; Figure 6-2 illustrates the sample distribution of cases by technical architecture and size. A more elaborate discussion of each case and contextual details takes places in Chapter 7.

A critical aspect, to an effective MDSO methodology, is homogeneity of other explanatory conditions which are not under study. The choice to keep the case selection to only DoD cases recognizes and conforms to the tenants of MDSO. Remaining within the already established DoD program boundaries pairs-down the number of cases and ensures regulatory, as well as statutory, homogeneity among cases. Fur-

Table 6.2: Description of Selected Cases

System	Size (files)	Dependencies		LOC[1]	SLOC[2]
		Direct	Indirect		
N	24,740	140,055	15,578,401	8,851,927	4,297,613
A	21,879	189,377	154,670,489	8,254,156	5,670,841
B	18,899	323,267	6,742,880	4,530,622	2,783,818
C	6,361	52,274	2,883,437	1,257,869	685,889
D	4,247	18,153	320,172	1,276,114	970,597
E	3,912	13,929	119,016	6,693,882	6,111,764
F	2,869	15,271	317,044	879,115	589,276
G	1,688	10,642	199,571	820,781	399,731
H	1,497	6,860	42,205	253,000	206,579
M	1,466	11,639	896,023	261,023	162,880
I	1,464	6,243	102,614	3,280,227	3,131,633
J	737	3,155	25,185	188,579	119,212
K	497	3,567	107,180	428,623	232,328
L	374	1,525	4,871	69,490	41,783
Totals	90,630	795,957	182,009,088	37,045,408	25,403,944

[1] Lines of Code (LOC)
[2] Source Lines of Code (SLOC)

thermore, the selected programs are from commensurate levels of technology maturity (i.e., not comparisons between programs in research and development against programs in production). Finally, both Geddes [134] and Yin [132] admonish against selecting cases based on the dependent variable; it is important to note that the cases set forth in Table 6.2 were not selected because they demonstrated characteristics of lock-in, but rather because they demonstrated an illustrative combination of architecture and intellectual property.

6.3 Qualitative Methods

Qualitative data collection methods empower theory building and increase exposure to alternate hypotheses. The primary means of qualitative data collection was semi-structured guide approach interviews. Sampling of interview subjects followed the recommendations of Eisenhardt and Graebner [133], which recommends collecting samples from different hierarchical levels, functional domains, and geographic loca-

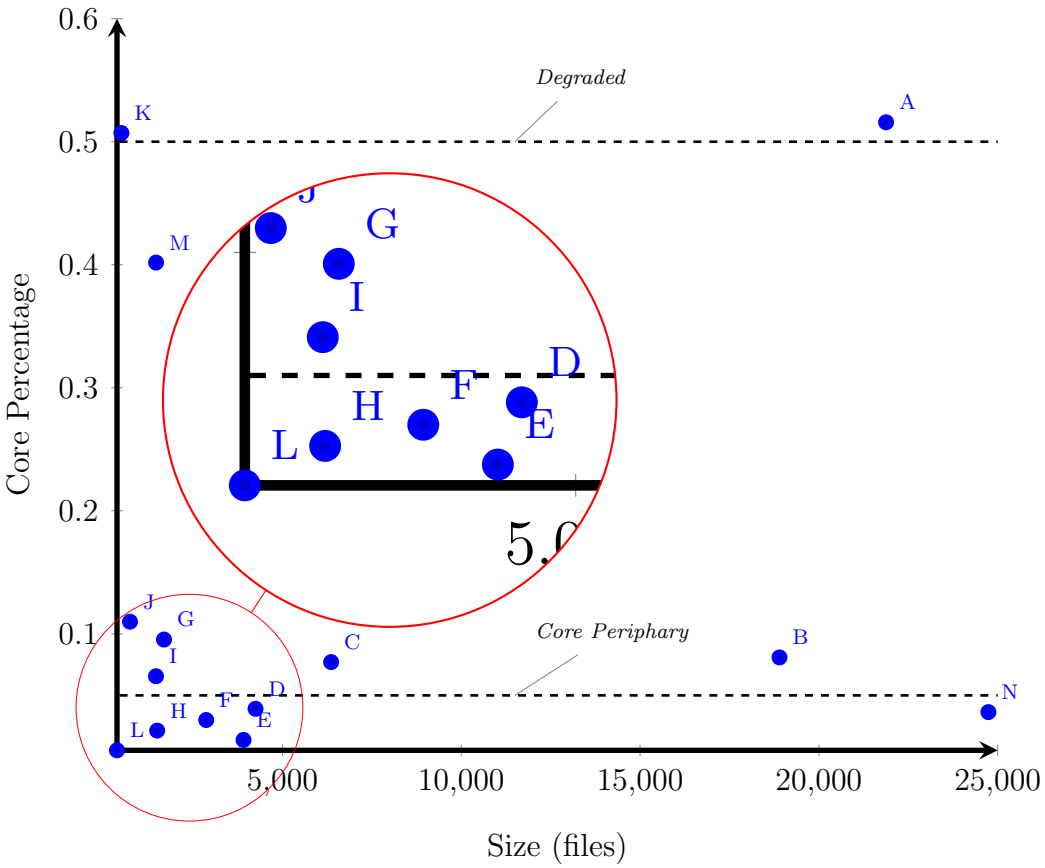


Figure 6-2: Case Sample Scatter Plot

tions to control for "retrospective sense-making" or localized biases. Where possible, Eisenhardt and Graebner also suggest interviewing subjects outside of the organization under study [133]. Consequently, subjects were interviewed from both engineering, management, and legal roles within selected DoD organizations, as well as from defense contractors for an external perspective.

In addition to interviews, archival documentation from each program was collected, where practical (i.e., not classified or proprietary). The document collection focused on each program's Acquisition Strategy and Acquisition Plan as well as each program's long-term assessment of technical data needs. Although such plans are required by statute, in many cases the documentation did not exist or was lost.

Interviewee Classification	Number of Interviews
Subject Matter Expert Interviews	
Legal	6
Engineering	3
Science	4
Policy	8
Logistics	1
Lobbyist	2
Case Specific Interviews	
Program Manager	15
Engineer	6
Oversight	3
Programmers	11
Total Interviews	59
Total Hours	71

Table 6.3: Summary of Interviews Conducted

6.3.1 Qualitative Data Collection

Table 6.3 summarizes the interviews which were conducted to build theory and case knowledge. The top half of the table lists the subject matter expert interviews conducted. These individuals are not tied directly to any of the cases researched; rather, they are considered experts on DoD intellectual property regulations and statute. The sample of subject matter experts was drawn from both within the DoD and private industry. The initial set of interviews were set-up using the list of members from the Section 813 Panel referenced in Chapter 2 and subsequent interviewees were identified using a snowball sampling method. In conducting interviews, many private industry parties were unwilling to be interviewed for this research. Consequently, industry lobbyists were interviewed to solicit the positions of private industry.

The bottom half of the table summarize case specific interviews. These are the individuals whose primary job it is to execute or administrate one or more of the cases under study. This sample represents primarily individuals within the DoD. Initial interviews were set up by contacting the lead engineer of each organization and using a snowball sampling methodology to identify the most knowledgeable manager(s) for

each case. In some cases up to 4 individuals were interviewed and in others only a single interview was necessary.

Interview data was treated as sensitive throughout this study. Many subject matter expert interviewees preferred to speak candidly under anonymous attribution. In the interest of protecting their anonymity, the discussions herein do not attribute interviewee remarks to a specific individual or job title. The community of experts on DoD intellectual property is small ($n < 30$); therefore, attribution is possible unless interviewees are referred using only ambiguous terms. Where possible, every effort is made to refer to the interviewees position in the hierarchy, domain, and affiliations (i.e., Senior Government Official in Acquisitions Policy or Private Industry Head Intellectual Property Attorney), but avoid more specific attribution.

For the subject matter expert interviews, each interview was tailored to the interviewee's specific domain expertise and hierarchical position. Interviews were accomplished using a semi-structured interview guide approach [135] and began with a basic explanation of the research as well as the researcher's background. Interviewees were initially asked about their educational background and professional experience, which permitted tailoring of the interviews. The length of each interview ranged from 1 - 1.5 hours. Appendix B provides a template of potential research questions used to guide the interviews.

For the case study interviews, each interview was tailored to the characteristics of the case. Interviews used a semi-structured guide approach to navigate the contextual details of each program, but a list of interview questions was not used. The choice not to use defined interview questions is a reflection of the contextual variance in each case. For example, Case A has a history which dates back to the 1980s, but Case G is only a few years old. Only three questions were uniformly asked across all cases:

1. Are you familiar with the concept of intellectual property lock-in?
2. Is the software in your program experiencing intellectual property lock-in?
3. Does the program office have an unlimited right to the software code?

The answers to these questions served as the basis for the unlimited right input condition and the outcome condition in the fsQCA model. If a program manager or

engineer answered "yes" to question two, the case was coded as $LI\{1\}$ or $LI\{0\}$ if the answer was "no". In all cases, the program managers were familiar with the concept of intellectual property lock-in. Similarly, an answer "yes" to question three was coded as $UR\{1\}$ or $UR\{0\}$ for a "no" answer. If the interviewee was unsure of the answer to the question, a snowball sample was taken to identify individuals who would know the answer. Using the snowball sample, interviews continued until an individual could address the questions. Interviewee answers to the questions above were never contradicted based on opinions of the researcher; however, if the researcher felt the empirical evidence of the case and the answers, to the questions above, conflicted a snowball sample was collected and other individuals were interviewed to help triangulate the results. An example of this occurred in interviews for Case M. The program manager for this case had been on the job less than six months and was not intricately familiar with the previous contract. When asked questions one and two, the program manager said the program was locked-in, but equivocated about the level of lock-in. To make sure the case was coded correctly the engineer for the case was also interviewed and confirmed that the answer was correct.

6.4 Quantitative Methods

The quantitative methods used in this research are discussed at length in Chapter 5. There is no need to recount the details of the quantitative framework, but a discussion of quantitative collection processes is necessary. This section focuses on processes of quantitative inquiry which are relevant to the validity of the research design and specific steps taken to reduce threats to validity.

6.4.1 Quantitative Data Collection

Although the basic framework for quantitative analysis used, herein, involves seven steps; the first two steps are data collection with the remainder being data manipulation:

1. Capture a network representation of software source-code

2. Capture all copyright information from software using regular expressions
3. Find all the paths (direct and indirect) between files in the network
4. Calculate visibility scores to each file
5. Organize files into one of four canonical groups based on visibility scores
6. Visualize the architecture
7. Visualize the overlay of copyright data on the architecture

To capture a network representation of each case the SciTool's Understand static code analysis tool was employed through a Python API. Since the majority of data is collected in this step (all files and dependencies), a validation check was used to ensure accuracy. Each potential code base was scanned using two different computers. This additional extraction was done to make sure the parameters in the static code analysis tool were properly configured; thus, reducing the likelihood of an instrumentation error. In addition to the system level metrics (i.e., number of files and connections between files), six file-level metrics are observed and an additional five are computed, see Table 6.4. The aggregation of the file-level data is the foundation for the system-level metrics discussed in great detail in Chapter 5

The copyright information is extracted using regular expressions, Appendix C contains a listing of all expressions used. To build and fine-tune the copyright extraction program a code base was scanned and then a random sample of files were hand coded to identify false negatives and false positives. Based on the results of the hand coding, changes were made to the regular expressions to reduce false negatives and positives. This iteration cycle continued until no classification errors were found in the random sample of files. The same iterative procedure was applied to three different code bases and iterated until no false negative or false positives were identified in the random sample. The copyright extraction collection process produces four file-level variables, see Table 6.4 for a description.

After quantitative data collection concluded for a specific case, the results were compiled and shown to the software programmer or engineer responsible for the case. Feedback was solicited on the accuracy of the observed variables and copyright variables. This informal qualitative triangulation methodology ensured that the file-level

Variable	Description
	Observed Variables
Filename	Location of file within directory structure
Language	Language of code in file (e.g. C++, C#, Java, Python, etc.)
Fan-In	Number of incoming direct dependencies
Fan-Out	Number of outgoing direct dependencies
LOC	Lines of Code
SLOC	Source Lines of Code
	Computed Variables
Visibility Fan-In	Number of incoming direct and indirect dependencies
Visibility Fan-Out	Number of outgoing direct and indirect dependencies
Cyclic Group ID	A rank order of cyclic group size for which file is a member
Cyclic Group Size	Number of files in cycle group which file is a member
Arch Type	Architectural location of file (i.e. Core, Control, Shared, Periphery)
	Copyright Variables
Owner	Name of copyright holder(s)
License	Type of copyright license(s) (i.e. BSD, MIT, GPL, etc.), if provided
Open Source	Binary classification
Year	Year(s) of copyright

Table 6.4: File-Level Variable Descriptions

metrics mirrored programmer perceptions of the system and were essential in identifying instrumentation errors early in the development of the quantitative analysis framework.

6.5 Threats to Validity

Campbell and Stanley outline twelve factors which threaten the validity of various experimental designs [136]. Beginning with internal validity, the largest single threat to the research approach outlined in this Chapter is instrumentation. The instrumentation threat to internal validity is concerned with miscalibration of measuring instruments (intentional or not). There are three points within this research approach where instrumentation error is possible:

1. Technical architecture extraction
2. Copyright extraction
3. fsQCA model specification

Each of these areas of potential hazard are addressed in the research approach. First, Section 6.3 addressed steps taken to ensure proper calibration of the technical architecture extraction and copyright extraction. These steps included use of multiple instruments, test of instruments against a control, and a qualitative triangulation of instrument results. The combination of these strategies does not completely remove threats to internal validity from instrumentation, but does reduce the risk to an acceptable level. Second, Section 6.1.2 outlined the steps taken to ensure proper calibration of the fsQCA model in accordance with researched parameters [129], [130]. However, it is important to point out that the cited research established parameters are for csQCA models. These ratios should extend to fsQCA models, but this was not a direct finding.

Other threats to internal validity exist, *selection bias* and *testing*. Although selection bias is traditionally concerned with test or control groups in experimental and quasi-experimental designs, there are obvious implications for any sample drawn using a theoretical sampling methodology. However, most of the selection bias threat is controlled through the use of an intermediate-N sample size and the QCA approach. The threats from testing are less obvious. The copyright extraction method only works if the copyright notices are unmolested. It is possible, albeit not probable, that a party may tamper with or remove copyright data before scanning. This testing interaction would result in a scan that returns many false negatives. There were no explicit steps taken in the research approach to control for this threat to internal validity. However, willfully removing a required copyright introduces many legal hazards, which should act as a sufficient disincentive to reduce the likelihood of occurrence.

Although Campbell and Stanley argue "the question of *external validity*, like the question of inductive inference, is never completely answerable"(original emphasis) [136, p. 5], it is still necessary to discuss any threats. The largest threat to external validity is alternate explanations of lock-in. The model used, herein, theorizes the explanation to be a conjunction of technical architecture, intellectual property architectures, and unlimited rights; however, other plausible explanations exist:

1. Failure to take delivery of technical data

2. Failure to adequately plan for long term intellectual property needs
3. Failure to enforce rights to intellectual property
4. Gross mismanagement of intellectual property needs

To control for this threat the literature review in Chapter 2 took a retrospective look at intellectual property statute history and captures many alternate explanations. Additionally, elicitation of alternate explanations for intellectual property lock-in was one of the primary objectives of the subject matter expert interviews. Each case was screened for the alternate explanations collected from both the retrospective analysis and subject matter expert interviews. Although subsequent detailed case analyses may reveal an alternate explanation exists, they were not obvious during case selection.

6.6 Conclusion

This Chapter drew from previous Chapters to outline a research approach to investigate the research questions and hypotheses. The benefits of a qualitative and quantitative approach were discussed, but the confounding effects of multiple conjunctural causation led to the adoption of a fuzzy-set Qualitative Comparative Analysis approach using three conditions, technical architecture, intellectual property architecture, and unlimited rights. The condition selection methodology was discussed in detail along with the case sample size and sampling methodology. Concluding, that an intermediate-N theoretically sampled fsQCA design was most appropriate given the nature of the research questions.

The second half of the Chapter discussed qualitative as well as quantitative data sources and collection methodologies, with a particular focus on the collection processes. Discussion of processes was important to identify how threats to internal and external validity were controlled. The Chapter concluded with an explicit discussion of the greatest threats to both internal and external validity and an outline of the steps taken to mitigate any impact.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 7

Contextual Case Summary

The intent of this Chapter is to provide contextual details for all of the cases included in the fsQCA analysis. These details will include a brief description of the history, architecture, and any case specific obstacles revealed during the interviews. As discussed in Chapter 6, the use of an intermediate-N case study design does not allow for a "thick case" approach to contextual details. This trade-off allows for a broader cross-case analysis, but a more concise discussion of contextual details. Consequently, what follows is a parsimonious discussion of only the relevant contextual details for each case and illustrations of the technical architecture as well as intellectual property architectures for each case. At a minimum, each case discussion will explain any contextual details which were relevant to coding input conditions, if any, as well as discuss illustrative interactions between conditions.

To facilitate an organized discussion, of so many cases, this Chapter is divided into three sections which map to the architectural patterns discussed in Chapter 5: Ideal Pattern, Boundary Pattern, and Hazard Pattern. A summary of the cases and their respective patterns, input conditions, and output conditions are illustrated below in Table 7.1 with a summary of the case metrics located in Appendix D. Discussion of each case will begin with a brief historical summary and transition to a discussion of the salient characteristics of the architecture. In all cases, either an open source architectural view or a copyright owner architectural view is provided. Recall from Chapter 5 that the open source view depicts the technical architecture with a binary

Table 7.1: Summary of Architectural Patterns by Case

Pattern	Case	Inputs			Outcome
		TA[1]	IP[2]	UR[3]	LI[4]
Hazard	Case A	0.52	0.01	0	1
	Case K	0.51	0.01	0	1
	Case G	0.10	0.00	1	0
	Case I	0.07	0.00	0	0
	Case J	0.11	0.01	1	0
	Case B	0.08	0.31	1	0
	Case M	0.40	0.00	1	0
Boundary	Case E	0.01	0.03	1	0
	Case H	0.02	0.00	1	0
	Case F	0.03	0.00	1	0
	Case N	0.04	0.40	1	0
Ideal	Case C	0.08	0.88	0	0
	Case L	0.01	1.00	1	0
	Case D	0.04	0.70	0	0

[1] Core size of technical architecture
[2] Percent of Core or D_8 which is accessible
[3] $UR\{1\}$: Unlimited Rights and $UR\{0\}$: Not Unlimited Rights
[4] $LI\{1\}$: Locked-in and $LI\{0\}$: Not Locked-in

classification for open source files overlaid. Where "True" implies a file carries an open source license and no label implies no open source license. Also, recall, the copyright ownership view overlays the copyright owner on the technical architecture.

7.1 General Context

Many of the cases selected were collected from DoD sustainment organizations[1]. This complicates the analysis, herein, as these cases have acquired sufficient rights to the original software to permit transfer to an DoD sustainment organization. If these programs were analyzed at the current point in time all would be categorized as having unlimited rights. However, this research is interested in the state of each case prior to entering DoD sustainment. At this point each case may or may not have unlimited rights, but would have had to make a judgment about acquiring rights

[1]Cases A, C, E, F, G, H, I, J, K, L, and M all came from DoD sustainment organizations. Cases B, N and D came from program offices which do not utilize DoD sustainment.

in order to transition the sustainment of the code. What follows is a discussion of each program prior to the point of transition to DoD sustainment. For almost all of the cases under study, the transition to sustainment happened one to four years ago. There was sufficient knowledge gathered from the interviews and archival documentation to recreate an accurate state of each case at this point.

It is also important to address the concept of *sustainment*. In the DoD lexicon, this concept refers to the continual process of fixing software defects, as well as making small enhancements. The sustainment model for each case is different, but generally users of the software submit defect reports to the sustainment organization, which then undertakes the effort to fix the defect, test the fix, and then release the fix to the user community. Depending on the case and software maturity, this could be hundreds of defects a month or only a handful. Similarly, a user could request a small feature enhancement through the same process. The line between a small feature enhancement and a development effort is typically based on the judgment of the program management team with input from the software sustainers.

7.2 Hazard Patterns

Hazard patterns were theorized in Chapter 5 as the conjunction of Degraded architecture, Core-Periphery architecture, and Closed IP Architecture.

7.2.1 Case A

Background

Case A is a software system that correlates radar tracks on a large scale; it processes, displays, and manages aircraft tracks as well as sends and receives track information via data links with other organizations. Additionally, Case A is a component under a larger umbrella program, which also encompasses Case K and two other sub-systems. However, the Case A portion comprises over 8 million lines of code and is, by far, the largest component of the system. The program dates back to the 1980's, but the

version of software under study dates back to 2003, when a large defense contractor won the contract for development. In early 2010, the program office began an initiative to transition the sustainment of Case A from contractor to DoD, in keeping with the DoD Acquisition Policy of the time.

Although Case A and K existed long before the attacks on 9/11, the use of commercial aircraft in acts of terrorism sparked a renewed interested in capabilities provided by the program. Interviewees noted, shortly after 9/11, the funding as well as requirements for the program were increased and the expectation was delivery as soon as possible. This rushed development may have attributed to the degraded state of the architecture (discussed below), a sentiment often acknowledged by interviewees.

Case A has a tenuous history with intellectual property rights. The eventual transfer of sustainment to DoD was the byproduct of a four-year disagreement over the Government's right to certain files of source code. At contract award, the contractor asserted the Government did not have a right to Case A because it was developed exclusively at private expense (contractor alleges $537 million in IRAD and international funding). The Air Force disagreed, citing over $223 million in funding for Case A. The contractor eventually revised their initial position, acknowledging the Air Force has an unlimited right to 30% of the source files, a government purpose right to 48% of the source files, and a restricted right to 22% of the source files. The Air Force disagreed with the proportion of files with a restricted right and insisted the contractor provide proof of private funding in accordance with DoD Cost Accounting Standards. The contractor was not able to provide adequate cost information, which led the contracting officer to issue a final determination that those files asserted by the contractor as restricted, in the absence of further proof, are now subject to an unlimited right. This determination allowed the entirety of Case A to transition to DoD sustainment.

It is important to note that the proceedings narrated above are not normal. In this case, the contractor failed to maintain required financial information in accordance DoD Cost Accounting Standards. This oversight permitted the Air Force to make a determination of unlimited rights. Had the contractor complied with financial record

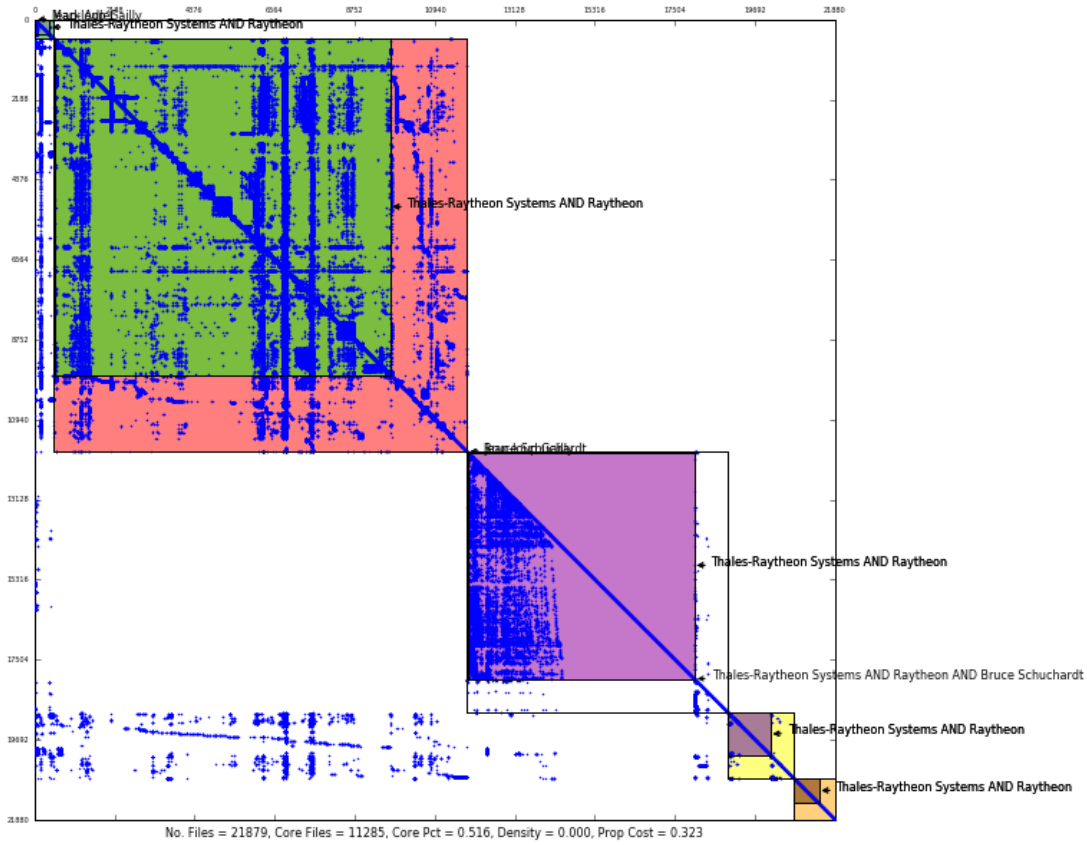


Figure 7-1: Case A Owner View

standards, the result could have been very different. In fact, interviewees acknowledge the outcome was partially due to "luck." Consequently, this case was not coded as a case with unlimited rights. As the DoD made no plans to acquire an unlimited right, but acquired them as a byproduct of contractor mismanagement. Had the mismanagement not occurred, the situation would have been drastically different.

Architectures

Case A has a degraded architecture with a closed intellectual property architecture, see owner architectural view in Figure 7-1. The core percentage of Case A is approximately 51% with 74 files containing permissive-type open source licenses. In addition, approximately 57% of Case A is written in the antiquated Ada programming language. The combination of such a degraded architecture, closed intellectual property architecture, and an antiquated programming language, make a hazard pat-

tern. Although not the scope of this research, Case A is arguably the archetype for a software system that needs a new-start. However, interviewees were quick to illustrate that the architecture of Case A is partially due to a trade-off between complexity and performance. To handle hundreds of radar tracks in real-time requires architectural choices that prioritize performance over design quality. Although rational, interviewees acknowledged this did not totally explain the "messy code base."

The causal conditions for Case A represent a conjuncture between a degraded architecture and a closed intellectual property architecture without unlimited rights resulting in lock-in.

7.2.2 Case K

Background

Case K is a protocol converter which converts Federal Aviation Administration message format radar tracks into the All-Purpose Structure Eurocontrol Surveillance Information Exchange (ASTRIX) format for input into Case A. Case K is only a fraction of the size of Case A, but is instrumental to the overall operation of the system. Beginning in 2003, Case K was won by the same defense contractor as Case A and was developed and sustained by that contractor until transition to DoD sustainment in early 2010. Interviewees noted that Case K, although designed specifically to input into Case A, is a commodity capability. There are many different protocol conversion solutions which offer the same functionality.

Interviewees for both Case A and K commented on the rushed nature of the acquisition. Both cases are part of the larger system which tracks air traffic over North America. After the events of 9/11, the requirements of the program were increased and the schedule for delivery of upgrades was rushed. Interviewees attribute some of the poor architecture to the speed at which it was developed; however, they offered no rationale for why the system was not fixed in subsequent years after initial fielding.

The intellectual property history of Case K is nearly identical to Case A. The only visible difference is the chain of copyright notices. Case K's copyrights began

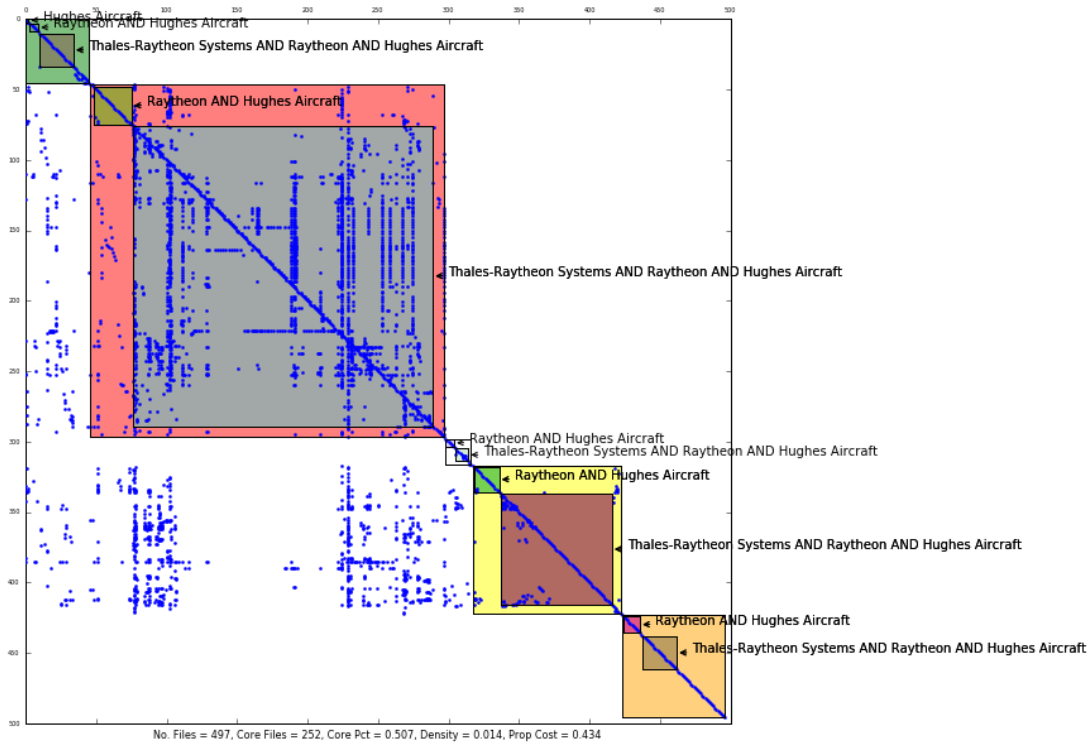


Figure 7-2: Case K Owner View

with Hughes Aircraft Company and transitioned to Raytheon Company and then finally Thales-Raytheon Systems. It was unclear from interviews if this system was acquired by Raytheon from Hughes Aircraft or if it was licensed. Additionally, in contrast to Case A, Case K is written primarily in the C programming language, with approximately 1% of the files written in Ada.

Architectures

Case K has a degraded architecture with a closed intellectual property architecture; see owner view in Figure 7-2. The core percentage of Case K is 50.7%, with only 5 permissive-type open source files. Note in Figure 7-2 the extent of copyright notices in the core. Had proper cost information been recorded for Case K, Figure 7-2 suggests it would have been difficult to transition this software program to DoD sustainment without procuring additional rights to the software because many of the copyright notices predate (1989-1997) the 2003 contract which the unlimited rights decision was based. Although not always a good indicator, the preexistence of a copyright notice

prior to the start of a contract demonstrates some part of the code was created prior to the beginning of the contract. Unless this code was developed under a previous DoD contract, it is likely this code carries a restricted right.

The causal conditions for Case K represent a conjuncture between a degraded architecture and a closed intellectual property architecture without unlimited rights resulting in lock-in.

7.2.3 Case J

Background

Case J is a software that integrates weather information into the Air Force's enterprise mission planning system. The system was developed in the early-2000s by an Air Force rapid acquisition organization to fulfill a capability gap for remotely piloted aircraft and was eventually transferred to a long-term program office in the late 2000s. The program operates with a defense contractor doing software development and DoD assets doing sustainment. This "develop externally, sustain internally" model is used because the defense contractor doing development is also the one responsible for the larger weather enterprise. Therefore, that contractor continues to update Case J on the same cycle as the main enterprise, and subsequent bug fixes or minor improvements are made at a later date by DoD sustainment organizations. For this model to work efficient, the DoD acquires an unlimited right to the software to enable the smooth and continual transition of contractor developed software to DoD sustainment assets.

The software is divided into three major sub-systems. One sub-system for the "handshake" between Air Force weather service and mission planning enterprise, one sub-system for visualization of weather information, and a final sub-system for metadata translation. Case J is unique in that it is one of the few cases that act as an interface between two large enterprises. Consequently, its development cycle, interfaces, and, to a certain extent, architecture is dictated by the design choices in the larger enterprises. This confines the possible architectural design space. For exam-

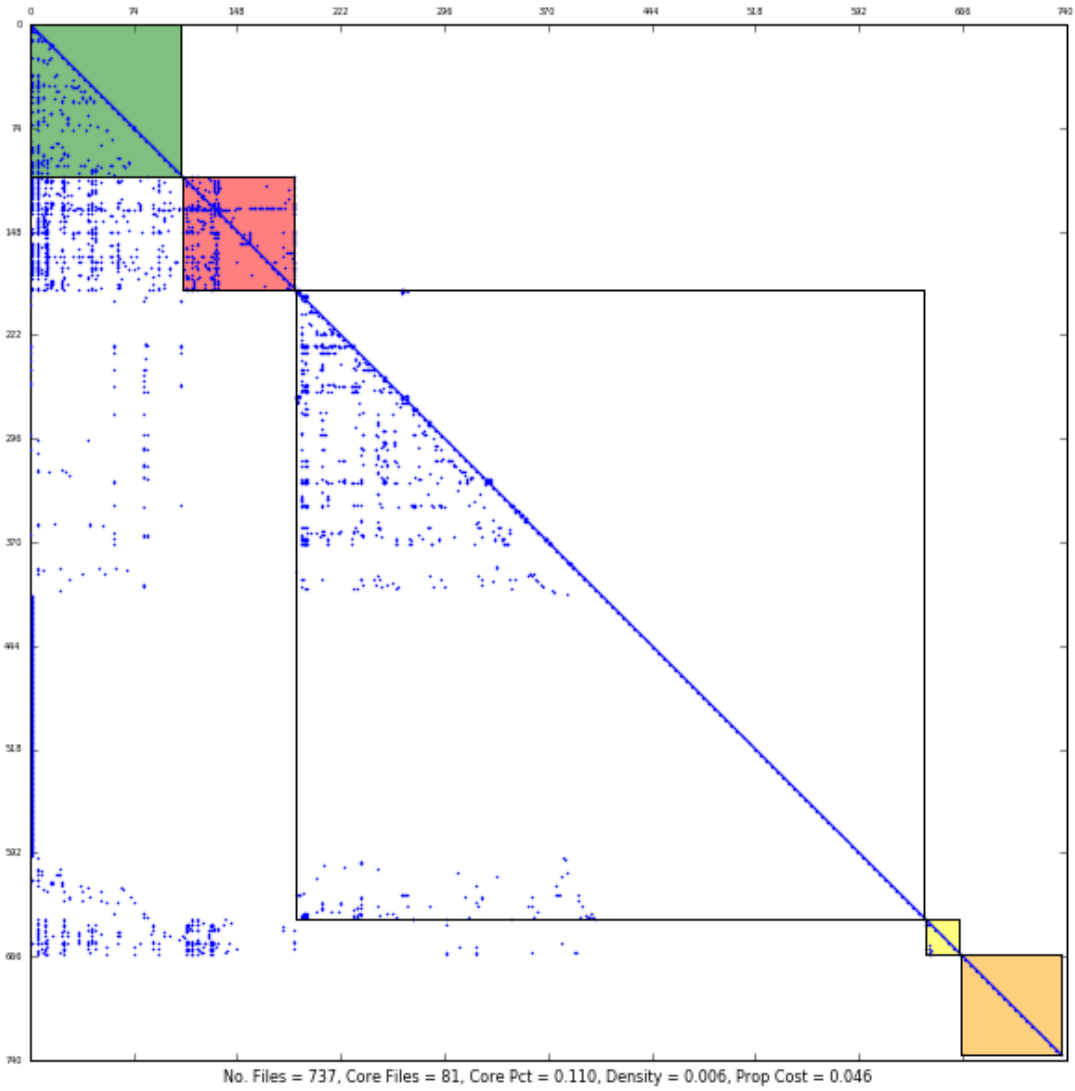


Figure 7-3: Case J Open Source View

ple, interviewees noted the data which comes from the weather enterprise uses the GRIB/GRIB2 standard format, which is a common format in meteorology for storing past and future weather forecasts. This raw message format must be read and parsed for use in the mission planning enterprise. Although a seemingly trivial example, this message format partially defines the input and output interfaces for Case J.

Architectures

Case J has a core-periphery architecture with a closed intellectual property architecture; see open source view in Figure 7-3. The core size of Case J is approximately

10.9%, with only five open source files. All of the open source licenses are permissive-type licenses and of such a small quantity that they have a trivial impact on the intellectual property architecture.

The three sub-system configuration of Case J adds complexity to the architectural evaluation. If the architecture is evaluated at a lower level of abstraction with each sub-system analyzed individually, the largest module, with 421 files in size, would have a core percentage of approximately 19% and the two smaller modules, 188 and 33 files in size respectively, would have hierarchical architectures with nonexistent cores. This sub-system structure is evident in the lower right corner Figure 7-3, where the connections are almost nonexistent. In such systems, an argument can be made for changing the unit of analysis to a lower level of architectural abstraction. However, to stay consistent with the research methodology outlined in Chapter 6, the differences in unit of analysis are discussed, but left for future research.

The causal conditions for Case J represent a conjuncture between a core-periphery architecture and a closed intellectual property architecture with unlimited rights resulting in avoidance of lock-in.

7.2.4 Case I

Background

Case I is a software that ingests and manages personnel information to facilitate personnel recovery operations. The information stored in this system are physical characteristics and descriptive features of each service member, (i.e., height, weight, tattoos, hair color, scars, etc.) as well as preformulated questions and answers entered by the service member to help verify his or her identity. This system started via DoD development in mid-2000. Interviewees noted the program had a "basement" start, implying that it was created by engineers tinkering to solve a problem. This start evolved into a fully funded program which was transitioned to a defense contractor for a short period of time and eventually to DoD sustainment; due to dissatisfaction with the defense contractor. Interviewees noted strong personality conflicts with the

defense contractor, commenting the decision to transition to DoD sustainment was "90% personality and 10% hardware costs."

Recently, Case I undertook an initiative to implement agile software development. Over a year was spent pursuing agile development, but eventually the program was forced to roll-back to sprint-0 due to poor requirements definition at the outset. The interviewee would not enumerate the requirements which were poorly defined, but did generally note the difficulty in assessing how well a contractor was delivering the requirements in an agile development environment. These comments suggest poor systems testing and verification requirements definition. The version used, herein, is post roll-back and does not contain material portions of code developed via agile.

Interestingly, interviewees noted there was no difference in cost between sustaining the system via a defense contractor or organically. Specifically arguing, any efficiency gained in lower cost was lost due to lower experience of DoD developers. It is important to note this sentiment was not echoed by any other cases and could be isolated to the sustainment team for Case I. However, an inspection of the architecture in the next section may explain why the DoD sustainment team is having difficulty.

Architectures

Case I has a core-periphery architecture with a closed intellectual property architecture; see open source view in Figure 7-4. The core size of Case I is approximately 6.5%, with about 147 open source files. Almost all of the open source files carry a permissive license and are located in the peripheral group. This license type implies that there are no provisions which transfer to a derivative work and the location implies that the licenses have no direct connections to the core.

Interestingly, Case I has four cores that are relatively similar in size, indicative of a multi-core architecture. The largest core is 96 files and the next three are 76, 58, and 51 in size. These are indicated in Figure 7-4 as small square groupings of dots above the diagonal. The presence of four similarly sized cores is not an ideal architecture. This is because these cores are all interconnected (single directional links versus bidirectional links), which implies a change in one core could propagate

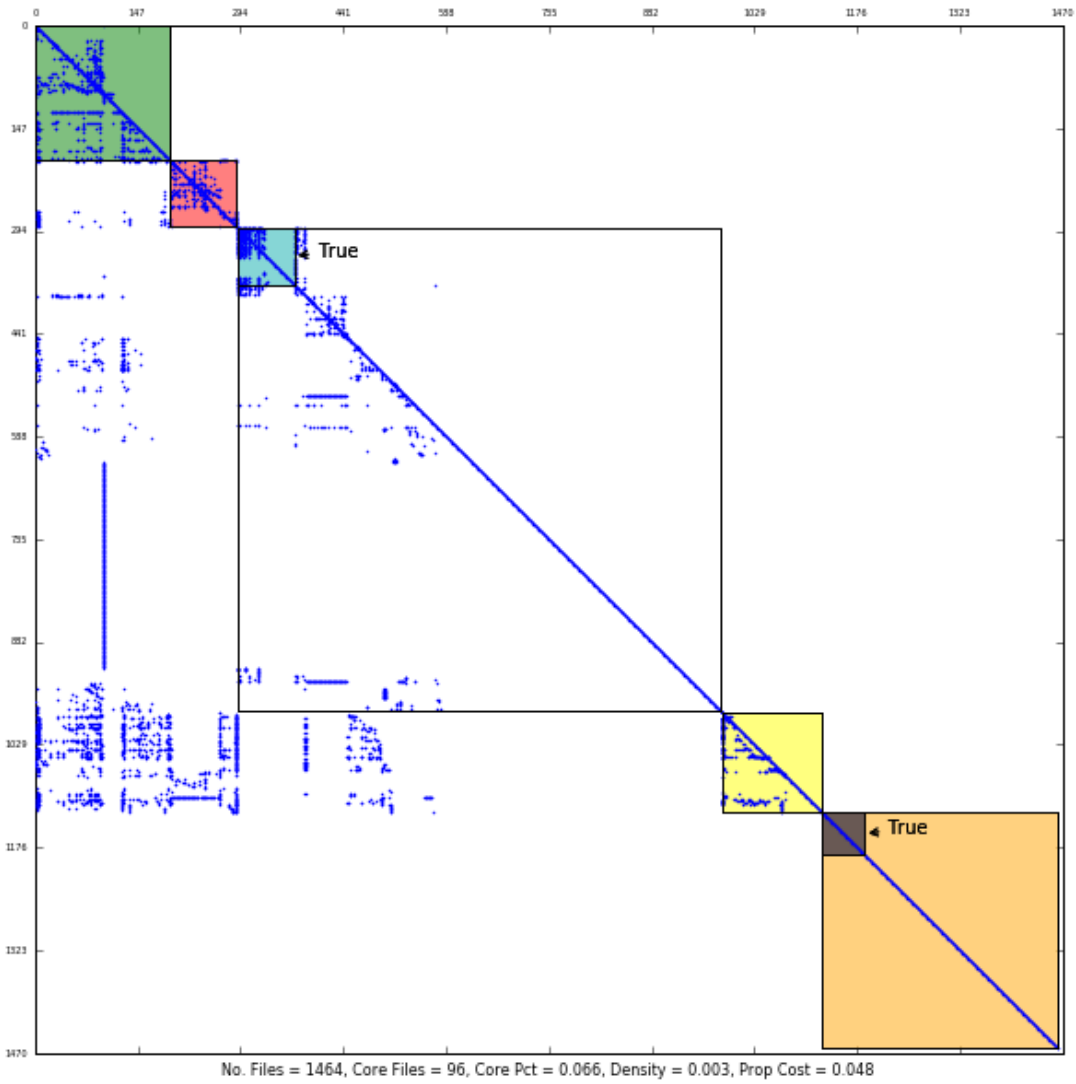


Figure 7-4: Case I Open Source View

to the next. One could make an argument for summing up the size of the similar cores (281 files) and using this new number to compute the relative core size (resulting size would be 19.2%). This change would push this system closer to the degraded threshold as opposed to core-periphery. However, the result would still be a Hazard architectural pattern regardless of the reclassification. Although this discussion is in the context of coding case studies, the impact on practice is a core size that is likely much larger than the reported 6.6%. This may partially explain why the DoD sustainment team is having difficulty, particularly since they were not the original organization who built the system. Though not the focus of this research, time spent

reducing the number of cores or core sizes, would likely improve the performance of the sustainment team.

The causal conditions for Case I represent a conjuncture between a core-periphery architecture and a closed intellectual property architecture with unlimited rights resulting in avoidance of lock-in.

7.2.5 Case G

Background

Case G is a tool suite that allows aircrew to conduct detailed mission planning for the full spectrum of possible missions. The data generated using this tool is then saved and uploaded to the aircraft for execution during a mission. Case E, G, and H are all instances of mission planning software, but differ in the specific aircraft for which the software is designed. Furthermore, Case G and H are mission planning software for different versions of the same aircraft. Although there are many similarities in the origins of the mission planning cases, the architecture of each is influenced by the required interfaces to the aircraft and design choices made in early development. Additionally, all mission planning cases (E, G, and H) carry an unlimited right, which was acquired by the program office at the outset to ensure long-term competition.

Similar to the other mission planning cases analyzed herein, Case G began in the aircraft program office and was transitioned to a consolidated mission planning program office in the late 2000s. During that program office transition, the sustainment for Case G was shifted from contractor sustainment to DoD sustainment. Interviews revealed the choice to switch sustainment organizations was "not a factor of satisfaction, but of contracting convenience." The interviewee's comment about convenience speaks to the difference in process between forming a contractual relationship with a DoD organization and a defense contractor.

The result is three programs which carry out the same functionality, but with three different architectural patterns. Interviewees acknowledged the architecture of Case G was not ideal, referring to it as "spaghetti architecture." A near-term goal

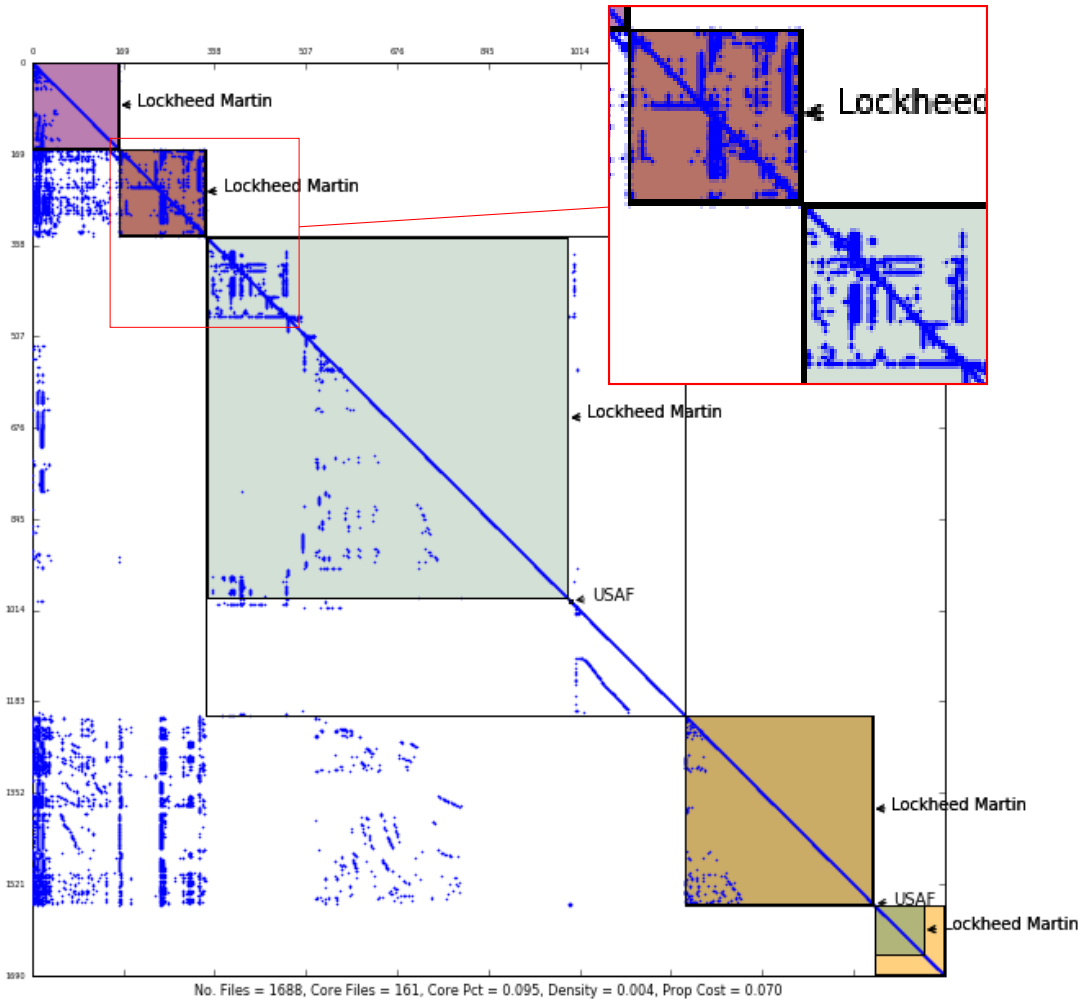


Figure 7-5: Case G Owner View

of Case G is to refactor the architecture to closer resemble that of Case H; ergo, moving from a core-periphery architecture to a hierarchical architecture. In changing the architecture interviewees believe there will be more opportunity for code reuse between Case G and H as well as lower the technical burden on new programmers.

Architectures

Case G is a core-periphery architecture coupled with a closed intellectual property architecture; see owner view in Figure 7-5. The core percentage of Case G is approximately 9.5%, with no open source software. However, Case G is an example of a multi-core architecture, which is evidenced by the zoomed inset in Figure 7-5. The

largest cyclic group is the core, which contains 161 files, and the second largest cyclic group is located in the periphery, which contains 125 files. This relatively rare form of technical architecture does not carry any special implications for lock-in because of the closed intellectual property architecture. However, had there been open source code, in this case, both cores would need to be examined to determine whether this case was an accessible intellectual property architecture.

The causal conditions for Case G represent a conjuncture between a core-periphery architecture and a closed intellectual property architecture with unlimited rights resulting in avoidance of lock-in.

7.2.6 Case B

Background

Case B is a decision support tool designed to aggregate data as well as generate courses of action for military and civilian exigencies. The tool is designed to gather information and present a common operating picture to commanders as well as generate courses of action to assist with mission planning. The contract for the program began in 2004 and was won by a large defense contractor. This contractor handled development efforts until 2013 when software sustainment was competed for and won by a different defense contractor. This competition was possible because the program office acquired an unlimited right to the software at the outset of the program. Interviews with the program office revealed the purpose for acquiring an unlimited right was not solely due to long-term competition needs, but also short-term integration needs. Interviewees did not note any singular aspects in development; however, code analysis revealed over 10 different programming languages were used in the system. This heterogeneous collection of programming languages makes it the most diverse under study. Some of the program language mix is explained by the design of the program, which has a Java back-end and a web-language based graphical user-interface.

Interviewees also noted the desire to migrate to a more "top-down" architecture in the upcoming release. The refactoring is catalyzed by an end-of-life situation

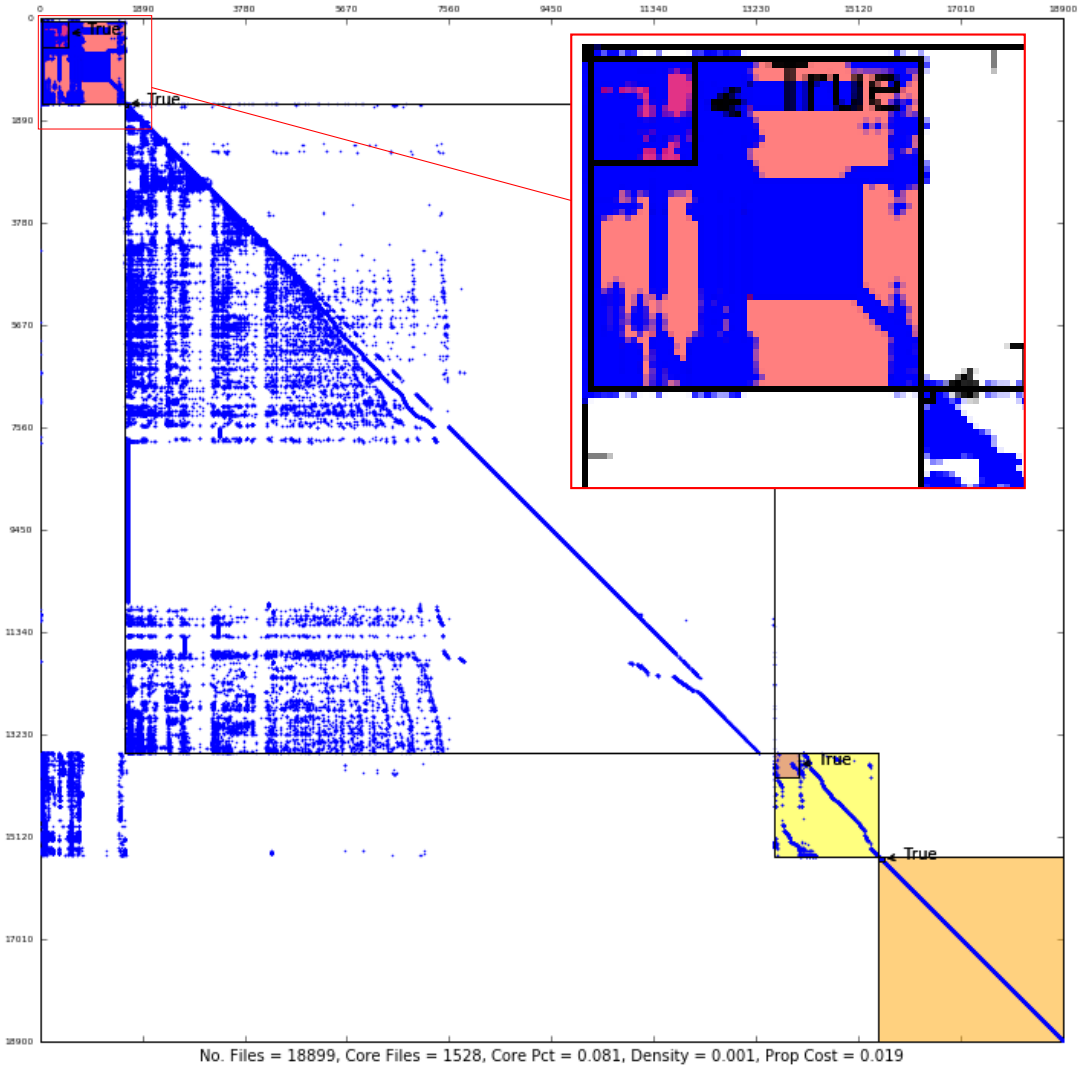


Figure 7-6: Case B Open Source View

for the front-end graphical user interface, but the program office would like to use that opportunity to migrate towards a more sustainable architecture. Specifically, interviewees spoke of "buying down technical debt." This is the only noted instance, in any of the cases, where the concept of technical debt was used by an interviewee in an architectural decision making process. Implying that quality and complexity of technical architecture is a factor in programmatic decisions.

Architectures

Case B has a core-periphery technical architecture with an accessible intellectual property architecture. Figure 7-6 illustrates the open source architectural view of Case B, using the notation "True" to indicate open source code and no label elsewhere. Case B has a core size of approximately 8%; placing it on the lower range of core-periphery architectures, with approximately 33% of the core being open source. See zoomed inset in Figure 7-6. Case B contains three distinct networks which are not statically connected. This is a common by-product in programs that contain extensive use of web-languages which are dynamically linked to the remainder of the software.

The causal conditions for Case B represent a conjuncture between core-periphery architecture and a closed intellectual property architecture with unlimited rights to avoid lock-in.

7.2.7 Case M

Background

Case M is a command and control system used across the Air Force to handle crisis situations at the base level. Specifically, Case M is part of a larger basket of tools that provide situational awareness and a common operating picture to base commanders for crisis response. Although there are many components in the overall tool suite, Case M specifically provides a decision support tool which generates a common operating picture in the form of geospatial overlays. These map overlays allow for timely sharing and updating of information to keep commanders informed. Case M is only a single program under a larger umbrella program that provides many crisis response capabilities. The history of umbrella program dates back to the early 2000s, where it was built and developed by a major defense contractor until 2013. In 2013, the development and sustainment were recompeted and won by a different defense contractor.

The move towards a different contractor was a shift in program office thinking. Interviewees noted the program wanted to migrate away from a thick-client solution,

which requires extensive hardware, to a thin-client based solution. A major part of this transition was rearchitecting of the basic framework of the system. However, Case M is one of the few tools in the overall program which has not migrated from thick-client to thin-client. Interviewees noted that no additional rights were purchased for this system above that which is conferred in standard DFARS clauses.

Although no additional rights were acquired, Case M differs from most of the umbrella program because it was developed organically by the DoD. Specifically, Case M was developed in an Air Force Research Laboratory. The result of this developmental choice is that all files developed are copyright to the U.S. Government, unless derivative works of open source software or licensed from other copyright holders. By virtue of developing Case M organically, the U.S. Government and, subsequently, the DoD is the copyright holder. In such a situation, there is no need for rights, since ownership implies an unlimited right. To account for this situation, Case M is coded as a case with unlimited rights.

Architectures

Case M has a core-periphery architecture with a closed intellectual property architecture; see the owner view in Figure 7-7. The core size of Case M is approximately 40% and there is only a single file with an open source license. Interestingly, Case M is written almost entirely in the Java programming language. This is interesting because most programs written in Java demonstrate a hierarchical architecture, but Case M is on the boundary between core-periphery architecture and degraded architecture. Interviewees did not have any explanation for the structure of the Java code and were unwilling to speculate. However, they did explain user satisfaction was moderate to high with the services provided by Case M.

The causal conditions for Case M represent a conjuncture between core-periphery architecture and a closed intellectual property architecture where unlimited rights avoided lock-in.

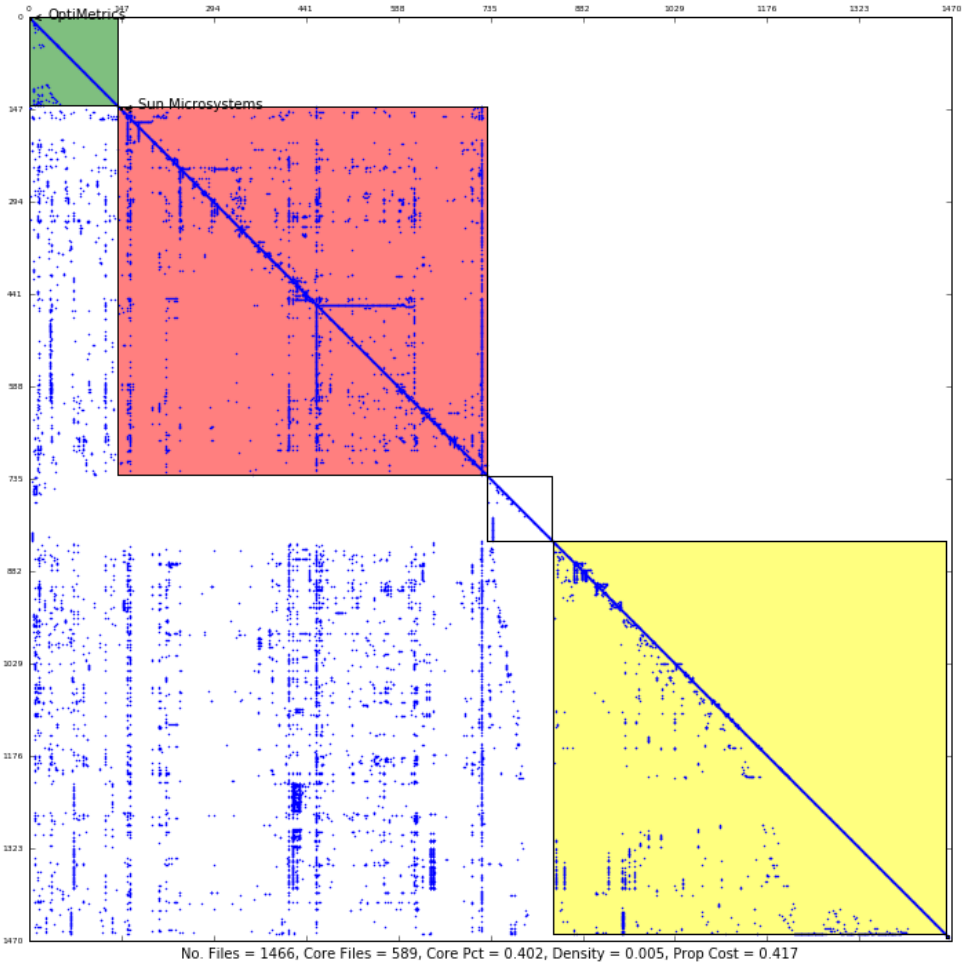


Figure 7-7: Case M Owner View

7.3 Boundary Patterns

Boundary patterns were theorized in Chapter 5 as the conjunction of degraded architecture and accessible architecture as well as hierarchical architecture and closed architecture.

7.3.1 Case N

Background

The history of Case N dates back to its creation in 2009, which was an effort to replace the legacy Air Force mission planning system. Specifically, Case N is a mis-

sion planning platform used to conduct detailed mission planning that supports the full spectrum of missions, ranging from simple training to complex combat scenarios. Multiple interviewees used the analogy of Waze, a popular automobile route-planning software, for aircraft to describe the functionality delivered by Case N. The primary differences being that Case N provides that route planning in three-dimensional space for aircraft, which introduces additional complexities: aircraft weight planning, fuel planning, weather avoidance, terrain avoidance, air traffic control frequencies, over-flight permissions, etc.

Additionally, because Case N is a platform it is important to note that there are greater than 50 external dependencies which rely on the services provided by Case N. This is salient because the analysis of Case N, herein, is only concerned with the internal quality and complexity of the architecture, but there are many external systems that also impose architectural constraints. To illustrate these constraints, it is first important to discuss how other cases in this research interact with Case N. Cases G, H, and E are all aircraft specific programs that use Case N to build missions. These three Cases use Case N to build mission plans based on the flight parameters of specific aircraft. Whereas, Case J and L are common utilities used by Case N for commodity functions (e.g., weather services for case J and GPS services for Case L). For any given mission, all of these different Cases work in unison to provide functionality to the end user. A change to one could have a rippling impact across all systems.

Interviewees noted a less than optimal architecture in Case N, using the analogy between making improvements to Case N to "making changes to an Atari." Although noting the hyperbolic nature of this analogy, interviewees argued that improvements to Case N were constrained by the architecture of the system. Consequently, the program is currently undergoing a major refactoring to decouple portions of the functionality from Case N and to improve the architecture. The interviewees would not go into great detail as this effort was still underway at the time of this writing, but the intent behind the major change was to spend time and money today to make a system that was easier and cheaper to maintain and improve in the future.

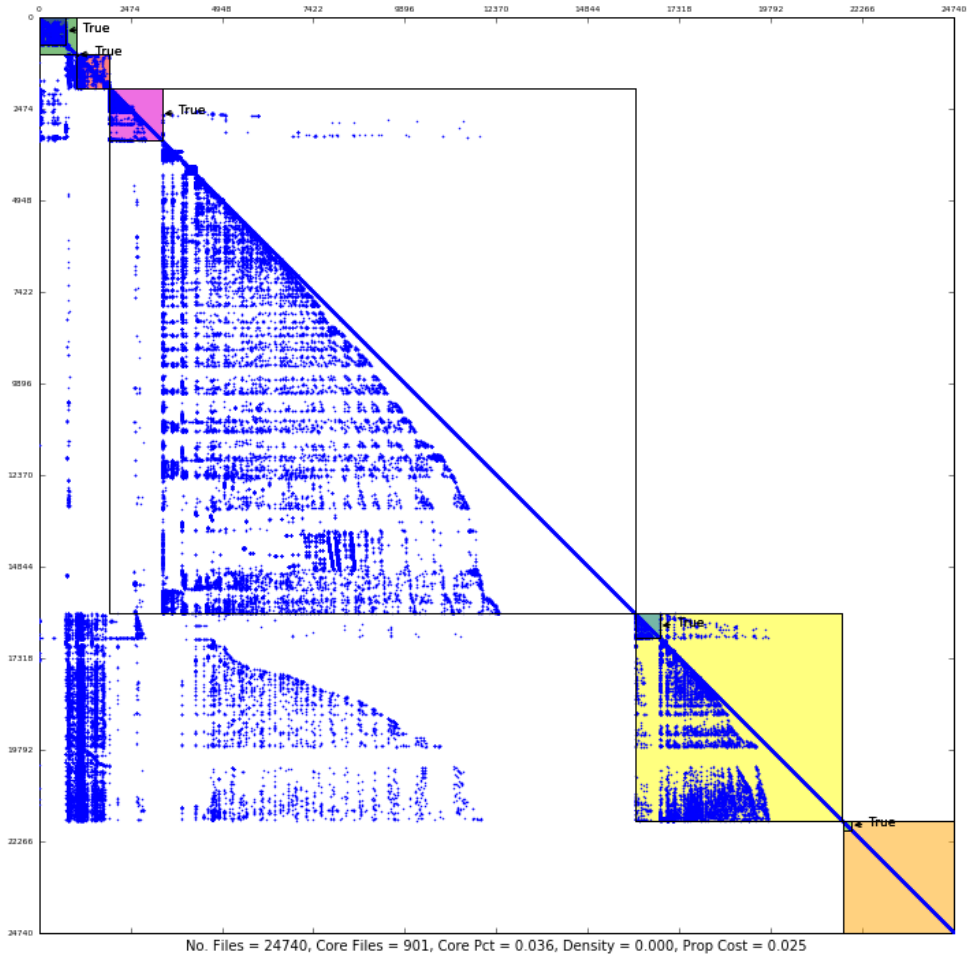


Figure 7-8: Case N Open Source View

When built, Case N chose to use, both, open source code and acquire an unlimited license to the code. However, interviewees noted that the open source code in the system was a persistent point of tension within the team. Some members argued for the benefit of using the code and others argued against the use of open source code due to the security risks. Although no long-term decision has been made regarding the future of open source code in Case N, interviewees would ultimately like to remove the open source code, but note budget constraints.

Architectures

Case N has a hierarchical architecture with a closed intellectual property architecture; see the open source view in Figure 7-8. The core size of Case N is approximately

3.6% with 40% of the D_8 files containing an open source license. Case N contains approximately 1,300 files which contain a reciprocal-type open source license and 1,788 files that contain a permissive-type open source license. Although this is a relatively high quantity of open source licenses, it falls below the 50% threshold to qualify as an accessible intellectual property architecture. Also of interest, Case N has two relatively large cores, the primary core which is 901 files and a secondary core which is 604 files. The secondary core is visible in the dense cluster of files located in the shared group of files in Figure 7-8. Since these cores are connected, but not bidirectionally connected, an argument could be made for counting both these cores in the calculation of the overall core size. This argument would yield a new core size of approximately 6%, which would push Case N into a hazard pattern.

The causal conditions for Case N represent a conjuncture between hierarchical architecture and closed intellectual property architecture where unlimited rights helped avoid lock-in.

7.3.2 Case E

Background

Case E is a tool suite that allows aircrew to conduct detailed mission planning for the full spectrum of possible missions. This data is then saved and loaded onto the aircraft for execution during the mission. Cases E, G, and H are all instances of mission planning software, but differ in the specific aircraft for which the software is designed. Furthermore, these specific instances of mission planning software are part of a larger enterprise that shares common components as well as a platform. Consequently, Case E, and all other mission planning cases, represents a set of modularized functionalities whose structure is partially dependent on the interfaces to both common components and the larger platform. These cases are also unique because at the outset of the program the Air Force acquired an Unlimited Right to the software source code. The acquisition of this class of rights ensured long-term competition for both the sustainment and development of the software.

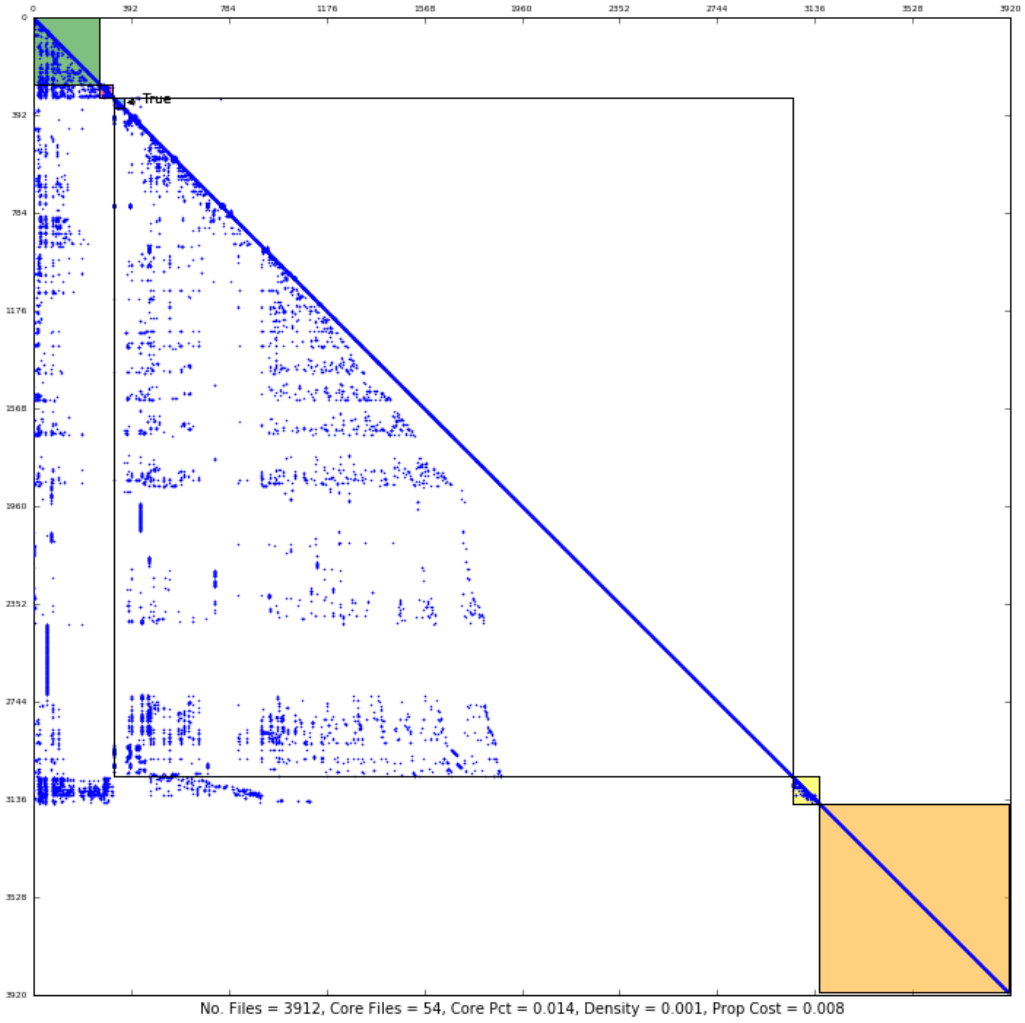


Figure 7-9: Case E Open Source View

Case E was originally built by the same program office that built the airframe which Case E services, but in the late 2000s the Air Force decided to consolidate all the mission planning software under a separate independent organization. Although each of the mission planning cases have a similar function, their architectures are influenced by vestiges from initial development under airframe specific program offices and interfaces with the airframe. This explains the technical architecture differences between Cases E, G, and H. Additionally, during the transition from geographically separated program offices into one consolidated program office, the mission planning cases were transitioned from contractor sustainment to DoD sustainment.

Architectures

Case E is a highly hierarchical technical architecture with a closed intellectual property architecture. The core percentage is approximately 1.4% and the D_8 is only 3.3% open. This core percentage makes Case E among the most hierarchical systems under study. Furthermore, the small section of files that are open source carry a copyleft-type license, but because these files are so small in number and do not have any direct connections to the core, it is unlikely the license would transfer to the derivative work. Although not strong separation, this case represents an interesting example where use of a copyleft-type license is inconsequential to the intellectual property architecture.

The causal conditions for Case E represent a conjuncture between a hierarchical architecture and a closed intellectual property architecture with unlimited rights resulting in avoidance of lock-in.

7.3.3 Case F

Background

Case F is a software program that parses complex mission tasking messages from United States Message Text Format (USMTF) into human readable format (e.g., a message may contain information such as the organization tasked, aircraft type, take-off time, air traffic control routes, etc.). The program started in 1996 as a "weekend project" built by a Federally Funded Research & Development Corporation engineer. Originally, the program was written in the C programming language and designed to work on Linux. Over the next two years, the program was officially funded, transferred to DoD sustainment, ported to the C++ programming language, and ported to a Microsoft operating system. Architecturally, the program remained unchanged until 2012 when the program was rewritten in the C# programming language and refactored into a "3-tier architecture."

The functionality behind Case F has not changed drastically from conception. However, the implementation of the functionality as well as integration of the functionality into various external enterprises have driven changes throughout the soft-

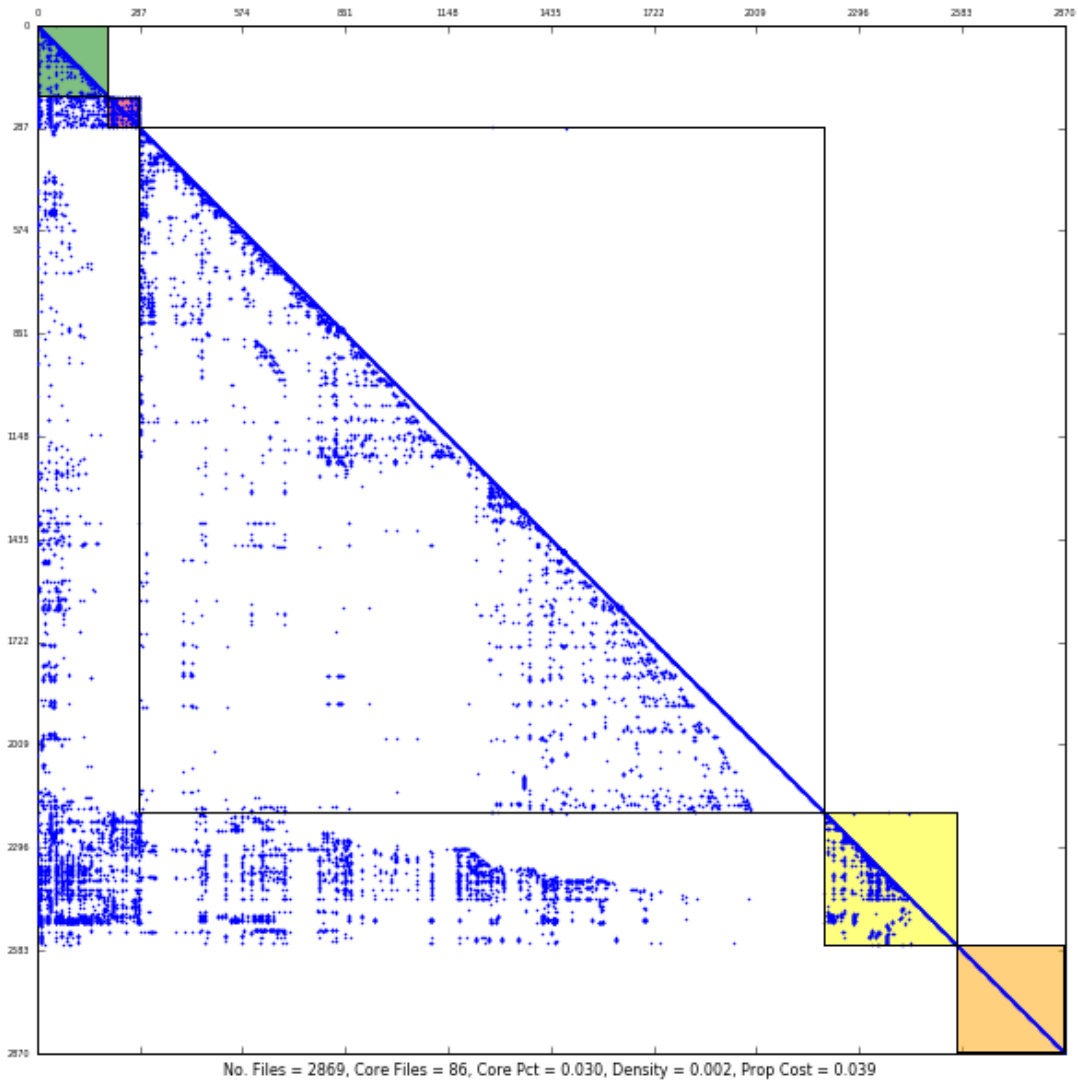


Figure 7-10: Case F Open Source View

ware's life-cycle. Specifically, when conceived, the program only parsed one type of message format. In the current version, the software is capable of parsing six distinctly different message formats and relaying that information to four different planning enterprises.

Architectures

Case F has a hierarchical technical architecture with a closed intellectual property architecture; see open source view in Figure 7-10. The core size of Case F is approximately 3%, with nine files that contain permissive-type open source licenses.

Given the background of Case F, the intellectual property architecture classification of Case F warrants further investigation. In all likelihood, the original engineer that created Case F maintained the original copyright, assuming this development was accomplished without any DoD funding. At some point the DoD acquired a right to the software and transitioned it to DoD sustainment. However, in one of the two programming language rewrites (i.e., C $\rightarrow$ C++ $\rightarrow$ C#) it is likely the original code was completely rewritten; ergo, the current copyright belongs to the DoD. Although unverifiable, if this were true, the intellectual property architecture would be accessible versus closed, which would reclassify Case F from a Boundary Pattern to an Ideal Pattern.

The causal conditions for Case F represent a conjuncture between hierarchical architecture and a closed intellectual property architecture where DoD had unlimited rights resulting in avoidance of lock-in.

7.3.4 Case H

Background

Case H is the last of the mission planning software cases. Recall from the discussions of Cases G and E that mission planning software is a program designed to aid aircrew in mission planning, then save the data and upload it to the specific aircraft. Additionally, Case H and Case G are software designed to interface with different versions of the same aircraft. All mission planning cases carry an unlimited right, which was acquired by the program office to ensure long-term competition.

Recently, Case H underwent a major refactoring effort. Recall from the discussion of Case E that the mission planning cases are part of a larger enterprise, which consists of common components and a larger platform. Each mission planning case interacts with the common components and the platform in slightly different ways. Interviews with the software engineers revealed the intent of the refactoring was twofold. First, to migrate the software to the C# programming language, and second to decouple as many of the common components as possible. Inspection of the ar-

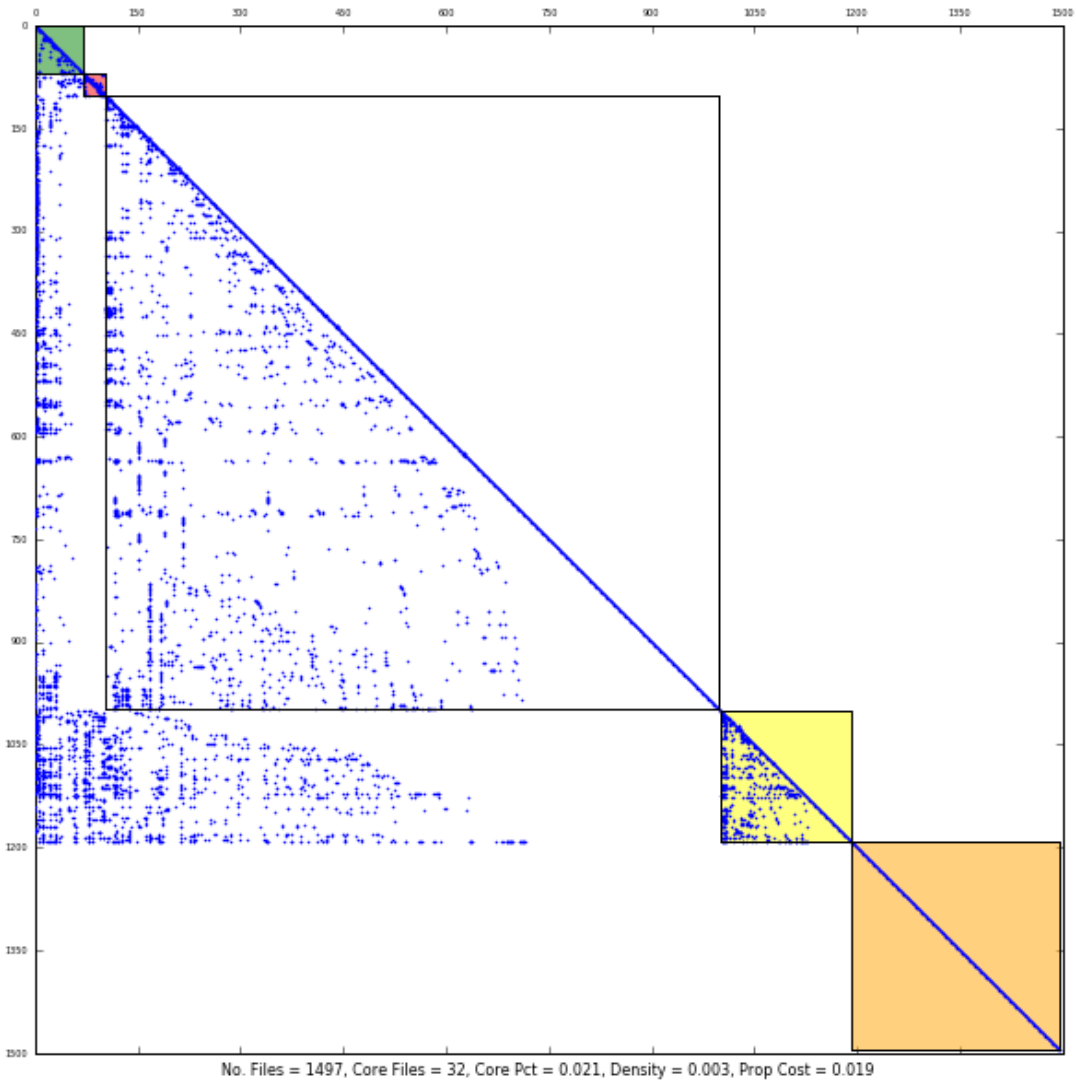


Figure 7-11: Case H Open Source View

chitecture shows this refactoring yielded a relatively well-architected system with a hierarchical architecture.

Architectures

Case H has a hierarchical technical architecture with a closed intellectual property architecture; see open source architecture view in Figure 7-11. The core percentage is approximately 2.1% and there is no meaningful amount of open source anywhere in the code.

The causal conditions for Case H represent a conjuncture between hierarchical architecture and a closed intellectual property architecture where DoD had unlimited rights resulting in avoidance of lock-in.

7.4 Ideal Patterns

Ideal patterns were theorized in Chapter 5 as the conjunction of core-periphery architecture and accessible architecture, as well as hierarchical architecture and accessible architecture.

7.4.1 Case L

Background

Case L is a software that provides mission planners with all Global Positioning System (GPS) prediction and reliability tools in one system. The software is the newest of the cases under study, with development beginning in August 2016. The capabilities provided by Case L existed previously, but were integrated into respective systems. Case L is an attempt to decouple the GPS functions from mission planning platform software and into a system that is easier to upgrade and sustain. Although critical to nearly all mission planning functions, Case L is the smallest case under study with only 374 files.

At the beginning of the Case L life-cycle, the decision was made to develop the system organically. The result of this decision is that all files developed are copyrighted to the U.S. Government, unless derivative works of open source software or licensed from other copyright holders (no derivative licences were identified in the software). By virtue of developing Case L organically the U.S. Government, and subsequently the DoD, is the copyright holder. In such a situation, there is no need for rights, since ownership implies an unlimited right. To account for this situation, Case L is coded as a case with unlimited rights.

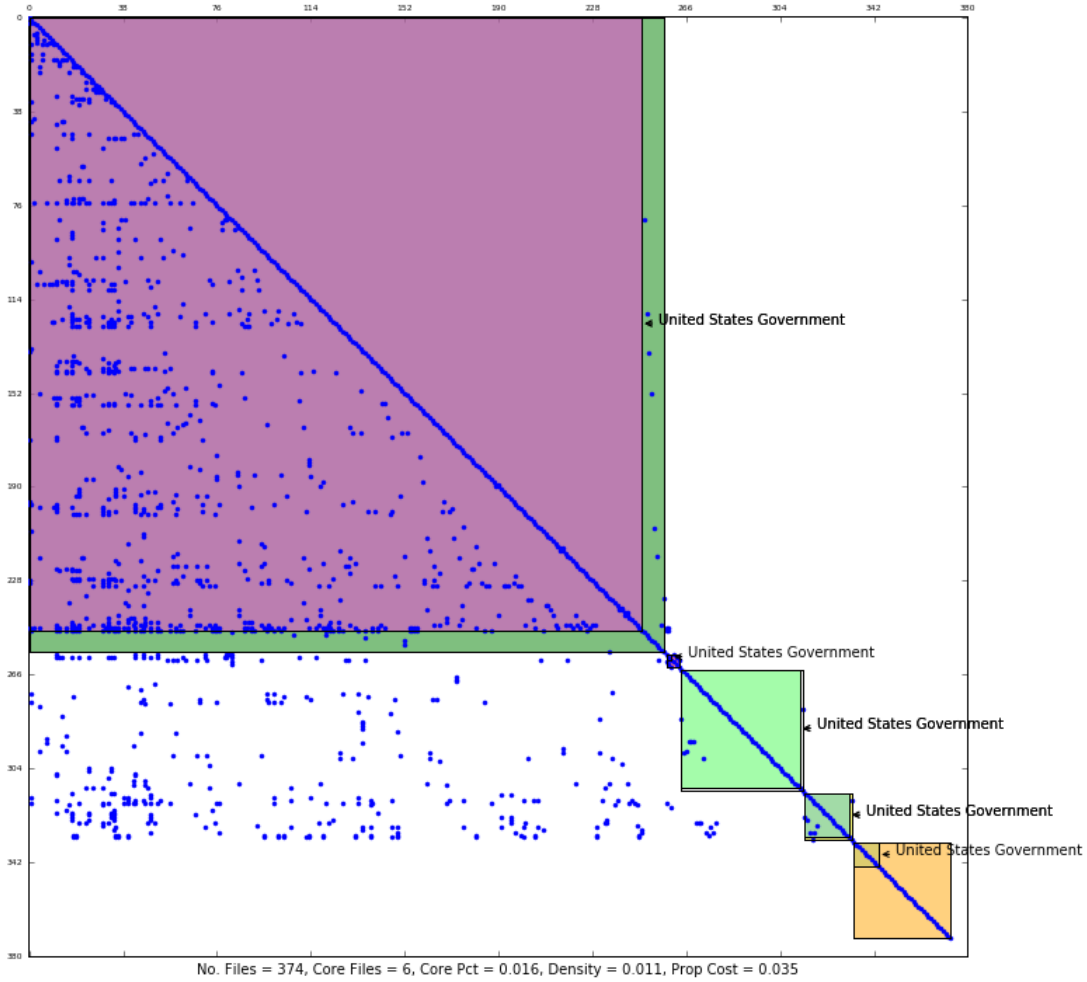


Figure 7-12: Case L Owner View

Architectures

Case L has a hierarchical architecture with an accessible intellectual property architecture; see owner view in Figure 7-12. The core size of Case L is approximately 1%, with no open source files. Figure 7-12 illustrates the copyrights which are owed by the U.S. Government, and account for approximately 87% of the entire system. Case L is also one of the few cases under study that uses a single programming language, C#. Lastly, it is interesting to note that Case L exposes a limitation of the visualization methods outlined in Chapter 5 in categorizing core files in highly hierarchical systems. This is because the largest cyclic group in Case L is two files, but there are 3 cyclic groups of size 2, each with varying VFI and VFO scores. Additionally, there

are another 22 files which are not in a cyclic group, but have a higher VFI and VFO than any of the cyclic groups. In these situations, it is difficult to tell which of the like-size cyclic groups are the core or if one of the highly connected single files is a more accurate core. Although interesting, the result makes little difference for the analysis, herein, as the result will always be a hierarchical architecture.

The causal conditions for Case L represent a conjuncture between hierarchical architecture and an accessible intellectual property architecture where DoD had unlimited rights resulting in avoidance of lock-in.

7.4.2 Case C

Background

Case C is a flight simulator used to train aircrew to respond to surface-to-air threats as well as terrain avoidance. The data herein largely comes from interviews with the contractor program manager, who worked on the program from its origins in the early 2000s until it was transitioned to DoD sustainment in 2014. The first contract was with a small technology firm and consisted of a small team of programmers who embraced a "hacker culture." As part of this culture, the team viewed it as their responsibility to save tax payer dollars wherever possible. Consequently, the program was built upon a foundation of open source code, some of which carried a GPL license. During the history of the program the government never acquired rights in addition to what were entitled in the standard DFAR clauses.

In 2014, the Air Force conducted a Source of Repair analysis on the program to determine if it were more cost effective to continue to sustain the system through the contractor or via DoD sustainment. Ultimately, the Air Force decided to transition the program from contractor sustainment to DoD sustainment. During the transition process the contractor program manager recounted the process of determining what parts of Case C could be sold to the DoD and what parts the DoD was entitled too. The prevalent use of open source code and lack of IRAD eventually led the contractor to conclude the DoD was entitled to the software at no cost.

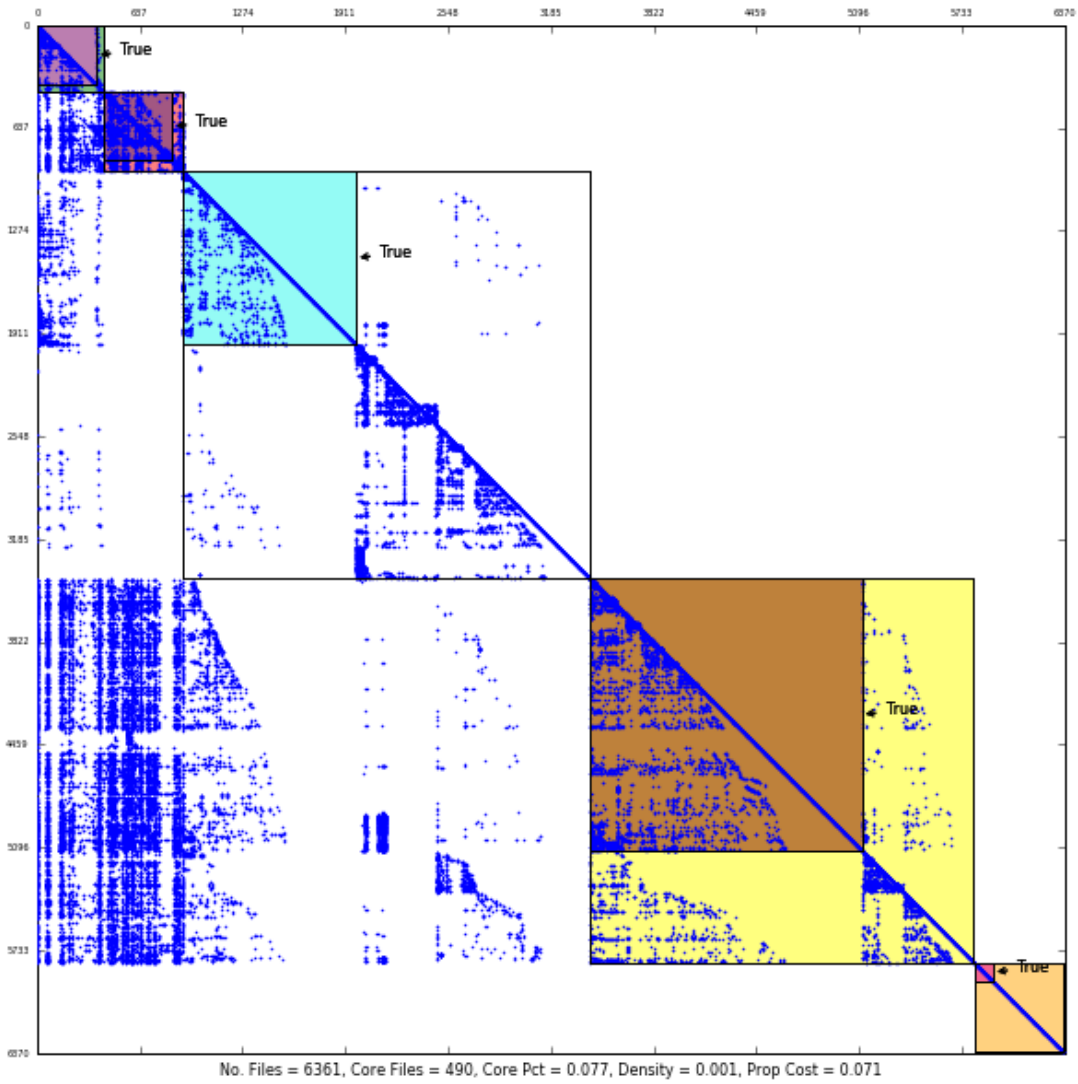


Figure 7-13: Case C Open Source View

Architectures

Case C has a Core-Periphery technical architecture as well as an accessible intellectual property architecture. Figure 7-13 illustrates the open source architectural view for Case C, where open source files are labeled as "True" and non-open source files are not labeled. Case C represents a relatively low complexity technical architecture combined with an intellectual property architecture where the preponderance of both, the shared and core groups, are open source; 87% and 86% respectively. Further, the type of open source in both the shared and core carry a copyleft-type license, implying

that the copyleft provision would transfer to the remainder of the software for reasons explained in Chapter 4.

The causal conditions for Case C represent a conjuncture between core-periphery architecture and an accessible intellectual property architecture where unlimited rights did not play a part to avoid lock-in.

7.4.3 Case D

Background

Case D is a software that provides a standards-based set of software services, documentation, and metadata schema to enable more efficient use of the DoD Intelligence, Surveillance, and Reconnaissance (ISR) enterprise. The program started in the mid-2000s as a method to allow for increased interoperability between ISR systems. A series of incremental baseline improvements carried the system through 2011. However, in 2012, Case D migrated to a modular open source OSGi based architecture. OSGi (Open Source Gateway initiative) is a set of specifications developed and maintained by the OSGi Alliance, to enable a software development model where applications are comprised of many reusable components. In addition to the open development model, Case D also utilizes the distributed data framework (DDF) maintained by the Codice Foundation, which is an "open source data layer that abstracts services and business logic from the underlying data structure to enable rapid integration of new data sources"[137].

Case D is the only software under study where the majority of the source code is downloadable via online software repositories and there is a public developer forum. The orientation towards open standards, open source frameworks, and community development parallel large scale open source projects in the commercial domain and represents somewhat of an outlier in the DoD domain. However, Case D is an important calibration point for the research model specified in Chapter 6, as it should always sit in the Ideal Architecture pattern.

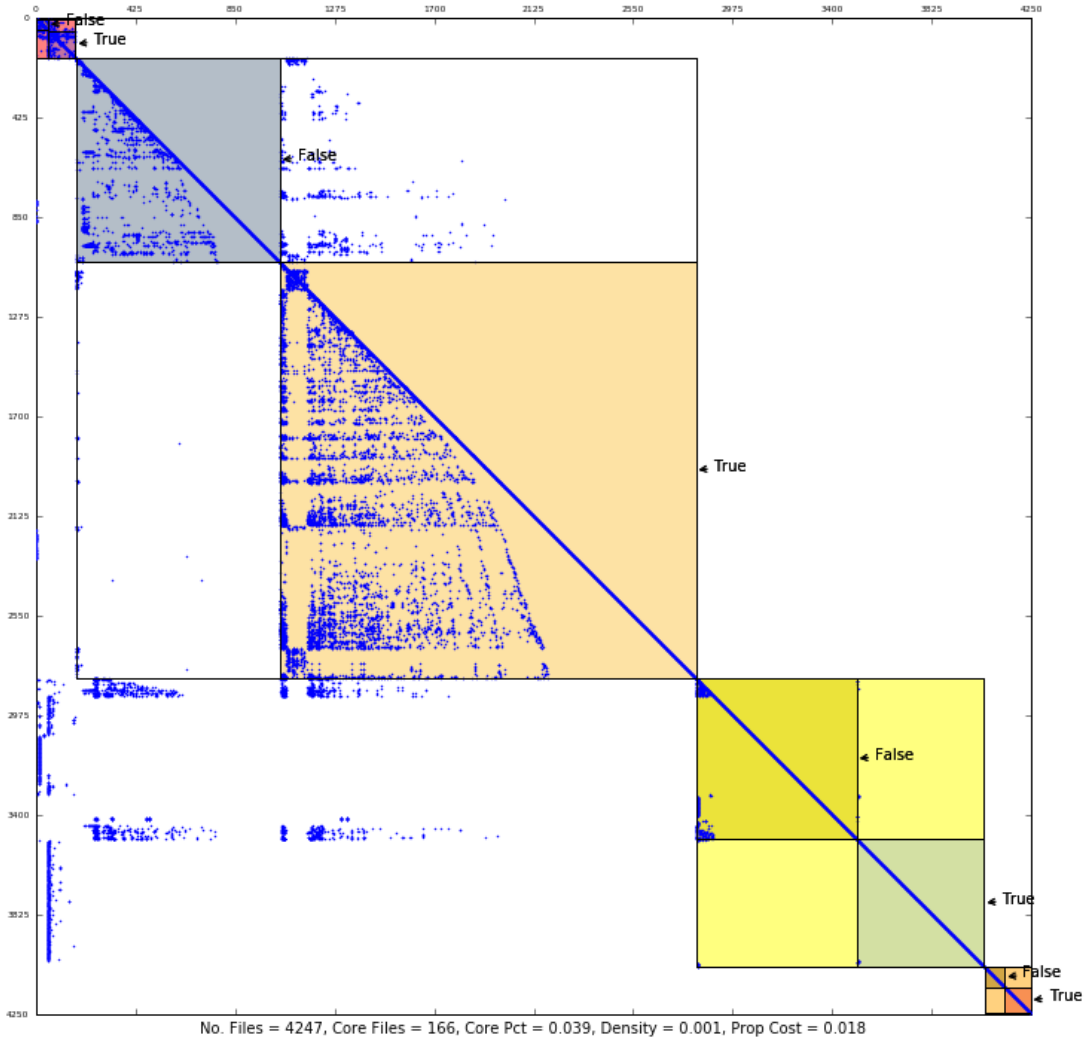


Figure 7-14: Case D Open Source View

Architectures

Case D has a hierarchical architecture with an accessible intellectual property architecture; see open source view in Figure 7-14. The core size of Case D is approximately 4%, with 2,549 files with varying types of open source licenses. Furthermore, approximately 70% of the files in the D_8 are open source, most of which carry a reciprocal type license. This proportion and location of open source licenses categorizes Case D as an accessible intellectual property architecture, but it is important to note that there are approximately 1,200 closed licenses as well.

The causal conditions for Case D represent a conjuncture between hierarchical architecture and an accessible intellectual property architecture where unlimited rights were not acquired leading to no lock-in.

7.5 Conclusion

This Chapter began by discussing overarching contextual details that apply to a majority of the cases. Specifically, for those cases which are currently sustained organically, the details discussed are pre-transition and the model conditions cover the pre-transition state. This was a necessary point of model calibration, as any system that is sustained organically is, by definition, not locked-in. This Chapter then outlined contextual details for each fourteen software cases from within the DoD. The cases range from a single year to twenty years of history, cover the spectrum of programming languages from Ada to C#, vary in size from as small as a couple hundred files to over twenty thousand files, as well as cover the spectrum of technical and intellectual property architectures.

Although not a "thick case" level of contextual details, the terse discussion of details, herein, is instrumental to drawing inferences from the cross-case analysis method outlined in Chapter 6. The subsequent Chapter will outline the results of the analysis and will rely upon the contextual details provided, herein, to draw conclusions.

Chapter 8

Cross-Case Analysis

This chapter will cover the full spectrum of fsQCA. Beginning with model calibration, each variable must be calibrated for fuzzy membership in the respective set. Two variables, technical architecture quality and accessible intellectual property architecture, will undergo calibration. Once all variables are calibrated the first step is identifying necessary conditions and identifying any conditions that are sufficient. Identification of both condition types involves logical tests and discussion of parameters of fit.

Although all the calculations that follow are primarily Boolean and Fuzzy algebra, scholars argue that QCA analyses, regardless of type, should be done using computational assistance to reduce the likelihood of errors in computation [113], [117], [125], [138], [139]. In adhering to the advice of fellow researchers, this analysis relies exclusively on the `QCA` software package [140]. Although calculations are shown below, which are both accurate and complete, these are examples designed to give the reader a better understanding of some of the more complex calculations made by the `QCA` package.

8.1 Model Calibration

Model calibration is the process by which raw data is transformed into fuzzy membership scores, sometimes also referred to as *fuzzification*. Ragin suggests two methods: indirect and direct methods of calibration [138] (other researchers refer to these

methods as: direct and transformation respectively [141]). In the indirect method, an external standard is adopted and all cases are assigned to one of n ($3 \leq n \leq 7$) ordinal groups (e.g., Likert Scales). After assignment, a simple estimation technique is used to assign scores to an interval scale. The direct approach uses three qualitative breakpoints that structure a set: full membership, full nonmembership, and the crossover point. These three breakpoints are used to transform the raw scores into fuzzy membership scores. Discussion in literature varies as to what method is most appropriate, with most authors carefully cautioning against a mechanistic approach [139] and emphasizing transparency as well as theoretical arguments for calibration method over prescriptive approaches [125], [126], [138].

8.1.1 Technical Architecture Quality

For the technical architecture quality condition the direct method of calibration is used. This method was chosen for the reason that external standards for set membership do not currently exist and because theoretical breakpoints are hypothesized in academic literature [75], [98], [101]. From literature, the theorized crossover point (referred to subsequently as τ_{cr}) between a hierarchical architecture and a core-periphery architecture is a core size of 5%. Additionally, literature shows that the closer the core size is to zero the more hierarchical the architecture. Therefore, a full membership inclusion point (referred to subsequently as τ_{in}) of 1% is adopted. This implies that all cases with a core size less than 1% are full members in the fuzzy set of quality technical architecture. The point least rooted in academic research is the full nonmembership exclusion point (referred to subsequently as τ_{ex}). Previous research is in agreement that the larger the core size the lower the quality of technical architecture, but no judgment is made on where this transition occurs. Consequently, a point of 25% was chosen as a reasonable standard because it allows for a core of sufficient size to cover any case that makes a quality versus performance trade-off.

Using these values, the fuzzy membership values were computed using a negative endpoint piecewise logistic function [142]; see Equation (8.1); where x_i is the base variable and all other variables are previously defined. Negative endpoint implies

that as raw values increase fuzzy membership score decreases. This phenomenon is illustrated in Figure 8-1, which shows a line plot of case core size versus fuzzy membership score.

$$\mu_{\mathbf{SET}}\left(x, \tau_{[\ldots]}\right) = \begin{cases} 1 & \text{if } \tau_{in} \geq x_i, \\ 1 - \frac{1}{2}\left(\frac{\tau_{in} - x_i}{\tau_{in} - \tau_{cr}}\right) & \text{if } \tau_{in} < 0 \leq \tau_{cr}, \\ \frac{1}{2}\left(\frac{\tau_{ex} - x_i}{\tau_{ex} - \tau_{cr}}\right) & \text{if } \tau_{cr} < x_i \leq \tau_{ex}, \\ 0 & \text{if } \tau_{ex} < x_i. \end{cases} \tag{8.1}$$

The result is a calibrated set of fuzzy membership scores and accompanying verbal cues to denote proper set membership; see Table 8.1. Those cases with core sizes less than 1% are said to be high quality technical architectures. Core sizes between 1% and 5% are said to be more or less high quality technical architecture and cores sizes between 5% and 25% are more or less not high quality technical architectures. Finally, those cases with core sizes greater than 25% are said to be not high quality technical architectures.

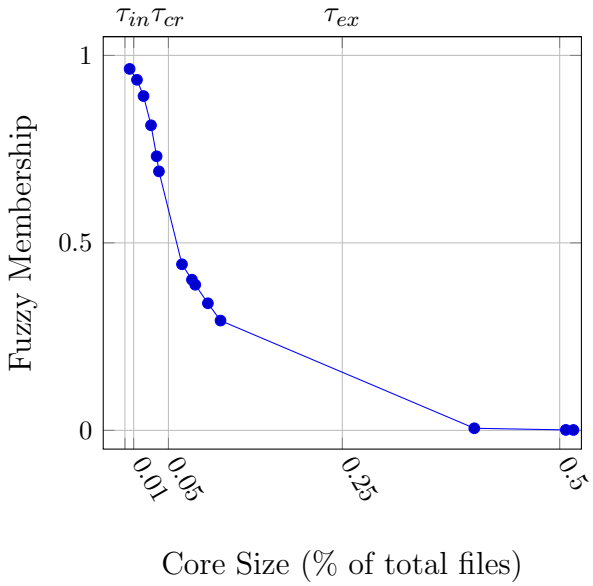


Figure 8-1: Plot of Membership in Set of Quality Technical Architecture

Raw Scores of Core Size (c)	Verbal Labels	Fuzzy Membership Score (m)
$c \leq 0.01$	High quality	m = 1.0
$0.01 < c < 0.05$	More or less high quality	$0.5 < m < 1.0$
$0.05 < c < 0.25$	More or less not high quality	$0 < m < 0.5$
$c \geq 0.25$	Not high quality	m = 0

Table 8.1: Fuzzy Membership in Set of Technical Architecture Quality

8.1.2 Accessible Intellectual Property Architecture

A direct method was also used to calibrate the set of accessible intellectual property architectures. However, there is not a substantive body of literature from which to derive theoretical breakpoints. Consequently, each breakpoint was established based on a conservative logical judgment by the author. Full membership is defined as an intellectual property architecture where greater than 90% of the core, or D_8, is open source or copyright to the U.S. Government (τ_{in}). Full nonmembership is defined as an intellectual property architecture with less than 10% of the core, or D_8, as open or copyright to the U.S. Government (τ_{ex}). Finally, the crossover point between full membership and nonmembership is defined as 50% (τ_{cr}).

Using these breakpoints, fuzzy set membership was calculated using a positive endpoint piecewise logistic functions; see Equation (8.2). Again, where x_i is the base variable and all other variables are previously defined. The positive endpoint implies that as values of accessible intellectual property architecture increase so do the fuzzy membership scores. This is illustrated in Figure 8-2, which shows a line plot of accessible intellectual property architecture scores versus fuzzy membership.

$$\mu_{\mathbf{SET}}\left(x, \tau_{[\ldots]}\right) = \begin{cases} 0 & \text{if } \tau_{ex} \geq x_i, \\ \dfrac{1}{2}\left(\dfrac{\tau_{ex} - x_i}{\tau_{ex} - \tau_{cr}}\right) & \text{if } \tau_{ex} < x_i \leq \tau_{cr}, \\ 1 - \dfrac{1}{2}\left(\dfrac{\tau_{in} - x_i}{\tau_{in} - \tau_{cr}}\right) & \text{if } \tau_{cr} < x_i \leq \tau_{in}, \\ 1 & \text{if } \tau_{in} < x_i \end{cases} \tag{8.2}$$

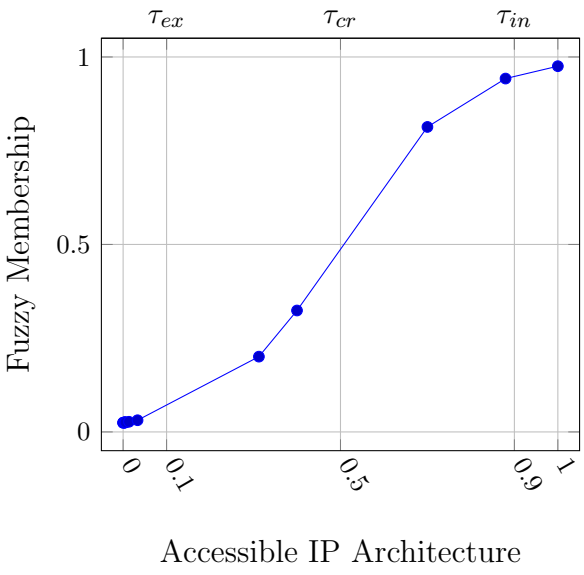


Figure 8-2: Plot of Calibrated Fuzzy Membership in Set of Accessible IP Architecture

Similar to the technical architecture set, the result of calibration is a set of fuzzy membership scores in the set of accessible intellectual property architecture with verbal descriptors; see Table 8.2. Those cases with an accessible amount less than 10% are referred to as "Not Accessible." If accessible values are between 10% and 50% it is referred to as "More or less not accessible" and values between 50% and 90% are referred to as "More or less accessible." Finally, those Cases with accessible values greater than 90% are referred to as "Accessible."

Open Source or U.S. Govt Copyright (c)	Verbal Labels	Fuzzy Membership Score (m)
$c \leq 0.1$	Not accessible	m = 0
$0.1 < c < 0.5$	More or less not accessible	$0 < m < 0.5$
$0.5 < c < 0.9$	More or less accessible	$0.5 < m < 1$
$c \geq 0.90$	Accessible	m = 1

Table 8.2: Fuzzy Membership in Set of Accessible Intellectual Property Architecture

There result of calibrating conditions TA and IP are visible below in Table 8.3. This Table, minus the "Pattern" column, represents the input for all subsequent calculations in the fsQCA model. The reader should note the change in the outcome variable from LI to 1-LI. With this change the fsQCA model is now set-up to investigate the negation of LI or the avoidance of lock-in.

Table 8.3: Calibrated Summary of Cases

		Inputs			Outcome
Pattern	Case	TA[1]	IP[2]	UR[3]	1-LI[4]
Hazard	Case A	0.00	0.03	0	0
	Case K	0.00	0.03	0	0
	Case G	0.34	0.03	1	1
	Case I	0.44	0.03	0	1
	Case J	0.29	0.03	1	1
	Case B	0.39	0.20	1	1
	Case M	0.01	0.02	1	1
Boundary	Case E	0.93	0.03	1	1
	Case H	0.89	0.02	1	1
	Case F	0.81	0.03	1	1
	Case N	0.73	0.32	1	1
Ideal	Case C	0.40	0.94	0	1
	Case L	0.96	0.98	1	1
	Case D	0.69	0.81	0	1

[1] Fuzzy membership in Set Quality Technical Architecture
[2] Fuzzy membership in Set Accessible Intellectual Property Architecture
[3] $UR\{1\}$: Unlimited Rights and $UR\{0\}$: Not Unlimited Rights
[4] $1 - LI\{1\}$: Not Locked-in and $1 - LI\{0\}$: Locked-in

8.2 Conditions

Analysis using any QCA, or set theoretic approach, is organized around identifying set relationships and drawing inferences from set relationships across many cases. Two specific set relationships or condition types are of particular interest: *necessary conditions* and *sufficient conditions*. A condition ($\mathbf{X}$) is described as necessary if it is required to produce the outcome understudy ($\mathbf{Y}$). Thinking about this relationship in terms of sets, for a condition to be necessary $\mathbf{Y}$ must be a subset of $\mathbf{X}$ (i.e., $\mathbf{Y} \subset \mathbf{X}$). Sufficient conditions are the inverse of necessary conditions. A condition ($\mathbf{X}$), or combination of conditions ($\mathbf{XZ}$), is said to be sufficient if it leads to the outcome understudy ($\mathbf{Y}$). In set theory, this relationship implies that $\mathbf{X}$ is a subset of $\mathbf{Y}$ (i.e., $\mathbf{X} \subset \mathbf{Y}$). Due to the nature of equifinality, it is likely that there are multiple conditions or groups of conditions that are sufficient for a particular outcome. Understanding, the relationships between sets allows for inferences about situations that are required for an outcome and situations that may lead to an outcome.

8.2.1 Parameters of Fit

However, due to the complex nature of equifinality and social interactions, conditions are rarely purely necessary or sufficient. Instead, they are more likely to be "almost always sufficient" or "more often than not necessary." To judge the quality of conditions two set theoretic parameters of fit are used: *consistency* and *coverage*. Set theoretic "*consistency* gauges the degree to which the cases sharing a given combination of conditions agree in displaying the outcome in question" [138, p. 44]. Said differently, consistency quantifies how close a perfect subset relation is approximated. Although the intent of the metrics is universal, the application of consistency differs between necessary and sufficient conditions, which will be discussed in further details in those specific sections. In contrast to consistency, "*coverage* assesses the degree to which a cause or causal combination accounts for the outcome" [138, p. 44] (e.g., when there are multiple paths to a specific outcome *coverage* gauges the empirical relevance or importance of each). Similar to consistency, the equation for coverage is different for necessary and sufficient conditions, which are discussed below in further detail. Lastly, it is important to note a sequential importance in these parameters of fit. Only after a set of conditions is determined to be reasonably consistent is calculation of coverage appropriate.

8.3 Necessary Conditions

For a condition to be necessary in a fuzzy-set all the fuzzy-set membership scores (X_i) must be greater than or equal to fuzzy-set membership in the outcome (Y_i) (i.e., $X_i \geq Y_i$). Because necessary conditions are required to be greater than or equal to the outcome, the goal is maximization of the fuzzy-set score. Consequently, the analysis begins by evaluating each of the conditions individually against the outcome, then all logical *OR* combinations, or unions, of the conditions against the outcome. In Boolean and fuzzy algebra, a membership score for OR combinations are calculated as the maximum value across the combinations (i.e., $A + B = max(A, B)$).

Each necessary condition is judged based primarily on two factors: consistency and coverage. The equations for consistency and coverage are listed below; see Equations (8.3) and (8.4) respectively.

$$Con_N(\mathbf{X}) = \frac{\sum_{i=1}^{n} min(x_i, y_i)}{\sum_{i=1}^{n} y_i} \tag{8.3}$$

$$Cov_N(\mathbf{X}) = \frac{\sum_{i=1}^{n} min(x_i, y_i)}{\sum_{i=1}^{n} x_i} \tag{8.4}$$

$$RoN(\mathbf{X}) = \frac{\sum_{i=1}^{n} (1 - x_i)}{\sum_{i=1}^{n} (1 - min(x_i, y_i))} \tag{8.5}$$

However, authors argue coverage fails to capture trivialness of necessary conditions under skewed distributions [125]. To address this deficiency, Schneider and Wagemann introduce Relevance of Necessity (RoN), Equation (8.5), which is less sensitive to skewed distributions than Equation (8.4). Low values of RoN suggest a trivial necessary condition and high values suggest a non-trivial necessary condition.

Calculations were conducted for all individual conditions and all possible logical OR combinations of variables. In an effort to discuss only the most parsimonious sample of results, Table 8.4 displays only those combinations with $Con_N > 0.9$. These are the combinations with the highest consistency. Upon inspection of the RoN metric, three of the four combinations have low scores suggesting they are only trivially necessary. However, the fourth combination (TA+IP+UR) has a consistency score of 0.93 as well as a RoN score of 0.98. Implying that this combination of conditions is potentially a superset of the outcome (i.e., $1 - LI \subset TA + IP + UR$).

To investigate this combination of conditions further, Figure 8-3 visually depicts the relationship between fuzzy membership in TA+IP+UR and membership in $1-LI$. If $1 - LI$ were a perfect subset of TA+IP+UR, all points on the scatter plot would be on or below the diagonal. Inspection reveals that only 11 of 14 points meet this criterion, with three Cases (D, C, and I) above the diagonal. The imperfection of this subset relationship is captured by Con_N, which indicates a less than perfect

subset relationship. Unlike mainstream statistics, there are no conventional standards in which a condition or group of conditions must meet to be considered necessary. Instead, it is left to the experimenter to make an assessment of the necessity of conditions based on theoretical evidence, which will take place in Chapter 9.

8.4 Sufficient Conditions

Each sufficient condition is judged primarily on two factors: consistency and coverage. The equations for consistency and coverage specific to sufficient conditions are listed below; see Equations (8.6) and (8.7) respectively.

$$Con_S(\mathbf{X}) = \frac{\sum_{i=1}^{n} min(x_i, y_i)}{\sum_{i=1}^{n} x_i} \tag{8.6}$$

$$Cov_S(\mathbf{X}) = \frac{\sum_{i=1}^{n} min(x_i, y_i)}{\sum_{i=1}^{n} y_i} \tag{8.7}$$

In addition to the two measures above, any analysis of sufficient conditions is also concerned with the *unique coverage* (Cov_U). When more than one condition or group of conditions is sufficient for an outcome (i.e., multiple conjunctural causality), the assessment of alternate combinations provides direct evidence towards the importance of relative empirical performance [138]. Equation (8.7) is for "raw" coverage or the coverage of the entire expression, where Cov_U is the coverage for each independent causal combination in any given expression. Understanding the change in coverage, based on the addition or removal of a variable on the outcome, is similar to the concept of explained variance in regression analysis. The equation for Cov_U of condition X is Cov_S of the full expression minus Cov_S of the expression without condition X (e.g., $Cov_S(XZ + AB) - Cov_S(XZ) = Cov_U(AB)$).

8.4.1 Conservative Expression

The process to test for sufficiency is slightly different than necessary conditions. This is because the goal is to identify groupings of causal combinations that lead to the

outcome. These groupings of causal combinations will be whole or partial subsets of the outcome. Consequently, for a condition to be sufficient in a fuzzy-set, all the fuzzy-set membership scores (X_i) must be less than or equal to fuzzy-set membership in the outcome (Y_i) (i.e., $X_i \leq Y_i$).

To identify sufficient conditions requires construction of a truth table. The function of a truth table is to categorize the membership level of each case across the range of 8 ($2^3 = 8$) corners of the three-dimensional vector space created by a 3 condition fsQCA. Fuzzy membership in the vector space is calculated by taking the logical AND of all possible conditions. In Boolean and Fuzzy algebra, a membership score for AND combinations is calculated as the minimum value across the combinations (i.e., A*B*C=min(A,B,C)). The results of these calculations are outlined in Table 8.5, which shows each case's fuzzy membership in all eight corners of the vector space (note, negation of a condition is indicated using lower cases letters: 1-IP = ip). All cases have greater than a 0.5 membership in only one corner of the vector space; these values are bold in Table 8.5. These bold values indicate which corner of the vector space each case is a member (e.g., corner ta*ip*UR has four Cases with a membership score above 0.5: M, B, G, and J). Therefore, Cases M, B, G, and J are said to be a member of set ta*ip*UR.

From the values in Table 8.5 a truth table is tabulated; see Table 8.6. The truth table for 1-LI shows five distinct combinations of causal conditions which lead to the outcome. One causal combination which leads to LI and two causal combinations for which there are no cases (denoted with a "?"). Because each case is a fuzzy member of each causal combination the consistency measure is important. If a causal combination demonstrated low consistency across the cases it would indicate that there is not agreement among the cases on the outcome, or that there is insufficient evidence in that specific group of causal conditions to make clear determination of the outcome. The threshold used to determine membership in 1-LI was a Con_S score greater than 0.95.

The truth table reveals there are five possible causal combinations which lead to 1-LI; see Equation (8.8). Although this is a complete solution for possibly sufficient

	Con_N	RoN	Cov_N
ta+IP+UR	0.94	0.26	0.85
TA+ip+ur	0.95	0.25	0.85
TA+ip+UR	0.92	0.34	0.85
TA+IP+UR	0.93	0.98	0.99

Table 8.4: Parameters of Fit for Necessary Conditions

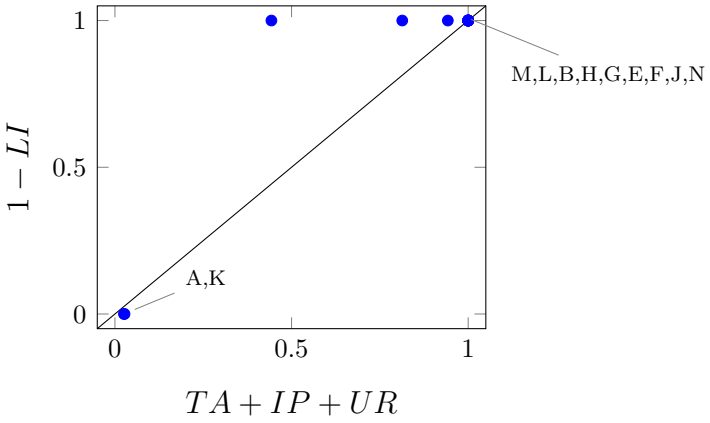


Figure 8-3: Scatter Plot of Set TA+IP+UR against Set 1-LI

	taipur	taipUR	taIPur	taIPUR	TAipur	TAipUR	TAIPur	TAIPUR
A	**0.97**	0.00	0.03	0.00	0.00	0.00	0.00	0.00
D	0.19	0.00	0.31	0.00	0.19	0.00	**0.69**	0.00
L	0.00	0.02	0.00	0.04	0.00	0.02	0.00	**0.96**
M	0.00	**0.98**	0.00	0.02	0.00	0.01	0.00	0.01
B	0.00	**0.61**	0.00	0.20	0.00	0.39	0.00	0.20
N	0.00	0.27	0.00	0.27	0.00	**0.68**	0.00	0.32
E	0.00	0.07	0.00	0.03	0.00	**0.93**	0.00	0.03
H	0.00	0.11	0.00	0.02	0.00	**0.89**	0.00	0.02
G	0.00	**0.66**	0.00	0.03	0.00	0.34	0.00	0.03
I	**0.56**	0.00	0.03	0.00	0.44	0.00	0.03	0.00
K	**0.97**	0.00	0.03	0.00	0.00	0.00	0.00	0.00
F	0.00	0.19	0.00	0.03	0.00	**0.81**	0.00	0.03
C	0.06	0.00	**0.60**	0.00	0.06	0.00	0.40	0.00
J	0.00	**0.71**	0.00	0.03	0.00	0.29	0.00	0.03

Table 8.5: Fuzzy membership in three-dimensional vector space (note: all causal combinations are expressed as conjunctions where "taipur" is equivalent to "ta*ip*ur")

conditions, the equation is unnecessarily complex and contains redundant descriptions of the same sample spaces.

$$\text{ta*ip*UR + ta*IP*ur + TA*ip*UR + TA*IP*ur + TA*IP*UR} \rightarrow 1 - LI \quad (8.8)$$

To reduce the complexity of the equation Boolean minimization is employed, specifically the enhanced Quine-McCluskey algorithm [143]. For example, Equation (8.8) contains the combinations TA*IP*ur as well as TA*IP*UR. These combinations account for membership in TA, membership in IP and both membership and non-membership in UR; therefore, the UR and ur conditions are logically redundant and the expression can be minimized to TA*IP. Figure 8-4 shows an example minimization using the enhanced Quine-McCluskey algorithm.

In this example, a less complex expression is reached which is logically equivalent to the original expression and the information contained in the truth table. In minimizing the complex solution at Equation (8.8), the nature of causal conditions is reduced to a *conservative expression* of the sufficient causal conditions. The result of the minimization are two possible solutions: Equations (8.9) and (8.10).

$$\text{ip*UR + IP*ur + TA*UR} \rightarrow 1 - LI \quad (8.9)$$

$$\text{ip*UR + IP*ur + TA*IP} \rightarrow 1 - LI \quad (8.10)$$

These expressions in Equations (8.9) and (8.10) can be further factored into Equations (8.11) and (8.12) respectively.

$$\text{IP*ur + UR*(TA + ip)} \rightarrow 1 - LI \quad (8.11)$$

$$\text{IP*(TA + ur) + ip*UR} \rightarrow 1 - LI \quad (8.12)$$

It is important to note that the two equations reached above (factored or unfactored) are logically equivalent expressions of the same sample space. Analysis of the parameters of fit in Table 8.7 offers empirical support in the identical fit measures

for both Equations (8.9) and (8.10). Both expressions demonstrate a high measure of consistency (0.99) and a relatively high measure of raw coverage (0.84). Unlike conventional statics, there is not an accepted threshold value for consistence and coverage that make a condition sufficient. Rather, in fsQCA determination is largely left to judgment based on theoretical and empirical evidence, which will be discussed in the subsequent Chapter.

8.4.2 Parsimonious Expression

The explanatory equations reached above are said to be *conservative expressions*, because they are logically equivalent expressions of only the data in the truth table. However, it is possible to reach a more parsimonious expression by using logical remainders. Logical remainders are those sets of conditions for which no empirical evidence exists (i.e., rows 4 and 5 in Table 8.6). These logical remainders allow for the use of *counterfactuals*. There are multiple categories of counterfactuals used in fsQCA; however, the analysis, herein, is concerned solely with *easy counterfactuals*. Easy counterfactuals are simplifying assumptions which are "in line with both the empirical evidence at hand and existing theoretical knowledge on the effect of the single conditions that compose the logical remainder" [125, p. 168]. The two available logical remainders in this analysis are sets ta*IP*UR (row 4) and TA*ip*ur (row 5). Theoretical and empirical evidence discussed in Chapters 2 and 6 support the simplifying assumption: ta*IP*UR$\rightarrow 1 - LI$, but not TA*ip*ur $\rightarrow 1 - LI$. To incorporate this simplifying assumption, the logical remainder at row 4 is recoded from "?" to "1" and the analysis is rerun. The result is the expression at the top of Figure 8-5.

The above expression is minimized in exactly the same procedure as previously explained. Although the minimization process was done using computational tools to reduce the likelihood of calculation errors, an illustration of the minimization is carried out manually in Figure 8-5, which results in Equation (8.13).

$$\text{UR+IP} \rightarrow 1 - LI \tag{8.13}$$

	TA	IP	UR	1-LI	n	Con_S	Cases
1	0	0	0	0	3	0.29	A,I,K
2	0	0	1	1	4	1	M,B,G,J
3	0	1	0	1	1	0.95	C
4	0	1	1	?	0	-	
5	1	0	0	?	0	-	
6	1	0	1	1	4	1	N,E,H,F
7	1	1	0	1	1	0.99	D
8	1	1	1	1	1	1	L

Table 8.6: Truth Table for outcome 1-LI

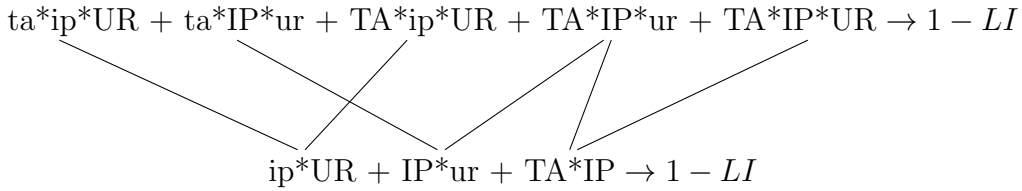


Figure 8-4: Example Boolean minimization of Equation (8.8) to Equation (8.10)

	Con_S	Cov_S	Cov_U	Cov_U(Eq 8.9)	Cov_U(Eq 8.10)	Cases
ip*UR	1.00	0.61	0.25	0.25	0.56	M,B,G,J; N,E,H,F
IP*ur	0.97	0.15	0.06	0.15	0.06	C; D
TA*IP	1.00	0.23	0.00		0.08	D; L
TA*UR	1.00	0.45	0.01	0.08		N,E,H,F; L
Eq 8.9	0.99	0.84				
Eq 8.10	0.99	0.84				

Table 8.7: Sufficient Conditions Parameters of Fit

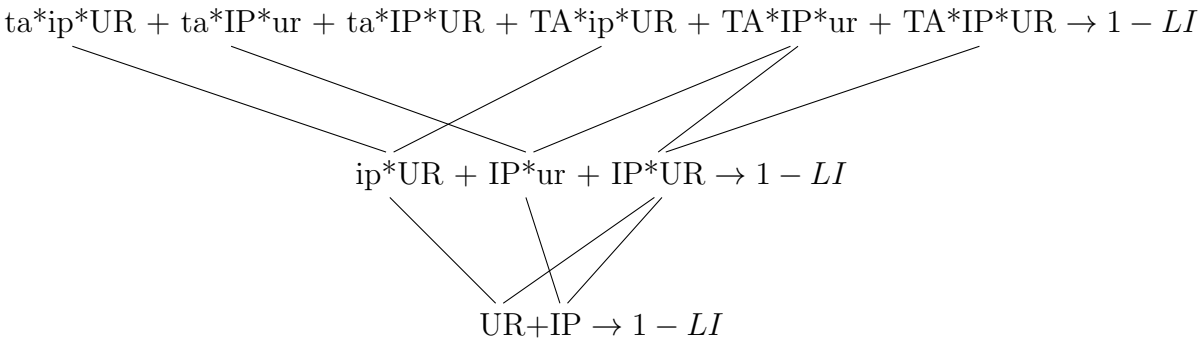
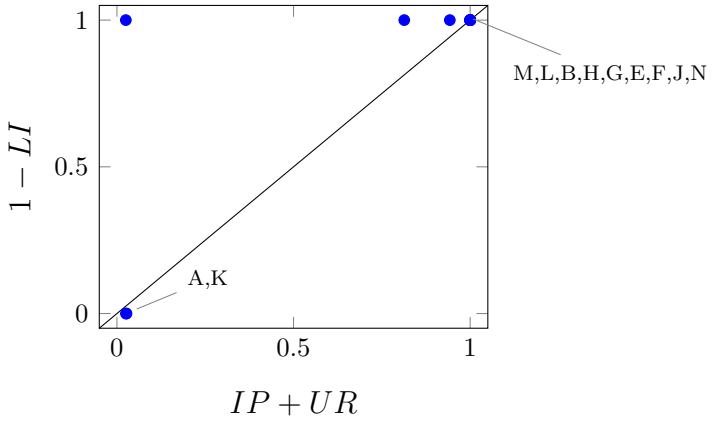


Figure 8-5: Boolean minimization resulting in Equation (8.13)

	Con_S	Cov_S	Cov_U	Cases
IP	0.98	0.29	0.15	C; D; L
UR	1.00	0.75	0.61	M,B,G,J; N,E,H,F; L
Eq 8.13	0.99	0.90		

(a) Parameters of Fit for parsimonious expression of sufficient conditions



(b) Plot of Membership in Set of IP+UR against Membership in Set 1-LI

Figure 8-6: Parameters of Fit for Parsimonious Expression

By incorporating the single logical remainder using an easy counterfactual, the conservative expression is reduced from an expression of multiple conjunctural causality to a parsimonious expression of multiple causality. The parameters of fit for the expression at Equation (8.13) are located in Table 8-6a. The consistency is nearly identical to the conservative expression and the coverage is 0.06 higher than the conservative expression. The increase in coverage is not a large enough for meaningful consideration; however, these parameters of fit show that the parsimonious expression for sufficient conditions demonstrates equal, if not marginally improved, consistency and coverage than the conservative expression.

Similar to the necessary conditions, a final inspection of the parsimonious expression is necessary against the outcome. Recall from earlier in this section that for a sufficient condition to exist it must be a subset of the outcome (i.e. $X_i \leq Y_i$). To conduct a manual inspection Figure 8-6b contains a scatter plot of IP+UR against 1-LI. Analysis of this figure shows only two cases marginally below the main diagonal. This relationship accounts for the high consistency score of the parsimonious

expression. Furthermore, the single point (Case I) in the upper left portion of Figure 8-6b demonstrates a false negative (see row 1 of Table 8.6) consequently this results in less than perfect coverage.

8.5 Conclusion

This Chapter began by discussing the *fuzzification* or calibration of two input conditions from raw scores to fuzzy-set membership scores. The first condition, quality technical architecture, used a negative-end piecewise logistic function for transformation and the second variable, accessible intellectual property architecture, used a positive-end piecewise logistic functions. In both cases, theoretical arguments were made for the calibration process. After calibration, the analysis proceeded to undergo the investigation of necessary conditions for outcome 1-LI. Concluding there was one potential combination of conditions which may be necessary, further discussion is deferred until the subsequent Chapter. The analysis then moved to the investigation of sufficient conditions for outcome 1-LI. Concluding two conditions were potentially sufficient, but again, discussion is deferred to the following Chapter.

Chapter 9

Discussion

The previous Chapter outlined the results of the fsQCA model, but offered little discussion on the meaningfulness or impact on practice of the results. This Chapter is dedicated to examining the results and comparing them against both parameters of fit as well as theory. The discussion, herein, will cover both the candidate necessary and sufficient conditions, looking at accuracy and respective impact of each type of condition on practice.

The second half of the Chapter is dedicated to discuss potential criticisms of the research methodology and outcomes. This explicit discussion of potential criticisms outlines areas of the research which impact validity and; subsequently, generalizability. Following the discussion of criticisms, the limitations of the research are discussed, which enumerate the various threats to generalizability.

9.1 Necessary Conditions

Analysis in Chapter 8 revealed four trilateral disjunctive combinations as candidate necessary conditions for 1-LI (i.e. ta+IP+UR, TA+ip+ur, TA+ip+UR, and TA+IP+UR). Of these four, TA+IP+UR carried the best parameters of fit. For this reason, TA+IP+UR was highlighted as a potential necessary condition for 1-LI. In addition to the high-quality parameters of fit, set TA+IP+UR is supported theoretically, based on the assumptions that high quality technical architecture (TA) or

highly accessible intellectual property architecture (IP) or acquired unlimited rights (UR) are all theorized as methods to avoid lock-in. Unfortunately, identifying necessary conditions is not that simple. The next sections will discuss the analysis of candidate necessary conditions and discuss empirical relevance for the findings.

9.1.1 Analysis and Accuracy

For a condition to be necessary it must make sense theoretically, regardless of parameters of fit. One of the candidate necessary conditions does not match this criterion, TA+ip+ur. This candidate necessary condition expresses the set of cases which are members in the set of high quality technical architecture (TA) or non-membership in accessible intellectual property architecture (ip) or non-membership in the set of cases which acquired an unlimited right (ur). Theoretically, the expression of technical architecture fits, but the two additional causal paths (ip + ur) go against theory and policy. It is irrational to expect a case which lacks an accessible intellectual property architecture or lacks unlimited rights to avoid lock-in. This conclusion is substantiated by the Relevance of Necessity (RoN) measure for TA+ip+ur, which indicates the condition may be necessary, but it is *trivially necessary*. Therefore, the remainder of the analysis of necessary conditions will omit set TA+ip+ur.

Recall from Chapter 8; for a condition to be necessary it must be a superset of the outcome. Expressed formally, all the values of Set **X** must be greater than set **Y**, (i.e., $X_i \geq Y_i$). Inspection of Figure 8-3 revealed three cases in which set membership of TA+IP+UR was less than set membership of 1-LI. To explain this outcome, Figure 9-1 illustrates a Venn Diagram of the three remaining candidate necessary conditions and the outcome, 1-LI. Those cases which had a membership score in sets ta+IP+UR, TA+ip+UR, and TA+IP+UR greater than 0.5 (i.e., the lowest possible threshold for set membership) were said to be members of those respective sets.

Based on this analysis, none of the candidate conditions under study satisfy the formal definition of a necessary condition. First, set TA+IP+UR only covers eleven of the twelve cases that avoid lock-in. Although this set is theoretically sound it is just as much a subset of 1-LI as a superset. Sets TA+ip+UR and ta+IP+UR offer

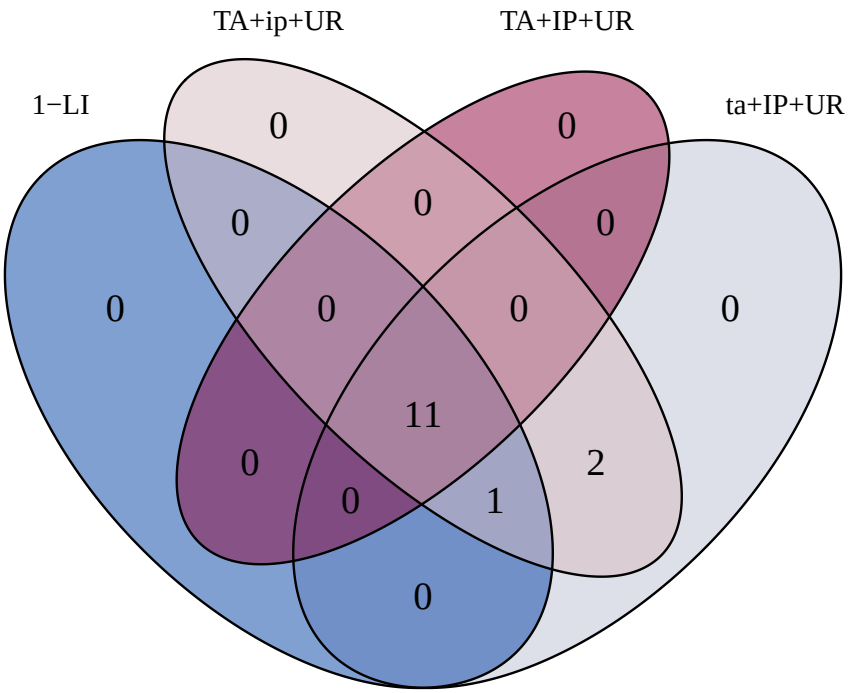


Figure 9-1: Venn Diagram of Candidate Necessary Conditions (where set membership > 0.5)

good coverage, but the causal conditions covered by TA+ip+UR and ta+IP+UR explains all cases, not just the ones where lock-in was avoided. Said in more common engineering terms, sets TA+ip+UR and ta+IP+UR have a low signal to noise ratio (i.e., they capture the signal of interest, but also a great deal of noise). Consequently, TA+ip+UR and ta+IP+UR are indeed supersets of 1-LI; however, they are trivial necessary conditions. This is evidenced by the empirical findings above and further corroborated by the relatively low Relevance of Necessity.

9.1.2 Sensitivity of Necessary Conditions

The only set which comes close to qualifying as a necessary condition is TA+IP+UR; however, as discussed previously it only covers eleven out of twelve cases with outcome 1-LI. QCA literature argues that clear-cut necessary conditions are rare in social science research and often confounded by three factors: causal complexity, imprecise measurement, and randomness [139]. To understand why TA+IP+UR only accounts for eleven of twelve cases, inspection of the fuzzy-set membership scores in

	Candidate Conditions			Outcome
	ta+IP+UR	TA+ip+UR	TA+IP+UR	1-LI
Case A	1.00	0.97	0.03	0
Case D	0.81	0.69	0.81	1
Case L	1.00	1.00	1.00	1
Case M	1.00	1.00	1.00	1
Case B	1.00	1.00	1.00	1
Case N	1.00	1.00	1.00	1
Case E	1.00	1.00	1.00	1
Case H	1.00	1.00	1.00	1
Case G	1.00	1.00	1.00	1
Case I	**0.56**	0.97	**0.44**	1
Case K	1.00	0.97	0.03	0
Case F	1.00	1.00	1.00	1
Case C	0.94	0.40	0.94	1
Case J	1.00	1.00	1.00	1

Table 9.1: Fuzzy-set membership scores in candidate Necessary Conditions

set TA+IP+UR is required; see Table 9.1. Fuzzy membership scores in TA+IP+UR are relative polarized except for Case I, which has a membership score of 0.44 (Case I: TA+IP+UR = $max(0.44, 0.03, 0) = 0.44$). A score so close to the 0.5 threshold for set membership indicates ambiguous membership.

Imprecise Measurement

The ambiguity of Case I membership in set TA+IP+UR is due to weak membership in Set TA (0.44), weak membership in Set IP (0.03), and non-membership in set UR (0). Sets IP and UR are so distant from membership that it is unlikely measurement error factored into the outcome. Conversely, membership in Set TA is largely based on the theoretical cut points used for *fuzzification*, which adopted architectural thresholds from Baldwin et al. [96]. However, Baldwin et al., qualify their choices by saying "in other environments, different choices may be appropriate" [96, p. 14]. For example, if τ_{cp} were adjusted from 0.05 to 0.07 (an arguably trivial 2% difference in core size) then Case I would be a member of set TA (assuming a sufficiency threshold of 0.5) and TA+IP+UR would be closer to a necessary condition for 1-LI. Unfortunately, there is no theoretical or empirical justification for altering the cross-over point for

membership and non-membership in Set TA. Nonetheless, it is important to illustrate the sensitivity of calibration, specifically cross-over point, has on the outcome of the model.

Randomness

Ragin also makes the argument, "it is difficult to model social phenomena with precision even when using measures that are error free" [139], attributing the confounding factor to *randomness* of social phenomena. Many authors argue the use of probabilistic criteria to control for randomness in results and qualifying language to more accurately capture random effects [125], [126], [139]. More specifically, a binomial test is used to assess the probability of observing a specific set of outcomes given an expected probability benchmark. The binomial probability formula is:

$$\binom{N}{r} p^r q^{N-r} \tag{9.1}$$

Where N equals the number of cases with nonzero membership in the outcome 1-LI, p is the selected benchmark proportion, q equals $1 - p$, and r is the number of cases with outcome membership scores that are less than or equal to their membership scores in the causal condition. Recall from Figure 8-3, set TA+IP+UR has nine cases which have a nonzero membership score in set 1-LI and a causal condition membership score that is less than or equal to their membership in 1-LI (i.e. $N = 12$ and $r = 9$). Ragin, drawing from Dion [144], identifies three benchmarks for *quasi-necessary* conditions; see Table 9.2. If the outcome of a binomial probability is significantly greater than the selected benchmark, for example 0.65, then a condition is said to be "usually necessary" [139]. Table 9.2 identifies the outcomes of Equation (9.1) for each benchmark. The only outcome which is significant to the conventional 0.1 or 0.05 standard levels is the linguistic qualifier "more often than not." Given this outcome, the Set TA+IP+UR is said to be a quasi-necessary condition for 1-LI more often than not.

Table 9.2: Outcomes of Equation (9.1) for TA+IP+UR

Linguistic Qualifier	Benchmark	Eq 9.1 Result
"Almost always"	0.8	0.794
"Usually"	0.65	0.347
"More often than not"	0.5	0.073*

* Significant to the $\alpha = 0.10$ threshold

Causal Complexity

Consider what happens when the criteria for necessary conditions (set herein as $Con_N > 0.9$) is slightly relaxed (i.e., $Con_N > 0.89$), see Table 9.3. Notice the emergence of a bilateral disjunctive combination, Set IP+UR, and juxtapose the parameters of fit for Set IP+UR against Set TA+IP+UR. The only change to occur is a reduction in consistency from 0.93 to 0.90. This small change in measure of consistency is arguably inconsequential. Therefore, an argument could be made for Set IP+UR as a necessary condition for 1-LI. Given that Set IP+UR also has nine cases where the set membership is less than or equal to the membership in outcome 1-LI, the same probability thresholds hold true for Set IP+UR as did for Set TA+IP+UR. Thus, making IP+UR a more often than not quasi-necessary condition for 1-LI.

Set IP+UR is a more attractive necessary condition because it is logically easier to comprehend and; ergo, apply to practice. The concept of quality technical architecture is not yet well understood in practice and the inclusion of such a measure in a candidate necessary condition makes that condition less accessible to those in practice. Conversely, IP+UR offers two relatively well understood phenomena in practice.

	Con_N	RoN	Cov_N
IP+UR	0.90	0.98	0.99
ta+ip+UR	0.91	0.36	0.84
TA+ip+ur	0.94	0.25	0.85
TA+ip+UR	0.92	0.34	0.85
Results Below From Table 8.4			
TA+IP+UR	0.93	0.98	0.99

Table 9.3: Necessary conditions under relaxed criteria ($Con_N > 0.89$)

Additionally, this work hypothesized that intellectual property architecture occurred at the conjuncture of technical architecture and contract structure. Given than Set IP partially takes into consideration technical architecture, by using only those files with a high VFI score, the inclusion of set TA seems partially redundant.

9.1.3 Practical Relevance of Necessary Conditions

Up until this point the discussion was largely centered on the model outcomes, with little focus on how the outcomes may impact practice. A necessary condition defines the sample space from which the outcome 1-LI occurs. This implies that 1-LI occurs in the presence of either accessible intellectual property architecture or unlimited rights. Implying, for membership in set avoidance of lock-in to occur, a case must maintain membership in either IP or UR. However, it is important to note that necessary conditions do not provide a "blueprint" or causal paths towards the avoidance of lock-in. Instead, they define the sample space in which the avoidance of lock-in occurred for the cases under study. Therefore, prescriptive recommendations are limited in any discussion of necessary conditions.

Implications for this finding are numerous. First at the portfolio level, this knowledge allows managers of portfolios to better understand what programs demonstrate the necessary conditions to avoid lock-in. This knowledge can aid portfolio managers to focus on those programs which do not demonstrate the necessary criteria to avoid lock-in and direct respective program managers to build more extensive long-term acquisition strategies. Furthermore, it outlines relative simplistic criteria for portfolio managers to look for when approving acquisition plans or strategies (e.g., if a portfolio manager is presented with an acquisition plan that does not utilize an accessible intellectual property architecture, or plan on acquiring unlimited rights, he or she knows that neither of the conditions which are necessary to avoid lock-in are present). This does not imply the acquisition strategy is incorrect or that the program will become locked-in, but instead allows the portfolio manager to ask more informed questions about how a program plans to avoid lock-in given the absent conditions which are necessary.

However, it is impossible to assess TA or IP until code is written for a given program. For some developmental programs that means a program may get all the way through the Critical Design Review before an opportunity exists to make an assessment on technical architecture or intellectual property architecture. For sustainment or modernization programs the ability to assess TA or IP is available much sooner, if not at contract start; making such cases stronger candidates for application of the practical findings discussed herein.

At the program level, this knowledge can help program managers decide whether it is necessary to make an initial purchase of data rights above and beyond what is outlined in the standard DFARS clauses or no purchase at all. For example, a program manager for a case in set TA+IP+ur or ta+IP+ur can conclude that the program maintains one of the conditions necessary to avoid lock-in. If membership in set IP is based on open source licenses, further investigation would be needed into the exact type of open source license. However, the knowledge of set membership would allow the program manager to prioritize his or her time investigating the type of open source licenses and build a long-term acquisition strategy around continued use of open source or U.S. Government copyright code; rather than undergoing a lengthy cost-benefit analysis on the purchase of unlimited rights. This does not imply that the purchase of unlimited rights would not assist the program, but rather that it may be an unnecessary step in order to avoid lock-in and; therefore, an unnecessary capital expenditure.

It bears discussing that the use of open source in DoD software is not a "free" decision. Program managers and engineers do have the ability to use open source in DoD software, as many of the cases examined, herein, demonstrate; those choices must weigh the security risks of employing code in the public domain against the subsequent cost savings. The choice to use open source may lead to a lengthier network accreditation process, in some cases the cost of the longer accreditation process may outweigh the cost savings of using open source. This cost-benefit decision is highly situation dependent. The main point being that using open source may not always be a good decision, even if it does increase the likelihood of avoiding lock-in.

The absence of necessary conditions in a case does not imply that a case will result in lock-in. Due to the assumption of asymmetry in equifinality it is incorrect to conclude that lock-in will occur in cases that lack either an accessible intellectual property architecture or unlimited rights. A separate analysis of cases with outcome LI is required to reach that conclusion; however, such analysis is limited, herein, by the small number of cases with membership in LI. Although all cases with outcome LI were members of set ta+ip+ur, there were not enough cases to consider this sample sufficient for analyses and consequently to draw inferences from. This is a clear point where future research is needed. Future research may well substantiate that set ta+ip+ur is a necessary condition for lock-in; however, the results outlined herein do not directly address that outcome.

Finally, it is once again important to point out that the set IP+UR as well as set TA+IP+UR were said to be quasi-necessary necessary conditions. This implies that there are still cases in which both sets may not be necessary. Although the findings on necessary conditions, herein, are a useful guide for practice, they require more certainty before they can be considered as empirical findings. Future research is needed where a larger sample of cases is used. One benefit of fsQCA is that it is extensible to a large-N sample size with few, if any, changes to the model specification. This is an area of potential future research.

9.2 Sufficient Conditions

Recall from Chapter 8, for a condition to be sufficient it must be a subset of the outcome. Expressed formally, all the values of Set **X** must be less than set **Y**, (i.e., $X_i \leq Y_i$). Analysis in Chapter 8 revealed two expressions for sufficient conditions: conservative and parsimonious. The conservative expression contained four prime implicants. The term *prime implicants* is used to describe each conjunctive combination in the final expressions. Similar to the analysis of necessary conditions, each prime implicant must be supported by theoretical evidence in addition to the parameters of fit. The prime implicants for the conservative expression were ip*UR, IP*ur, TA*IP,

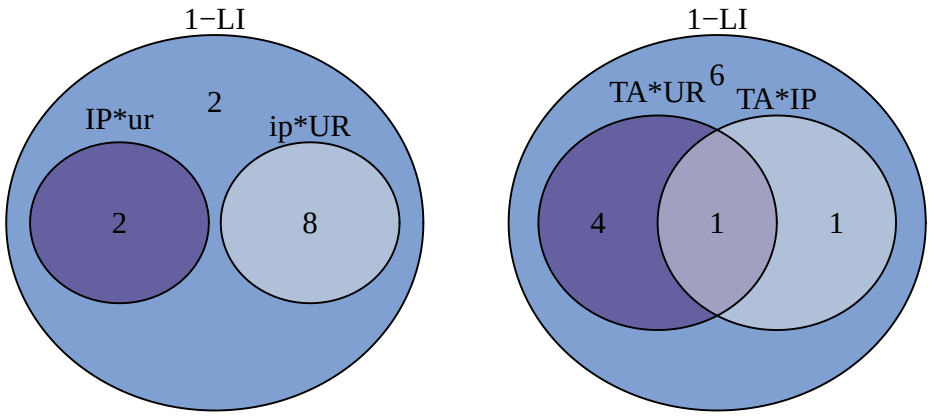
	Cases				
	M,B,G,J	C	N,E,H,F	D	L
ip*UR	x	-	x	-	-
IP*ur	-	x	-	x	-
TA*IP	-	-	-	x	x
TA*UR	-	-	x	-	x

Table 9.4: Prime Implicant Chart

and TA*UR. Two of these prime implicants are said to be *essential prime implicants* (ip*UR and IP*UR) and two are said to be *inessential prime implicants* (TA*IP and TA*UR). Essential prime implicants imply fundamental products which are not covered by any other prime implicants.

The concept of essential prime implicants is easiest explained using Venn Diagrams; see Figure 9-2. In these Venn Diagrams, Figure 9-2a shows the sample space of the two-essential prime implicants, which represent 10 out of twelve cases[1]. Each essential prime implicant covers a unique group of cases; whereas, Figure 9-2b shows the inessential prime implicants which cover mostly redundant cases. The only case unique to the inessential prime implicants is Case L, which is covered by both inessential prime implicants. Therefore, the addition of TA*IP or TA*UR only adds a single case to the coverage of sufficient conditions and the inessential prime implicants are interchangeable given that they both add coverage for the same case. These relationships are also succinctly displayed in Table 9.4, which shows the coverage of each prime implicant. Table 9.4 also shows that the only coverage unique to TA*IP and TA*UR is redundant coverage of Case L. This examination of prime implicants is empirical proof that both conservative expressions are logically equivalent expressions of the same solution. Given this logical equivalence, it is necessary to examine whether one of the conservative expressions is better explained by theory.

[1]Recall from Table 8.6 that Case I, which has outcome 1-LI, had membership in a corner of the vector space that was also occupied by three Cases which had outcome LI. Therefore, the twelfth Case of 1-LI is not explained by the sufficient conditions



(a) Essential Prime Implicants (b) Inessential Prime Implicants

Figure 9-2: Venn Diagrams for Sufficient Conditions (not drawn to scale)

9.2.1 Sensitivity Analysis and Accuracy

Each of the prime implicants must pass theoretical examination for consideration as a sufficient condition. Beginning with IP*ur, this is the set of cases that have membership in the set of accessible intellectual property architectures (IP) and non-membership in the set of unlimited rights (ur). This conjunctive combination of conditions represents one possible causal path for avoidance of lock-in. Although this set does not have an unlimited right, a majority of software code with the highest VFI scores are either open source or copyright to the U.S. Government. For reasons argued in Chapter 4, this set of cases can avoid lock-in by using one of many competitive copying techniques or by transference of copyleft licenses to the derivative work. While this has not been proven empirically, prior to this research, the logic of the argument is both testable and falsifiable. Furthermore, analysis of the parameters of fit for IP*ur, suggest a high score consistency (0.97), but a relatively low score for coverage (0.15). Many authors make the point that high parameter of fit scores are often a trade-off between coverage and consistency [117], [125], [138], [139]. However, both Figure 9-2a and Table 9.4 indicate that IP*ur only covers two out of twelve cases with outcome 1-LI. The low raw coverage and unique coverage scores indicate that IP*ur is not the most common causal path to the outcome, given the cases examined. Nonetheless, IP*ur covers a set of cases which would otherwise be left out of the

conservative solution. Consequently, IP*ur is determined to be a theoretically sound prime implicant.

The second prime implicant, ip*UR, represents the set of cases with membership in the condition of not accessible intellectual property architecture and membership in the condition of unlimited rights. For reasons outlined in Chapter 2, the possession of an unlimited right grants the DoD the ability to do whatever it chooses. This is supported both by statute and empiric observation. The existence of an unlimited right, regardless of membership in the set of accessible intellectual property architectures, should always yield outcome 1-LI. The results of the fsQCA model used, herein, match that theoretical argument. Consequently, ip*UR is determined to be a theoretically sound prime implicant. Furthermore, an analysis of the parameters of fit for ip*UR indicate a maximal consistency score (1.0) and a moderate coverage score (0.61). The relatively higher coverage scores for ip*UR over IP*ur indicate this prime implicant plays a larger part in explaining the outcome. This is reflected in the unique coverage score of ip*UR, which indicates ip*UR explains the largest proportion of the outcome. Consequently, ip*UR is determined to be a theoretically sound prime implicant. Said plainly, ip*UR is a more important sufficient condition than IP*ur, as it explains a greater proportion of the outcome than any of the other prime implicants.

Since TA*IP and TA*UR are both inessential prime implicants less time is dedicated to their discussion. Given the arguments outlined in Chapter 5, both pass a test of theoretical sufficiency. One would expect membership in the set of high quality technical architecture (TA) in conjunction with membership in either the set of accessible intellectual property architecture (IP) or membership in the set of unlimited rights (UR) to result in the avoidance of lock-in. The discussion of each essential prime implicants demonstrated that both IP and UR lead to avoidance of lock-in absent the other condition. The inessential prime implicants suggest that both IP and UR, when combined with TA, help explain an additional causal path towards avoidance of lock-in. However, an analysis of the parameters of fit show relatively high consistency (TA*IP: 0.99 and TA*UR: 1.0), but relatively low raw coverage

scores (TA*IP: 0.23 and TA*UR: 0.45). Both raw coverage scores are higher than the raw coverage score of IP*ur, but for reasons already explained they are inessential prime implicants because they yield only redundant cases. Consequently, both TA*IP and TA*UR are considered theoretically sound prime implicants; although, inessential given the sample of cases studied herein. However, since both inessential prime implicants add redundant coverage, only a single one is needed. Given the cases under study, implicant TA*UR demonstrates greater parameters of fit and is, therefore, likely a more important sufficient condition than TA*IP.

9.2.2 Findings

The four prime implicants (ip*UR, IP*ur, TA*IP, and TA*UR) identified are all sufficient conditions to avoid lock-in. Each of these paths offers an empirically proven combination of conditions which was successful on one or more previous cases at avoiding lock-in. The first sufficient condition, ip*UR, is a somewhat obvious finding. The outcome ip*UR $\rightarrow 1 - LI$ is not an experimental or socio-technical phenomena, but rather one directed by statute. Given this directed outcome, the sufficient condition ip*UR is not considered a finding of this research. Although not a direct finding, the appearance of ip*UR, in both the conservative and parsimonious expressions, is a good triangulation of the fsQCA model specification. Had ip*UR not been present it would have indicated a fsQCA model with a poor theoretical fit.

The second sufficient condition identified, IP*ur, is a direct finding of the research herein. At the outset of this investigation, intellectual property architecture was postulated as a mechanism of action for intellectual property locked-in. Specifically postulating that those systems with a highly accessible intellectual property architecture would avoid lock-in. The sufficient condition, IP*ur, provides empirical, but not statistically significant, confirmation for this theory. Accessible Intellectual Property Architecture, absent any other explanatory conditions, is a sufficient condition; ergo, a mechanism of action for the avoidance of lock-in. Much of the DoD literature on "Open Architectures" argued the advantages of using open source code as well as open standards and theorized it would avoid lock in, but was unable to empirically

prove there was a connection [145]. The finding herein is the first time the outcome was proven empirically; thus, this finding is neither obvious nor trivial.

Finding 1: Accessible intellectual property architecture, absent unlimited rights, is an empirically and theoretically sufficient condition for the avoidance of lock-in

Although discussed in terms of inessential prime implicants, both TA*IP and TA*UR are sufficient conditions to avoid lock-in, albeit partially redundant. The inclusion of condition TA in both sets is important because it acknowledges the role high quality technical architecture may contribute to the avoidance of lock-in. Chapter 5 briefly discussed the theories of modularity and hierarchy, arguing systems with both high levels of modularity and hierarchy are easier to maintain and have less complexity. Although set IP measures the proportion of areas with accessible files, it does not capture the quality of the architecture. The quality of the architecture was hypothesized in Chapter 5 as a key factor in creating a taxonomy of architecture based on both quality of technical architecture and accessibility of intellectual property architecture. Recall from Figure 9-3, Hazard Patterns were theorized as set ta*ip*ur and Ideal Patterns as set TA*IP*ur; with boundary patterns being sets TA*ip*ur and ta*IP*ur. Inspection of the truth table, Table 8.6, reveals two of three cases in set ta*ip*ur were locked-in. For reasons explained in the section on necessary conditions, Case I had arbitrary membership in this set and with a relatively small change in set calibration could change sets. Furthermore, ta*ip*ur is absent from the conservative expression of 1-LI, Equation (8.8). This empirically supports the taxonomic formulation of hazard patterns. Additionally, all cases in set TA*IP*ur avoided lock-in and TA*IP*ur is present in the conservative expression for 1-LI, Equation (8.8). This empirically supports the taxonomic formulation of ideal patterns. Unfortunately, no cases were present in set TA*ip*ur, making conclusions on the suitability of this formulation difficult. Furthermore, a single case was present in set ta*IP*ur, which avoided lock-in. A single case in the Boundary Pattern makes

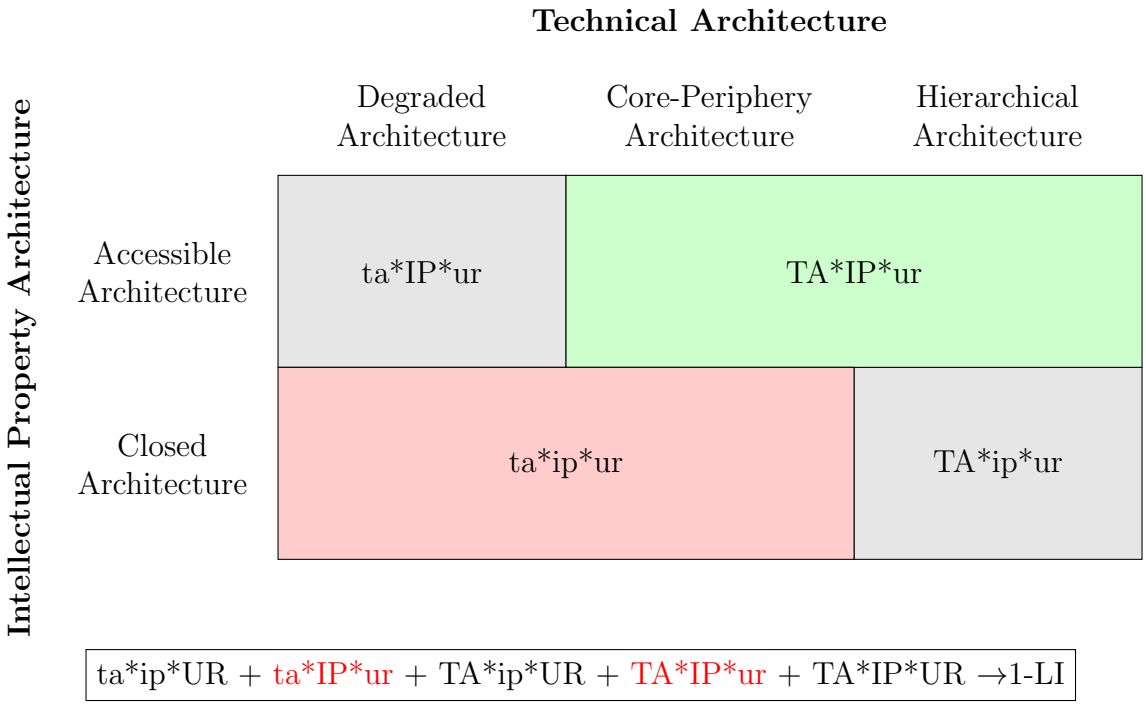


Figure 9-3: Taxonomy of Architectural Patterns (assuming no Unlimited Rights) compared against the Conservative Expression, Equation for 1-LI, (8.8)

the formulation difficult to substantiate given the current sample. The occurrence of TA*IP and TA*UR in the conservative solution supports the formulation of the taxonomy, but ideally more research is needed to confirm the exact breakpoints between Hazard, Boundary, and Ideal architecture patterns. The research, herein, does not contradict the hypothesized taxonomy, but the sample size of cases is too small to make broad claims of generalizability. This is an area of future research.

Finding 2: Taxonomy of Hazard Patterns and Ideal Patterns of Architecture are supported empirically

Lastly, a discussion of the parsimonious expression identified in Chapter 8 is necessary. The addition of a single logical remainder (Set ta*IP*UR), reduced the conservative expression from three bilateral conjunctive arguments to two arguments. Although the parsimonious expression is not supported empirically, it is rational to expect an outcome of 1-LI given the strong theoretical support. Thus, the parsimonious expression is supported theoretically. The minimization down to two conditions

makes the impact on practice much simpler to understand than the conservative expression. The implications of the parsimonious expression are important because if the true expression for avoidance of lock-in is IP+UR, then these conditions may be both sufficient and necessary (recall from the discussion on necessary conditions that IP+UR was a candidate necessary condition). The existence of conditions, which are both necessary and sufficient, imply IP+UR $\leftrightarrow$ 1-LI or that IP+UR equivalent expressions of the same sample space as 1-LI. This is a much stronger finding and allows for a larger potential impact on practice. However, this research does not conclude IP+UR $\leftrightarrow$ 1-LI, due to the weak support for IP+UR as a necessary condition and the lack of empirical support for IP+UR as a sufficient condition. Although strong enough theoretical support exists to declare the parsimonious expression a finding, future research should endeavor to provide empirical support for the expression ta*IP*UR $\rightarrow$ 1-LI.

Finding 3: Accessible intellectual property architecture OR unlimited rights are theoretically, but not empirically, sufficient conditions to avoid lock-in

9.2.3 Practical Relevance of Sufficient Conditions

Each finding has a direct impact on practice. Beginning with Finding 1, the empirical proof that accessible intellectual property architecture, absent an unlimited right, is a sufficient condition to avoid lock-in offers portfolio and program managers new options when confronted with the task of long-term planning. Previously, managers were confined to discussing long-term acquisition strategy options largely in terms of classes of rights (e.g., a manager for a large system would have to foresee the rights the program needed over the life-cycle to ensure future competition). Consequently, the acquisition of rights was often thought of as the only way to avoid lock-in. Finding 1 does not countermand this previous guidance, but instead adds viable options for the manager to achieve the same desired outcome, but with a different approach.

Prescriptive Impact

Given Finding 1, managers are now capable of taking preventive actions to avoid lock-in without securing any additional rights to intellectual property. Prescriptively, managers that seek to avoid lock-in, without securing additional rights to intellectual property, may choose to formulate system requirements in a way to require or incentivize the use of open source or U.S. Government owned code in certain parts of the software. As an example, managers may use prescriptive contract language such as the following:

1. The core size of the software (measured in file size) shall not exceed 10% of the entire system (measured in file size).

2. The core of the software should contain greater than 50% open source software.

The first requirement above is an example of a threshold requirement (i.e., compliance is mandatory); whereas, the second requirement is an example of an objective requirement (i.e., compliance is optional, but oftentimes incentivized). The introduction of objective requirements formulated towards creating an accessible intellectual property architecture, coupled with an incentive structure, provide a means to both objectively measure the outcome and incentivize the creation of an architecture to avoid lock-in (e.g., a contractor could be incentivized by offering $1 thousand for every tenth of percentage point the core size is below 10% or $50 thousand for every percentage point increase in open source above 50% of the core). This would imply a 1% reduction in core size and would net a contractor $100 thousand. Prior to this research, there was no means to objectively measure the accessibility of software in a meaningful way for contract language; nor was there a way to specify an architectural location for the desired open source software. The contract language exemplified above provides both an objective measure and the ability to specify location.

It is important to note, there is no guarantee contractors will accept the incentive or work towards the incentivized goal. Although these types of requirements are advantageous for the DoD, a contractor may conduct a cost-benefit analysis on the revenue accrued as a result of incentives compared to the long-term revenue accrued as

a result of creating a system without an accessible intellectual property architecture. Such a cost-benefit analysis may favor creation of a system that ignores accessible intellectual property architecture and potentially yields lock-in. Depending on the incentive amounts and life-cycle of a given system, this may be a rational economic choice. To avoid this situation, further research is needed into the optimal quantity of economic incentives which would give contractors adequate incentive to build systems with accessible intellectual property architectures.

Care must also be taken to ensure the requirements outlined above do not create the wrong incentive (e.g., a contractor could pollute a code base with unnecessary open source code). While this increased level of open source may increase the likelihood of avoiding lock-in, it may also impact the performance of the system. Consequently, any use of the above contracting requirements must be employed in conjunction with a rigorous performance testing scheme to ensure no system degradation results. Although a real threat, it is partially mitigated by the use of core size. Which files will end up in the core is not always obvious to programmers when building the code. In fact, it is often an emergent property based on the incomprehensible indirect dependencies. Understanding the full set of indirect dependencies in a piece of software is beyond human cognition; therefore, accurately predicting which files will end up in the core is also a difficult task for humans. The inability to foresee and predict which files will be core or shared offers some mitigation against open source pollution or gaming incentives.

Taxonomy of Architectures

Earlier discussion of architecture argued the role of the system architect was to make performance trade-offs in the mapping of function to form. Managing these trade-offs is imperative to creating a system that meets defined requirements and stakeholder needs. Given Finding 2, the taxonomy of architecture outlined in Chapter 5 provides system architects and engineers with a rough outline of the available "trade space." Consider an example where an engineer must trade quality technical architecture to achieve a specific performance goal. In this example, architectural complexity is

traded for performance. The taxonomy of architectural patterns, outlined herein, provides a guide for the engineer to make these trade-offs while understanding the implications on long-term competition and lock-in. In continuance of the example, suppose a degraded architecture is required to secure a specified threshold of performance, using the taxonomy the engineer knows that to ensure long-term competition, he or she can create a boundary pattern by using an accessible architecture. Failure to take this action relegates the architecture to a hazard pattern, which research, herein, has demonstrated as a pattern unsuccessful at avoiding lock-in.

The next logical task for the system architecture becomes a question of how to create an accessible intellectual property architecture (e.g., where to place the open source code). Recall from Chapter 5, that the presence of open source is not necessarily meaningful. Rather, the type of open source and the location of the open source are of importance. Chapter 5 introduced those files which will endure the longest in a code base as those with the highest "hardness-to-kill" or files with the highest VFI scores. In core-periphery architectures these are the shared and core groups of files. These are the files system architects should target for inclusion of open source. No argument is made, herein, for prioritizing open source in share over core or vise-versa. Logical arguments could be made for either prioritization. For hierarchical systems, the boundaries are less clear. Instead of using core or shared groups, the goal becomes introducing open source in those files with the highest VFI or those highest in the hierarchy.

Understanding the potential "trade space" allows system architects, engineers, and program managers to make more informed decisions throughout the life-cycle of a program about the likelihood of long-term sustained competition. The addition of intellectual property to the trade-space of engineers and system designers is a novel concept. This does not imply that engineers and system designers must now become experts in the legal domain, but that choices they make are part of a larger scheme of multiple conjunctural causality that could eventually lead to lock-in. Knowing even the rough implications of architectural choices on lock-in and knowing the impact of interactions between choices, should allow engineers and system designers the ability

to make rational decisions, that both allows for performance and complexity trade-offs, while still considering the goal of long-term sustained competition.

Acquisition Planning

Understanding the list of conditions which are sufficient to avoid lock-in, helps both portfolio and program managers refine the sample of options available when conducting long-term planning. Consider the example, where a portfolio manager is presented with an acquisition plan for a program with planned membership in set ta*ip*ur. He or she knows based on this research that this configuration of conditions is not in the sample space of cases that avoid lock-in. That is not to suggest avoidance of lock-in is impossible, but that this specific acquisition plan may require further scrutiny on long-term competition above and beyond the normal standard. Whereas, an acquisition plan with membership set ta*IP*ur is well within the sample space of cases that avoid lock-in and may not require additional scrutiny. In essence, the combinations of conditions and sample spaces outlined in the findings, herein, provide a subjective measure for portfolio managers to assess the planning of program managers. Although these are subjective guidelines at this point, they are an improvement over the current state of practice which relies on anecdotal heuristics, domain experience, and intuition to inform judgment.

9.2.4 Impact Outside DoD

The literature review in Chapter 2 enumerated the many ways in which the intellectual property schema in the DoD differs from standard commercial intellectual property schema. Most obvious is the concept of Unlimited Rights, there is no such class of compulsory Government rights in commercial intellectual property statute. Also in stark contrast, in some situations non-DoD firms consider the existence of open source software as a potential liability as opposed to an asset. Consider the lawsuit initiated by the Free Software Foundation FSF against Cisco Systems on December 11, 2008,

in the United States District Court for the Southern District of New York[2], where Cisco was accused of violating the terms of the GPL license. Cisco acquired Linksys in March 2003 for $500 million and shortly thereafter, in June 2003, received complaints that Linksys violated the terms of GPL. Cisco eventually settled the lawsuit out of court with the Free Software Foundation, but the scenario has become a commercial acquisition parable of "buying a lawsuit." This problem is starkly different from those faced by the DoD; however, the methods introduced, herein, could prove equally useful in identifying such copyright notices and assessing their transfer in derivative works.

Specifically, the methods outlined, herein, could be used to conduct due diligence on software acquisitions to identify potential open source copyrights and their objective location within the architecture. For reasons discussed in Chapter 5, a GPL license in the core would be far more alarming than a MIT license in the control. Having this knowledge during the acquisition process would be invaluable in calculating the associated risks. Additionally, the methods, herein, are both automated and have proven scalable up to 8 million lines of code. This lowers the cost, in terms of time and money, to identify potential copyright violations. Such a relatively low cost due diligence process, could have a dramatic impact on future software acquisitions.

9.3 Potential Criticisms

The first potential source of criticism is the use of crisp-set outcome condition in a fuzzy-set QCA. By using a dichotomous outcome condition, it makes identifying truly necessary conditions difficult. In necessary conditions the fuzzy-set membership score must be greater than or equal to the outcome; however, if all the outcomes are "1" then all fuzzy-set membership scores must also be "1" to qualify as a necessary condition. Given the nature of fuzzy-set calibration it is unlikely that many fuzzy-set membership scores will meet that threshold. To address this criticism, the dichotomous outcome condition could have been calibrated into fuzzy-set membership scores, but the thresholds for calibration would be arbitrary and the resulting scores would

[2]Full complaint available at: `http://www.fsf.org/licensing/complaint-2008-12-11.pdf`

still represent polarized set membership (i.e., instead of "1", fuzzy-set membership score may be 0.95). Such a forced calibration is not supported by QCA literature and does not address the heart of the criticism.

The only way to properly address the above criticism is to identify a variable measure of lock-in. However, during the course of this research no such measure was obvious. One potential solution, instead of asking program managers if their programs were locked-in, would be to ask program managers to rate what degree they are locked-in on a scale of one to ten. These scores could then be calibrated into fuzzy-set membership scores. However, it would be difficult to maintain accurate scoring across program managers and the meaningfulness of a three out of ten score versus a four out of ten score is arbitrary, at best.

Another potential method is to create a variable measure of lock-in based on the cost of the technical data package. For example, a locked-in program with a technical data package costing 25% of the total program cost (i.e., $25 million on a $100 million program), is arguably more locked-in than a program with a technical data package costing 5% of total program cost (i.e., $5 million on a $100 million program). This normalized measure of lock-in, as a percentage of total program cost, is arguably more objective than the method outlined above. However, the pricing of data rights and technical data packages is more of an art than a science. Subject matter expert interviewees noted one of the biggest tensions between the DoD and Industry is the lack of a "common methodology for pricing of data rights." This sentiment is echoed in an Air Force Handbook on Technical Data:

> "There is no statute, regulation, or policy applicable to DoD contracts mandating that the parties utilize a specific approach for calculating a fair and reasonable price for the rights in technical data and computer software delivered under DoD contracts." [146, p. 59]

The lack of a common method to value intellectual property will partially confound this approach to measuring lock-in. The value calculations for data rights may be $10 million using one methodology and $5 million using another. Which is the correct

measure? No argument is made herein. Instead this work points to this as a viable method of measuring lock-in if a common method to value intellectual property is ever adopted by the DoD. This is an area which should be targeted in future research.

A second area of criticism is the calibration of Set TA. Although already discussed in the section on necessary conditions, it bears further discussion as this small change could change the final result of the research. Simply by changing the τ_{cr} point from 0.05 to 0.07 the conservative sufficient expression becomes:

$$\text{TA} + \text{ip*UR} + \text{IP*ur} \rightarrow 1 - LI \tag{9.2}$$

This reduces the prime implicants from four to three and reduces the number of inessential prime implicants to zero. Although the consistency and coverage parameters of fit are similar between the conservative solution reached in this research and that in Equation (9.2), set TA is now an essential prime implicant. The small change in cross-over point recategorized Case I from ta*ip*ur to TA*ip*ur, providing an existence proof that set TA*ip*ur also leads to 1-LI.

This new essential prime implicant is important because the existence of TA, as a sufficient condition, suggests that one possible path to avoid lock-in is to use a high quality technical architecture absent of an accessible intellectual property architecture or unlimited rights. This outcome was not theorized at the outset of this research and could have strong implications on practice. The existence of one case is only proof that this outcome is possible; however, it is impossible to argue that the classification of one case could not be a result of one or more threats to external validity discussed in Chapter 6. Although this research highlights the significance of this potential set of conditions, the research, herein, lacks evidence to support tampering with the cross-over point already established in literature and leaves this area for future research.

The operationalization of the set UR is a point of certain criticism from DoD legal experts. There are many different paths to secure membership in set UR; however, for this research set UR was operationalized as the purposeful *acquisition* of an unlimited right. This set did not include cases that received an unlimited right due to

expiration of a restricted right, mismanagement on the part of the contractor, or any other method that did not directly acquire an unlimited right as a condition to, or a deliverable of, a contract. Operationalization of this type narrows membership in set UR and ignores some cases which may otherwise have an unlimited right. This choice for operationalization sought to intentionally exclude those cases which obtained an unlimited right by accident. Cases A and K were examples of such a scenario. These cases obtained an unlimited right arguably due to contractor mismanagement, not foresight or planning of the program office. In excluding such cases, it also excludes other cases with unlimited rights. Future research may identify an improved method to capture the set of cases which purposefully acquired an unlimited right and exclude those that did not, but at the time of writing this were the most effective and logical operationalization of set UR.

9.4 Limitations

Limitations are those factors of the research which may impact the generalizability of the research results. The first threat to generalizability is the number of cases present in some of the sets of conditions. Specifically, sets ta*IP*ur, TA*IP*ur, and TA*IP*UR are only covered by a single case. With only once case present it is difficult to argue that these are unique results and not a byproduct of one or more threats to external validity or a byproduct of randomness in data. In QCA research, each set of conditions would ideally be supported by multiple cases (n) and only those sets with a number of cases above a selected threshold would be factored into the results (e.g., $n > 5$ or $n > 10$). If five or ten cases all demonstrate the same configuration it is more difficult to argue these results are random as this allows for control against randomness in the data.

Perhaps the largest threat to generalizability is the possibility of alternate hypothesis. Although QCA research uses a rigorous method of investigation across many cases, the outcomes are limited to only those conditions included in the model, which is a criticism of any variable oriented approach. This limits the findings of

this research to three possible conditions. In all likelihood, there are more than three conditions which contribute to lock-in. However, this condition space was kept parsimonious, herein, to avoid the problem of limited diversity and reduce the likelihood that the model would fit an outcome to random data. The inclusion of additional conditions is recommended for future research, but to do so requires a larger sample of cases.

9.5 Future Research

There are general avenues of investigation which would improve the validity of the findings, herein, and specific avenues of investigation which would lead to additional findings. Generally, an increase in the case sample size, as well as case diversity, would address many of the threats to generalizability and criticisms outlined in previous sections. Unfortunately, the sample of cases used, herein, demonstrated limited diversity in outcome LI. However, limited diversity in a sample is a risk anytime cases are not selected based on the outcome condition (dependent variable). At first glance, this may seem like a biased sample, but recall from Chapter 3 that cases of lock-in only occur as a single digit percentage of all DoD contracts. Consequently, the case sample, herein, may well mirror practice or in-fact represent an oversampling of cases with outcome LI. Nonetheless, a larger sample of cases with outcome LI is needed, followed by an exhaustive investigation into the specific conditions or sets of conditions, that specifically yield LI. Doing so is the only way to properly investigate the conditions which directly result in lock-in.

Furthermore, adding even as few as ten cases to the sample, regardless of outcome condition, strengthens the findings outlined herein, but also provides ample "condition space" (similar to degrees of freedom) for the inclusion of another causal condition. This research makes no argument for which potential causal condition should be investigated, but the inclusion of any causal condition, regardless of a positive or negative finding, allows for findings which are more generalizable and addresses one of the largest threats to validity. Even if another potential causal condition is not

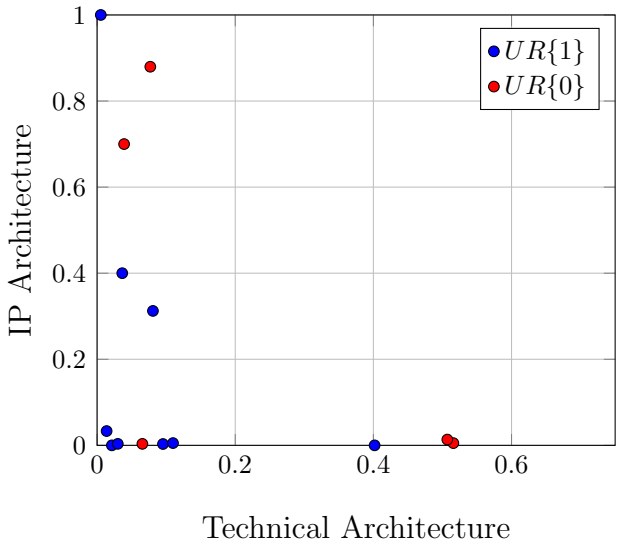


Figure 9-4: Plot of "Condition Space" Coverage

used, rerunning the same experiment as the one herein, with ten more cases, would increase the validity of the findings.

The addition of any configuration of cases would improve the research; however, special attention should be paid to collecting more cases which do not posses an unlimited right. Figure 9-4 is a visualization of the "condition space" covered by the cases herein. From this figure it is clear the cases with an unlimited right may be oversampled and cases without an unlimited right offer only sparse coverage of the condition space. To address this any future research should investigate more cases without an unlimited right to better understand what impact, if any, these cases would have on the findings.

Although future research is suggested in many areas, two avenues of specific investigation are prioritized above all others: further investigation of necessary conditions and research on calibration of set IP. The research, herein, was able to identify many candidate necessary conditions, but was unable to conclude with any certainty the existence of a necessary condition. It is plausible that no necessary conditions exist given the set of conditions studied. However, without a better operationalization and calibration of the set LI, it makes further analysis on necessary conditions difficult. What is needed is further research into a gradient scale measure of lock-in that is

generalizable across DoD or, ideally, across all software systems. This scale could be an operationalization of the current spectrum of rights outlined in the DFARS or a completely novel approach. However, until a non-dichotomous condition is used, the identification of necessary conditions with any certainty remains out of reach.

The second area of specific future research is the calibration of Set IP. The three calibration points chosen for the model, herein, are logical, but arguably arbitrary. Further, the cross-over point of 0.5 is less than ideal for fuzzy-set membership because it may yield ambiguous set membership. The choice for a cross-over point of 0.5 is not drawn from academic research, but rather the author's judgment based on both experience in the domain and theoretical support. Although absent from peer-reviewed academic literature, the answer to effective calibration may exist somewhere in legal precedent. Analysis of legal cases and subsequent precedents was outside the scope of the literature review conducted herein and the skills of the author; however, more thorough legal research is needed to determine if such a precedent exists (e.g., legal precedent may suggest more than a simple preponderance [0.6 versus 0.5] of open source code is needed to cross-over from non-membership to membership in the Set IP or perhaps a lower cross-over point is prudent). The calibration of set IP carries more significance than an input to a QCA model. It can also be used to refine the architectural pattern taxonomy outlined herein, provide clearer guidance on the "trade space" available to engineers and system designers, and inform drafting of prescriptive contract requirements to avoid lock-in.

Finally, it bears discussing the extension of the methodology and approach outlined, herein, outside of the software domain. With the exception of open source software rights, the same methodology should be theoretically capable of analyzing hardware systems. Although neither the technical architecture extraction nor the intellectual property architecture extraction would be automated in a hardware system, the presentation and exploration of intellectual property architecture views are possible. Researchers have already demonstrated the extraction of architecture in large complex hardware systems (see generally [147]); the only novel obstacle which remains is the extraction of intellectual property. However, this step may be more

difficult in hardware systems because there is usually no explicit intellectual property notice similar to a copyright in software. Furthermore, hardware protected by intellectual property is likely covered by a patent, which operates under an arguably more complex intellectual property protection schema than copyrights. Future research is needed to identify the challenges in extracting intellectual property information from hardware, as well as identifying commonalities across lock-in or the avoidance of lock-in between hardware and software.

9.6 Conclusion

This Chapter began with discussion of the candidate necessary conditions previously identified. Discussing both the theoretical rational for the candidate conditions as well as the usefulness of the parameters of fit. Given the design of the research model, none of the candidate necessary conditions had enough empirical support to meet the criteria of a necessary condition. In an effort to provide a quasi-necessary condition a binomial probability test was used to investigate the top candidate under probabilistic criteria.

Following the discussion of necessary conditions, each sufficient condition was discussed in detail. Generally, the parameters of fit for all essential prime implicates met the subjective standards for consideration as a sufficient condition. Consequently, more discussion was focused on the theoretically foundations of each condition. Ultimately concluding with three findings, followed by a lengthy discussion of the implications of each finding on practice. Where possible, examples of how the findings would be applied to practice were provided.

The chapter concluded with a discussion of both the limitations of the research as well as the anticipated criticisms. The discussion of these two factors served to both explain the choices made by the researcher, but also to inform a discussion on future research.

Chapter 10

Conclusion

This research set the general goals of investigating the mechanisms that link formal software architecture and intellectual property to induce intellectual property lock-in as well as to develop a method for objectively measuring and identifying potential incidences of lock-in.

A formal investigation was organized around these two goals. Beginning with a review of literature on intellectual property within the DoD, which concluded by highlighting gaps in literature, as well as research. One such gap was an understanding of the magnitude of the problem; to address this Chapter 3 was dedicated to a quantitative framing of the intellectual property lock-in problem in DoD contracts, which established both a magnitude of the problem as well as temporal trends in the problem. Both the literature review and quantitative framing informed the subsequent conceptual model of lock-in, which was based on the theoretical concept of bottlenecks and the conjunction of architectures. Around this conceptual model, scientific methods were erected to capture intellectual property architecture and objectively measure accessibility. These scientific methods were the basis of a research methodology used to investigate the two research goals outlined above. Finally, an intermediate-N fuzzy-set Qualitative Comparative Analysis was used as a basis to draw cross-case inferences and conclude findings.

10.1 Contributions

The substantive contributions from this work are listed concisely below, with further discussion of each in the subsequent sections. In the interest of organization, each contribution was classified as either a methodological, empirical, or theoretical contribution to understanding of the research goals outlined above.

Theoretical Contributions

1. Formulated conceptual model for lock-in as the conjuncture of Contract Structure and Technical Architecture
2. Introduced the concept of intellectual property architecture
3. Proposed taxonomy of architectural patterns for avoidance of intellectual property lock-in

Methodological Contributions

1. Method to quantify magnitude and trends in intellectual property lock-in
2. Novel method for objectively measuring intellectual property architecture
3. Novel method for operationalizing accessibility of architecture

Empirical Contributions

1. Quantified the magnitude and trends of intellectual property lock-in for DoD contracts
2. Substantiated the hypotheses that intellectual property architecture is a mechanism of action for avoidance of lock-in
3. Evidenced the theoretical formulations for the taxonomy on architectural patterns

Although there were other lesser contributions, the above list captures those which have the potential to impact either future research or practice in a meaningful way.

10.1.1 Theoretical

Theoretical contributions are those which are the result of novel concepts or formulations. Beginning with the idea of intellectual property architecture, Chapter 4 drew from Baldwin's work on bottlenecks [76] to theorize that *de jure* strategic bottlenecks

or intellectual property lock-in occurred at the conjunction of the technical architecture and the contract structure. The importance of this novel theoretical approach is not necessarily the conceptual novelty, but the ability to apply measurement to the theory. Previous conceptual formulations of intellectual property lock-in were not measurable and subsequently not quantitatively testable. However, the concept of intellectual property lock-in at the conjuncture of architectures provides future researchers a means to base further quantitative inquiry that is rooted in variable-oriented approaches versus case-oriented approaches.

This work extended the conceptual model of intellectual property lock-in to theorize the concept of intellectual property architecture as one specific measurement approach. Intellectual property architecture recognizes that lock-in occurs at the conjuncture of architectures and allows for a prioritization of each piece of intellectual property based on location in the overall architecture. Herein, the measure of Visibility Fan-in (VFI) was used to prioritize based on work by MacCormack et al., [94]. However, this is only one potential choice for operationalization of intellectual property architecture, which is discussed in further detail in the section of methodological contributions.

The final theoretical contribution is the culmination of both previous theoretical contributions, which led to a taxonomy of architectural patterns for intellectual property lock-in avoidance. Recognizing the role of the system architect to make design trade-offs, a 2 x 3 matrix was theorized to contain architectural patterns which are categorized as either: Hazard, Boundary, or Ideal patterns of architecture. This taxonomy is important because it provides an abstraction of the underlying theory which is both approachable and meaningful to those in practice. Although the boundaries are fuzzy by nature of the experimental design, the taxonomy enables practitioners to easily understand what architectural patterns are more likely to avoid lock-in.

10.1.2 Methodological

Methodological contributions are those which introduced a new research approach; beginning first with the method to quantify magnitude and trends in intellectual

property lock-in within the DoD. The literature review in Chapter 2, was unable to identify any previous research which endeavored to capture the magnitude of the problem or recent trends. To address this gap, Chapter 3 erects a methodology to quantify the number of FAR 6.302-1(b)(2) Justifications and Authorizations as an indicator for the number of contracts which are locked-in. The introduction of this method is important because it is replicable and will help inform policy makers on the impacts from changes to intellectual property regulation or statute.

The second methodological contribution is the introduction of a method to objectively measure intellectual property architecture. This method builds on the work of Baldwin [76], [91], [92], [96], [99], [100], MacCormack [75], [94], [101], Rusnak [102], and Sturtevant [98] to erect a methodology which results in understanding, not only the various intellectual property within a code base, but also the significance of each piece of intellectual property relative to location within the architecture. Chapter 5 makes the argument that understanding both the type and location of intellectual property allows for more informed decision making about the potential for lock-in.

Chapter 5 builds upon the method of objectively measuring intellectual property architecture and erects an operationalization of accessible intellectual property architecture. This methodological operationalization introduces a means to quantify the level of ownership in those files which were shown to have the highest "hardness-to-kill" within an architecture. The importance of this method lies in the ability to compare the accessibility of an architecture against lock-in, which is the basis for the of fsQCA research method used herein.

10.1.3 Empirical

Empirical contributions are those which introduced evidence previously undocumented in literature. First, Chapter 3 underwent a quantitative framing of the intellectual property lock-in problem in the context of DoD contracts. Concluding that over the fiscal years under study, the order of magnitude for the problem is approximately $6 billion and that the temporal trends are noisy, with some measures increasing and other decreasing. This empirical contribution is important because it marks the first

DoD-wide assessment of the problem. Previous studies only offered a glimpse into a subset of DoD contracts [34] or used incomplete sources of data [35]. The new framing of the problem should set a baseline, which future policy decisions should be compared against.

Although important, the quantitative framing of the problem was not a main thrust of the research herein. The investigation of the research goals outlined above yielded three main findings which were discussed in Chapter 9. Amongst those was the finding that intellectual property architecture is a mechanism of action for avoidance of lock-in. Specifically, this research revealed empirical evidence where intellectual property architecture, absent unlimited rights, yielded avoidance of lock-in. This finding is significant due to the potential impact on practice. Empirical support for a new causal path to the avoidance of lock-in provides practitioners with a new strategy. These impacts are discussed in great detail in Chapter 9.

Finally, the research, herein, also provided empirical evidence in support of the theorized taxonomy of architectural patterns for the avoidance of lock-in. The fsQCA model provided empirical evidence for both the existence of Hazard and Ideal patterns, as well as support that these patterns avoid lock-in (i.e., Ideal Patterns) or do not avoid lock-in (i.e., Hazard Patterns). These results are important because they mark the first theorized taxonomy of intellectual property architectural patterns which are supported empirically. Previous attempts to identify taxonomies, such as the DoD's Open Source Architecture initiatives, offered strong theoretical support, but only anecdotal evidence.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A

OLS Regression Results

This appendix provides the full set of results for the OLS regressions outlined in Chapter 3. These analyses were accomplished using Python 3.4 as well as the Statsmodels, version 0.6.1, and Numpy, version 1.11.1, libraries. See Listing A below for actual code used in analysis.

```
import statsmodels.api as sm
import numpy as np
import pandas as pd

regList =       (df.ratiojnatoawd,testDF.ratiojnatoawd,
                 df.ratiopdrtoawd,testDF.ratiopdrtoawd,
                 df.ratiopdrtojna,testDF.ratiopdrtojna)

regx    =       sm.add_constant(np.arange(1,97))

for reg in regList:
    regr        = sm.OLS(reg, regx).fit()
    print(regr.summary().as_latex())
```

Listing A.1: Python OLS Regression Code

Dep. Variable:	Ratio JnA to Awd	**R-squared:**	0.001
Model:	OLS	**Adj. R-squared:**	-0.010
Method:	Least Squares	**F-statistic:**	0.06929
Date:	Wed, 17 Aug 2016	**Prob (F-statistic):**	0.793
Time:	11:07:09	**Log-Likelihood:**	102.23
No. Observations:	96	**AIC:**	-200.5
Df Residuals:	94	**BIC:**	-195.3
Df Model:	1		

	coef	**std err**	**t**	**P > |t|**	**[95.0% Conf. Int.]**	
const	0.4258	0.0173	24.550	0.000	0.391	0.460
x1	8.173e-05	0.0003	0.263	0.793	-0.001	0.001

Omnibus:	48.377	**Durbin-Watson:**	1.443
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	195.545
Skew:	1.607	**Prob(JB):**	3.45e-43
Kurtosis:	9.209	**Cond. No.**	113.

Table A.1: OLS Regression Results for Ratio of JnA to Total Awards ($)

Dep. Variable:	Ratio JnA to Awd	**R-squared:**	0.322
Model:	OLS	**Adj. R-squared:**	0.315
Method:	Least Squares	**F-statistic:**	44.61
Date:	Wed, 17 Aug 2016	**Prob (F-statistic):**	1.67e-09
Time:	11:07:09	**Log-Likelihood:**	225.42
No. Observations:	96	**AIC:**	-446.8
Df Residuals:	94	**BIC:**	-441.7
Df Model:	1		

	coef	**std err**	**t**	**P > |t|**	**[95.0% Conf. Int.]**	
const	0.2230	0.005	46.394	0.000	0.213	0.233
x1	-0.0006	8.61e-05	-6.679	0.000	-0.001	-0.000

Omnibus:	0.150	**Durbin-Watson:**	0.561
Prob(Omnibus):	0.928	**Jarque-Bera (JB):**	0.023
Skew:	0.038	**Prob(JB):**	0.988
Kurtosis:	3.012	**Cond. No.**	113.

Table A.2: OLS Regression Results for Ratio of JnA to Total Awards (Contract Actions)

Dep. Variable:	Ratio PDR to Awd	**R-squared:**	0.041
Model:	OLS	**Adj. R-squared:**	0.030
Method:	Least Squares	**F-statistic:**	3.986
Date:	Wed, 17 Aug 2016	**Prob (F-statistic):**	0.0488
Time:	11:07:09	**Log-Likelihood:**	488.09
No. Observations:	96	**AIC:**	-972.2
Df Residuals:	94	**BIC:**	-967.0
Df Model:	1		

	coef	**std err**	**t**	**P > |t|**	**[95.0% Conf. Int.]**	
const	0.0016	0.000	5.163	0.000	0.001	0.002
x1	1.114e-05	5.58e-06	1.997	0.049	6.16e-08	2.22e-05

Omnibus:	80.884	**Durbin-Watson:**	1.921
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	632.805
Skew:	2.704	**Prob(JB):**	3.87e-138
Kurtosis:	14.356	**Cond. No.**	113.

Table A.3: OLS Regression Results for Ratio of PDR to Total Awards ($)

Dep. Variable:	Ratio PDR to Awd	**R-squared:**	0.662
Model:	OLS	**Adj. R-squared:**	0.658
Method:	Least Squares	**F-statistic:**	184.0
Date:	Wed, 17 Aug 2016	**Prob (F-statistic):**	7.33e-24
Time:	11:07:09	**Log-Likelihood:**	642.86
No. Observations:	96	**AIC:**	-1282.
Df Residuals:	94	**BIC:**	-1277.
Df Model:	1		

	coef	**std err**	**t**	**P > |t|**	**[95.0% Conf. Int.]**	
const	0.0018	6.21e-05	29.647	0.000	0.002	0.002
x1	-1.509e-05	1.11e-06	-13.566	0.000	-1.73e-05	-1.29e-05

Omnibus:	1.908	**Durbin-Watson:**	1.383
Prob(Omnibus):	0.385	**Jarque-Bera (JB):**	1.411
Skew:	0.083	**Prob(JB):**	0.494
Kurtosis:	3.570	**Cond. No.**	113.

Table A.4: OLS Regression Results for Ratio of PDR to Total Awards (Contract Actions)

Dep. Variable:	Ratio PDR to JnA	**R-squared:**	0.031
Model:	OLS	**Adj. R-squared:**	0.021
Method:	Least Squares	**F-statistic:**	3.029
Date:	Wed, 17 Aug 2016	**Prob (F-statistic):**	0.0851
Time:	11:07:09	**Log-Likelihood:**	395.89
No. Observations:	96	**AIC:**	-787.8
Df Residuals:	94	**BIC:**	-782.7
Df Model:	1		

	coef	**std err**	**t**	**P > |t|**	**[95.0% Conf. Int.]**	
const	0.0040	0.001	4.930	0.000	0.002	0.006
x1	2.536e-05	1.46e-05	1.740	0.085	-3.57e-06	5.43e-05

Omnibus:	82.637	**Durbin-Watson:**	1.869
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	728.952
Skew:	2.719	**Prob(JB):**	5.13e-159
Kurtosis:	15.356	**Cond. No.**	113.

Table A.5: OLS Regression Results for Ratio of PDR to JnA ($)

Dep. Variable:	Ratio PDR to JnA	**R-squared:**	0.576
Model:	OLS	**Adj. R-squared:**	0.571
Method:	Least Squares	**F-statistic:**	127.7
Date:	Wed, 17 Aug 2016	**Prob (F-statistic):**	3.30e-19
Time:	11:07:10	**Log-Likelihood:**	484.68
No. Observations:	96	**AIC:**	-965.4
Df Residuals:	94	**BIC:**	-960.2
Df Model:	1		

	coef	**std err**	**t**	**P > |t|**	**[95.0% Conf. Int.]**	
const	0.0087	0.000	27.100	0.000	0.008	0.009
x1	-6.531e-05	5.78e-06	-11.300	0.000	-7.68e-05	-5.38e-05

Omnibus:	0.453	**Durbin-Watson:**	1.107
Prob(Omnibus):	0.797	**Jarque-Bera (JB):**	0.164
Skew:	0.081	**Prob(JB):**	0.921
Kurtosis:	3.122	**Cond. No.**	113.

Table A.6: OLS Regression Results for Ratio of PDR to JnA (Contract Actions)

Appendix B

Interview Question Template

Christopher Berardi
Engineering Systems Division
Phone: [redacted]
Email: [redacted]

Massachusetts Institute of Technology
77 Massachusetts Ave, Bldg 33-419
Cambridge, Massachusetts 02139

Interview Questions on Department of Defense (DoD) Intellectual Property (IP) issues and solutions:

1. Is there a problem with DoD IP policy and management? If so, how would you frame the problem?

2. In the simplest terms, how would you frame the IP tensions between private industry and the DoD?

3. From a DoD perspective, what would you characterize as the most significant IP problem or issue facing the DoD?
    a. How would you frame a solution to this problem?

4. From a DoD perspective, what would you characterize as the most significant IP problem or issue facing industry?
    a. How would you frame a solution to this problem?

5. Is there a consensus among DoD leaders on how to solve or positively impact the DoD IP problems or issues?

6. Is the DoD's taxonomy of data rights (i.e. unlimited rights, government purpose rights, limited rights, etc.) effective? If not, what should change?

7. Are there any contracting trends which are not quantified that you would like to see?

8. What are strategies employed by contractors to lock-in the DoD? Do you have any war stories?

9. What are the strategies employed by the DoD to avoid lock-in? Do you have any war stores?

Christopher Berardi
Engineering Systems Division
Phone: [redacted]
Email: [redacted]

Massachusetts Institute of Technology
77 Massachusetts Ave, Bldg 33-419
Cambridge, Massachusetts 02139

10. Are you aware of any case studies of either exemplary IP management or poor IP management I should review?

11. Is there anyone else I should interview who is either an expert on DoD IP or has an interesting perspective on DoD IP management?

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix C

Copyright Regular Expressions

This appendix provides the full list of regular expressions used to extract copyright information. These analyses were accomplished using Python 3.4.5 as well as the `re` library, version 2.2.1. All regular expressions were initially compiled using the `re.compile` function; after which, matching was done using the `re.search` function.

```
import re

#Copyright extraction regular expressions
p_copyright      = re.compile("Copyright", re.IGNORECASE)
p_copyr          = re.compile("\sCopyr|\sCopr\D", re.IGNORECASE)
p_cbracket_l     = re.compile("copyr.+\(c\)", re.IGNORECASE)
p_cbracket_r     = re.compile("\(c\).*copyr", re.IGNORECASE)
p_cbracket_o     = re.compile("\(c\)", re.IGNORECASE)
p_year           = re.compile("\D([12]\d\d\d)\D")
p_two_years      = re.compile("\D([12]\d\d\d)\D{1,5}([12]\d\d\d)")

#License extraction regular expressions
GNU              = re.compile('\sGNU\D')
MIT License      = re.compile('MIT License|Licensed MIT|\sMIT\D')
OpenSceneGraph   = re.compile('\sOSGPL\s|OpenSceneGraph Public License',
                              re.IGNORECASE)
Artistic License = re.compile('Artistic License', re.IGNORECASE)
LGPL             = re.compile('\sLGPL\s|Lesser General Public License|Library General
                              Public License',
                              re.IGNORECASE)
BSD              = re.compile('\sBSD\D')
```

```
Unspecified OS   = re.compile('free of charge|open source license', re.IGNORECASE)
GPL              = re.compile('\sGPL\D|(?<!Lesser|ibrary)\sGeneral Public License',
                              re.IGNORECASE)
Apache License   = re.compile('Apache License', re.IGNORECASE)
Creative Commons = re.compile('\sCC\s|Creative Commons')
```

Listing C.1: Python Regular Expression Code

Appendix D

Case Summary

.

Table D.1: Case Summary (Sorted by System File Size)

	File Size			Dependencies		Propagation			Copyright Category		
	System	Core	Core Pct[1]	Direct	Indirect	Density[1]	Cost[1]	Copyrights	Open	Accessible[2]	Closed[3]
N	24,740	901	3.6%	140,055	15,578,401	0.0%	2.5%	24,740	3,100	0	21,640
A	21,879	11,285	51.6%	189,377	154,670,489	0.0%	32.3%	21,879	74	0	21,805
B	18,899	1,528	8.1%	323,267	6,742,880	0.1%	1.9%	18,899	1,102	0	17,797
C	6,361	490	7.7%	52,274	2,883,437	0.1%	7.1%	6,361	3,674	0	2,687
D	4,247	166	3.9%	18,153	320,172	0.1%	1.8%	4,247	2,549	0	1,698
E	3,912	54	1.4%	13,929	119,016	0.1%	0.8%	3,912	46	46	3,820
F	2,869	86	3.0%	15,271	317,044	0.2%	3.9%	2,869	9	0	2,860
G	1,688	161	9.5%	10,642	199,571	0.4%	7.0%	1,688	2	12	1,674
H	1,497	32	2.1%	6,860	42,205	0.3%	1.9%	1,497	0	1	1,496
M	1,466	589	40.2%	11,639	896,023	0.5%	41.7%	1,466	1	0	1,465
I	1,464	96	6.6%	6,243	102,614	0.3%	4.8%	1,464	147	0	1,317
J	737	81	11.0%	3,155	25,185	0.6%	4.6%	737	5	0	732
K	497	252	50.7%	3,567	107,180	1.4%	43.4%	497	5	0	492
L	374	2	0.5%	1,525	4,871	1.1%	3.5%	374	0	331	43

[1] All percentages rounded to the nearest tenth of a percent
[2] To differentiate between Open Source, in this table, Accessible implies the copyright owner is the U.S. Gov't
[3] Closed copyrights are only those files which contain a copyright that is not open or accessible. It does not include the assumption that files without a copyright are closed.

Bibliography

[1] H. W. J. Rittel and M. M. Webber, "Dilemmas in a General Theory of Planning," *Policy Sciences*, vol. 4, no. 2, pp. 155–169, 1973.

[2] J. C. Camillus, "Strategy as a Wicked Problem," *Harvard Business Review*, vol. 86, no. 5, pp. 98–106, May 2008.

[3] L. M. Lavenue, "Technical Data Rights in Government Procurement: Intellectual Property Rights in Computer Software and the Indicia of Information Systems and Information Technology," *University of San Francisco Law Review*, vol. 32, no. 1, pp. 1–75, 1998.

[4] W. C. Anderson, "Comparative Analysis Of Intellectual Property Issues Relating To The Acquisition Of Commercial And Noncommercial Items By The Federal Government," *Public Contract Law Journal*, vol. 33, no. 1, pp. 37–61, 2003.

[5] C. P. Dungan, "Less Is More: Encouraging Greater Competition In Computer Software Procurement By Simplifying The DFARS Licensing Scheme," *Public Contract Law Journal*, vol. 39, no. 3, pp. 465–482, 2010.

[6] D. M. Sidebottom, "Intellectual Property In Federal Government Contracts: The Past, The Present, And One Possible Future," *Public Contract Law Journal*, vol. 33, no. 1, pp. 63–97, 2003.

[7] M. S. Simchak, "Protecting Rights In Technical Data And Computer Software: Applying The Ten Practical Rules And Their Corollaries," *Public Contract Law Journal*, vol. 33, no. 1, pp. 139–162, 2003.

[8] G. S. Sharp, "A Layman's Guide To Intellectual Property In Defense Contracts," *Public Contract Law Journal*, vol. 33, no. 1, pp. 99–137, 2003.

[9] N. O. Dix, F. A. Lavallee, and K. C. Welch, "Fear And Loathing Of Federal Contracting: Are Commercial Companies Really Afraid To Do Business With The Federal Government? Should They Be?" *Public Contract Law Journal*, vol. 33, no. 1, pp. 5–36, 2003.

[10] D. Larson, "Yesterday's Technology, Tomorrow: How The Government's Treatment Of Intellectual Property Prevents Soldiers From Receiving The Best Tools To Complete Their Mission," *The John Marshall Review of Intellectual Property Law*, pp. 171–201, 2007.

[11] B. Lill, "Restrictive Legends In Federal Procurement: Is The Risk Of Losing Data Rights Too Great?" *Public Contract Law Journal*, vol. 38, no. 4, pp. 895–911, 2009.

[12] C. C. Trend, "Killing The Goose That Laid The Golden Egg: Data Rights Law And Policy In Department Of Defense Contracts," *Public Contract Law Journal*, vol. 34, no. 2, pp. 287–336, 2005.

[13] M. A. Heller, "The tragedy of the anticommons: Property in the transition from Marx to markets.," *Harvard Law Review*, vol. 111, no. 3, p. 621, Jan. 1998.

[14] A. R. Whale, "Government Rights to Technical Information Recieved Under Contract," *The George Washington Law Review*, pp. 289–362, 1957.

[15] M. R. M. Hinrichs, "Proprietary Data and Trade Secrets Under Department of Defense Contracts," *Military Law Review*, pp. 61–90, 1967.

[16] W. G. Reed, *Department of Defense Spare Parts Procurement Practices*, Washington, DC, 1983.

[17] R. Murray, "Intellectual Property and Technical Data Rights, It's About the Money," PhD thesis, United States Army War College, 2012, pp. 1–64.

[18] Office of the Undersecretary of Defense for Acquisition Technology and Logistics, "Intellectual Property: Navigating Through Commercial Waters," Department of Defense, Tech. Rep., 2001, pp. 1–106. [Online]. Available: `http://www.acq.osd.mil/dpap/Docs/intelprop.pdf`.

[19] Section 800 Committee, "Streamlining Defense Acqusition Laws," Tech. Rep., 1993, p. 1674.

[20] Section 807 Committee, *Government-Industry Technical Data Advisory Committee*, Washington, DC, Apr. 1994.

[21] N. Reynolds, *Section 807 Committee Minority Report*, Washington, DC, Dec. 1993.

[22] E. Mazour, "If You Fix It, They Will Come: Drawing Lessons Learned From Patents For Dealing With Rights In Technical Data," *Public Contract Law Journal*, vol. 38, no. 3, pp. 667–688, 2009.

[23] J. J. Petrillo, "Repealing Complex Procurement Laws May Not Be So Wise," *Gov't Computer News*, vol. Sept 13, p. 104, 1993.

[24] A. C. Lazure and J. H. Church, "The Shadow And Substance Of Intellectual Property In Defense Department Research And Development Contracts," *The Federal Bar Journal*, vol. 14, pp. 300–320, 1954.

[25] K. Pages, "Rights in Technical Data and Computer Software - A Tenuous Balance Between the Government and the Contractor," *Contract Management*, vol. 53, no. 2, pp. 42–50, 2013.

[26] K. Baker, "Intellectual Property Clauses to Watch for in Government Contracts," *Contract Management*, vol. 52, no. 5, pp. 50–55, 2011.

[27] G. Pickarz, "Data Rights Assertions Give Clarity - Without One, You Have No Clue," *Contract Management*, vol. 55, no. 1, pp. 22–33, 2015.

[28] M. S. Simchak and D. A. Vogel, "A Few Words of Advice: Protecting Intellectual Property When Contracting with the Department of Defense According to the October 1988 Regulations," *Public Contract Law Journal*, vol. 23, no. 2, pp. 141–167, 1994.

[29] D. Duberstein, "Validation Of Proprietary Data Restrictions : How Contractors Can Protect Their Rights in Technical Data Against Government Challenge," *National Contract Management Journal*, vol. 22, no. 1, pp. 25–32, 1988.

[30] J. M. Baker, S. Haque, J. Mccarthy John E, and A. Raddock, "GOVERNMENT DATA RIGHTS CHALLENGES: Tips on How to Avoid & Respond," *Contract Management*, vol. 55, no. 11, pp. 20–28, 2015.

[31] T. J. Madden, J. Marvin Charles R, and J. T. Reams, "The Role of Independent Research and Development Funds in Securing Contractor Rights to Intellectual Property," *Contract Management*, vol. 46, no. 7, pp. 6–12, 2006.

[32] J. H. Rabitschek and N. J. Latker, "Reasonable Pricing - A New Twist for March-in Rights under the Bayh-Dole Act," *Santa Clara Computer & High Technology Law Journal*, vol. 22, no. 1, pp. 149–168, 2005.

[33] L. Muzzelo and C. Arndt, "Data Rights for Science and Technology Projects," *Defnese Acquisitions Research Journal*, vol. 21, no. 2, pp. 625–650, 2014.

[34] E. J. Liedke and J. D. Simonis, "Increasing Competitive Actions: A Focus On Technical Data Rights Associated with Non-Commercial Hardware Items," PhD thesis, Naval Post Graduate School, 2014, p. 79.

[35] C. W. Berardi, B. Cameron, D. Sturtevant, C. Y. Baldwin, and E. Crawley, "Architecting Out Software Intellectual Property Lock-In: A Method to Advance the Efficacy of BBP," in *Proceedings of the 13th Annual Acquisition Research Symposium*, Monterey, CA, 2016, pp. 184–201.

[36] J. Hasik, "Better Buying Power or Better Off Not? Purchasing Technical Data for Weapon Systems," *Defense Acquisitions Research Journal*, vol. 21, no. 3, pp. 693–713, 2014.

[37] A. Dixit, "Incentives and Organizations in the Public Sector: An Interpretative Review," *The Journal of Human Resources*, vol. 37, no. 4, pp. 696–727, Oct. 2002.

[38] N. J. Ross, "Technical Data Packages: When Can They Reduce Costs for the Department of Defense?" *Defense Acquisitions Research Journal*, vol. 22, no. 4, pp. 450–471, 2015.

[39] S. Head and J. Nelson, *Data Rights Valuation in Software Acquisitions*, Alexandria, Virginia, 2012. [Online]. Available: `https://www.cna.org/CNA_files/PDF/DRM-2012-U-001825-Final.pdf`.

[40] A. P. Hunter, R. Mccormick, J. Ellman, G. Sanders, K. Johnson, and G. Coll, "Defense Acquisition Trends, 2015," Center for Strategic & International Studies, Tech. Rep., Jan. 2016, p. 118.

[41] A. Hunter, G. Sanders, R. Mccormick, J. Ellman, and M. Riley, "Measuring the Success of Acquisition Reform by Major DoD Components," in *Proceedings of the 12th Annual Acquisition Research Symposium*, vol. 1, Monterey, CA, 2015.

[42] J. B. Liebman and N. Mahoney, "Do Expiring Budgets Lead to Wasteful Year-End Spending? Evidence from Federal Procurement," *National Bureau of Economic Research Working Paper Series*, vol. No. 19481, 2013.

[43] F. Kendall, "Implementation Directive for Better Buying Power 3.0 - Achieving Dominant Capabilities through Technical Excellence and Innovation," Under Secretary of Defense for Acquisitions Technology and Logistics, Washington, DC, Tech. Rep., 2015.

[44] ——, "Implementation Directive for Better Buying Power 2.0 - Achieving Grater Efficiency and Productivity in Defense Spending," USD AT&L, Washington, DC, Tech. Rep., 2013.

[45] A. B. Carter, "Better Buying Power: Mandate for Restoring Affordability and Productivity in Defense Spending," USD AT&L, Washington, DC, Tech. Rep., 2010.

[46] Office of the Undersecretary of Defense for Acquisition Technology and Logistics, "Performance of the Defense Acquisition System, 2015 Annual Report," Under Secretary of Defense, Acquisition, Technology, and Logistics (USD[AT&L]), Washington, DC, Tech. Rep., Sep. 2015, p. 210.

[47] G. Sanders, J. Ellman, and S. Cohen, "Competition and Bidding Data as an Indicator of the Health of the U.S. Defense Industrial Base," Center For Strategic & International Studies, Tech. Rep., Oct. 2015, p. 78.

[48] Defense Acqusition University, *Justification and Approval (J&A)*, 2015. [Online]. Available: `https://dap.dau.mil/acquipedia/Pages/ArticleDetails.aspx?aid=70a60a2f-c14b-4513-b32f-49afa3434999`.

[49] W. P. Rogerson, "Economic Incentives and the Defense Procurement Process," *The Journal of Economic Perspectives*, vol. 8, no. 4, pp. 65–90, 1994.

[50] National Science Foundation, *WebCASPAR: Integrated Science and Engineering Resources Data System*, 2016. [Online]. Available: `https://ncsesdata.nsf.gov/webcaspar/`.

[51] Defense Contract Audit Agency, *Independent Research and Development and Bid and Proposal Costs*, 2016. [Online]. Available: `http://www.acq.osd.mil/dpap/cpic/cp/independent_research_and_development_and_bid_and_proposal_costs.html`.

[52] C. Shearer, "The CRISP-DM Model: The New Blueprint for Data Mining," *Journal of Data Warehousing*, vol. 5, no. 4, pp. 13–22, 2000.

[53] N. Y. Moore, C. A. Grammich, and J. D. Mele, "Findings From Existing Data on the Department of Defense Industrial Base: Guided Missile and Space Vehicle Manufacturing Example," in *Proceedings of the 12th Annual Acquisition Research Symposium*, Monterey, CA: Naval Postgraduate School, 2015, pp. 347–365.

[54] Center for Strategic & International Studies, *Methodology, The Techniques, Assumptions, and Limitations Underlying the Analysis of Defense Contracting Trends*, 2016. [Online]. Available: `https://www.csis.org/programs/international-security-program/defense-industrial-initiatives-group/methodology`.

[55] W. T. Woods, "Reliability of Federal Procurement Data," United States General Accounting Office, Tech. Rep., 2003, p. 5.

[56] USASpending.gov, *USASpending.gov Data Downloads Data Dictionary*, 2015. [Online]. Available: `https://www.usaspending.gov/DownloadCenter/Documents/USAspending.govDownloadsDataDictionary.pdf`.

[57] J. Ellman and J. Bell, "Analysis of Defense Products Contract Trends, 1990-2014," Center For Strategic & International Studies, Tech. Rep., Oct. 2015, p. 65. [Online]. Available: `http://csis.org/publication/analysis-defense-products-contract-trends-1990-2014`.

[58] W. P. Rogerson, "Overhead Allocation and Incentives for Cost Minimization in Defense Procurement," *The Accounting Review*, vol. 67, no. 4, pp. 671–690, 1992.

[59] ——, "Chapter 12 Incentive Models of the Defense Procurement Process," in *Handbook of Defense Economics*, B. T. .-.-. H. o. D. Economics, Ed., vol. Volume 1, Elsevier, 1995, pp. 309–346.

[60] J. M. Karpoff, D. S. Lee, and V. P. Vendrzyk, "Defense Procurement Fraud, Penalties, and Contractor Influence," *Journal of Political Economy*, vol. 107, no. 4, pp. 809–842, 1999.

[61] W. P. Rogerson, "Profit Regulation of Defense Contractors and Prizes for Innovation," *Journal of Political Economy*, vol. 97, no. 6, pp. 1284–1305, 1989.

[62] W. Adams and W. J. Adams, "The Military-Industrial Complex: A Market Structure Analysis," *The American Economic Review*, vol. 62, no. 1/2, pp. 279–287, Mar. 1972.

[63] T. Riddell, "Concentration and Inefficiency in the Defense Sector: Policy Options," *Journal of Economic Issues*, vol. 19, no. 2, pp. 451–461, Jun. 1985.

[64] J. P. Dunne, "Chapter 14 The defense industrial base," in *Handbook of Defense Economics*, vol. Volume 1, Elsevier, 1995, pp. 399–430.

[65] M. G. Jacobides and J. C. Tae, "Kingpins , Bottlenecks , and Value Dynamics Along a Sector," *Organization Science*, vol. 26, no. 3, pp. 889–907, 2015.

[66] M. G. Jacobides, "Ripples through the value chain: how upstream innovation shapes profit and scope in a sector," *Wharton Technology Conference*, 2013.

[67] G. P. Pisano and D. J. Teece, "How to Capture Value from Innovation: Shaping Intellectual Property and Industry Architecture," *California Management Review*, vol. 50, no. 1, pp. 278–296, 2007.

[68] R. N. Langlois, "Modularity in technology and organization," *Journal of Economic Behavior and Organization*, vol. 49, no. 1, pp. 19–37, 2002.

[69] M. G. Jacobides, T. Knudsen, and M. Augier, "Benefiting from innovation: Value creation, value appropriation and the role of industry architectures," *Research Policy*, vol. 35, no. 8 SPEC. ISS. Pp. 1200–1221, 2006.

[70] D. J. Teece, "Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy," *Research Policy*, vol. 15, no. 6, pp. 285–305, 1986.

[71] B. W. Arthur, *The Nature of Technology: What It Is and How It Evolves*. New York, NY: Free Press, 2011, p. 256.

[72] K. Ulrich and S. D. Eppinger, *Product Design and Development*, Fourth. New York, NY: Irwin/McGraw-Hill, 2008.

[73] H. A. Simon, "The Architecture of Complexity," *Proceedings of the American Philosophical Society*, vol. 106, no. 6, pp. 33–42, 1962.

[74] K. Ulrich, "The role of product architecture in the manufacturing firm," *Research Policy*, vol. 24, no. 3, pp. 419–440, May 1995.

[75] A. MacCormack, J. Rusnak, and C. Y. Baldwin, "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code," *Management Science*, vol. 52, no. 7, pp. 1015–1030, Jul. 2006.

[76] C. Y. Baldwin, "Bottlenecks, Modules and Dynamic Architectural Capabilities," 2015.

[77] C. Y. Baldwin and K. B. Clark, *Design Rules: The Power of Modularity, Volume 1*. Cambridge, MA: MIT Press, 2000.

[78] W. Scacchi and T. Alspaugh, "Emerging Issues In The Acquisition Of Open Source Software," in *Proceedings of the 5th Annual Acquisition Research Symposium*, Monterey, CA: Naval Postgraduate School, 2008, pp. 230–245.

[79] Free Software Foundation, *What is free software?* 2015. [Online]. Available: `http://www.gnu.org/philosophy/free-sw.html`.

[80] Apache Software Foundation, *Apache License Version 2.0*, 2004. [Online]. Available: `https://www.apache.org/licenses/LICENSE-2.0`.

[81] P. A. Czech, "The Myth of Data Rights," *Defense Acqusitions Technology & Loistics*, no. December, pp. 46–50, 2015.

[82] R. C. Nash, S. L. Schooner, and K. R. O'Brien-DeBakey, *The Government Contracts Reference Book: A Comprehensive Guide to the Language of Procurement*, 4th ed. Riverwoods, IL: Wolters Kluwer Law & Business, 2013.

[83] M. E. Sosa, T. Browning, and J. Mihm, "Studying the Dynamics of the Architecture of Software Products," in *ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Las Vegas, Nevada, 2007, pp. 329–342.

[84] S. K. Ethiraj and D. Levinthal, "Modularity and Innovation in Complex Systems," *Management Science*, vol. 50, no. 2, pp. 159–173, Feb. 2004.

[85] A.-L. Barabasi and R. Albert, "Emergence of Scaling in Random Networks," *Science*, vol. 286, no. 5439, pp. 509–512, Oct. 1999.

[86] D. L. Parnas, "A technique for software module specification with examples," *Communications of the ACM*, vol. 15, no. 5, pp. 330–336, 1972.

[87] ——, *On the criteria to be used in decomposing systems into modules*, 1971.

[88] ——, "Designing Software for Ease of Extension and Contraction," *IEEE Transactions on Software Engineering*, vol. SE-5, no. 2, pp. 128–138, 1979.

[89] K. Ulrich, "Fundamentals of Product Modularity," in *Management of Design*, S. Dasu and C. Eastman, Eds., Dordrecht: Springer Netherlands, 1994, ch. 12, pp. 219–231.

[90] J. Barney, "Firm Resources and Sustained Competitive Advantage," *Journal of Management*, vol. 17, no. 1, pp. 99–120, 1991.

[91] C. Y. Baldwin and J. Henkel, "The Impact of Modularity on Intellectual Property and Value Appropriation," 2012.

[92] ——, "Modularity and Intellectual Property Protection," 2014.

[93] J. Henkel, C. Y. Baldwin, and W. C. Shih, "IP Modularity: Profiting from Innovation by Aligning Product Architecture with Intellectual Property," 2012.

[94] A. MacCormack, J. Rusnak, and C. Y. Baldwin, "The Impact of Component Modularity on Design Evolution: Evidence from the Software Industry," 2007.

[95] E. Ravasz and A.-L. Barabasi, "Hierarchical organization in complex networks," *Physical Review E*, vol. 67, no. 2, p. 26 112, Feb. 2003.

[96] C. Y. Baldwin, A. MacCormack, and J. Rusnak, "Hidden Structure: Using Network Methods to Map System Architecture," Oct. 2014.

[97] Random House Inc, *Equillbirium*, Oct. 2016. [Online]. Available: `http://www.dictionary.com/browse/equilibrium`.

[98] D. Sturtevant, "System Design and the Cost of Architectural Complexity," PhD thesis, Massachusetts Institute of Technology, 2013, pp. 1–166.

[99] C. Y. Baldwin, "When Open Architecture Beats Closed : The Entrepreneurial Use of Architectural Knowledge," 2010.

[100] ——, "Where do transactions come from? Modularity, transactions, and the boundaries of firms," 2008.

[101] A. Maccormack, "THE ARCHITECTURE OF COMPLEX SYSTEMS: DO 'CORE-PERIPHERY' STRUCTURES DOMINATE?" In *Academy of Management Annual Meeting Proceedings*, Academy of Management, Aug. 2010, pp. 1–6.

[102] J. Rusnak, "The Design Structure Analysis System: A Tool to Analyze Software Architecture," PhD thesis, Harvard University, 2005, p. 293.

[103] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*, ser. Structural Analysis in the Social Sciences. New York, NY: Cambridge University Press, 1994, p. 825.

[104] S. Eick, T. Graves, and A. Karr, "Does Code Decay? Assessing The Evidence from Change Management Data," *IEEE Transactions on Software Engineering*, vol. 27, no. 1, pp. 1–12, 2001.

[105] M. Cataldo, P. a. Wagstrom, J. D. Herbsleb, and K. M. Carley, "Identification of Coordination Requirements: Implications for The Design of Collaboration and Awareness Tools," in *Proc. ACM Conf. on Computer-Supported Work*, Banff, Canada, 2006, pp. 353–362.

[106] S. Rusovan, M. Lawford, and D. L. Parnas, "Open Source Software Development: Future or Fad?" In *Perspectives on Free and Open Source Software*, Joseph Feller, B. Fitzgerald, S. A. Hissam, and K. R. Lakhani, Eds., Cambridge, MA: MIT Press, 2007, ch. 6, pp. 107–121.

[107] D. M. Sharman and A. A. Yassine, "Characterizing complex product architectures," *Systems Engineering*, vol. 7, no. 1, pp. 35–60, Jan. 2004.

[108] J. Warfield, "Binary Matrices in System Modeling," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 5, pp. 441–449, 1973.

[109] A. M. St. Laurent, *Understanding Open Source and Free Software Licensing*, S. S. Laurent, Ed. Sebastopol, CA: O'Reilly Media, 2004, p. 208.

[110] J. A. Maxwell, *Qualitative research design: an interative approach*. Thousand Oaks, Calif.: Sage Publications, 1996, p. 152.

[111] D. Berg-Schlosser, G. De Meur, B. Rihoux, and C. C. Ragin, "Qualitative Comperative Analysis (QCA) as an Approach," in *Configurational Comparative Methods*, B. Rihoux and C. C. Ragin, Eds., Los Angeles: SAGE Publications, 2009, ch. 1, pp. 1–19.

[112] J. S. Mill, *A System of Logic: Ratiocinative and Inductive*, 1 edition. CreateSpace Independent Publishing Platform, 1843.

[113] C. C. Ragin, *The Comparative Method: Moving Beyond Qualitative and Quantitative Strategies*. Berkeley, CA: University of California Press, 1987.

[114] ——, *Constructing Social Research*. Thousand Oaks, Calif.: Pine Oaks Press, 1994, p. 194.

[115] ——, "Core versus tangential assumptions in comparative research," *Studies in Comparative International Development*, vol. 40, no. 1, pp. 33–38, 2005.

[116] ——, "Set Relations in Social Research: Evaluating Their Consistency and Coverage.," *Political Analysis*, vol. 14, no. 3, pp. 291–310, Jul. 2006.

[117] B. Rihoux, "Qualitative Comparative Analysis (QCA) and Related Techniques: recent advances and challenges," in *Methoden der vergleichenden Politik- und Sozialwissenschaft: Neue Entwicklungen und Anwendungen*, S. Pickel, G. Pickel, H.-J. Lauth, and D. Jahn, Eds., Wiesbaden: VS Verlag für Sozialwissenschaften, 2009, pp. 365–385.

[118] ——, "Bridging the Gap between the Qualitative and Quantitative Worlds? A Retrospective and Prospective View on Qualitative Comparative Analysis," *Field Methods*, vol. 15, no. 4, pp. 351–365, Nov. 2003.

[119] ——, "Qualitative Comparative Analysis (QCA) and Related Systematic Comparative Methods," *International Sociology*, vol. 21, no. 5, pp. 679–706, Sep. 2006.

[120] B. Rihoux and B. Lobe, "The Case for Qualitative Comparative Analysis (QCA): Adding Leverage for Thick Cross-Case Comparison," in *The SAGE Handbook of Case-Based Methods*, C. C. Ragin and D. Byrne, Eds., Thousand Oaks, CA and London: SAGE Publications Ltd, 2009, ch. 12, pp. 222–242.

[121] F. Varone, B. Rihoux, and A. Marx, "A New Method for Policy Evaluation? Longstanding Challenges and the Possibilities of Qualitative Comparative Analysis (QCA)," in *Innovative Comparative Methods for Policy Analysis: Beyond the Quantitative-Qualitative Divide*, B. Rihoux and H. Grimm, Eds., Boston, MA: Springer US, 2006, ch. 10, pp. 213–236.

[122] B. Rihoux, P. Álamos-Concha, D. Bol, A. Marx, and I. Rezsöhazy, "From Niche to Mainstream Method? A Comprehensive Mapping of QCA Applications in Journal Articles from 1984 to 2011," *Political Research Quarterly*, vol. 66, no. 1, pp. 175–184, 2013.

[123] B. Rihoux and G. De Meur, "Crisp-Set Qualitative Comparative Analysis (csQCA)," in *Configurational Comparative Methods*, C. C. Ragin and B. Rihoux, Eds., Los Angeles: Sage Publications, Ltd., 2009, ch. 3, pp. 33–68.

[124] L. Cronqvist and D. Berg-Schlosser, "Multi-Value QCA (mvQCA)," in *Configurational Comparative Methods*, C. C. Ragin and B. Rihoux, Eds., Los Angeles: SAGE Publications Ltd, 2009, ch. 4, pp. 69–86.

[125] C. Q. Schneider and C. Wagemann, *Set-Theoretic Methods for the Social Sciences: A Guide to Qualitative Comperative Analysis*. Cambridge, UK: Cambridge University Press, 2012.

[126] C. C. Ragin, "Qualitative Comparative Analysis Using Fuzzy Sets (fsQCA)," in *Configurational Comparative Methods*, B. Rihoux and C. C. Ragin, Eds., Los Angeles: SAGE Publications Ltd, 2009, ch. 5, pp. 87–121.

[127] D. Berg-Schlosser and G. De Meur, "Comperative Research Design," in *Configurational Comparative Methods*, C. C. Ragin and B. Rihoux, Eds., Los Angeles: Sage Publications, Ltd., 2009, ch. 2, pp. 19–32.

[128] E. Amenta and J. D. Oulsen, "Where to Begin: A Survey of Five Approaches to Selecting Independent Variables for Qualitative Comperative Analysis," *Sociological Methods & Research*, vol. 23, no. 1, pp. 22–53, Aug. 1994.

[129] A. Marx, "Crisp-set qualitative comparative analysis (csQCA) and model specification: Benchmarks for future csQCA applications," *International Journal of Multiple Research Approaches*, vol. 4, no. 2, pp. 138–158, Aug. 2010.

[130] A. Marx and A. Dusa, "Crisp-Set Qualitative Comparative Analysis (csQCA), Contradictions and Consistency Benchmarks for Model Specification," *Methodological Innovations Online*, vol. 6, no. 2, pp. 103–148, Aug. 2011.

[131] K. M. Eisenhardt, "Building Theories from Case Study Research.," *Academy of Management Review*, vol. 14, no. 4, pp. 532–550, Oct. 1989, ISSN: 03637425.

[132] R. K. Yin, *Case Study Research Design and Methods*, 5th. Thousand Oaks, California: SAGE Publications, 2014.

[133] K. M. Eisenhardt and M. E. Graebner, "Theory Building From Cases: Opportunities and Challenges.," *Academy of Management Journal*, vol. 50, no. 1, pp. 25–32, Feb. 2007.

[134] B. Geddes, "How the Cases You Choose Affect the Answers You Get: Selection Bias in Comparative Politics," *Political Analysis*, vol. 2, no. 1, pp. 131–150, 1990.

[135] M. Q. Patton, *Qualitative Evaluation and Research Methods*, 2nd ed. Newbury Park, CA: SAGE Publications, inc, 1990.

[136] D. T. Campbell and J. C. Stanley, *Experimental and Quasi-Experimental Designs for Research*, 1966.

[137] C. Foundation, *Distributed Data Framework*, 2017. [Online]. Available: `http://codice.org/ddf/`.

[138] C. C. Ragin, *Redesigning Social Inquiry: Fuzzy Sets and Beyond*. Chicago: University of Chicago Press, 2008.

[139] ——, *Fuzzy-Set Social Science*. Chicago: University of Chicago Press, 2000.

[140] A. Thiem and A. Dusa, "QCA : A Package for Qualitative Comparative Analysis," *The R Journal*, vol. 5, no. 1, pp. 87–97, 2013.

[141] J. Verkuilen, "Assigning Membership in a Fuzzy Set Analysis," *Sociological Methods & Research*, vol. 33, no. 4, pp. 462–496, May 2005.

[142] A. Thiem and A. Dusa, *Qualitative Comperative Analysis with R*. New York, NY: Springer, 2013.

[143] A. Dusa, "A mathematical approach to the boolean minimization problem," *Quality & Quantity*, vol. 44, no. 1, p. 99, 2008.

[144] D. Dion, "Evidence and Inference in the Comparative Case Study," *Comperative Politics*, vol. 30, pp. 127–145, 1998.

[145] Department of Defense, "MODULAR OPEN SYSTEMS ARCHITECTURE (MOSA) AND THE DEPARTMENT OF DEFENSE: Intellectual Property Practices , Policy , Trends , And Recommendations," Tech. Rep., 2015.

[146] Office of the Staff Judge Advocate SMC, "Acquiring and Enforcing the Government's Rights in Technical Data and Computer Software Under Department of Defense Contracts: A Practical Handbook for Acquisition Professionals," Air Force Space and Missile Center (SMC), Tech. Rep., Aug. 2015, p. 291.

[147] T. Browning and S. D. Eppinger, *Design Structure Matrix Methods and Applications*. Cambridge, MA: MIT Press, 2012, p. 334.