

Coordination Strategies and Individual Behavior in Complex Engineered Systems Design

by

Arianne Xaviera Collopy

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Design Science)
in the University of Michigan
2019

Doctoral Committee:

Associate Professor Eytan Adar, Co-Chair
Professor Panos Y. Papalambros, Co-Chair
Professor Bogdan I. Epureanu
Professor Richard Gonzalez

Arianne X. Collopy

acollopy@umich.edu

ORCID iD: 0000-0001-9343-6310

© Arianne Xaviera Collopy 2019

To my family
past and present

ACKNOWLEDGMENTS

I want to thank the many people that made sure I would be successful and that provided support, mentorship, and advice throughout my graduate career. I am grateful for the opportunities to learn from my mistakes and learn to look fearlessly to the future, and I am proud of who I have become in the last four years.

This work was completed thanks to financial support from the Rackham Graduate School, the Division of Integrative Systems and Design, the NASA Systems Engineering Consortium at the University of Alabama Huntsville, and the Automotive Research Center. I am grateful for the support and the opportunities afforded from each, including teaching, community engagement, and industry partnership.

To my advisors: thank you for your mentorship and patience. Panos, for helping me see the big picture and become confident in my place in it. Eytan, for helping me learn to focus on the details and enjoy the process. I have learned so much from you both and I am glad to have had the opportunity to learn from each of you what it means to be a designer, scientist, researcher, teacher, and mentor.

Thanks also to my committee members: Bogdan, for being a role model of how to work efficiently and always at a high standard, and for the experience working with the Automotive Research Center. Rich, for suggesting new angles to my work and for being a cornerstone of the Design Science program.

To my ODE family, especially Melissa, Emrah, Alex, Yanxin, Namwoo, Vignesh, Tianyi, Vanessa, Rugved, and Sanjana, thank you for your willingness to collaborate, learn together, and have fun. Thanks also to all of the ODE alumni I have met and will continue to meet who are sources of inspiration.

I owe a significant amount of my success to the Design Science program and ISD for being a welcoming and supportive community. It is too easy to become adrift during graduate school, and the strength of our community is beyond what I imagined. Thanks especially to Diann Brei, Colleen Seifert, Matt Reed, Gail Carr, and Elena Chesney for making sure our program is successful. Design Science would not be what it is without the students, and I thank especially my colleagues and friends Meira, John, Clover, Matt V., Raíssa, Koray, Ilka, Marianna, Matt N., Sean, Shannon, Xinhui, Maya, Gerardo, and Vincent. Thank you for the conversations, walks, parties, and optimism.

I would not be where I am without my experiences in graduate school prior to Michigan: thank you to the faculty and mentors who supported me at the University of Maryland and the University of Alabama Huntsville. Particular thanks are due to Mary Bowden, Alison Flatau, Bryan Mesmer, and Dale Thomas for being excellent role models.

Last, but not at all least, thank you to my family for your love and support. Wesley, you have been an outstanding supporter of my journey through the best and worst times and I can't thank you enough. Alejandra, you have always set a great example and I'm so proud of you. Mom and Dad, you gave me every opportunity I have and taught me to take advantage of them with sincerity. Thank you for the constant reminder that life is what you make of it.

TABLE OF CONTENTS

Dedication	ii
Acknowledgments	iii
List of Figures	viii
List of Tables	ix
List of Appendices	x
Abstract	xi
 Chapter	
1 Introduction	1
1.1 Research Questions	2
1.2 Approach	3
1.2.1 Methodology	4
1.2.2 Discipline background	4
1.3 Summary of Findings	5
1.4 Contributions	7
1.5 Dissertation Outline	8
2 Background and Terminology	9
2.1 System Representations	9
2.1.1 Matrix-based Representations	10
2.1.2 Graph-based Representations	11
2.1.3 Partitioning and Coordination	12
2.2 Review of Literature: Coordination	13
2.2.1 Design Optimization	14
2.2.1.1 Coordination Methods in MDO	15
2.2.2 Organization Science	16
2.2.2.1 Coordination Methods in Organization Science	17
2.2.2.2 Comparing Organizational Coordination to MDO	18
2.2.3 Software and Engineering Design	19
2.2.3.1 Coordination in Software and Engineering Design	20
2.2.4 Systems Engineering	22

2.2.5	Measuring Coordination	24
2.2.5.1	Conway’s Law	24
2.2.5.2	Mirroring	25
2.2.5.3	Socio-Technical Congruence	25
2.2.5.4	Challenges in Measuring Coordination	25
2.3	Summary	27
3	Coordination in Industry Practice	29
3.1	Introduction	29
3.2	Literature: Coordination Methods	30
3.3	Methodology	31
3.4	Thematic Analysis	34
3.4.1	Data Preparation	34
3.4.2	Deductive Coding	34
3.4.3	Inductive Coding	35
3.4.4	Theme Identification	39
3.4.4.1	Authority	42
3.4.4.2	Management	44
3.4.4.3	Empathetic Leadership	46
3.4.4.4	Facilitation of Coordination	48
3.4.4.5	Themes	52
3.4.5	Reflection	55
3.5	Discussion	56
3.5.1	Active Facilitation of Coordination	56
3.5.2	Relationship between Active and Passive Themes	58
3.6	Summary	60
4	Coordination in Design Teams	63
4.1	Introduction	63
4.2	Study Design: Data Collection and Analysis Approach	64
4.2.1	Data Collection	64
4.2.2	Analysis Approach	66
4.2.2.1	Keyword Identification and Analysis Method	67
4.2.2.2	Network Representations and Measures	70
4.2.2.3	Clustering Analysis	73
4.3	Characterization of Data	74
4.3.1	Characterization of Keyword Data	74
4.3.2	Characterization of Network Data	77
4.4	Clustering Results and Interpretation	82
4.4.1	Clusters as Described by Keywords	82
4.4.2	Clusters as Described by Network Measures	85
4.5	Discussion	87
4.5.1	Coordination Roles	87
4.5.2	Correlation to Interview Data	89
4.6	Summary	90

5 Using Agents to Model Coordination	92
5.1 Introduction	92
5.2 Model Description	95
5.2.1 Objectives	95
5.2.2 Network Structure	95
5.2.3 Decision-Making Task	98
5.2.3.1 Data Partitioning	100
5.2.3.2 Distributed Classification	102
5.2.4 Coordination Problem	107
5.2.5 Agent Behavior and Interaction	108
5.2.5.1 Interaction Probability	108
5.2.5.2 Information Exchange	110
5.2.5.3 Agent Distribution	111
5.2.6 Summary: Model Parameters	112
5.2.7 Model Process	113
5.2.8 Model Outputs: Performance and Costs	116
5.2.8.1 Performance	116
5.2.8.2 Cost	117
5.3 Hypotheses	117
5.4 Model Behavior	118
5.4.1 Direct Solution	118
5.4.2 Performance	120
5.4.2.1 Local Classification Accuracy	120
5.4.2.2 Global Classification Accuracy	125
5.4.3 Active vs. Passive Agent Accuracy	129
5.4.4 Cost	129
5.5 Parametric Analysis: Agent Concentration	130
5.6 Discussion	134
5.7 Validation plan	136
5.8 Summary	138
6 Conclusions	140
6.1 Review of Dissertation	140
6.2 Dissertation Contributions	142
6.3 Future Work and Extensions	142
6.3.1 Limitations	143
6.3.2 Extensions	144
Appendices	145
Bibliography	207

LIST OF FIGURES

Figure

1	Illustrations of Conway's Law, Mirroring, and Sociotechnical Congruence . . .	24
2	Illustration of the concept of misalignments	27
3	Map of interrelations between inductive codes	39
4	Selected subcodes highlighted in theme analysis discussion.	41
5	Map of subcodes related to Authority	42
6	Map of subcodes related to Management	45
7	Map of subcodes related to Empathetic Leadership	46
8	Map of subcodes related to Facilitation of Coordination	49
9	Illustration of Themes	52
10	Full map of all subcodes	53
11	Photo of cooperative game played with Robotic Machine Players (RMPs) . . .	65
12	Category distribution of keywords used in clustering analysis	75
13	Degree distributions calculated from generated networks and survey data . . .	80
14	Distributions of centrality and clustering coefficient calculated from generated networks and survey data	81
15	Top 20 keywords in each cluster	84
16	Cluster descriptions based on network measures	86
17	Two networks generated using the LFR algorithm	97
18	Confusion matrix of all 20,000 items in Letter Recognition Data Set	100
19	Illustration of how classification data is partitioned	101
20	Illustration of agents' distributed classification task	103
21	Illustration of threshold calculation	106
22	Flowchart of agent model process	115
23	Initial T=0 accuracy for agents on local and global classification tasks	120
24	Example local task accuracy, without threshold applied	121
25	Example local task accuracy, with threshold applied	121
26	Example of challenging local classification task	124
27	Example global task accuracy, with threshold applied	126

LIST OF TABLES

Table

1	Summary of interviewee demographics	33
2	Deductive codes and definitions used for second step of thematic analysis, divided by topic	36
3	Inductive codes, definitions, and selected subcodes, divided by topic	38
4	Categories of keywords identified from survey responses, with examples	69
5	Description of five network representations of thresholded survey data	71
6	Top 50 keywords identified from text analysis	75
7	Descriptive statistics for network measures from survey data	77
8	Kullback-Leibler Divergence calculated for Degree measures	80
9	Kullback-Leibler Divergence calculated for centrality and clustering coefficient measures	81
10	Category distribution of top keywords in each cluster	84
11	Parameters used as input for generated LFR networks	96
12	Example of threshold application and calculation of thresholded accuracy.	104
13	Local, community, and global parameters that govern agent model	112
14	K-NN classification results with varied training data size	119
15	Local classification results	123
16	Global classification results	128
17	Global classification task (aggregate) accuracy, reported for varied combinations of Passive and Active agents	131
18	Average number of interactions across all agents, reported for varied combinations of Passive and Active agents.	132

LIST OF APPENDICES

Appendix

A Interview Protocol	145
B Team Coordination Survey Protocol	149
C Agent Model Code	152

ABSTRACT

Coordination – ensuring interfacing groups are working with consistent data, interpretations of that data, and consistent goals – becomes necessary when design work is distributed or partitioned among different individuals or teams. System design optimization algorithms are well-established for the coordination of analytical design problems. However, in actual system design, individuals are unlikely to coordinate by following algorithmic procedures. In a design organization, distributed tasks must be first partitioned so that they can be worked in parallel, and then coordinated so that the results can be joined together to effect the overall project goal. In this organizational context, coordination is primarily a communicative process focused on information sharing among parallel tasks. This research focuses on these individual communication behaviors and demonstrates their impact on system-level coordination.

This dissertation addresses two research questions. First, what approaches and behaviors do individuals use to facilitate system-level coordination of distributed design work? This is answered with a pair of exploratory studies. A qualitative study based on interviews of industry experts found that proactive, empathetic leadership-based behaviors and more passive, authority-based behaviors are complementary approaches to facilitating coordination. A quantitative study based on a survey of novice designers resulted in identification of coordination roles within teams. Text analysis and network analysis of survey data identified five roles: some are more communicative and focused on integrative tasks and leadership, whereas others are less communicative and focused on documentation and detailed engineering tasks. The findings suggest a parallel to the active and passive behaviors identified in the qualitative study.

The second research question asks what is the quantitative impact of these behaviors on system-level performance. A descriptive agent-based model was developed that simulates the impact of individual behaviors on the system-level performance of a distributed design task requiring coordination. Results from this model indicate a correlation between more active agent behaviors and higher performance, but at the cost of increased peer-to-peer interactions.

This dissertation illustrates the importance of a balance between proactive, empathetic-leadership-based and more passive, authority-based processes and behaviors for the effective coordination of decomposition-based design work. There are three primary contributions of this work. This dissertation highlights that coordination is a central task of systems engineering personnel in engineering design organizations. This work uniquely shows how theory pertaining to coordination applies to and describes systems engineering activities, particularly in relation to engineering design. The second contribution of this work is the development of quantitative approaches to describe and evaluate coordination practice in decomposition-based system design. This is shown through an agent-based simulation model to assess the impact of communicative and information-seeking behavior on coordination and design task outcomes. This model also serves as a platform that allows extensive parametric analysis, permitting exploration of a variety of design tasks, organizational structures, and agent behaviors. Finally, this work illustrates the importance of coordinator roles in effective distributed design tasks. Frequently, managerial and integrative roles are not included in measures of coordination effectiveness, which we argue is not representative of true systems engineering practice. This dissertation is a step towards the inclusion of systems engineering and management roles in models of coordination effectiveness.

CHAPTER 1

Introduction

This dissertation is about the role that individuals play in the system-level coordination of large-scale and complex engineered systems (LSCES). Coordination – ensuring interfacing groups are working with consistent data, interpretations of that data, and consistent goals – becomes necessary when design work is distributed among different individuals or teams. This is generally the case in the design of aircraft, rockets, automobiles, and other LSCES. These projects involve thousands of people in multiple locations, and often multiple companies, who need to make internally consistent decisions on millions of design variables in order to produce a single artifact.

A widely-used approach to the design of LSCES is decomposition-based design. Decomposition or *partitioning* of a system into subsystems creates internal interfaces between subsystems. *Coordination* requires working across those internal interfaces. Central to this coordination task is communication and information exchange among engineers and designers. These stakeholders must develop a clear understanding of how subsystems relate to each other and affect overall system performance. The goal of coordination is successful system integration.

Successful *system integration* refers to the assurance that the design of the recomposed system is equivalent to the original system design that would have been achieved without partitioning (Papalambros and Wilde, 2017; Blanchard and Fabrycky, 2011). The final system’s capability should include all desired technical functions, and all interfaces should be

well defined (Sage and Lynch, 1998). System integration is technical in nature. Yet, the coordination required for system integration is also a management activity. Within the design organization, integration and coordination requires different disciplines to work together effectively (Sage and Lynch, 1998) and to share information between the tasks of analysis, design, and test (Johnson, 1997; Blanchard and Fabrycky, 2011). Integration is challenging due to the many interdependent functions required of LSCES and the heterogeneous expertise required of the design organization (Madni and Sievers, 2014).

Coordination and integration are particularly challenging in the design of LSCES. Complexity in natural and engineered systems is described as unpredictable behavior during design or operation (Bloebaum and McGowan, 2012; Pennock and Rouse, 2016). Unpredictable behavior at the system level indicates that there is no direct mapping between subsystem behavior and system behavior (Bloebaum and McGowan, 2012). This can be due to dynamic behavior of subsystems, incomplete understanding of natural phenomena or novel technology, or the process of learning about the design during development (Gao, Barzel, and Barabási, 2016; Pennock and Rouse, 2016; Bloebaum and McGowan, 2012; Page, 2015).

Errors in systems integration and interface definition have been attributed to the “fuzzy” nature of complex engineered systems or errors in the flow-down (target cascading) of requirements during early conceptual design (Madni and Sievers, 2014). When problems arise at interfaces during the system integration phase of design, long after early conceptual design is complete, one possible outcome is that the system as designed cannot function as intended. Failure of verification or validation requires costly redesign efforts.

1.1 Research Questions

This research identifies and characterizes how individuals contribute to coordination in LSCES design work. Understanding how individual actions impact coordination outcomes

is a first step toward improved system design practices and more successful system integration in the future. The focus of this dissertation is on the strategies – sets of actions and behaviors – individuals use to facilitate coordination during LSCES design work.

Much of the literature on coordination methods and best practices uses a top-down perspective: methods tend to be prescriptive and implemented from the top of the organization. This is consistent across literature in organizational and management science, design optimization, systems engineering, and software engineering.

We challenge this perspective by acknowledging that the work to accomplish coordination is done by people. We argue that an improved understanding of how coordination is accomplished comes from studying individuals' actions and behaviors and their connection to system-level coordination efforts. This bottom-up perspective complements existing literature on coordination in decomposition-based design by considering the role of individual actors' behaviors in successful coordination.

This dissertation addresses the following research questions:

1. What approaches and behaviors do individuals use to facilitate effective system-level coordination of distributed design work?
2. What impact do individuals' behaviors to facilitate coordination have on system-level coordination?

1.2 Approach

This research is informed by ideas from multidisciplinary design optimization, organization theory, and social network theory. These perspectives together allow a holistic qualitative and quantitative analysis of coordination in complex engineered systems design.

1.2.1 Methodology

The research was conducted in two stages: the first stage was exploratory, aiming to define and refine the research questions. This was accomplished through qualitative and quantitative data collection and analysis. The second stage was descriptive, developing a simulation model to explore the likely practical impacts of exploratory findings.

This dissertation presents three studies that cover the two stages mentioned. The first is qualitative, based on semi-structured interviews of industry experts. Thematic analysis of the interview responses helped shape the research questions and our understanding of what individual behaviors are used during design.

The second study is quantitative and is based on surveys of novice designers. We analyzed survey responses using text analysis and network analysis to discover what roles, or combinations of tasks and communication, are adopted to support coordination in small design projects. This study also contributes to our understanding of individual behaviors used in support of coordinating distributed design work.

The third study is based on a descriptive agent-based model that simulates the impact of individual behaviors on the system-level performance of a distributed design task requiring coordination. The model itself serves as a platform for parametric analysis and exploration of what behaviors are more and less impactful on system outcomes. Results from this model contribute a quantitative analysis of individuals' coordination-facilitation behaviors.

1.2.2 Discipline background

The research conducted is inherently interdisciplinary and integrative. The research methods selected draw from social science, information science, and engineering. Thematic analysis is a typical stage of grounded theory development, common in social science research (Braun and Clarke, 2006; Nowell et al., 2017). We use it as a standalone method, suitable for our purpose of exploring and refining research questions. Text analysis is a common tool in information retrieval, frequently used in products and services such

as search engines and automated document analysis (Manning, Raghavan, and Schütze, 2008).

Network analysis is used in physics and mathematics-based study of natural and complex engineered systems, ecological and epidemiological modeling, and sociological modeling of people, groups, and organizations (Newman, 2018; Wasserman and Faust, 1994; de Weck, Roos, and Magee, 2011). We use social network analysis, employed by social scientists to study the connection between individual and aggregate behavior. Social network analysis is used to develop theories of how communities behave, and to inform the development of tools to leverage and support those communities (Easley and Kleinberg, 2010; Tichy, Tushman, and Fombrun, 1979; Borgatti and Foster, 2003; Temdee and Korba, 2001; Ogata et al., 2001).

Finally we use agent-based simulation modeling to model organizational processes. Simulation modeling is common in engineering research as a way to test a wide range of parameters with carefully controlled environments. It is difficult to study organizations as part of a controlled experiment, thus we adopt simulation. Agent modeling is the study of how simple rules adopted by individuals can cause aggregate or emergent behaviors. It is used often in the study of complex systems, including ecological models (Railsback and Grimm, 2012; Wilensky and Rand, 2015) and social systems (Epstein and Axtell, 1996; Miller and Page, 2007; Gero, 2002).

1.3 Summary of Findings

The two exploratory studies presented in this dissertation identified individual actions and behaviors used to support the coordination of distributed design tasks. We found that industry practitioners use a combination of authority-based actions (setting rules, plans, and structures for work) and what we call empathetic leadership-based actions (asking questions, using and encouraging empathy, and tailoring interactions) to support coordination.

We found that student designers adopt roles within their teams that are more and less proactively communicative (initiators and non-initiators), as well as focused on more general or more specific tasks. Teams also had a combination of initiators and non-initiators, and typically had a combination of generalists and specialists. Teams with a single initiator appear to be more hierarchical team organizations and teams with multiple initiators appear to be more non-hierarchical team organizations. These exploratory studies indicated that multiple communication approaches are used in support of coordination work, and specifically, that a combination of different coordination strategies may be most beneficial.

The exploratory studies together identified complementary strategies of more active communication, based on empathetic leadership and including proactive communication and engagement with peers, and more passive communication, based on authority and including formalized or standardized interaction with peers. These strategies we call *Active* and *Passive*, terms which we define and describe first in Chapter 3 and revisit in Chapter 5. We developed computational agents that embody simple versions of these Active and Passive behavioral archetypes. Agents were tasked with completion of a distributed computational design task, a multiclass classification problem. We found that Active and Passive agents achieve similar individual performance on partitioned tasks by following their prescribed coordination strategies. The aggregate performance, however, is typically improved with the addition of Active agents. We identified several input parameters to the model as impactful. These include the network structure agents interact along, what fraction of each type of agent is included, and where each agent type is located within the network. We also include a plan for model validation. Limitations and directions for future work are presented throughout.

1.4 Contributions

The intended audience for this work is the systems engineering and design community, including both researchers and practitioners. The first contribution of this dissertation is to the development of systems engineering theory and principles underlying effective systems engineering practice. We do this by highlighting that coordination is a central task of systems engineering personnel in engineering design organizations, and pointing out the theory relevant to coordination in multiple disciplines. This work uniquely shows how this theory applies to and describes systems engineering activities, particularly in relation to engineering design. This includes coordination as described in the literature, as well as theory related to behaviors identified through this research as supporting system-level coordination.

The second contribution of this work is the development of quantitative approaches to describe and evaluate coordination practice in decomposition-based system design. We do this by demonstrating a novel method to identify coordination roles, or the communicative behaviors and tasks individuals are engaged in during design work. We also use an agent-based simulation model to assess the impact of communicative and information-seeking behavior on design task performance. This model also serves as a platform that allows extensive parametric analysis, permitting exploration of a variety of design tasks, organizational structures, and coordination behaviors.

Finally, this work illustrates the importance of coordinator roles in effective distributed design tasks. Frequently, managerial and integrative roles are not included in measures of coordination effectiveness, which we argue is not representative of true systems engineering practice. This dissertation is a step towards the inclusion of systems engineering and management roles in such models of coordination effectiveness.

1.5 Dissertation Outline

Chapter 1 discussed the challenge of large-scale and complex engineered system design, and particularly the challenge of coordinating diverse distributed tasks. Chapter 2 presents terminology relevant to the discussion of decomposition-based design and a review of literature on coordination practices. Chapters 3-5 present each of the three studies contained in this dissertation. Chapter 3 describes our process of semi-structured interviews and thematic analysis, as well as a discussion of where our findings are consistent and inconsistent with existing literature. Chapter 4 describes the process of surveys, text analysis, and network analysis we used to identify coordination behaviors adopted by novice designers. Chapter 5 describes the agent-based simulation model that is based on results of the first two studies in Chapters 3 and 4. We describe the model and results from it, as well as next steps, including a comprehensive validation plan. Chapter 6 summarizes conclusions and discusses several avenues for future work.

CHAPTER 2

Background and Terminology

In Chapter 1, we introduced decomposition-based design, which proceeds through partitioning of work and coordination of its completion. In this chapter, we introduce terminology relevant to the discussion of partitioning and coordinating design work. First, we review different system representations. Through this we introduce partitioning terminology. Then we review how coordination is described in several disciplines. We introduce literature in Multidisciplinary Design Optimization (MDO), Organization Science, Software Engineering, Engineering Design, and Systems Engineering. Then we summarize how the coordination of partitioned tasks in decomposition-based design is accomplished according to the literature.

2.1 System Representations

A system can be represented by its physical *components* or by its *functions*. The former focuses on the system's embodiment, whereas the latter focuses on the system's purpose. If you consider the design of an airplane, the component-based view yields a composition of wings, fuselage, and engines, joined by mechanical bolts, material welds, and electrical wiring. This part-based decomposition is also referred to as *Object Decomposition*. The function-based view may depict a vehicle that carries cargo, produces thrust, and provides lift. These functions interact through the design variables that enable those functions, for example the curvature of the wing, the size of the engine, and the diameter of the fuselage.

Function-based or discipline-based decomposition is also referred to as *Aspect Decomposition*. These representations complement each other by shedding light on different aspects of the same system.

In this dissertation, a system's *subsystem* refers generically to a subset of the system, which could be either a function or a physical component. To depict these subsystems, both matrix-based and graph-based representations of systems can be used. Each has advantages and disadvantages, which will be discussed in the following sections. In addition, as in this simple example, the method of representing system elements also dictates how the interactions and interfaces between those elements are represented.

2.1.1 Matrix-based Representations

The information of the system elements and their relations can be displayed as a matrix. The functional dependency table, or FDT, is a function-based representation of the system that shows the system functions and the variables on which those functions depend (Wagner and Papalambros, 1993). This representation therefore requires some knowledge of the mathematical equation that relates variables that describe the system and the function that describe its behavior. A second matrix-based representation is the design structure matrix, or DSM. The design structure matrix can be thought of as an adjacency matrix, meaning that the rows and columns of the DSM are the same system elements. The binary values within the matrix therefore indicate relationships between elements. Due to its versatility, the elements included in the DSM can be functions, components, or process steps.

The DSM was originally conceived as a planning tool for the design of complex systems, where the system elements represented are the design process steps, or design decisions, needed to complete the system (Steward, 1981). In this format, the marks within the DSM indicate whether the row element depends on the column element, meaning that if the matrix cell ij is marked, then decision j must be made before decision i can be made. If decision i precedes decision j , then the process is clearly cyclical, and may require several

iterations to converge to a final design. The value of the DSM as a visualization tool is that these cycles can easily be identified – these out of step decisions sit above the diagonal – and further may be eliminated via basic row and column transformations on the matrix. However, the utility of this tool depends on the accurate determination of precedence, i.e. which decisions impact other decisions (Steward, 1981).

The initial conceptualization of the design structure matrix was as a tool to refine the design process; it has since been used for system architecture design and organizational design (Browning, 2001; Sosa, 2007; McCord and Eppinger, 1993; Pimmler and Eppinger, 1994). The system elements included in the DSM may be functions, technical components, or social components, therefore marks in the DSM indicate some connectivity or dependency between those elements. As a design tool, the architecture or organization DSMs are used to group rows and columns of system elements to minimize the interactions between other elements, in effect modularizing the system (McCord and Eppinger, 1993; Pimmler and Eppinger, 1994).

2.1.2 Graph-based Representations

The information in a functional dependency table or a design structure matrix can also easily be translated into a graph or network representation. For a technical system, the *vertices* or *nodes* of the corresponding graph are technical elements, and the *edges* between them represent the shared variables or interfaces. For a social system, the vertices are people or groups of people, and the edges between them represent social relationships. Networks are used to model structures of systems, behaviors resulting from those structures, and the mechanisms by which edges facilitate transfer of resources, information, and knowledge between nodes of the network (Newman, 2003). Graph or network representations are also useful for performing clustering or partitioning operations to identify structures within the system (Newman, 2006). This can be particularly useful as a complement to a matrix-based representation of a system. For example, one method for partitioning a FDT is to perform

graph partitioning on the equivalent representation (Wagner and Papalambros, 1993; Krishnamachari, 1996).

Network representations of systems have the potential to provide insight into the mechanisms by which networks are formed as well as the behavior that emerges from particular structures or dynamic entities within the network (Newman, 2003; Wasserman and Faust, 1994; Strogatz, 2001). In design research, network representations are valuable to study design processes, organizational behavior, and more (Chen et al., 2018; Parraguez and Maier, 2016; Mosleh, Ludlow, and Heydari, 2016). Several heuristics have been developed as a result of these studies, including the concepts of transitivity and bridges. *Transitivity* describes the tendency for triadic groups to close, i.e. the tendency of your friend's friend to be your friend as well (Newman, 2003; Granovetter, 1973). *Bridges* are nodes within networks that span disjoint groups and therefore have access to diverse information, filling what are known as structural holes (Burt, 1992; Granovetter, 1973). Both of these properties are of interest in the study of coordination across interfaces in the design of LSCES. For example, it is the role of systems engineers and integrators to facilitate triadic closure across design groups, i.e. to ensure that critical interfaces are addressed. Their ability to do so may depend on their ability to identify structural holes within the organization and bridge them effectively. Identifying a link between a system's structure and its behavior is of particular interest in the study of LSCES (de Weck, Roos, and Magee, 2011), which may be accomplished through the use of complex networks as representations of both the design organization and the technical system.

2.1.3 Partitioning and Coordination

Matrices and graphs or networks show how a system is partitioned into subsystems. Partitioning and coordination strategies are not independent: the selection of a partitioning strategy is connected to coordination requirements, and the selection of a coordination strategy is connected to required partitioning (Allison, 2008). This research will focus primarily

on coordination activity, but with the awareness that this activity is not free of the context provided by partitioning.

2.2 Review of Literature: Coordination

Coordination is defined in the dictionary as “the organization of the different elements of a complex body or activity so as to enable them to work together effectively” (Stevenson and Lindberg, 2011). Literature on coordination spans several disciplines. We focus here on four in particular: multidisciplinary design optimization (MDO), organization science, engineering design, and software and systems engineering. This selection of disciplines considers respectively, the analytical formulation and solution of system design problems, the structure and nature of organizations that may be engaged in system design work, and the design and development of systems alongside methods and tools to support that design.

McGowan (2014) distinguishes between four types of interactions that occur across organizational and technical interfaces during system design: *Connecting* is akin to the assembly of parts. Connected items work together, but are designed and developed separately. *Collaboration* is a process of bringing together heterogeneous parts to form an integrated system. Ownership of the parts remains distinct, but individuals and the artifacts they design are tailored to facilitate integration. *Collective* design is even more collaborative; the result is a fully homogeneous and co-created artifact that results from shared and integrated expertise. Finally, *Coordination* is described as a process of ensuring diverse system elements remain integrable as they evolve, ensuring that the system-level needs can be met as the component parts are defined. Coordination is described as negotiation or orchestration among constituent parts and people (McGowan, 2014; Ryschkewitsch, Schaible, and Larson, 2009).

We mention these distinctions between coordination, collaboration, connecting, and collective work because these terms are often used interchangeably in literature. The pro-

cesses as described by McGowan are similar, and even complementary, but are ultimately distinct. Here we focus on coordination, which is differentiated by the focus on an overall system objective, or intent.

2.2.1 Design Optimization

Multidisciplinary Design Optimization or MDO arose from the need to analyze systems whose constituent elements are modeled with various discipline-based analysis tools (Sobieszczanski-Sobieski, 1995). The analysis models integrated under MDO use different equations, assumptions, and variables. However these subproblems cannot be solved on their own: maximizing individual subproblems without regard to how the solutions need to fit together will generally not yield a feasible system-level solution, or one that is most desired (optimal) (Allison, 2008). Thus the solution of subproblems requires *coordination*: “the task of guiding subproblem solutions toward an optimal system design” (Allison, 2008).

If all interfaces between analysis models are identified, they can be combined. It is only then that the system behavior can be modeled, and further, optimized. An optimized system model yields not only optimal values for subsystem and system parameters, but also *consistent* values of variables used as inputs to or outputs from multiple models.

The organization of discipline-specific models under an optimization scheme in order to coordinate the search for an optimal system solution is called the MDO *architecture*. A review of common MDO architectures is given in (Martins and Lambe, 2013), where architectures can be divided generally into monolithic or distributed formulations. Some of the distinguishing features of MDO architectures are the location and purview of the decision maker(s), and the approach to coordinating the partitioned analysis models. As an example, the all-at-once (AAO), analytical target cascading (ATC), and collaborative optimization (CO) strategies are compared. The AAO problem statement includes all linking variables and discipline constraints into a single problem statement, requiring no parti-

tioning or coordination (Cramer et al., 1994). ATC and CO both make use of a hierarchical partitioning strategy, dividing the optimization problem into subproblems, and a centralized coordination strategy (Kim et al., 2003; Kroo and Manning, 2000). In ATC a target value is selected for the overall system which is distributed to each subproblem according to the partitioning strategy. Each subproblem is individually optimized, coordinated through the use of this target value. This process iterates until the targets and responses are consistent. Collaborative optimization also distributes optimization across discipline subproblems, but does so by creating local copies of all shared variables along with consistency constraints for each subproblem. Coordination is accomplished through iterative updates of targets and responses or consistency constraints among subproblems.

2.2.1.1 Coordination Methods in MDO

The selection of MDO architecture is dependent on the physical system's architecture and the couplings that exist between discipline models (Martins and Lambe, 2013). Coordination methods in MDO have been developed to take advantage of unique problem properties, particularly the existence and strength of coupling between subproblems. Examples include the combination of ATC and CO to coordinate across two different partitioning strategies (Allison et al., 2005), the use of global sensitivity equations and modified GSEs to calculate coupling strength (Hajela, Bloebaum, and Sobieszczanski-Sobieski, 1990; Alyaqout et al., 2011), improved computational efficiency through suspension of weakly coupled subproblems or reformulation of subproblem optimizers (Alyaqout et al., 2011; Alexandrov and Lewis, 2002), and the inclusion of uncertainty (Yao et al., 2011). However, MDO remains unable to fully account for the impacts of humans on the design process (Simpson and Martins, 2011; Bloebaum, Collopy, and Hazelrigg, 2012).

A solution to a MDO problem yields a set of fully consistent coupling variables between modeled systems and insight into system behavior. What is difficult to model includes factors such as incomplete information, unknown or emergent couplings, and the role of

humans as designers that sometimes make mistakes. For example, a central decision maker in an optimization routine is capable of simultaneous processing of information and simultaneous delivery to connected analysis functions. This ideal decision maker also has full information about the system. Neither the ability to simultaneously process multiple inputs nor the assumption of complete information is reasonable for a human decision-maker (Simon, 1955).

Coordination in MDO is a deterministic method (McGowan, 2014); the choice of method coupled with the choice of partitioning (Allison, 2008). Coordination is a pre-meditated strategy to transfer information between concurrent problem solving processes, selected based on existing information about how a problem is divided into smaller parts. The end goal is subproblem solutions that together comprise an optimal system design solution.

2.2.2 Organization Science

Coordination activity during design is contextualized by organization, which dictates roles, lines of communication, trust, authority, and accountability (Galbraith, 1974; Simon, 1973; Gulati and Singh, 1998). Scott and Davis describe three perspectives of organizations: rational systems where all actors work in concert to achieve a common objective, natural systems where actors have diverse goals but use the organization as a common source of information and knowledge, and open systems where actors selectively join and separate to achieve their goals (Scott and Davis, 2006).

Most of the foundational literature on coordination in organizations builds on the rational system model of organizations. In the rational system model of organizations, every organization has a central goal (March and Simon, 1958; Blau, 1974; Scott and Davis, 2006). A defining feature of a formal organization is “the existence of procedures for mobilizing and coordinating the effects of various, usually specialized subgroups in the pursuit of joint objectives” (Blau, 1974). Similar to distributed optimization, individual expertise

is partitioned into groups and teams. Their work is then coordinated through the development of rules, plans, and channels for rapid feedback (March and Simon, 1958; Thompson, 1967; Van De Ven, Delbecq, and Koenig, 1976).

2.2.2.1 Coordination Methods in Organization Science

Another foundational theory in organization science is *Contingency Theory*. Contingency Theory states that the best approach to organizational design, or coordination strategy, is dependent on an organization's goals and environment (Thompson, 1967; Lawrence and Lorsch, 1967; Pugh and Hickson, 2007). Contingency theories of coordination have focused on the appropriate coordination methods given the degree of task uncertainty or non-routineness and the interdependence between those tasks. Empirical studies have found that the combination of task uncertainty and task interdependence, or *complexity*, is correlated with the use of more coordination methods. Methods are added roughly additively (Van De Ven, Delbecq, and Koenig, 1976) with increasing task complexity. Setting standard rules and procedures (including formalizing team roles) is the most basic method, followed by the addition of planning to pace the timing of simultaneous work, followed by feedback or mutual adjustments between teams to continuously update work (Thompson, 1967; Van De Ven, Delbecq, and Koenig, 1976). A central component of the uncertainty and interdependence of tasks is the information processing required of individuals and teams in order to actually complete a task (March and Simon, 1958; Galbraith, 1974; Tushman, 1979; Malone, 1987).

In addition to known coordination needs, unknown but anticipated coordination needs have been discussed in the context of organizational alliances and the design of visualization tools (Gulati and Singh, 1998; Dossick and Neff, 2010). This work indicates that the frequency of anticipated coordination work and trust between parties are significant factors in determining how that coordination is carried out. New technologies may improve awareness of how system elements are related, however trust in the technology as an accu-

rate representation of those interactions is key to its adoption. In addition, the identification of coordination needs does not necessarily translate to realized coordination if the organization structure makes developing those communication channels expensive (Dossick and Neff, 2010). Coordination across groups is typically reserved for specific coordination roles within the organization (Cataldo and Herbsleb, 2008). Anticipated coordination may also drive the creation of new organizational channels for information transfer in a new alliance (Gulati and Singh, 1998), but unplanned or unexpected coordination needs are more difficult to address. These works reinforce the impact of organization structure on coordination activity, both in identifying dependencies and in coordinating across those dependencies.

Other approaches to understanding coordination between individuals specifically in the context of LSCES design include economic theory (Mosleh, Ludlow, and Heydari, 2016), game theory (Vermillion and Malak, 2015), and exploration of the cognitive processes of systems engineers (Greene, Papalambros, and McGowan, 2016; McGowan, 2014).

Coordination in organizations is about managing interdependencies between groups of people (Malone and Crowston, 1994). Methods for coordination are prescribed top-down through the structure and partitioning of the organization into divisions, teams, and individual roles on teams, as well as the characteristics of the tasks those divisions, teams, and individuals are asked to complete. The more complex the tasks undertaken by the organization, the more rules, plans, and channels for feedback are incorporated into formal organization design. Practically, these rules, plans, and feedback channels may be interpreted respectively as design and manufacturing processes, project management and schedules, and office layouts and meetings designed to bring interfacing groups together.

2.2.2.2 Comparing Organizational Coordination to MDO

One approach to interpreting coordination mechanisms among designers is to compare observed activity to established MDO architectures. Past research has illustrated that some decision makers within organizations use a process that is similar to a hierarchical MDO al-

gorithm termed a ‘hybrid MDO–game theoretic’ model (Austin-Breneman, Yu, and Yang, 2015; Honda et al., 2015). While information sharing among individuals in the organization was found to follow a pattern similar to an algorithm, it was not guaranteed to converge due to the incomplete information shared between subsystems (Austin-Breneman, Yu, and Yang, 2015). This work illustrates the importance of both the information sharing architecture as well as the quality of that information in understanding the effectiveness of coordination activity within organizations.

2.2.3 Software and Engineering Design

The disciplines of software engineering and engineering design consider the design work an organization undertakes. While MDO is an approach to systems design that focuses on coordination of partitioned elements, another approach is to start from the process of partitioning to effect desired coordination. Several best practices for the design of products and systems come from modularization or partitioning of work, primarily to reduce the need for coordination (Parnas, 1972; Souza et al., 2004; Maier and Rechtin, 2009; Panchal, 2010). In addition, the need for frequent inter-group communication is considered one of the primary challenges of highly integrative design work (Pimmler and Eppinger, 1994; Cataldo et al., 2006), and is one driver for developing a more modular design process (Bosch and Bosch-Sijtsema, 2010). Another set of objectives might be based on life-cycle properties of the product or system, suggesting partitioning strategies based on maintenance requirements, changeable technologies, or product differentiation (Bayrak et al., 2018; Dahmus, Gonzalez-Zugasti, and Otto, 2001; Asikoglu and Simpson, 2012).

Module identification or design is often accomplished using a DSM- or FDT-based approach, where the goal is to rearrange the matrix of interactions to reduce interactions across some set of modules. One key step in this process is the identification of *linking variables*, which are those that bridge system elements as coupling or shared variables. In a complex system, many variables will be linking variables, but to cleanly divide a

system into elements, some of the most-connected variables will need to be set aside as integrating elements. The selection of this set of linking variables or functions to treat as an integrative element by the introduction of design rules (Baldwin and Clark, 2000) or graph partitioning (Wagner and Papalambros, 1993; Krishnamachari, 1996) defines the set of remaining variables or functions to divide into modules. The selection of integrating or coordinating elements within the system therefore can have a large impact on the resulting architecture: few linking variables may result in a small number of larger modules, and many linking variables may result in a large number of smaller modules.

Once a partitioning objective has been selected, modules can be identified by matrix transformations on the DSM or FDT. This is typically done by clustering based on interaction type. Modifications and improvements to this approach include clustering based on multiple interaction types simultaneously (Pimmler and Eppinger, 1994), weighting values based on likelihood of technology change (Asikoglu and Simpson, 2012), using an optimizer to select module groupings based on system-level performance measures (Bayrak et al., 2018), and the combination of DSM coupling measures with functional similarity measures (Borjesson and Hölttä-Otto, 2013). Other approaches focus specifically on assembling elements into new products, which include graph or design grammars (Schmidt and Cagan, 1998), functional modeling (Dahmus, Gonzalez-Zugasti, and Otto, 2001), and the use of a component-function catalog to assemble novel products (Bryant et al., 2005).

2.2.3.1 Coordination in Software and Engineering Design

Design via modularization illustrates a focus on ensuring that the subsystem interfaces are simple connections between complex elements. Modular design is considered good practice both in systems architecting (Maier and Rechtin, 2009) and software design (Parnas, 1972); however it presents challenges as systems become more complex. The implementation of simple interfaces between technical interfaces also promotes the practice of hiding the complex nature of a technical element behind an interface (Parnas, 1972). In practice,

this is often accomplished in software by the use of application-program interfaces (APIs), which have been shown to cause coordination challenges due to the increasing complexity of both APIs and the modules behind them (Souza et al., 2004). Communication tends to follow organizational boundaries (Kleinbaum, Stuart, and Tushman, 2008), and interfaces across those boundaries are often sources of inter-team communication lapses (Sosa, Eppinger, and Rowles, 2003; Sosa, Eppinger, and Rowles, 2004; Dossick and Neff, 2010; Kraut and Streeter, 1995; Galviņa and Šmite, 2012). Geographically distributed teams face further challenges due to communication delays and differing norms (Herbsleb and Mockus, 2003; Olson and Olson, 2000).

A study of inter-team collaboration at a software development company found that individuals with more experience or those with management roles are more likely to identify interdependencies that require coordination; however, individuals of different roles focus on different kinds of interdependencies (Grubb and Begel, 2012). This is consistent with the observation that communication within an organization tends to fall along organization lines (Kleinbaum, Stuart, and Tushman, 2008). While this work investigates known dependencies, it does not address the question of the motivations or mechanisms for coordination work. One explanation is that coordination tends to be reactive rather than ongoing, and is seen more often after an identified failure (Panjer, Damian, and Storey, 2008). Another explanation is that coordination across groups is typically reserved for specific coordination roles within the organization (Cataldo and Herbsleb, 2008).

External mechanisms also have an effect on coordination. Research on coordination activity during the design of software systems has focused on the impact of geographic dispersion. One perspective is that informal communication is critical for issue resolution, and dispersion within and between design teams is problematic because it disrupts this informal communication (Herbsleb and Grinter, 1999; Herbsleb and Mockus, 2003). Two mechanisms are proposed for this disruption. The first is lack of context or ‘teamness’, which includes understanding of who is responsible for what work and use of compatible

processes for accomplishing work (Herbsleb, 2007; Herbsleb and Mockus, 2003). The second mechanism is the sheer number of people involved in a project, supported by the observation that more people tend to be involved in distributed work than in co-located work, and therefore resolution of issues takes longer (Herbsleb and Mockus, 2003). The observation that fully connected teams require more time to resolve issues is supported by the finding that highly connected groups or cliques are particularly vulnerable to defects in social systems (Piccolo, Lehmann, and Maier, 2018) and technical systems (Zimmermann and Nagappan, 2008). This suggests that some similar mechanisms may be at play in technical and social networks.

2.2.4 Systems Engineering

Systems engineering is a process to enable the design of a system, with a focus on managing the complexity inherent in the design of LSCES (Maier and Rechtin, 2009; Blanchard and Fabrycky, 2011; Hazelrigg, 1996; de Weck, Roos, and Magee, 2011; Johnson, 2002). Most systems engineering process standards include some variant of requirements definition, detailed design, test and integration, verification and validation, and operation and maintenance (Johnson, 2002; United States Department of Defense, 2017; National Aeronautical and Space Administration, 2007; International Council on Systems Engineering (INCOSE), 2004; Doran, 2006). These processes formalize partitioning and coordination through technical control processes including requirements management, interface management, and configuration management (Johnson, 2002; National Aeronautical and Space Administration, 2007; Blanchard and Fabrycky, 2011). This documentation supplies the information needed to make coordinated, or internally consistent, decisions at every step of design. However documentation alone does not ensure coordination: the documentation must be written, stored, accessed, and understood the same way by others. If systems engineering is to manage complexity from both technical and social sources, it must be more than good documentation (Ryschkewitsch, Schaible, and Larson, 2009; Griffin, 2010; Tri-

antis and Collopy, 2014; Bloebaum and McGowan, 2012).

These tasks reiterate the notion that systems engineers have a managerial role, with responsibility to ensure successful system integration throughout the life-cycle (Sage and Lynch, 1998). Focusing on the design aspect of systems engineering, systems engineers can be considered as decision-makers whose task is to select the design that is most preferred based on evaluation of available options under risk and uncertainty (Hazelrigg, 1996) or as enablers of elegant design (Griffin, 2010).

As described above, systems engineering requires continual integration of technical elements via design and management activities. However, there is little consensus as to the actions systems engineers and coordinators ought to undertake to achieve an integrated, optimal, design (Bloebaum and McGowan, 2012; Bloebaum, Collopy, and Hazelrigg, 2012). The existence of a social component to systems engineering has been recognized in multiple systems engineering competency models (Hutchison, Henry, and Pyster, 2016; Metzger and Bender, 2007; Woodcock, 2010; Williams and Derro, 2008; Frank, 2012; Pietrzyk and Handley, 2016). These competency models have identified skills and behaviors exemplified by effective systems engineers in defense and commercial industry. Skills include basic engineering know-how, holistic thinking, familiarity with systems engineering lifecycle and process, management and leadership abilities, and the ability to collaborate and communicate across diverse groups (Hutchison, Henry, and Pyster, 2016; Metzger and Bender, 2007; Woodcock, 2010; Williams and Derro, 2008; Frank, 2012; Pietrzyk and Handley, 2016). What is not directly included in these competency models is an illustration of how the interpersonal skills identified are used to support system-level coordination of design work.

2.2.5 Measuring Coordination

2.2.5.1 Conway's Law

Finally, there is another body of literature that asks what coordinated work processes *should* look like. One of the first treatises on the relationships between the technical system and the organization that designs it is Conway's law, which states that "organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations" (Conway, 1968). The implicit assumption made here is that there exists a mapping between elements of the organization and elements of the technical system, as shown in Figure 2.1(a). The hypothesis presented by Conway is illustrated in Figure 2.1(b). Conway's argument is that the process of partitioning a design organization into teams implicitly constrains the set of design alternatives that organization is capable of creating. The implication is that for a static organization, tasks will be delegated along the organizational boundaries that exist, therefore creating a system design that has interfaces along those same lines. Further, Conway argues that degradation of inter-organizational communication likewise directly leads to breakdown across technical interfaces, resulting in cases of integration failure.

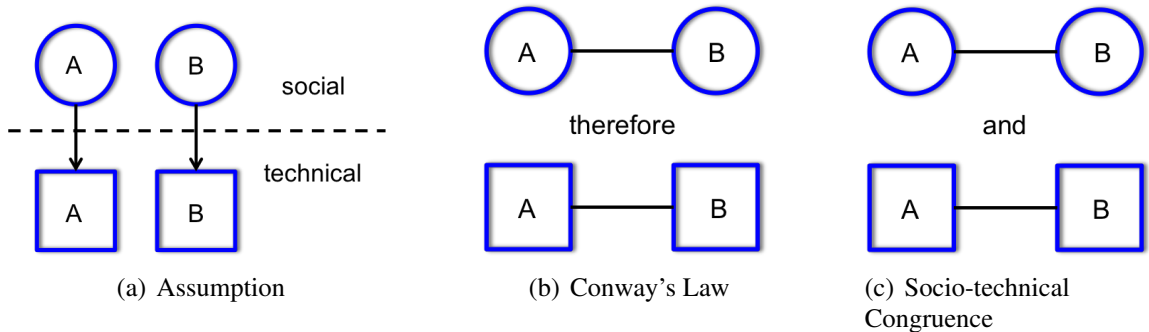


Figure 1: Illustrations of a) assumption of one-to-one mapping between organizational groups (social) and subsystem elements (technical); b) hypothesis given by Conway's Law; c) expected mapping between organization and technical system given socio-technical congruence

2.2.5.2 Mirroring

Conway's law makes explicit the hypothesis of a directional mechanism between organizational structure and product structure. Other research has suggested the reverse mechanism may also be at play: that the design organization's structure is constructed to match the proposed system architecture (MacCormack, Baldwin, and Rusnak, 2012; Pugh and Hickson, 2007). This has been in turn called the Mirroring hypothesis (Colfer and Baldwin, 2016) and is supported by empirical evidence of matching structures (MacCormack, Baldwin, and Rusnak, 2012; Le and Panchal, 2012) and that matched structures correlate to higher project success (Vrolijk and Szajnfarber, 2015). This work provides evidence in support of the hypothesis that there is a relationship between organizational and product architectures.

2.2.5.3 Socio-Technical Congruence

Building on the assumption that there is a relationship between groups in the design organization and elements in the technical system, several approaches have been taken to try to characterize these technical and social interfaces in some way. Socio-technical congruence is a measure meant to illustrate the degree of similarity between the task dependency present in a technical system and the communications that occur within the design organization (Cataldo et al., 2006). Congruence is calculated by creating a DSM of element interdependencies within the technical system, and comparing it to a DSM constructed of social interactions within the organization. Perfect congruence is achieved when the matrices are identical, yielding the interpretation shown in Figure 2.1(c).

2.2.5.4 Challenges in Measuring Coordination

The measure of socio-technical congruence however has several limitations. The first is that there is no consistent method for weighting the interface types of the technical or organizational system. Unweighted interactions, weighted interactions according to frequency, and weighted interactions due to work on a shared element result in structural congruence

values ranging from 25 to nearly 75 percent for projects considered successful (Cataldo, Herbsleb, and Carley, 2008; Ehrlich et al., 2008; Kwan, Schroter, and Damian, 2011). The second limitation regards the type of interfaces considered: in many implementations of STC, only functional dependencies within software are considered, which focuses only on implementation decisions within software and neglects any partitioning or architecture decisions made prior. In addition, directional interfaces are not considered in the original formulation (Cataldo et al., 2006), but are considered in other formulations (Valette et al., 2007; Ehrlich et al., 2008). Finally, socio-technical congruence is a static measure which has been shown to decrease with time (Le and Panchal, 2012), suggesting that the measure could be improved by a distinction between resolved design dependencies that do not require coordination activity and yet to be resolved design dependencies that do require coordination.

Given the observation that not every successful system design has 100% congruence, some research has been directed at identifying the cause of incongruence as defined (Sosa, Eppinger, and Rowles, 2004; Ehrlich et al., 2008). Sosa, Eppinger, and Rowles (2004) proposed the use of design structure matrices for the organization and the technical system to identify unaddressed interfaces, examples of which is illustrated in Figures 2.2(a) and 2.2(b). A similar question was posed by Ehrlich et al. (2008), asking what is the source of 'gaps' in the organizational network. In this research, a gap is identified as the lack of inter-team or intra-team interactions predicted by technical interfaces. Sosa et al.'s case study suggests that technical interfaces that span disciplines tend to be those that are not addressed in the organizational DSM.

Some of those missed interactions may exist as indirect communications, e.g. as shown in Figures 2.2(b) and 2.2(d) (Sosa, Eppinger, and Rowles, 2004). These indirect interactions that are not directly matched may exist on the technical or social side, as illustrated in Figure 2. These findings suggest that avenues to improve this work is to examine the methods of social communication and coordinated work, as well as explicitly account for

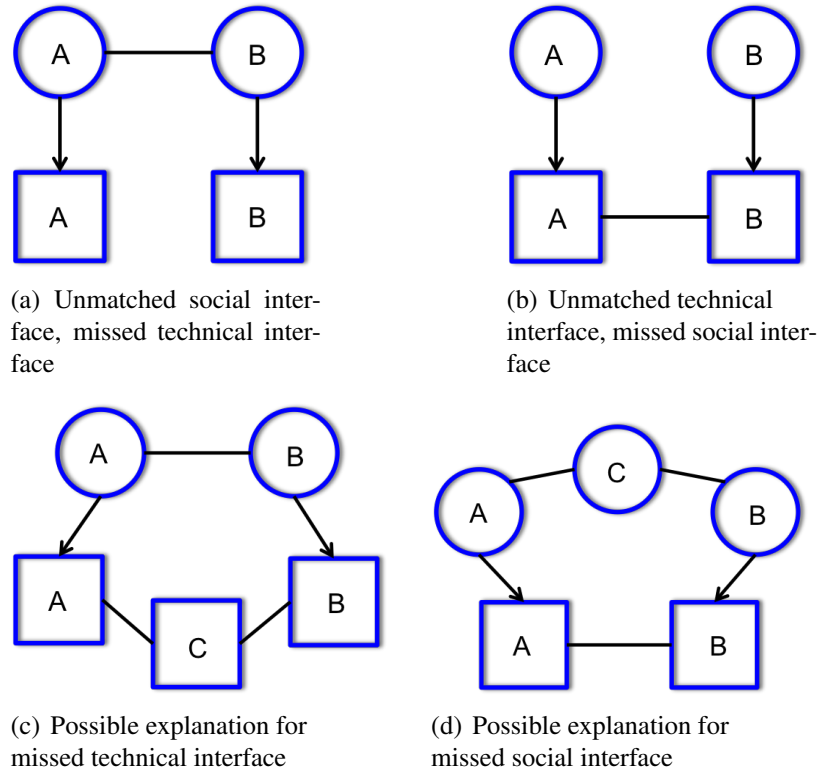


Figure 2: Illustrations of the concept of misalignments as in Sosa, Eppinger, and Rowles (2004), with examples of a) an unmatched social interface; b) an unmatched technical interface; c) an unmatched social interface due to indirect technical interactions; d) an unmatched technical interface due to indirect communications

the role of indirect integrator-type roles within the organization. These integrator roles may have a large impact on measures of socio-technical congruence (Collopy et al., 2017).

2.3 Summary

Coordination is described as a preconceived strategy or set of rules that addresses the uncertainty and interdependence between diverse tasks completed in support of a common goal. Coordination is often defined through top-down prescriptions of work procedures, development of schedules and plans, and provisions for divisions and teams to interact through their work environments and meetings. Our interest is the design of large-scale and complex engineered systems, which are by nature of their size and use of novel technologies,

very complex (Bloebaum and McGowan, 2012; de Weck, Roos, and Magee, 2011). Another prominent source of complexity is the social system engaged in system design and development work (Grogan and de Weck, 2016; de Weck, Roos, and Magee, 2011; Bloebaum and McGowan, 2012; Sheard et al., 2015; McGowan, 2014). Distributed knowledge (McGowan, 2014; Cumming, 2002), inconsistent preferences (Kannan, Mesmer, and Bloebaum, 2017; Bhatia, Mesmer, and Weger, 2018), and multiple incentives (Grogan et al., 2018; Vermillion and Malak, 2015; Meluso and Austin-Breneman, 2018) all contribute to complexity, and hence challenges for coordination.

The reviewed literature has shown the existence of coordination modes, work processes and plans instituted by the organization; a systems engineering process is an example. However, as expressed in the literature, the facilitation of coordination is through top-down prescription of rules and procedures for the organization as a whole. The inclusion of individual behaviors and skills adds a new dimension to the discussion of coordination that shifts from a rational closed-system view of organization to a more natural system, in which individuals are acknowledged as having diverse goals (Scott and Davis, 2006). This research seeks to contribute to this conversation by exploring the connection between individuals' coordination-facilitation behaviors and performance on a design task requiring coordination.

CHAPTER 3

Coordination in Industry Practice

3.1 Introduction

In this study, we identified strategies used by systems engineering and management personnel to support the coordination of distributed design work in LSCES design projects. We interviewed professionals with expertise in systems engineering, project management, and technical leadership at two large aerospace design organizations. Through qualitative thematic analysis, we identified two strategies used to facilitate coordination. The first is enabled by technical know-how and the use of authority, and the second is enabled by interpersonal skills, leadership traits, and empathy, which comprise a concept we term empathetic leadership. These strategies emerged from analysis as complementary sets of actions and behaviors used by individuals to facilitate system-level coordination of design work. These findings connect existing models of systems engineering and project management skill sets to actions and behaviors used in practice to accomplish systems design work.

The contribution in this chapter is the identification of *coordination strategies*, namely, actions and behaviors reported by individuals as useful to support coordination activity in LSCES design work. We find that skills identified in competency models enable these coordination strategies, which in turn support the organizational tasks of management and coordination of interdependent work. While literature suggests coordination is accomplished through prescriptive processes and plans, this study focuses on the contribution of individ-

uals' behaviors to facilitation of system-level coordination. This contributes a new picture of how coordination is accomplished in LSCES design work, and the role that systems engineering and management personnel play in facilitating that coordination.

3.2 Literature: Coordination Methods

Coordination is described in the literature of several disciplines as a programmatic or prescriptive process; its purpose to address the information and communication challenges of working on interdependent tasks simultaneously. In classic organization science literature, ensuring coordination is considered a large part of organizational design, including defining group structure, lines of hierarchy, and schedules and plans (Blau, 1974; March and Simon, 1958; Thompson, 1967; Van De Ven, Delbecq, and Koenig, 1976). In the multidisciplinary design optimization literature, coordination serves the same purpose for solving interdependent analytical problems, with the addition of ensuring a global objective is found (Papalambros and Wilde, 2017). Similar to defining aspects of organization structure, coordination is accomplished in MDO through the definition of linking variables between problems, and iteration through a programmatic routine that shares information between problems at regular intervals. In engineering design, systems engineering, and software engineering, coordination is discussed in terms of work process and project management approaches. Coordination is improved through the development and use of tools to support cooperative work (Cataldo et al., 2006), and the architecting of products and schedules to minimize required interactions (Baldwin and Clark, 2000; Steward, 1981). The underlying needs and causes of coordination challenges include large distances between distributed teams and different work styles (Herbsleb and Mockus, 2003), misunderstandings of shared product or software interfaces (Souza et al., 2004), and challenges of working across the organization where there are no formal paths to do so (Sosa, Eppinger, and Rowles, 2004). In sum, coordination is accomplished in the literature primarily using a

prescriptive process or tools. We expand this perspective to include the role of individuals' behaviors in facilitating coordination.

3.3 Methodology

To address the question of how coordination in LSCES design is carried out in practice, we developed a semi-structured interview protocol. Semi-structured interviews consist of *a priori* questions to guide the general discussion, but are open-ended to allow for fluid conversation and exploration of unexpected topics that may arise (Given, 2008). Our interview protocol was designed to elicit individuals' skills and behaviors used as part of LSCES design work. Our focus on the individual led us to narrow our questioning to look at communicative and cognitive skills and behaviors. Questions in the protocol progress from general to specific, setting a basic understanding of an individual's job and typical functions before discussing how they complete those functions. Of particular focus were job functions that pertain to partitioning and coordination processes during system design. Examples include determining work breakdowns, delegation, as well as the need to recombine and make sense of distributed information. Our questions centered on strategies for partitioning and coordination tasks that are both communicative (e.g., How would you characterize interactions between groups you work with regularly? What are typical communication methods used within the organization and what works best for you?) and cognitive (e.g., Was there any stage during the system design process in which you found it difficult to process and integrate the information available?). The full interview protocol is included in Appendix A.

We interviewed twenty professional engineers, managers, and systems engineers at two large aerospace design and manufacturing organizations as part of this study. Interviewees were selected by their organizations as representative of expertise in a range of positions within the organization. The gender diversity of interviewees is roughly consistent with

national trends: three of twenty interviewees (15%) were women, slightly higher than the 9-10% typical of mechanical and aerospace engineering workforce in the United States in 2015 (National Science Board, 2018). Our sample is biased, however, as 19 of 20 interviewees were in management positions. This does not reflect the typical depth of engineering organizations. Interviewees were also selected as representative of exemplar systems engineering practice within the organization, meaning that their approaches to coordination are not necessarily shared by all members of the organization. This study examines what are considered best practices.

All twenty interviews were analyzed together due to the generality of our questions as well as the similarity between the two organizations studied. Both companies are matrix-organized, design and manufacture large aerospace systems, and have a systems engineering process in place. These organizations partition their system design work both by aspect into *disciplines* and by object into *subsystems*. This dual partitioning is realized as a matrix organization structure, with management and leadership roles overseeing both the discipline analysis work (aspect partitions) and the technical subsystem design work (object partitions). We refer to the design and manufacturing work to develop and produce a single system as a *project*.

We grouped the actual titles of our interviewees into six general titles to preserve anonymity: Engineer, Discipline Lead, Chief Engineer, Project Manager, Systems Engineer, and Senior Management. *Engineer* reflects a non-management position that supports a single discipline or subsystem within a single project. *Discipline Lead* and *Project Manager* titles refer to management and leadership positions for a single project, overseeing a single aspect or object partition, respectively. *Chief Engineer* and *Systems Engineer* titles are also positions for a single project, overseeing both dimensions of partitioning for a subsystem or system. Those with Systems Engineer titles tended to be responsible for interfaces between system partitions throughout project work, while those with Chief Engineer titles tended to be responsible for the success of the project overall. The *Senior*

Manager title refers to management and leadership positions whose scope extends to multiple projects. Table 1 presents a summary of interviewee demographics including current titles and years of industry experience. This summary gives an idea of the types of people we interviewed, noting that the exact function expected of an individual with any given title may vary, and may differ between organizations.

Table 1: Summary of interviewee demographics based on title and years of industry experience

		Years of Experience			Total
		10-19	20-29	30+	
Position	Engineer			1	1
	Discipline Lead	2		2	4
	Project Manager	2	2	1	5
	Chief Engineer		1	1	2
	Systems Engineer	2	1	2	5
	Senior Manager		1	2	3
Total		6	5	9	

Our main goal of analysis was to identify and describe coordination methods used in practice which can then inform future hypotheses. Our sample size is small, and is therefore appropriate for an exploratory study but not necessarily the development of generalizable theory. Thematic analysis is well suited for this exploration. Thematic analysis focuses on the identification of emergent concepts or *themes* from the aggregate analysis of data (Patton, 2015). Our thematic analysis approach follows the methodology outlined by Braun and Clarke in (Braun and Clarke, 2006). Our implementation consists of five steps:

1. Prepare data for analysis by transcription of audio recordings.
2. Deductive coding based on initial broad categories of interest derived from original research question.
3. Inductive coding of the segments coded in step 2 to organize emergent ideas.
4. Identification of themes from review of inductive codes and their interrelationships.

5. Reflection and review of themes to ensure they are characteristic of entire interview corpus.

Our methodology as stated differs from that given by Braun and Clarke (2006) in that we separate our initial coding process into two steps, and our search, review, and naming of themes are combined within our theme identification step. The findings at each step of our analysis are presented in the following section.

3.4 Thematic Analysis

3.4.1 Data Preparation

To prepare interview data for analysis, we created verbatim transcripts of raw audio files. Our transcription focused foremost on recording words accurately. We then also added punctuation according to pauses and emphasis in interviewees' speech. Following transcription, the raw transcripts were anonymized by replacing or redacting any names mentioned throughout. Qualitative coding assistants prepared for analysis by reviewing the organizations' structures, the products they design and manufacture, their design processes, and notes from interviews regarding general impressions of the work environment. This served to provide the coders – who did not participate in interviews – with context for their review of interview data.

3.4.2 Deductive Coding

Our first stage of analysis was deductive coding, meaning to structure qualitative data with existing categories in mind (Patton, 2015, p. 64). We started with four initial categories based on the initial research question: personal and organizational description, language and information usage, process description, and technical design context. Two coders independently coded three interviews based on these categories. Afterward, the categories

were reviewed and collaboratively refined into fifteen *deductive codes* to focus on specific aspects of each category. The final set of deductive codes grouped into four new topics of personal, interpersonal, design process, and technical. The main change made in this refinement was to divide the original category of personal and organizational description into two topics. Both topics focused on personal preferences for doing work; the first topic centered on individual tasks and the second on interacting with others. Table 2 shows definitions of each topic and the deductive codes within each topic. The same two coders then used the final deductive codes to independently code (or re-code) all interviews by tagging each sentence or paragraph with the deductive code or codes they felt most appropriate. Coders used NVivo (NVivo for Mac Version 11), a Computer Assisted Qualitative Data Analysis Software (CAQDAS) tool.

The resulting deductively coded segments were analyzed using Cohen's Kappa as a measure of inter-rater reliability. Cohen's Kappa is increased when two independent coders code the same text segments the same way, but is decreased according to the calculated likelihood that coders code the same way by chance (Salkind, 2010). While our deductive codes were developed and defined collaboratively, the Kappa values were below 50% for all codes. The low values are consistent with observed difference in coding styles (e.g., coding entire paragraphs at once compared to portions of sentences), as well as different interpretations of the codes that became apparent through discussion. This meant the coders were each picking up on different nuances of the same code, despite agreeing on the overall code definition. These discrepancies indicated that the definitions for each deductive code were not sufficiently capturing the nuances of interviewees' responses.

3.4.3 Inductive Coding

Together, both coders identified a total of 2,388 deductive codes. The coded segments were aggregated and used as a basis for the next step of analysis. We used inductive analysis to identify the multiple concepts within each of our deductive codes. Whereas deductive

Table 2: Deductive codes and definitions used for second step of thematic analysis, divided by topic

Code	Definition
Personal	Attributes of person, innate or imposed by organization
Role, Function	Role or function within organization, formal or informal
Personality	Personality traits
Training, expertise	Experience or training that impacts how a person approaches their job
Personal style, work process	Preferences for approach to own work, including organization of information
Interpersonal	Attributes of people's interactions with others, innate or imposed by organization
Relationships	Relationships between people, formal or informal
Emotional and social awareness	Approach to interactions with others, including empathy
Communication methods and modes	Method, medium, or context of communication described
Communication purpose	Purpose and motives for communication, including whose purpose and motives
Communication style	Preferences regarding communication
Design Process	Attributes of design process
Information	What information is shared by communication and what form it takes
Meetings	How meetings are formed and their purpose
decision-making	decision-making processes, what information informs decisions and risk analysis
Iteration	Feedback during design, including formal and informal iteration
Technical	Descriptions of technical design work
Discipline identity	How discipline identity is characterized
Level of abstraction	At what level of detail design work is approached, addressed, or understood

coding is a process of fitting qualitative data to a structure, inductive coding is a process of fitting a structure to the data (Patton, 2015, p. 64). This is an iterative process, where potential codes are created, merged, and potentially discarded as the concepts in the data are organized.

Our process included two iterations: the first to identify potential or *initial codes* out of deductively coded segments, and the second to refine and organize those codes into final *inductive codes*. We collaboratively reviewed each deductive code one at a time, tagging the segments with initial codes and *subcodes*. The initial codes represent common threads that emerged from review of the deductively coded segments, and subcodes are specific examples of those common threads. An example is an initial code we called ‘Formally Structured Information’ which included subcodes of presentations, reports, technical documentation, database, schedule, budget, and deliverables.

Throughout analysis, codes and subcodes were added and combined as necessary. After reviewing all the deductively coded segments, the initial codes and subcodes were laid out on cards and overlaps and connections between them considered. Rearranging cards, we grouped initial codes and their respective subcodes into final inductive codes. Continuing the above example, the subcodes within the initial code of ‘Formally Structured Information’ were reorganized into final inductive codes of ‘Information Usage’ and ‘SE Process’. A map of the interrelations we observed between inductive codes is shown in Figure 3, with inductive codes grouped into four topics: Precursors, Methods, Purpose, and Context. The general flow of this map is that precursors inform the choice of methods used to accomplish a purpose, all within the context of a design process. While the deductive codes are organized by type of task, the inductive codes and topics are instead organized along a temporal axis separating actions and outcomes. The final inductive codes, their definitions, and some example subcodes are given in Table 3.

Table 3: Inductive codes, definitions, and selected subcodes, divided by topic

Code	Definition
Precursors	Innate attributes of person that impact how they approach their job functions
Personality and Leadership	Personality and behavioral traits, e.g., extroversion, curiosity, system awareness, leadership
Values, Ideology	Attitude towards work and valued traits in others, e.g., favors simplicity or efficiency, problem or solution orientation, value technical curiosity, lifecycle experience
Experience	Education, on-the-job training, or mentorship
Skills	Skills learned through experience, e.g., discipline expertise, technical analysis, work with people, workflow
Methods	Preferences and approaches to completing job functions
Communication preferences	Preferred way of interacting with people, e.g., build and maintain relationships, use empathy, keep people informed, use authority
Communication modes, setting	Choice of setting for communication, e.g., face-to-face, email, meetings, one-to-one or group
Job methods, approach, style	Actions used to accomplish job functions, e.g., establish norms and process, delegate, ask questions, facilitate brainstorming, engage with experts
Purpose	Job functions, goals of communication
Role/Function	Job functions, e.g., integration, coordination, facilitation, enablement, negotiation
Decision-making	Aspects of decision-making process, e.g., responsibility, oversight, set objectives, create trust and buy-in
Context	Systems Design Context
Information Usage	Information created by or used to make a decision, e.g., technical data, formal documents, plans, updates and feedback, lessons learned, homegrown tools
SE Process	Elements of systems engineering process, e.g., formal reviews, formal documentation, change management, requirements, verification and validation plans
Complexity	Sources of complexity, e.g., lots of parts, lots of disciplines, new approaches, change of scope, culture clash

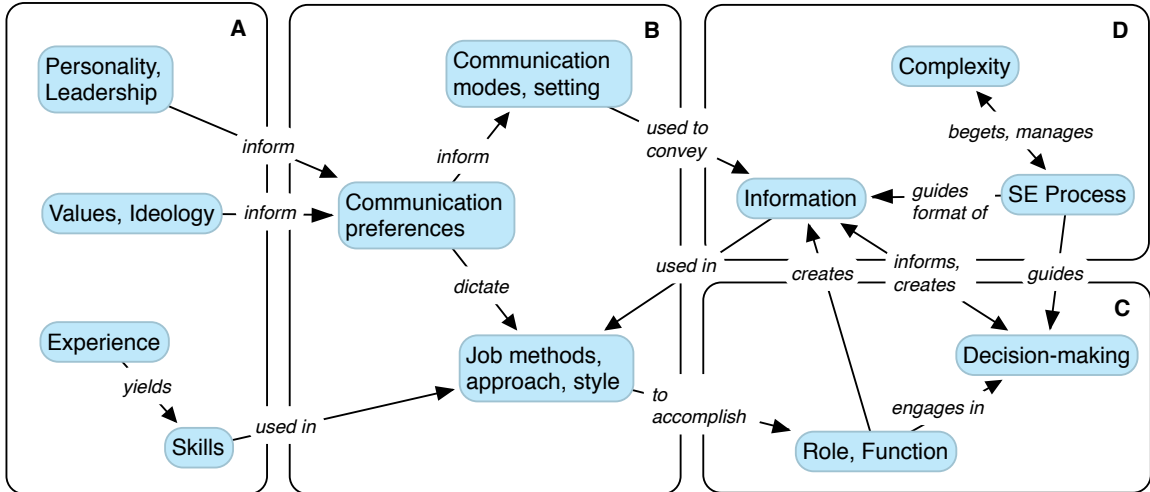


Figure 3: Map of interrelations between inductive codes. Codes are marked in blue, and organized into topics of (A) Precursors, (B) Methods, (C) Purpose, and (D) Context. Arrow labels are general characterizations of the relationship between inductive codes. Inductive codes are defined in Table 3.

3.4.4 Theme Identification

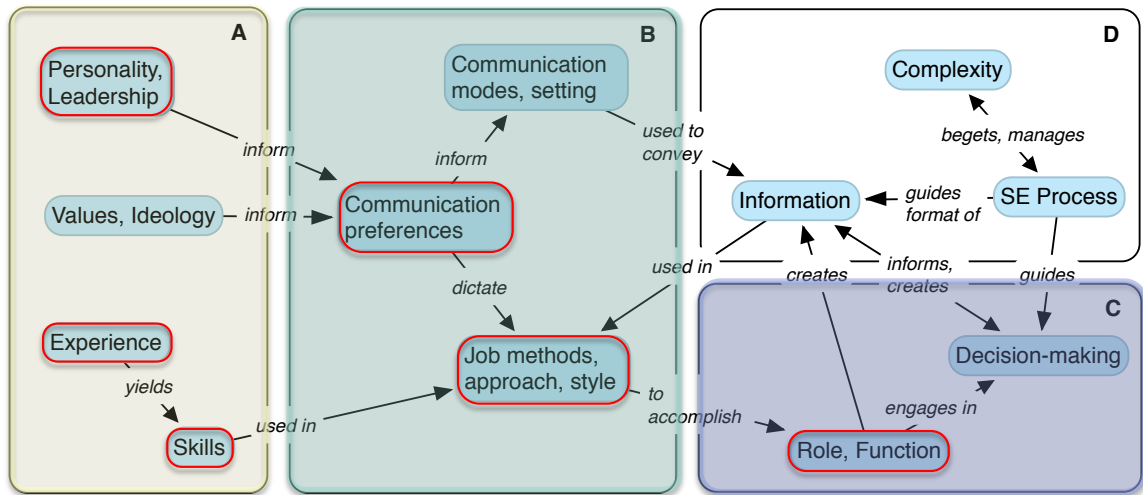
Inductive codes describe the content of the interviews, but do not necessarily give insight into the meaning behind the content. Our next step of analysis was to look in more detail at the relationships between inductive codes as shown in Figure 3. For example, which communication preferences dictate which job methods or approaches will be used? What job functions create what kind of information? To address the research question of how coordination is accomplished in practice, we focused on one specific question: what job methods, approaches, and styles are used to accomplish specific roles and functions? In other words, we looked at the connections between inductive codes ‘Communication preferences’, ‘Job methods, approach, and style’, and ‘Role, Function’ (which includes facilitation of coordination).

We looked for evidence of relationships between subcodes by reviewing text segments within each inductive code. For every subcode, we reviewed again the coded segments and identified other subcodes that were overlaid or connected through the interviewee’s language. Subcodes that tended to appear together coalesced into four concepts which

became focal points of our analysis: Authority, Empathetic Leadership, Management, and Facilitation of Coordination. Authority and Empathetic Leadership are names we gave to groups of subcodes under the ‘Communication preferences’ code. Management and Facilitation of Coordination are two subcodes under the ‘Role/Function’ code. Our analysis identified which subcodes under the ‘Job methods, approach, and style’ code connect these concepts; i.e., the link between communication preferences and job functions. Inductive codes and subcodes that appear in the following analysis are highlighted in Figure 4.

Our analysis focused on codes within the three topics of Precursors, Methods, and Purpose. While Context, the fourth topic, is not explicitly present in the analysis, it is pervasive throughout. An example is the use of a systems engineering process as a tool to organize what, how, and when work is done: this appears as ‘establish norms and process’ under the ‘Job methods, approach, and style’ code. Under the ‘SE Process’ code, not included in this analysis, are the specific kinds of documentation and reviews that comprise the process itself.

We discuss the central concepts of Authority, Management, Empathetic Leadership, and Facilitation of Coordination in turn. For each, we highlight the subcodes relevant to each, drawn from the inductive codes indicated in red in Figure 4. Our analysis focuses on identifying connections between subcodes according to our interviewees’ responses. An example we walk through below in Section 3.4.4.3 is how Personality and leadership traits (extroversion and integrity) impact Communication preferences (use of empathy and knowing people; or Empathetic Leadership), which in turn dictate actions such as asking questions and translation as strategies for Facilitation of Coordination and Management. Final themes emerge from the composition of these analyses, which show a dichotomy between authority-driven and empathetic leadership-driven strategies for facilitation of coordination.



Selected subcodes:

Personality, Leadership

- Extroversion; proactivity
- Reliable; has integrity

Experience

- Job responsibility, job training

Skills

- Technical understanding

Communication Preferences

- Use authority
- Build and maintain relationships
- Have empathy
- Humanize relationships
- Know people as people
- Listen

Job Methods, Approach, Style

- Establish norms, process
- Delegation
- Action-tracking
- Planning
- Set deliverables
- Call standing meetings
- Co-locate groups
- Use standard process
- Translation; sensemaking; create common language
- Ask questions
- Work across personalities; encourage empathy

Role, Function

- Coordination, Facilitation
- Technical Management
- Resource Management
- Technical Leadership

Figure 4: Selected subcodes included in theme analysis discussion. Many subcodes are common to the examples given in Table 3. Colored boxes around each topic of codes are consistent with those in subcode maps in the following sections: Precursors are yellow (left), Methods are green (middle), and Purpose codes are purple (bottom right).

3.4.4.1 Authority

Authority is defined as the ability to give orders and make decisions (Stevenson and Lindberg, 2011). Authority, its enablers, and the tasks it is used for as identified from our interviews are shown in Figure 5. Each block in this map represents a subcode. Arrows between subcodes indicate that the attribute or action described by one subcode enables another. Squared boxes in this map indicate specific instantiations of each subcode: for example, instantiations of Authority identified through analysis include *technical authority* based on recognized technical expertise and *positional authority*, based on delegated responsibility.

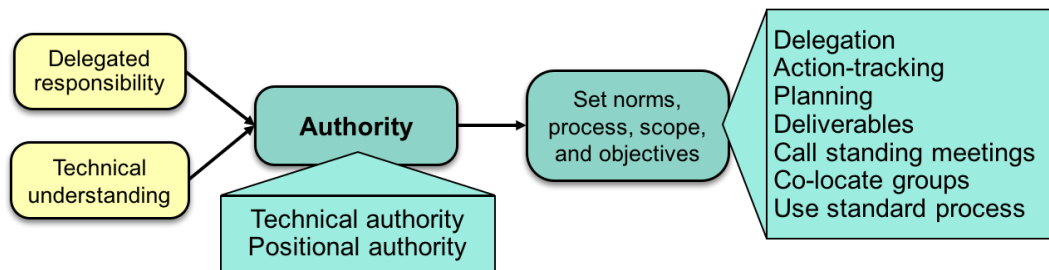


Figure 5: Map of subcodes related to Authority subcodes, with specific instantiations of ‘Authority’ and ‘Set norms, process, scope, and objectives’ subcodes shown in squared boxes. Subcode colors refer to the topic its parent inductive code belongs to (see Figure 4).

Technical Expertise

Authority comes from assigned responsibility for decision-making, which in turn is supported by experience and technical understanding. As pointed out by one of our interviewees, technical understanding is essential to effective decision-making: “I don’t understand how you can be accountable for the decisions you’re making if you don’t understand the thing you’re building, and how it works, and what the trades are.” The benefit of technical knowledge was mentioned by all interviewees, but its importance was emphasized for situations with high technical or programmatic uncertainty. Decision-making under uncertainty requires a risk assessment, based on experience: “If things feel like [they are] low risk, I don’t spend a lot of time looking at them. If they’re new, or novel, or something we

haven't done in a while, then I'll pay special attention to that." In addition to experience, one interviewee pointed out that "it helps to have some ... level of domain knowledge" to estimate technical and programmatic risk.

Positional Authority

Technical authority to make decisions can be delegated to any member of the organization where experience and technical understanding assists that decision-making. However we mostly spoke with those in management and leadership roles, whose positional authority enables directing the work and decision-making of others. Direction includes setting the scope and objectives of work (what work is done) as well as norms and processes for doing and reporting work (how work is done). Specific instantiations of norms, process, objectives, and scope are shown in Figure 5, and include delegation, action tracking, planning, setting deliverables, calling standing meetings, co-locating groups, and using a standard design process.

Set Norms, Process, Scope, and Objectives

Methods such as action tracking and deliverables are used to structure the vast amount of information that our interviewees have coming to them daily, regardless of their role. One interviewee explained how structured deliverables help them stay on top of their group's work: "I need to get this information from everybody so I can understand it so I can make sure it's all coming together, and I need them to provide it to me in some kind of consistent format, otherwise it takes me too long to digest all of it." For some, this consistent format is a bulleted list, others, documents posted to SharePoint, and still others use a custom action-tracking document to centralize information about who is working on what tasks and each task's status.

Planning and establishing a formal design process supports management tasks by enabling tracking how closely actual work adheres to those plans. Plans also support coordination by scheduling concurrent design work. As one interviewee explains, "I think

just having that process in place that everybody's bought into and everybody kind of understands and follows, it helps ensure that everybody's communicating and they're on the same page." At the organizations we visited, these design processes – including implementations of systems engineering, lean manufacturing, and design review schedules – are an integral part of organizational culture and shape how work is done. While a design process applies to the project as a whole, an individual's authority allows them to enforce how the process is followed. An example is using standing meetings and co-location to support design work: "There are situations where ... somebody [has] the experience [needed to resolve an issue] , but it may not be communicated. So what we do to foster that communication is essentially have open-ended discussions, weekly meetings [and] we also try to co-locate teams."

Authority enables the setting of formal structures of work and standards for accomplishing that work. The management and coordination-facilitation tasks these authority-based actions support are discussed further in Section 3.4.4.2 and Section 3.4.4.4 below.

3.4.4.2 Management

Management in the design of LSCES can be both technical, ensuring the right work is done to meet technical requirements, and resource-based, ensuring constraints of time and budget are met. A map of the subcodes related to Management functions is shown in Figure 6. Again, the various instantiations of subcodes are shown in squared boxes. Management is supported by the use of authority to set norms, process, objectives, and scope of work, instantiations of which are discussed in Section 3.4.4.1. Management tasks are also supported by one's knowledge of others.

Knowing People

An approach to delegation mentioned by interviewees relies on knowing people well enough that they can delegate tasks knowing how that person will respond to situations. As one interviewee explained: "I'll split my work across ten people. ... I know where my work

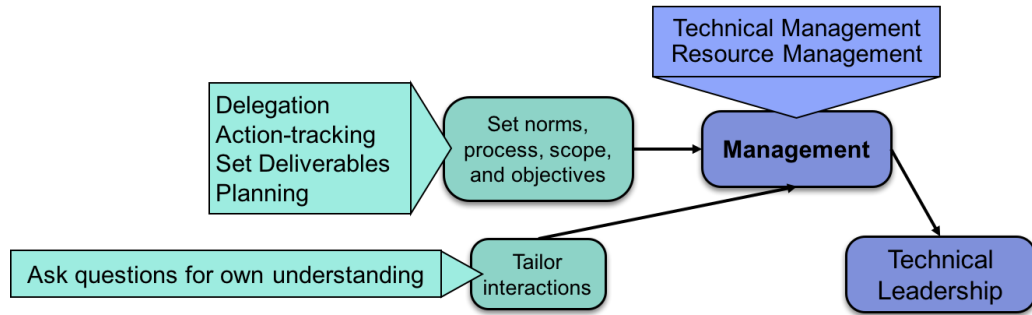


Figure 6: Map of subcodes related to Management subcodes, with specific instantiations of ‘Set norms, process, scope, and objectives’, ‘Tailor interactions’, and ‘Management’ subcodes shown in squared boxes. Subcode colors refer to the topic its parent inductive code belongs to (see Figure 4.)

is delegated to, and I basically self-replicate my principles to them.” Another frequently mentioned approach to ensuring that the correct work is done is asking questions. One interviewee described one of their roles as an ‘interrogator’, “[asking] you as many hard questions as I can to make sure that you are [doing the right work].” Asking questions of multiple people is also a tactic used to reduce uncertainty: “It’s not uncommon for me to ask the same question three times either in different ways, or [of] different people, just to check consistency.” The actions of asking questions and relying on knowledge and trust in people to ensure work is being done stand in contrast to using authority to set up-front rules and procedures for doing work. Both strategies are used widely by our interviewees to support technical management tasks.

Technical Leadership

We also found that management tasks support *technical leadership*. While technical management as we define it here is about ensuring the technically correct work is done to meet requirements, technical leadership is the complementary guiding vision that defines what the resulting system should be. This leadership concept is also about maintaining focus on the end designed system: “[E]verybody has to as much as possible consistently implement leadership’s intent. Because everybody can’t be going different directions.”

When designing large hardware systems, the final physical artifact is out of sight for much of the design process, and perhaps the entirety for designers in remote locations. Maintaining this forward “solutions-oriented” focus was cited as an important part of their job for several of our interviewees: “You have to be able to share a vision. ... Being optimistic and solution oriented is so critical. Without that, then the team basically loses confidence.”

3.4.4.3 Empathetic Leadership

We use the term *empathetic leadership* to describe the tailored, individualized approach to working with and leading or managing people described by several interviewees. A map of the subcodes related to empathetic leadership is shown in Figure 7. These subcodes include personality traits, social skills, and the ability to work with people effectively.

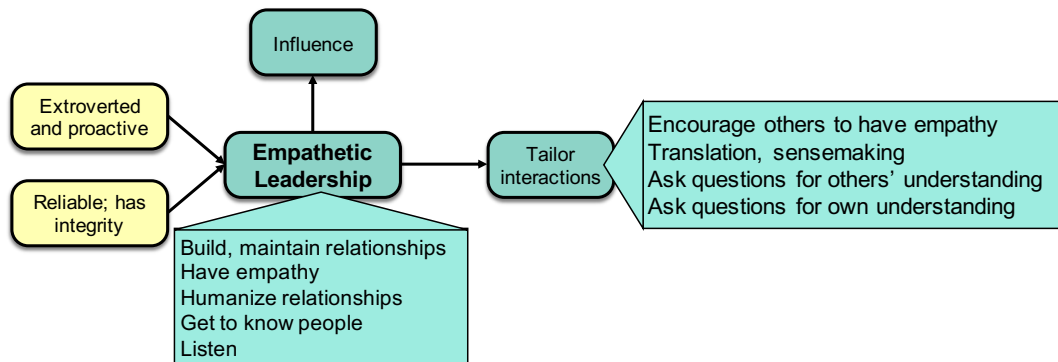


Figure 7: Map of subcodes related to Empathetic Leadership subcode, with specific instantiations of ‘Empathetic Leadership, Social Capital’ and ‘Tailor interactions’ subcodes shown in squared boxes. Subcode colors refer to the topic its parent inductive code belongs to (see Figure 4).

Building Relationships

One instantiation of empathetic leadership is an emphasis on getting to know people. This way of working with people was described by one interviewee: “I spend time knowing people, talking to people, I know what they do outside of work, I know what their interests are, I get to know them, like if they have families and what they’re doing, and taking that time – there’s a lot of engineers that will tell you that you don’t need to do that, that’s not

part of your job. — and I strongly disagree. ... I don't see how you can lead a group of people without seeing that whole human side.” Acknowledging that people have different needs and different perspectives helps these interviewees to get the most out of their teams and build trust within the team.

Influence

Building individual relationships with people is also helpful for working outside one's team. As one interviewee expressed, “When we say please and thank you and treat each other with respect in that way and we do relationship building, [when] the next project or the next thing comes along, people have a more positive understanding.” Several interviewees mentioned that many technical challenges benefit from knowing the people who are involved: “[U]sually it's not that we don't know how to do something technically, it's usually a conflict that's more at a personal level or it's an opinion or something like that. ... [Y]ou need to steer toward understanding how individuals operate or their perspectives.” Maintaining connections and building trust in those relationships also helps to support technical leadership through the development and use of influence. According to our interviewees, influence can support or supplant authority to guide decision-making: “[y]our ability to influence the final design is much more dependent upon your personal influence and your personal integrity than it does on your position of authority.”

Empathetic Leadership

Our synthesis of all interviews suggested an underlying concept that is supported by these actions of building and maintaining connections with others, using empathy to get to know people, building trust in relationships with others, and personality traits and values of extroversion, reliability, and integrity – all shown on the left side of Figure 7. We call this concept empathetic leadership as explained above. We also note this description evokes the concept of social capital. Social capital is the resource afforded to an individual based on their connections and their ability to navigate those connections (Adler and Kwon, 2002;

Burt, 1992). This connection to social capital is discussed further in Section 3.5.

On the right side of Figure 7 are several actions and behaviors that emerged as supported by empathetic leadership. These actions, such as encouraging others to have empathy, asking questions so that others gain understanding of situations, and translating or engaging in sensemaking to help others navigate discipline or cultural boundaries, are all supported by the ability and willingness to humanize and tailor reactions with others. These proactive behaviors are valuable for supporting management tasks and delegation as discussed previously, but also support the facilitation of coordination by helping others work together productively. Facilitation of coordination and the strategies used to accomplish coordination are discussed more in Section 3.4.4.4.

3.4.4.4 Facilitation of Coordination

According to our interviewees, facilitation of coordination is about “[getting] the right information to the right people at the right time so that they can be enabled to [do] what needs to be done.” We heard from our interviewees several actions they use to facilitate coordination in their daily work. The majority of interviewees mentioned one of their top responsibilities in their position was communication, and in many cases followed closely by ensuring others are communicating. In the design of large and complex systems, communication across disciplines is key due to the inherent interdependencies between parts that are designed by different people and groups, often in different locations and sometimes in different organizations. We found that both setting norms, process, scope, and objectives for work (through use of authority) and tailored interactions (through use of empathetic leadership) are used to facilitate coordination. These authority-based and empathetic leadership-based strategies, actions that comprise each strategy, and example instantiations of each are shown in Figure 8.

We found that the authority-based actions of calling standing meetings, co-locating teams, and the use of a standardized process are complementary to the actions based on

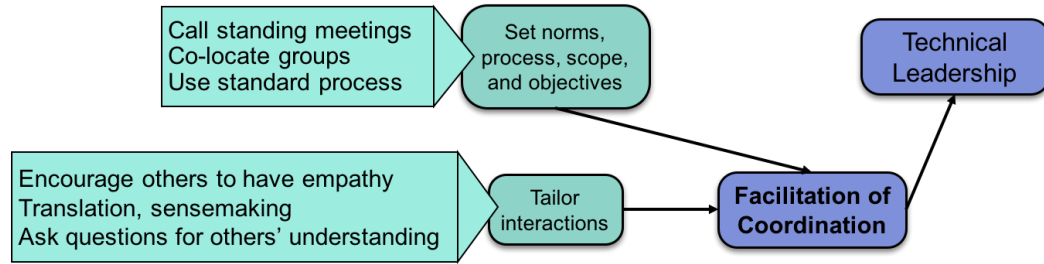


Figure 8: Map of subcodes related to Facilitation of Coordination subcode, with specific instantiations of ‘Set norms, process, scope, and objectives’ and ‘Tailor interactions’ subcodes shown in squared boxes. Subcode colors refer to the topic its parent inductive code belongs to (see Figure 4).

empathetic leadership: encouraging others to have empathy, translation and sensemaking, and asking questions for others’ understanding. We discuss each pair in turn.

Meetings

One oft-used mechanism for ensuring discourse across groups and disciplines is meetings. For some, meetings are where their work gets done: “[W]e have so many different design [groups], we’ve got to get them to talk to each other. ... [T]here are some days I have a half hour that I’m not in the meetings throughout the day. But really that’s how work gets done.” Meetings can take several forms, but two main functions were apparent from our interviews. One typical purpose of a meeting is to provide awareness of team member’s current status and issues. A second common meeting function is bringing together multiple teams whose work is impacted by a design change to make and agree to that change. In both cases the meeting is a forum for disseminating information to many parties at once, ensuring all parties receive the same information.

Encouraging Empathy and Asking Questions

The success of meetings and their utility depends on what the individuals bring to the table: what makes these meetings go well centers around having the “right people” there. This means people with the right experience and information to contribute to decisionmaking, as well as a clear sense of purpose of what they want to get out of being at a meeting.

The effective transfer of information at a meeting, central to coordination, then depends on both having regular meetings and the awareness to make use of the meeting time. Coordination is in part facilitated through asking questions of people to ensure they understand why their meeting participation is important and encouraging them to have some empathy and awareness of how their work impacts others; “helping them try and connect the dots”. Empathetic leadership, specifically having empathy and knowing people’s individual strengths and weaknesses, supports these two actions of asking questions and encouraging the use of empathy.

Co-Location

Another approach used to facilitate coordination is to co-locate people who have related or interdependent work. One interviewee expressed their goal of co-locating a multidisciplinary team is to help them “self-integrate, because [then they’re] just the people that naturally communicate with each other on a day-to-day basis.” Another interviewee expressed a similar sentiment, that co-location for their team ensures that problems can be solved by walking to a nearby desk: “[Y]ou can go to your neighbor or you would go to a guy that you see almost every day who is ... the expert in that discipline or methodology. Or [say] hey, I think that I’ve seen on [someone’s] screen [something] that I’m trying to solve.”

Proactivity and Translation

In highly interdependent design work, people are likely to have multiple interdependencies across both aspect (discipline) and object (subsystem) partitions. Co-location partitions a group of people based on one kind of dependency. In turn, that group likely has additional dependencies with other individuals who are farther away. Proactively seeking out those more remote interdependencies was identified by our interviewees as important to effective facilitation of coordination: “You can’t sit in your office, you’ve got to go [and] interact with all of these design disciplines. You’ve got to communicate.” Being able to communicate across discipline and cultural divides, or “translate”, is essential. As one interviewee

explained, the lack of consistent terminology and language across disciplines can cause miscommunication and ultimately problems that take extra time to resolve. Translating discipline-specific language to something more universally understood is key to ensuring groups work well together. “I end up being the translator. I end up being the kid that says hey wait a minute, I think this is what you just said. Is that right? ... I specifically really try and push enough of a plain language that all of the groups can understand it. ... It’s a hard thing, but that actually is really, really critical.” Again, empathetic leadership includes building relationships across the organization and knowing individuals and their styles. Proactively seeking out others and translation to develop a common language are central to the coordination strategy based on empathetic leadership.

Standard Process

Finally, the complexity introduced by the multitude of parts and people involved in large-scale and complex engineered systems design is often managed through a formal systems engineering or design process. Simply having a process and a standard way of documenting work is not always considered sufficient or the most efficient way to support coordination in all cases, though: “[I]t may not be as effective sometimes to just look through a bunch of documentation, and sometimes knowing the right person to ask the informal route [is more effective].” As mentioned by several interviewees, the “real work” happens prior to documentation being written and approved, and that prior work is where coordination is needed. “[I]t takes a lot of communication, because there has to be a level of day to day communication that gets the message across of what’s about to occur, not stacks and stacks of documents and review, but it’s about having that right level of cognizance in that discipline and say, hey we have a design issue that we’re having, a challenge meeting our requirements, it looks like we’re making a change over here, it’s probably going to affect discipline X and Y, we need to bring them in.” An individual’s authority gives them the ability to set a design process in place and mandate certain kinds of documentation. However, the actions and behaviors of proactive interaction with peers, developing a

shared understanding of the technical system, and knowing the right people to help identify and resolve design problems are part of the empathetic leadership concept introduced in Section 3.4.4.3.

3.4.4.5 Themes

As we explored these interrelationships we found two clear themes emerged: *authority-based* and *empathetic leadership-based* strategies for job functions, particularly the facilitation of coordination work. This dichotomy is illustrated on the composite subcode map shown in Figure 9. A full subcode map including all previously discussed instantiations is shown in Figure 10.

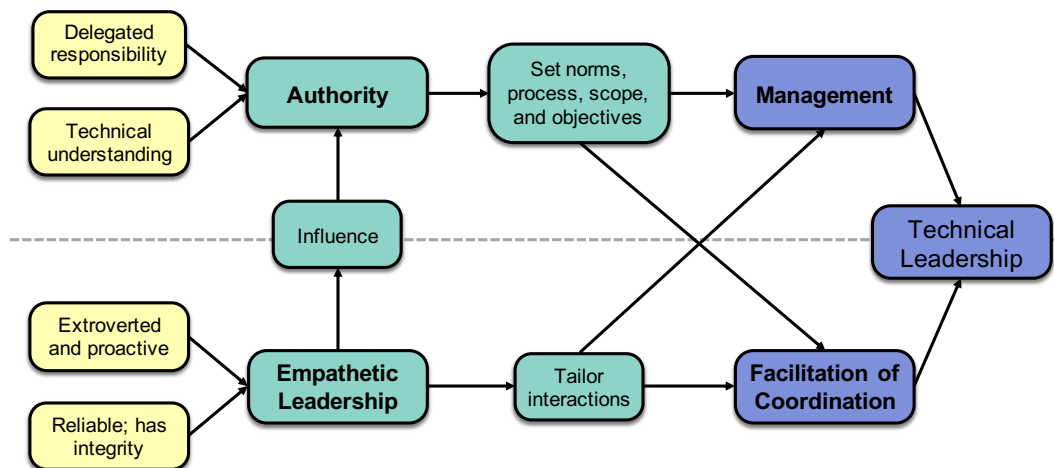


Figure 9: Illustration of themes overlaid on full subcode map, illustrating empathetic leadership and social capital as complementary strategies used for facilitation of coordination and management. Subcode colors refer to the topic its parent inductive code belongs to (see Figure 4).

The authority based strategy is comprised of actions to set norms, processes, scope, and objectives for technical work, and behavior that relies on the use of authority to get things done. As mentioned in previous sections, these actions are used in support of both management (ensuring the right work is done within organizational constraints) and facilitation of coordination (ensuring that distributed work is being done towards the same goals).

Authority supports actions and behaviors including delegation, planning, co-location of

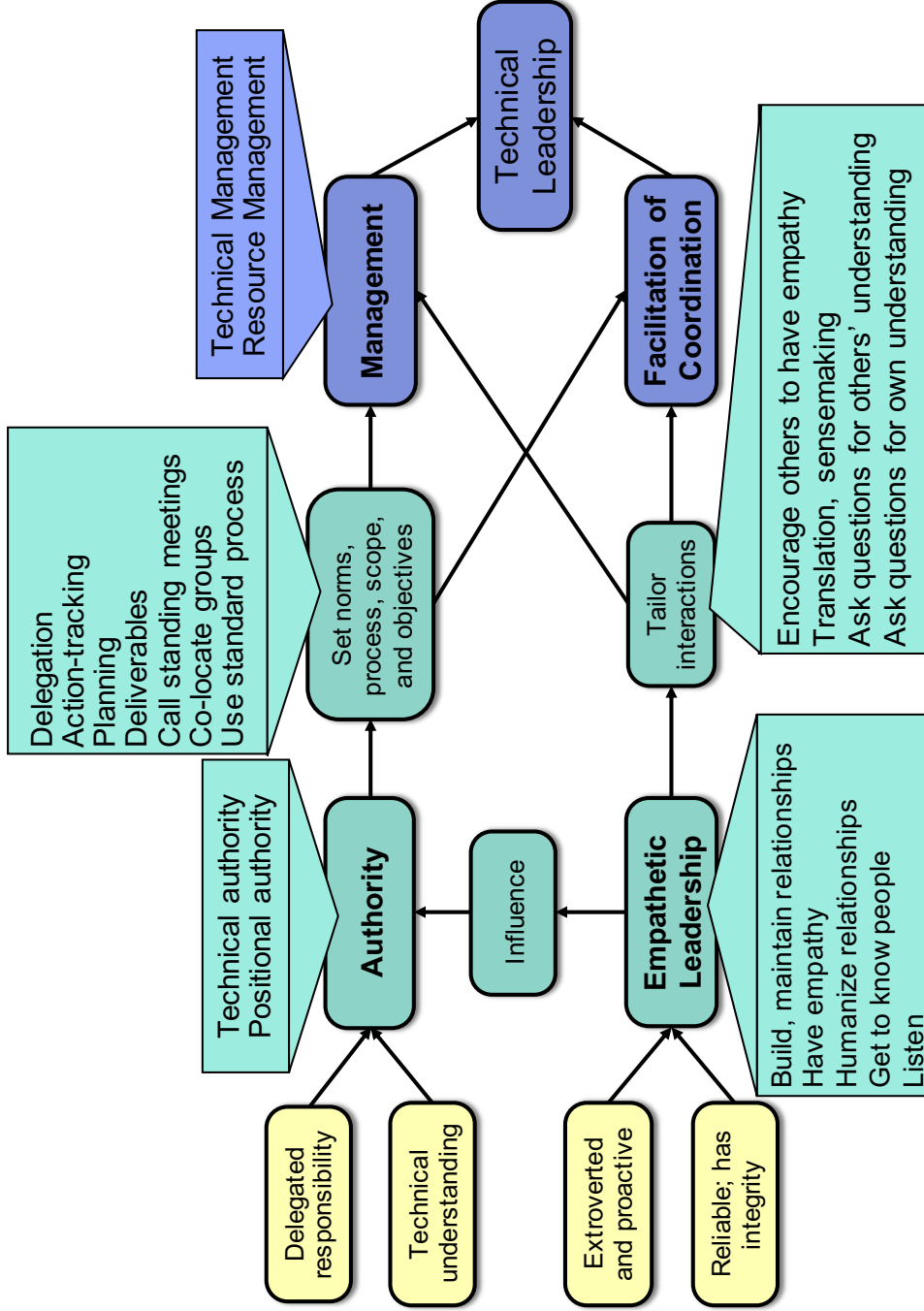


Figure 10: Full map of all highlighted subcodes, including all specific instantiations mentioned in previous sections. Subcode colors refer to the topic its parent inductive code belongs to (see Figure 4).

groups, and setting and enforcing a standard process. They are enacted through the exercise of authority, and shape the system of how work is done within the organization. These authority-based actions bear strong similarity to how coordination is presently described in multiple disciplines: a process prescribed at the outset of work to ensure that information can be communicated across the organization to those working on complementary parts of a larger design problem.

In turn, empathetic leadership enables a strategy that includes actions and behaviors to tailor interactions with others. These actions include building and maintaining a social network of information resources, tailoring interactions with others, encouraging others to have empathy in interactions, and asking questions to support a common understanding of what work is being done and how it fits together. These actions and behaviors help to ensure information is shared and meaningfully translated between disciplines and assist sensemaking (Weick, Sutcliffe, and Obstfeld, 2005) in order to facilitate the coordination of diverse work. Again, as mentioned in previous sections, these actions are used by our interviewees in support of both management and facilitation of coordination tasks.

The two themes that emerged from our analysis comprise what appear as complementary strategies for the facilitation of coordination work. These strategies differ in terms of their enabling preferences (use of authority or use of empathetic leadership), or the resultant actions and behaviors (centered on plans and procedures, or centered on tailored one-on-one interactions). While multiple terms could be used to describe each strategy, we choose to call these strategies *Passive* and *Active*. The first strategy for facilitation of coordination we call *Passive* due to its top-down implementation through the use of authority. The second strategy we call *Active*, distinguished by the proactive behavior interviewees mentioned repeatedly as helpful. Knowing when to “jump in” and get involved and taking the time to tailor interactions with each person they talk to both require going above and beyond the minimum required for a given task. The denotation of the authority-based actions and behaviors as *Passive* is not meant to belittle the work that goes into these tasks,

but rather serve as contrast to the extra proactivity requisite for the Active strategy.

3.4.5 Reflection

The previous conclusions are drawn from the analysis of inductive codes in turn drawn from a subset of the original data. As a final step in our thematic analysis, we reviewed the entire interview corpus to draw out a description of each theme reflective of the full dataset.

Our thematic analysis suggests the existence of two archetypical strategies, or sets of behaviors, based on the use of authority or empathetic leadership to accomplish management and the facilitation of coordination tasks in LSCES design work. Referring to Figures 9 and 10, the Passive archetype describes individuals who use their authority to dictate what tasks are to be done (e.g., delegation), standards for task reporting (e.g., documentation standards), plans for work completion (e.g., through project management), the process by which work is completed (e.g., a design process), and the environment in which work is done (e.g., location of teams).

The Active archetype describes individuals who are extroverted, proactive, and reliable, which allows them to build and maintain a robust social network and use it to access information. By using empathy in interactions with others, these individuals tailor their interactions to support coordination. This includes encouraging the use of empathy and systems awareness (e.g., through mentorship and coaching), translating information to bridge discipline and cultural boundaries, creating a positive and collaborative work environment, and asking questions to ensure a common understanding of work and its purpose.

Archetypes are stereotypes of behavior, and thus purely active and purely passive behavioral archetypes are not likely to match the behavior of any one individual all the time. Most of our interviewees are best described by a combination of both archetypes, using both Active and Passive strategies, or a combination, as they believe the situation merits.

3.5 Discussion

Our qualitative analysis identified two different sets of actions and behaviors used to facilitating coordination, which we call *Passive* and *Active*. We focus on two main findings from our analysis: the concept of empathetic leadership and related concepts in literature, as well as the juxtaposition of *Passive* and *Active* coordination strategies.

3.5.1 Active Facilitation of Coordination

The literature on coordination emphasizes the use of standards for documentation and work processes, clearly partitioned work assignments and lines of communication, schedules, and meetings for the effective coordination of complex tasks (March and Simon, 1958; Thompson, 1967; Van De Ven, Delbecq, and Koenig, 1976; Tushman, 1979; Souza et al., 2004; Sosa, Eppinger, and Rowles, 2003; Herbsleb and Mockus, 2003; Cataldo, Herbsleb, and Carley, 2008; Malone and Crowston, 1994). This emphasis is fairly consistent across disciplines, and aligns clearly with the *Passive* strategy for facilitation of coordination we found in our analysis. Our analysis suggests that other processes are also used to facilitate coordination throughout a complex design project, supported by concepts of empathetic leadership and social capital. These concepts are not new in the literature, but have not been closely connected to coordination.

Informal processes, referring to personal values and individuals' interactions within a formal organization, have long been recognized as important in organizations (Blau, 1974; Barnard, 1964). It is natural that the findings from our questioning, focused on individuals' cognitive and communicative actions and behaviors, might correspond to informal processes studied in organizations. The concepts of 'soft' systems methodology emphasizing systems thinking (Checkland, 2000), social capital emphasizing development and leverage of social networks (Adler and Kwon, 2002; Esser, 2008; Van Deth, 2008; Agneessens and Wittek, 2012), brokerage emphasizing bridging communities or 'structural holes' (Burt,

1992; Obstfeld, Borgatti, and Davis, 2014; Burt and Merluzzi, 2014), and positive leadership emphasizing empathy and optimism (Baker, Cross, and Wooten, 2003; Terrasi, 2015) are well known informal organizational processes but are generally not connected to coordination of technical tasks, typical of LSCES design. One of few exceptions is Larsson (Larsson, 2007) who illustrates how social capital is important for effective collaboration. These concepts resonate with the Active theme identified in our analysis, including empathetic leadership, how it is developed, and how it is used to facilitate coordination.

There is also some overlap between the Active theme we identified and the skills identified in systems engineering and project management competency models. Extroversion or proactive communication, leadership abilities, and the ability to collaborate and communicate across diverse groups are included in several competency and behavioral models of systems engineering practice (Hutchison, Henry, and Pyster, 2016; Williams and Derro, 2008). These skills and behaviors are reflected in the Precursors topic of our inductive code; those that we found to support empathetic leadership. Our analysis connects these skills and empathetic leadership to the facilitation of coordination through proactive and tailored communication. We thus show a link between skills and behaviors identified of successful systems engineers and tasks they may be engaged in, particularly the facilitation of coordination.

The overlap between concepts of systems thinking, social capital, brokerage, and positive leadership in sociology and organizational literature, skills like empathy, leadership, and communication in existing competency models, and the actions that comprise our identified Active coordination strategy suggests three things. The first is that *empathetic leadership* may serve as an encompassing term to bring these various ideas together to describe the informal processes happening in large organizations in support of coordination. Second, as coordination is one major task for which these skills are used in practice, we may consider in turn coordination to be a central component of systems engineering practice, and one that not only those with the title ‘systems engineer’ are engaged in. Finally, the

connection between the concept of empathetic leadership and facilitation of coordination is important to recognize that there are existing theories and concepts that help to describe coordination in technical organizations engaged in large-scale and complex design work. These concepts may contribute to a growing set of theories about the nature of systems engineering and what goes into effective systems engineering practice.

3.5.2 Relationship between Active and Passive Themes

Through our analysis, we identify both Active and Passive strategies used to facilitate the coordination of LSCES design work. The discussion above suggests that the ‘active’ actions and behaviors we identified are similar to informal organizational processes, whereas the ‘passive’ actions and behaviors we identified are similar to formal organizational processes. This dichotomy suggests that the two sets of coordination methods are related, in the sense that ‘passive’ or ‘formal’ approaches establish a structure or culture of work, and ‘active’ or ‘informal’ approaches are used to work within and across that structure. Parallels may also be drawn to a handful of other dualities: strategies to set long-term goals and tactics to accomplish near-term objectives, design of an artifact and control to determine its behavior, and partitioning to divide a problem and coordination to solve it. The nature of each of these pairs is that they are intricately linked parts of the same problem. This has been illustrated for partitioning and coordination decisions (Allison, 2008), design and control decisions (Reyer and Papalambros, 2002), formal and informal organization (McEvily, Soda, and Tortoriello, 2014), and strategy and tactics selection (Mackay and Zundel, 2017), to name a few. Generally, making decisions about one without consideration of the other leads to a poorly performing or, at worst, infeasible system design. What this suggests for our findings is that the Passive and Active coordination strategies should be considered together when making decisions about how to best facilitate the coordination of distributed work.

Some ideas for how to consider Active and Passive coordination strategies together

come from studies focused on unexpected challenges in collaborative work. Modularization relies on clearly defined tasks in order to partition work cleanly and reduce the need for coordination. However, this focus on clean partitions can result in challenges putting the results of those tasks back together. Even if tasks are defined so that the results should be easy to integrate, the awareness of how the parts combine to operate as a whole can be lost (Souza et al., 2004; Grubb and Begel, 2012). Similarly, the use of a standard process or set of rules is better able to support coordination if the individuals whose work is to be coordinated are bought into the efficacy and purpose of the process (Espinosa, Armour, and Boh, 2010). Recommended resolutions to these apparent challenges with the Passive strategy for facilitating coordination focus on increasing communication and awareness of parallel work (Herbsleb and Mockus, 2003; Espinosa, Armour, and Boh, 2010).

There are also evident challenges in executing the Active strategy. In highly interdependent system development, the logic that all pairs of individuals with interdependent tasks should be communicating regularly becomes infeasible (Brooks, 1995). In an organization of several thousand or more individuals, it is difficult to develop a robust social network that spans a significant portion of those individuals. Relying on individuals to coordinate their own work faces challenges from the difficulty of communicating across divisional boundaries (Sosa, Eppinger, and Rowles, 2003; Dossick and Neff, 2010) and identifying and accessing relevant information out of a large possible set (Salzberg and Watkins, 2016). To address these challenges, defining formal coordinator roles are often recommended (Parraguez, Eppinger, and Maier, 2016; Strode et al., 2012; Poleacovschi and Javernick-Will, 2016). Formal roles come with authority, thus enabling the Passive coordination-facilitation strategy.

Active and Passive strategies for facilitating coordination both have benefits and drawbacks. The review of literature presented here suggests drawbacks from one set of actions may be mitigated by use of the other set of actions. Following from this, we hypothesize that a balance between Active and Passive actions and behaviors for the facilitation of co-

ordination is needed in large-scale and complex engineered systems design. These projects require many individuals with high technical skill in order to design and develop a system. The large organization benefits from Passive coordination strategies to unify standards of doing work, while also benefiting from empathetic leaders who use Active coordination strategies to leverage extensive networks across the organization. An individual may use either strategy, or a combination. In our study, technical managers, project managers, and systems engineers all made use of some combination of these strategies in order to facilitate coordination.

3.6 Summary

We presented in this chapter the results of an exploratory qualitative study, identifying two strategies – sets of actions and behaviors – used in practice to facilitate the coordination of distributed design work. Through thematic analysis, we found evidence of both Passive and Active strategies for the facilitation of coordination. Passive actions and behaviors are supported by the use of authority to enforce what work is done and how it is done through formal delegation, setting standardized documentation and deliverables, a common design process, developing plans and project management, and co-locating groups with interdependent tasks. These actions provide common norms of how work is to be done with the expectation that this environment enhances coordination. Active actions and behaviors are supported by a concept we call empathetic leadership, a proactive approach to developing and maintaining a diverse social network across the organization. The ability to develop these connections is used to tailor communications to successfully share technical information across discipline and cultural boundaries, ask deep questions to help oneself and others develop a common understanding of work to be done and how it fits together, and encourage others to use empathy in their own interactions to facilitate effective collaboration. Identification of these two strategies presents a new way to look at coordination in

LSCES design. Our findings suggest that implementing and enforcing a systems engineering process (a Passive action) is complemented by the actions and behaviors of systems engineers and managers above and beyond that process (Active actions), both in support of successful coordination.

The Passive strategy we identified is consistent with existing literature on coordination in multidisciplinary design optimization, organization science, and software and engineering design. This literature tends to describe coordination as a preconceived set of processes meant to address the uncertainty and interdependence between diverse tasks that support a common goal. Empathetic leadership and the actions and behavior that comprise the Active coordination strategy we identified are similar to concepts in organizational sociology. Related concepts include the use of social capital, brokerage, systems thinking, and positive leadership. The precursors we identified as associated with the active facilitation of coordination, including extroversion and leadership traits, are consistent with skills and behaviors found in systems engineering competency models. Together these findings suggest that coordination is a central task of systems engineers and systems engineering, and concepts in social science such as brokerage, social capital, and positive leadership are likely to contribute to the further development of systems engineering theory and best practice.

This work has limitations in that this study purposely focused on experts in systems engineering, management, and leadership roles. Followup interviews or surveys are recommended to ensure the validity of findings across all levels of the organization. Our findings are in addition limited to large aerospace design organizations, and extension to other sectors that engage in large-scale design requires additional study.

In this study, we do not aim to develop a full theory of coordination, instead we seek to develop hypotheses for further exploration. Our findings here are supported by existing literature in a variety of domains, thus suggesting that Active and Passive strategies for facilitating coordination are representative of important organizational processes. We hypothesize that Active and Passive actions and behaviors used to facilitate coordination

are interrelated and that there exists a balance between them in practice. We explore this idea further in Chapter 4 by looking at the coordination approaches adopted by novice designers. This is in contrast to the experts studied here. Chapter 5 follows the exploratory studies with a simulation model that quantifies design outcomes from simplified versions of Active and Passive behaviors.

In each of the three studies presented in Chapters 3-5, there is evidence of a balance point between Passive and Active coordination-facilitation strategies. As discussed in Chapter 6, future work should aim to identify this balance point and explore what environmental parameters (e.g., group size and degree of interconnectedness between teams as in Tushman (1979), and design activity as in Strode et al. (2012)) and behaviors (e.g., Active and Passive) impact the balance point and the performance outcome of the designed system.

CHAPTER 4

Coordination in Design Teams

4.1 Introduction

Decomposition-based design of engineered systems is achieved by partitioning into subsystems, assignment to individual engineers, and coordination. The goal of coordination is overall system compatibility when all subsystems are integrated. System design optimization algorithms (also referred to as multidisciplinary design optimization) are well-established for performing portions of this process. Algorithms for decomposition and coordination of system design problems are given in Papalambros and Wilde (2017) and Martins and Lambe (2013).

However, in actual system design, individuals are not likely to coordinate their work following algorithmic procedures. In a design organization, distributed tasks must be first partitioned so that they can be worked in parallel, and then coordinated so that the results can be joined together to effect the overall project goal. In this organizational context, coordination is primarily a communicative process focused on information sharing among parallel tasks. Critically, the roles of individual participants influence this coordination process.

To better understand these roles, we analyze coordination in the controlled environment of student design projects (159 students in 32 teams). In the previous chapter, we identified some approaches used by industry experts use to facilitate the coordination of design work.

Here, our goal is to identify how these novice designers coordinate their design work in a design task requiring partitioning. We used self-report surveys and communication as a proxy for coordination. Using network representations of the survey data, we identify structures and patterns in team communication. By looking at the correlation between network structures and self-reported team roles, responsibilities, and teamwork, we are able to identify what we term *coordination roles* comprising responsibilities and communication behaviors found in these design teams.

Numerous sets of team roles have been identified in the literature, focused on team member behaviors (Aritzeta, Swailes, and Senior, 2007), personality (Stewart, Fulmer, and Barrick, 2005), and cognitive styles (Kress and Shar, 2012). Roles pertaining to coordination between groups have been identified (Sonnenwald, 1996; Sheard, 1996), but do not necessarily translate to coordination activity within groups or teams.

The structure of this chapter is as follows: We describe the study design, including our data collection, processing, and analysis approaches, in Section 4.2. A brief analysis of the survey data after post-processing into text keywords and networks is in Section 4.3. Finally the results of clustering analysis and description of each cluster are given in Section 4.4 and discussed in Section 4.5.

4.2 Study Design: Data Collection and Analysis Approach

4.2.1 Data Collection

Study participants were second year undergraduates at the University of Michigan enrolled in the course *Design and Manufacturing I* in Winter 2017. As part of the course, students work in teams of five to design and build Robotic Machine Players (RMPs). In each class section, four teams design and build their own RMPs, which collaboratively compete as a squad. The squad's RMPs are remotely piloted by students with a goal of scoring the most points. An overhead view of an in-progress game is shown in Figure 11.

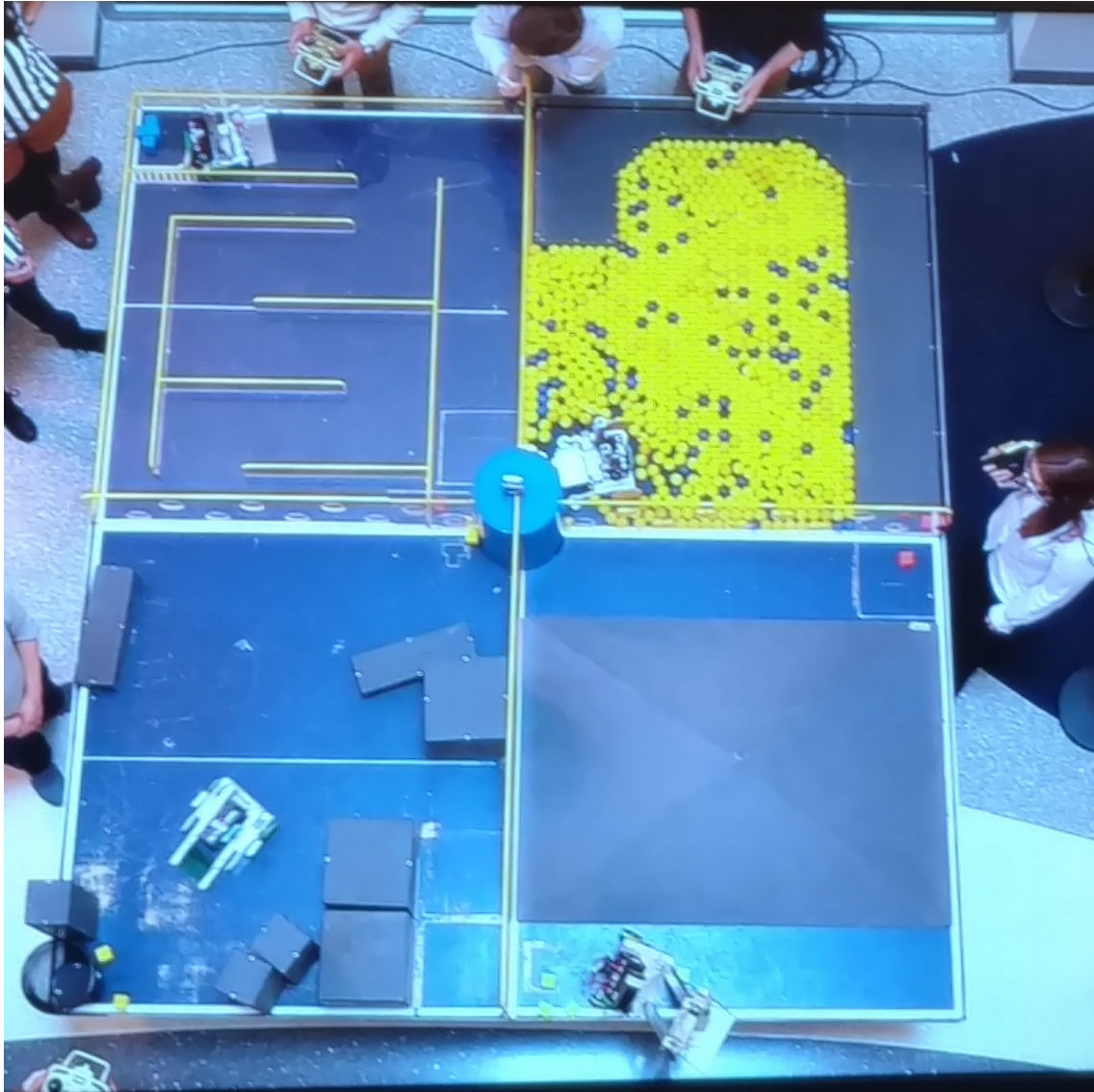


Figure 11: Overhead photo of game showing four-zone board, each with its own remotely-piloted RMP. Each zone requires navigation of different obstacles: clockwise from top left obstacles are a maze, ball pit, pyramidal ramp, and heavy blocks. (photo taken April, 2017)

RMPs are built from a standard kit though additional parts can be added or substituted. Typical features of an RMP include chassis, wheels, drivetrain, and a shovel or scoop to collect small cubes and move them to scoring zones. This course is an ideal situation to study coordination. Each team must coordinate its actions to design and manufacture an RMP and simultaneously coordinate with the other teams in their squad to implement a winning strategy. The course and typical project is described in more detail in Papalam-

bros (2015). In this study, the focus is on the coordination required within each team to design and manufacture an RMP rather than the coordination within the squad to design complementary RMPs.

We administered paper-form self-report surveys at the conclusion of the semester. The survey asked students to: reflect on the semester's project work; report their self-determined roles; identify tasks they were responsible for; describe the roles and responsibilities of their teammates; and recount the frequency and usefulness of task-focused communications. Existing survey instruments designed to conduct social network analysis within organizations (Cross, Borgatti, and Parker, 2002; Krebs, 2000) were adapted to create the survey instrument. The survey was voluntary but we received at least partial responses for 140 students in the course (88% of 159). In this study, our goal is to *characterize* the different approaches taken to coordinate distributed design work, rather than to *evaluate* those approaches. Accordingly, we did not correlate responses to grades or game results.

4.2.2 Analysis Approach

Our survey asked each respondent to share both their own, and their teammate's, main roles and responsibilities. These questions were open-response to avoid artificially limiting the scope of described responsibilities offered by the students. The survey then asked two sets of questions about inter-team communication, both focused on frequency and usefulness of communication with other team members. However, the questions differed in their focus on communication content. *Self Questions* asked about communication pertaining to the respondent's own work and *Peer Questions* about communication pertaining to their peers' work. Respondents were asked to indicate frequency of communication on a five-point scale ("4+ days per week", "2-3 times per week", "<2 times per week", "Monthly", and "Never"), and usefulness of communication on a four-point scale ("N/A", "Not Needed", "Helpful", and "Essential"). Because each team member was asked for this information, each pairwise interaction is described by up to two responses. In the case where one team

member did not respond, there are still responses from their peers to at least partially represent their participation on the team. Responses were collected without names using alphanumeric labels for team members within each squad, then aggregated across all squads by assigning a single unique number to each respondent.

We processed the survey data using two different analyses, each applied to different data subsets. The first used text analysis (described fully in Section 4.2.2.1) to identify common keywords representative of the roles and responsibilities respondents reported within their team. This allowed us to use participant language and terminology to describe types of tasks each undertook. We included in text analysis all individuals that gave responses to questions about the role and/or responsibilities of themselves or their peers. This gave us data from 140 individuals across all 32 teams (with 19 non-respondents). However, because surveys also reflect peer roles and communications, we could infer roles and responsibilities for 154 individuals.

Our second analysis focused on the communication network within the teams (detailed in Section 4.2.2.2). We model each team member as a node, and edges connecting two nodes represent communication between those individuals reported in the survey. We excluded individuals with no responses ($n = 19$) or partial responses for the communication questions ($n = 11$). Entire teams were thus excluded if there were missing responses from two or more members out of five ($n = 10$ teams). The resultant networks used for further analysis include 109 individuals from 22 teams (21 teams of five, and one team of four).

4.2.2.1 Keyword Identification and Analysis Method

Keywords were identified from responses using the Term Frequency and Inverse Document Frequency (TF-IDF) weights (Salton and McGill, 1986). TF-IDF measures relevance and importance of words or terms found in a specific document. In our case, each student's response to a question is treated as a document. We use the TfidfVectorizer module (Developers, 2018) implemented in the open source scikit-learn package to calculate TF-IDF

scores. The term frequency, tf , calculated for a word or phrase w in a given survey response r , is the raw count of that term in the given response. The inverse document frequency, idf , is calculated as a function of the total response count (N) and the number of those responses r that contain a word w . The variant of IDF we used is given in Equation 4.1:

$$idf(w) = \log\left(\frac{1 + N}{1 + \sum_r w}\right) + 1 \quad (4.1)$$

TF favors frequent words within each document, while the IDF favors rare words across all documents. The final TF-IDF score is a product of these frequencies, in turn favoring words that have relatively high scores for both frequencies. The TF-IDF weighting identifies common jargon in our survey responses as well as specific terms that may only be used within one or two teams. Technical jargon common across all teams is used to describe specific tasks; these terms receive high term frequency scores since they are identified by many respondents but low inverse document frequency scores since the terms are common. Language used frequently within a single team to describe each others' tasks receives a high inverse document frequency score due to its localization to a single team and a high term frequency score within those members' responses due to its commonality within the team. We first calculated the TF-IDF scores of all words in the raw survey responses to identify and rank top words used in descriptions of team roles, responsibilities, and project tasks.

We then reviewed the data iteratively to identify overlapping keywords such as 'manufacturing', a verb, and 'manufacturing plan', a noun, which we separated into distinct keywords. We also identified dissimilar terms that can be interchanged with little loss of meaning such as 'initial design' and 'preliminary concept' or 'team member' and 'teammate', which were combined under a single label for consistency. We manually consolidated terms into distinct keywords based on these two rules, in effect transforming the open response data into closed form representations of respondent language.

Table 4 gives examples from the final keyword set, categorized into six dimensions that reflect approaches to problem partitioning (aspect and object), approaches to project organization and coordination (team roles, management tasks, and reflection on how work was coordinated, e.g., split or shared), and deliverables or documentation (documents).

Table 4: Categories of keywords identified from survey responses, with examples

Aspect	Object	Documents	Team Role	Management	Coordination
CAD, design, manufacturing, assembly, brainstorm, analysis	Part, lift, system, scoop, chassis, drivetrain, axle	Report, manufacturing plan, sketch, video, assignment	Leader, member, general engineer, specialist	Manage, organize, make sure, ensure/verify, on track	Split work equally, everybody did everything, work together

The new set of transformed data used 471 distinct words across all responses. We recalculated TF-IDF scores using the transformed data. For this we used as a single response unit all survey responses about an individual, whether from that individual or their peers. This resulted in a vector of TF-IDF scores of length 471 for each individual. Each vector of length 471 was then normalized by its Euclidean norm. A value of 0 at any position in this vector indicates the term was unused in describing that individual’s role. Individuals with the highest normalized TF-IDF scores for a word were those whose teammates described their role or responsibilities the same way, and for whom few other words were used to describe their role and responsibility.

However, high TF-IDF scores can also come from frequently used words, such as prepositions, conjunctions, and indefinite pronouns. We removed such *stop words*, examples of which are ‘and’, ‘all’, ‘of’, and ‘did’, according to the word’s part of speech. The words remaining were nouns, verbs, and some adjectives. We retained for further analysis the top 50 remaining words as ranked by the average normalized TF-IDF scores across all individuals. These top 50 words are given in Table 6. The result was a vector of length 50 with weighted values representing the dominance of a word in the responses about an

individual's work.

4.2.2.2 Network Representations and Measures

To model the responses to communication questions, we represented the data as networks. Our survey data is on a scale, which we transform to a binary edge indicator using a fixed threshold. A given threshold – or combination of frequency and usefulness levels – means we included data that meets any combination of levels defined in the threshold. We compared several potential thresholds and ultimately selected a threshold of 4+ days per week under frequency, and ‘helpful’ or ‘essential’ under usefulness. This threshold yielded a dataset that includes 221 of 426 reported edges from the Self Questions, and 190 of 430 reported edges from the Peer Questions. This threshold was chosen over others as it presents connected team structures while preserving distinctive features. A lower threshold, including less frequent communication, forms team networks that are nearly all identical. A higher threshold, including only communication reported as essential, did not give connected teams in most cases. We also chose to separate the more objective frequency levels as opposed to the more subjective usefulness levels, including only ‘4+ days/week’ frequency and both ‘essential’ and ‘helpful’ usefulness in our chosen threshold. Under this threshold, only survey responses from respondent *A* about communication with peer *B* which follow the pattern below were counted as an edge:

- Self: *A* reports talking to *B* about *A*'s work 4+ days per week and it is essential or helpful to *A*.
- Peer: *A* reports talking to *B* about *B*'s work 4+ days per week and it is essential or helpful to *A*.

The Self and Peer network data cannot be directly superimposed as the edges in each network refer to communication about different things. To avoid oversimplification of our data, we created five networks, each telling us something different about the data. The *Self*

Network and *Peer Network* were created directly from the survey responses to each communication question, the former about communication with peers about the respondent’s own work and the latter about communication with peers about their peer’s work. We then combined the results of these two questions in three different ways. The first is a *Purpose-directed Network*, where there is an edge from *A* to *B* if either respondent *A* or *B* reports talking to the other about *B*’s work. The second is an *Initiator Network*, where there is an edge from *A* to *B* if respondent *A* reports talking to peer *B* about both *A*’s own work as well as *B*’s work. Finally, in an *Agreement Network*, there is an edge from *A* to *B* if both *A* and *B* respond that they talked to the other about *B*’s work; in other words, the responses from both members agree. In each network, an edge is defined by the direction and content of communication, given that the edge reported meets the selected threshold of frequency and usefulness. The network definitions are summarized in Table 5.

Table 5: Description of five network representations of thresholded survey data

Network	Edge from $A \rightarrow B$ in this network if:
Self	<i>A</i> reports talking to <i>B</i> about <i>A</i> ’s work
Peer	<i>A</i> reports talking to <i>B</i> about <i>B</i> ’s work
Purpose-directed	<i>A</i> reports talking to <i>B</i> about <i>B</i> ’s work OR <i>B</i> reports talking to <i>A</i> about <i>B</i> ’s work
Initiator	<i>A</i> reports talking to <i>B</i> about <i>A</i> ’s work AND <i>A</i> reports talking to <i>B</i> about <i>B</i> ’s work
Agreement	<i>A</i> reports talking to <i>B</i> about <i>B</i> ’s work AND <i>B</i> reports talking to <i>A</i> about <i>B</i> ’s work

In the Self, Peer, and Initiator Networks, a node’s out-edges represent their communication with someone else. In contrast, in the Purpose-directed and Agreement Networks, a node’s out-edges can be interpreted as that person having information about what their peers at the other end of those edges are working on – the direction of the edge indicating the topic of conversation. We characterize the data by calculating several measures from each network. These measures look at common features of social networks: degree, centrality, and clustering coefficient (Newman, 2018). We used node-level measures rather than network-level measures as this analysis focused on communication behaviors of indi-

viduals rather than of the team as a whole.

Degree is a count of the number of edges inbound to or outbound from a node. *Out-degree* is the number of people a respondent is communicating with or whose work they are receiving information about. *In-degree* is the number of people a respondent is on the receiving end of communication from. In the Initiator Network, out-degree carries a special meaning we call *initiator-ness*: the behavior of communicating with others about both one's own work and their peers. Those with maximum initiator out-degree we call *top initiators*. We find that most teams have one top initiator and rarely have more than two. Overlaying the edges in the Agreement Network with the Purpose-directed Network, we can also calculate what percentage of directed edges are reported by both respondents.

Centrality measures of nodes in a network quantify the importance of a node. *Betweenness centrality* is a measure of how many shortest paths between pairs of nodes a certain node lies on. Path length is the number of edges in the network traversed to get from one node to another. In a directed network, the paths are also directed. In networks where edges indicate information flow, a person with high betweenness centrality indicates they are receiving a lot of information and are also serving as a bridge or source from which others might be able to access that information. *Closeness centrality* measures how closely connected a node is to all others in the network. *Harmonic closeness centrality* is the ratio of the number of peer nodes in a network to the minimum path length required to reach all nodes. The highest value of harmonic closeness centrality is achieved for a person who reported initiating communication with all other members of their team.

Finally, the *clustering coefficient* measures the tendency of nodes to be closely linked, or clustered (Watts and Strogatz, 1998). Here we used the local clustering coefficient measure defined for the ego-network of a node: a person and their immediate connections. The local clustering coefficient is the ratio of links that exist within a node's ego-network to the number of links that could exist. In small teams, a node's local clustering coefficient is strongly correlated to overall team density.

Summary statistics of the node-level measures calculated from our data are given and discussed in Section 4.3.2. We used these measures as inputs for clustering analysis to identify groups of similar nodes based on the structural features of their team networks. Some detail on how we conducted our clustering analysis is provided in Section 4.2.2.3 and the results of the analysis are in Section 4.4.

4.2.2.3 Clustering Analysis

The text and network-based analyses result in a numeric vector that represents every individual in the analysis. We concatenate the weighted the TF-IDF scores for each of the top-50 keywords and the 17 network measures calculated for each individual. This results in a vector of 67 numbers representing task responsibilities and communication behaviors for each of 109 individuals in 22 teams as reported by themselves and their peers. We use these vectors as inputs to clustering analysis.

Since this clustering is exploratory, we selected an unsupervised clustering algorithm. We used hierarchical clustering based on the Euclidean distance calculated between each pair of vectors. In hierarchical clustering, each individual starts out in her own cluster. Then clusters are joined one at a time to their most similar neighbor based on the contents of their vector representation. Once each cluster has more than one individual, new calculations are measured relative to the cluster centroid. This proceeds until all individuals are joined into one final cluster. Recording the within-cluster similarity at each step allows the creation of a plot of similarity versus number of clusters. A sharp drop in within-cluster similarity as the number of clusters decreases suggests a good stopping point for clustering; continuing to join dissimilar clusters yields less interpretable and less meaningful results (Thorndike, 1953). We used the “knee” or “elbow” found from hierarchical clustering as an input for k-means clustering. In k-means clustering, clusters are calculated the same way as in hierarchical clustering except individuals can be moved from one cluster to another as clustering proceeds until each individual is closer to its cluster centroid than any others.

The results were obtained from clustering with $k = 5$ clusters. We describe the network measures and keyword statistics for each cluster in Section 4.4.

4.3 Characterization of Data

4.3.1 Characterization of Keyword Data

The TF-IDF measure is calculated for each word and each respondent, based on their roles and responsibilities as described by themselves and their peers. To form an aggregate score for each word, we average TF-IDF scores across all individuals. The top 50 keywords we use in clustering analysis are listed in Table 6, along with the number of documents the word appeared in (nonzero contributions to the average), as well as the inverse document frequency (IDF) of each keyword and the average term frequency (TF) across all responses. Note that the IDF and TF as reported in this table are unnormalized values, as the TF-IDF measure is normalized after the two components are multiplied together. Therefore $TF * IDF$ does not equal the TF-IDF value.

The top 50 terms represent each of the six categories we identified previously in Table 4, though they are not evenly distributed. Figure 12 shows the distribution of these top 50 keywords among the six categories. The Aspect and Object partitioning terms are the two categories with the most terms; there are 14 Object partitioning terms and 10 Aspect partitioning terms. That these categories are dominant reflects the variety of tasks individuals engaged in and parts those tasks applied to during the design process. The categories with the next most terms are Team roles and Coordination terms, each with 8 terms or 16 percent of this set. Seven Documentation terms make up 14 percent of the total, and three Management keywords make up the last 6 percent of the total. We present these results to describe the data collected about the roles and responsibilities of $n = 154$ individuals. All 50 keywords are used as input to clustering analysis, though only for a subset of individuals ($n = 109$) for whom both network and keyword data is available.

Percent of Top 50 Keywords in each Category

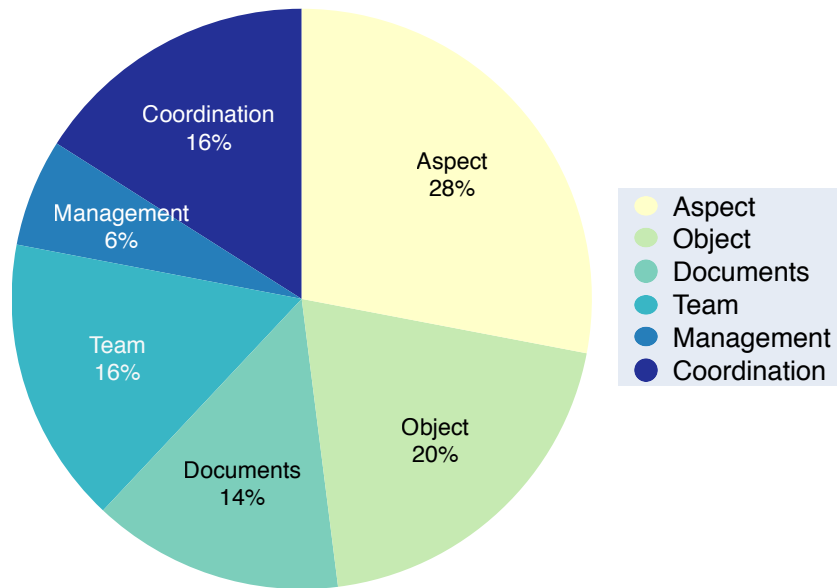


Figure 12: Percent of top 50 Keywords used in clustering analysis that are in each of six categories: Aspect, Object, Documents, Team, Management, and Coordination.

Table 6: Top 50 keywords identified from text analysis, ranked by their average TF-IDF score across all responses. Also shown are the number of documents each keyword appeared in, average TF score across all responses, and the IDF score for each keyword. These data represent the roles and responsibilities of n = 154 individuals.

Keyword	Mean TF-IDF †	No. Docs	Mean TF ‡	IDF ‡
1. Manufacturing	0.153	122	1.830	1.103
2. CAD	0.144	97	1.654	1.202
3. Design	0.119	112	1.409	1.140
4. Team member	0.104	83	1.038	1.269
5. Team leader	0.087	46	0.742	1.521
6. Part	0.083	84	1.006	1.264
7. Shared responsibilities	0.068	77	0.704	1.301
8. Everything	0.066	59	0.597	1.415
9. Help	0.063	71	0.654	1.336
10. Assembly	0.062	56	0.604	1.437
11. Machining	0.061	58	0.623	1.422
12. Mill	0.058	46	0.472	1.521
13. Manufacturing plans	0.055	45	0.547	1.530
14. Lathe	0.053	40	0.453	1.580
15. Engineering drawings	0.053	47	0.491	1.512

Continued on next page

Table 6 – *Continued from previous page*

Keyword	Mean TF-IDF †	No. Docs	Mean TF ‡	IDF ‡
16. Organize	0.052	37	0.434	1.613
17. Create	0.047	57	0.528	1.430
18. Work	0.047	53	0.453	1.461
19. Split the work	0.044	35	0.390	1.637
20. Assignments	0.044	34	0.415	1.649
21. Work together	0.038	38	0.340	1.602
22. Analysis	0.034	25	0.252	1.778
23. No specific role	0.034	19	0.239	1.892
24. Report	0.033	30	0.233	1.702
25. RMP	0.033	42	0.327	1.560
26. Scoop	0.032	19	0.258	1.892
27. Complete	0.030	31	0.252	1.688
28. Drivetrain	0.029	20	0.245	1.871
29. Writing	0.028	31	0.252	1.688
30. Chassis	0.027	24	0.245	1.795
31. Brainstorming	0.026	26	0.214	1.762
32. Team	0.026	27	0.239	1.746
33. Supports	0.026	22	0.176	1.831
34. Specialist	0.024	24	0.182	1.795
35. Milestones	0.024	26	0.208	1.762
36. Axle	0.023	22	0.164	1.831
37. Video	0.023	12	0.157	2.079
38. Everybody did everything	0.022	14	0.145	2.017
39. Lift	0.021	17	0.151	1.938
40. Manage	0.020	13	0.107	2.047
41. No overlap	0.020	16	0.126	1.963
42. Ensure/Verify	0.019	18	0.151	1.914
43. Lead	0.018	23	0.164	1.813
44. Engineer	0.018	9	0.017	2.193
45. Arm	0.017	10	0.113	2.152
46. Everyone	0.017	21	0.132	1.851
47. Build	0.017	20	0.138	1.871
48. Initial design	0.016	18	0.119	1.914
49. None (no role)	0.016	11	0.088	2.114
50. Tasks	0.016	18	0.113	1.914

† Values normalized before average

‡ Values unnormalized

4.3.2 Characterization of Network Data

Summary statistics of the measures calculated from network representations of survey data are given in Table 7. These measures were calculated for every node out of 109 included in this representation. First we note that most of the measures calculated span the range of possible values for a five-node network. Notable exceptions are the values of betweenness centrality calculated from the Self and Peer networks, which have maximums of 5 and 3, respectively. The maximum betweenness centrality possible, 12, is seen only in the Purpose-Directed network. Lower values of betweenness centrality indicate that each team member has connections with multiple peers, which results in reduced betweenness centrality for all nodes in the group. High values of betweenness centrality are typical of team networks that have a single focal hub who communicates with all other team members.

Table 7: Descriptive statistics for network measures calculated from survey data. These data represent the communication patterns of n = 109 individuals.

Measure (n = 109)	Min. value	Max. value	Mean (stdev)
Self Network			
In-degree	0	4	2.03 (1.00)
Out-degree	0	4	2.03 (1.89)
Betweenness centrality	0	5	0.17 (0.69)
Harmonic closeness centrality	0	1	0.56 (0.48)
Peer Network			
In-degree	0	4	1.74 (0.99)
Out-degree	0	4	1.74 (1.93)
Betweenness centrality	0	3	0.08 (0.45)
Harmonic closeness centrality	0	1	0.47 (0.49)
Initiator Network			
Out-degree (initator-ness)	0	4	1.53 (1.87)
Out-degree = max. possible? (binary)	0	1	0.36 (0.48)
Purpose-directed Network			
In-degree	0	4	2.90 (1.27)
Out-degree	0	4	2.90 (1.22)
Total degree	0	8	5.80 (2.30)
Pct. agreement edges out of total degree	0	0.75	0.24 (0.24)
Betweenness centrality	0	12	0.81 (1.91)
Harmonic closeness centrality	0	1	0.84 (0.24)
Local clustering coefficient	0	1	0.77 (0.31)

We also used the same network measures to compare our data to that expected of random 5-node networks. For larger networks we may opt to generate a selection of networks representative of the networks possible; our 5-node networks are small and therefore we can easily enumerate all possible networks. We generated the 2^{n^2-n} or 1,048,576 directed networks possible, and calculated degree and centrality measures for each. The resulting distribution of measures we call the *random network distribution*. We then use the random network distribution as reference distribution Q and our collected data as distribution P to calculate the Kullback-Leibler Divergence, a measure of difference between two distributions of data (Kullback and Leibler, 1951). The K-L Divergence is calculated for two distributions P and Q as in Equation 4.2:

$$D(P||Q) = - \sum_x P(x) \log \frac{Q(x)}{P(x)} \quad (4.2)$$

The measure is based on Shannon’s measure of entropy, and represents the information gained by describing data using distribution P as compared to distribution Q . K-L Divergence is measured in units of information entropy, or nats. If there is little difference, the reference distribution, here the random network distribution, is a reasonable description of the data collected. If values of the divergence are high, the collected data is demonstrably different from random networks.

We calculate the K-L Divergence for each of the measures listed in Table 7, excepting the percent of edges in the Purpose-directed network that were in agreement, i.e., reported by both individuals. Percent agreement is a derivative measure and cannot be calculated directly from the network itself. The remaining seven measures are: in-degree, out-degree, total degree, maximum degree (binary), betweenness centrality, harmonic closeness centrality, and local clustering coefficient. The cumulative distributions for each of these measures as calculated from our data and the generated networks (random network distribution) are plotted in Figures 13 and 14. The K-L Divergence calculated for each network

and measure combination is tabulated in Tables 8 and 9.

We briefly review some insights gained from this analysis, focused on where our data differs or is quite similar to the generated networks. We start with the plots of degree distribution in these networks. The distribution of in-degree found in our data follows closely the distribution of in-degree found in random networks, shown in Figure 4.13(a). The exception is for the Purpose-Directed network, where the majority of nodes in our data have an in-degree of four out of four. While the in-degree calculated for for the Self and Peer networks is similar to that of the random networks, the out-degree differs substantially. This is shown in Figure 4.13(b). The Self and Peer networks are equally likely to have an out-degree of zero or four out of four possible out-edges in our data, where the typical out-degree is two for nodes in random networks of size five. In the case of out-degree, the Purpose-Directed network and Initiator network more closely resemble the distribution from random networks.

Next, we look at total degree and maximum degree in Figures 4.13(c) and 4.13(d). In the purpose-directed network, a total degree of 8 is over-represented as compared to the random networks. This is consistent with the over-representation of both out-degree and in-degree of 4 in this network as compared to the random networks. Maximum degree is calculated in our data only for the Initiator network, and we see in Figure 4.13(d) that top initiators are over-represented compared to what we see from random networks. This may be an artifact from the way this network was constructed; it is a layering of pairs of edges from the in-degree and out-degree networks. However, this result is consistent with the high fraction of nodes with out-degree of four in the Self network.

The centrality measures shown in Figures 4.14(a) and 4.14(b) show that nodes with centrality values of 0 are much more common in our data than in random networks. This indicates many of the teams in our analysis are well-connected, more so than is typical of random five-node networks. This makes sense as some amount of connectedness is expected of individuals working together on a project, whereas the generated networks

Table 8: Values of Kullback-Leibler Divergence calculated for In-Degree, Out-Degree, Total Degree, and Maximum Degree network measures.

Measure	Network	K-L Divergence (nats)
In-Degree	Self Network	0.008
In-Degree	Peer Network	0.042
In-Degree	Purpose-Directed Network	0.656
Out-Degree	Self Network	1.396
Out-Degree	Peer Network	1.697
Out-Degree	Purpose-Directed Network	0.621
Out-Degree	Initiator Network	0.621
Total Degree	Purpose-Directed Network	1.668
Maximum Degree? (binary)	Initiator Network	0.381

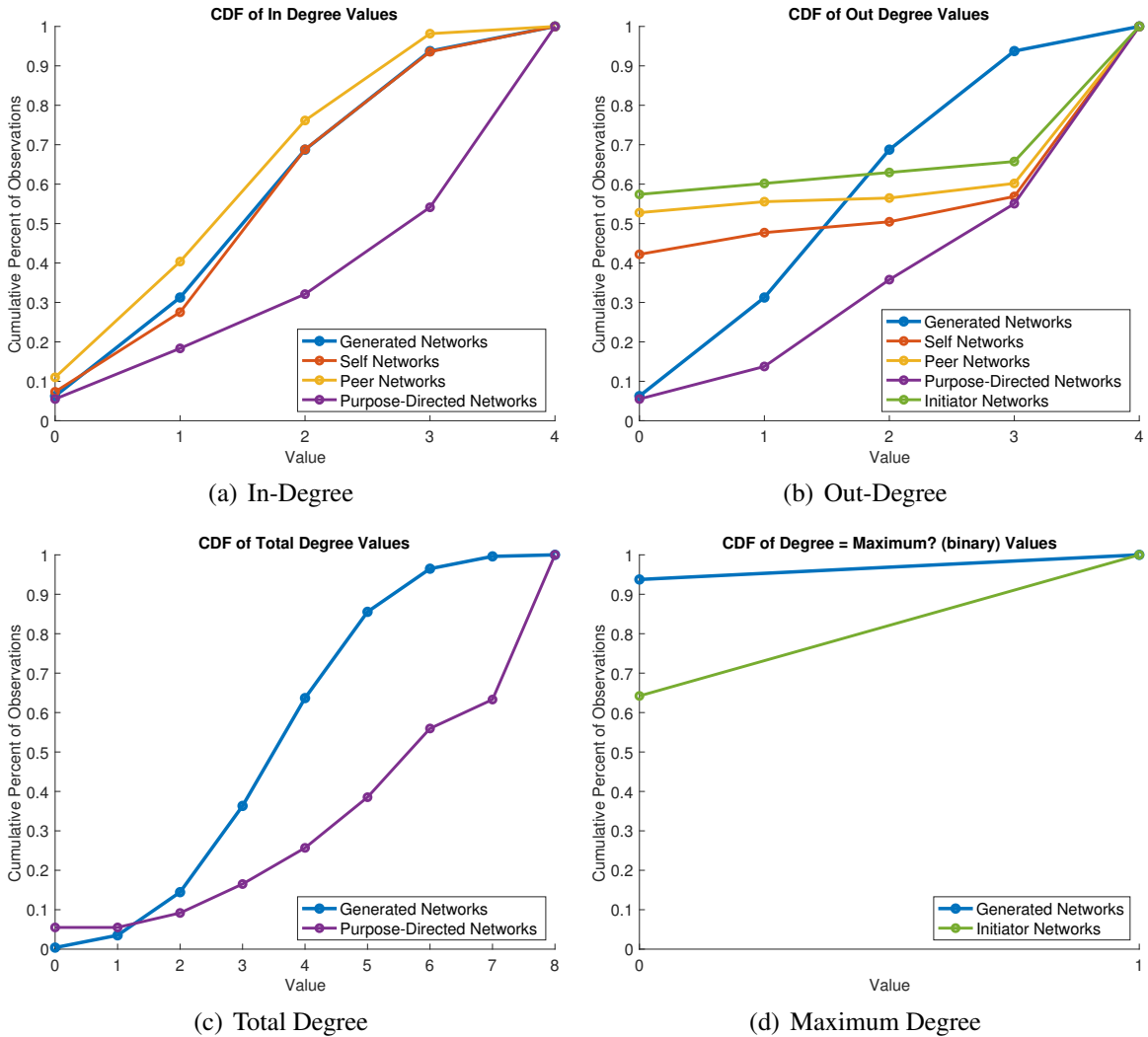
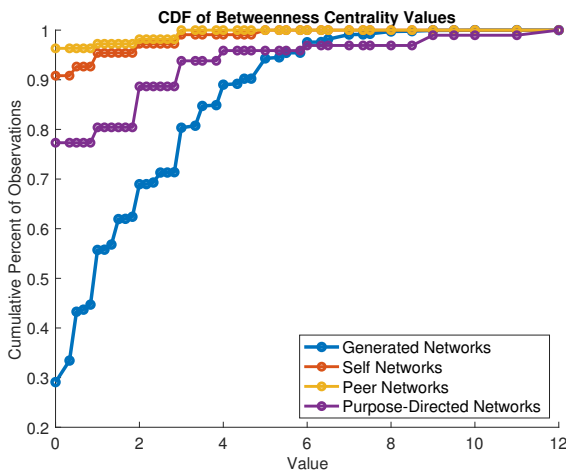


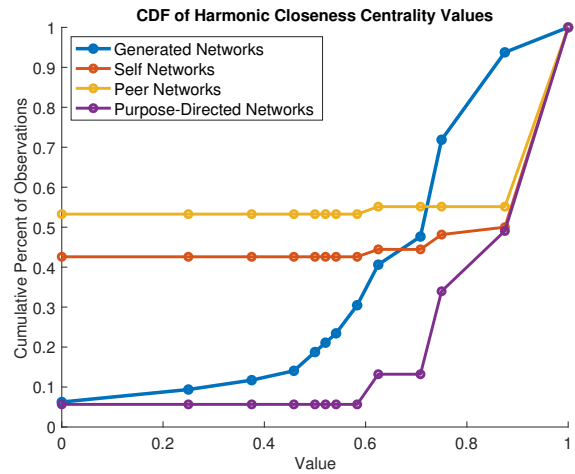
Figure 13: Distributions of In-Degree, Out-Degree, Total Degree, and Maximum Degree measures calculated from generated networks and network representations of survey data.

Table 9: Values of Kullback-Leibler Divergence calculated for Betweenness Centrality, Harmonic Closeness Centrality, and Local Clustering Coefficient network measures.

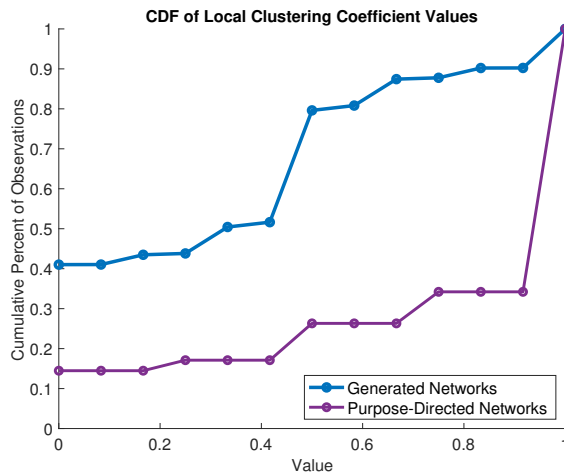
Measure	Network	K-L Divergence (nats)
Betweenness Centrality	Self Network	0.899
Betweenness Centrality	Peer Network	1.083
Betweenness Centrality	Purpose-Directed Network	0.834
Harmonic Closeness Centrality	Self Network	1.710
Harmonic Closeness Centrality	Peer Network	1.994
Harmonic Closeness Centrality	Purpose-Directed Network	0.953
Local Clustering Coefficient	Purpose-Directed Network	1.305



(a) Betweenness Centrality



(b) Harmonic Closeness Centrality



(c) Local Clustering Coefficient

Figure 14: Distributions of Betweenness Centrality, Harmonic Closeness Centrality, and Local Clustering Coefficient measures calculated from generated networks and network representations of survey data.

include all possible configurations of five members.

Finally we consider the local clustering coefficient as shown in Figure 4.14(c). We see that a local clustering coefficient value of 1 in the Purpose-Directed network is highly represented relative to random networks. Values of 0 and 0.5 are more common in random networks. Similar to low centrality, high local clustering coefficient values are an indicator of well-connected networks for small networks, which we see again are more common in our data than random networks.

4.4 Clustering Results and Interpretation

Clustering individuals based on both network measures and keywords appears to give a balance between clusterings obtained individually. We present here the results of this composite clustering based on the representative keywords and typical values of network measures in each cluster. Section 4.5 discusses further the composite cluster characteristics.

4.4.1 Clusters as Described by Keywords

The top 20 keywords in each cluster are given in Figure 15. Several keywords are highly ranked in all clusters: CAD, manufacturing, design, machining, manufacturing plans, team leader, and shared responsibilities. This finding highlights core aspects of coursework. Project work involves designing parts in CAD, putting together documentation for those parts, and manufacturing or machining them. It therefore makes sense that these terms would be widely used. Team leaders also were not performing just dedicated roles but were also engaged in many other tasks. It is then also reasonable that those serving as team lead may be grouped in various clusters with others serving different primary functions on the team. Finally, shared work is common, likely due to the inclusion in analysis of the responses to a survey question specifically asking respondents to indicate which of a peer's roles overlap with their own. In most cases, the students reported working together

or having shared responsibilities. Common to four of the five clusters is the keyword “split the work”. Combined with “shared responsibilities” appearing in every cluster, this may suggest a variety of working patterns within each cluster.

Figure 15 illustrates the top twenty keywords, sorted by average TF-IDF score of the terms across all cluster members. The darker the text color, the lower the average TF-IDF score, indicating less frequent appearance in describing the individuals in each cluster. We also present the distribution of keyword categories within each cluster in Table 10. This is calculated as a percentage of terms out of the top twenty that belong to each of six categories, introduced in Table 4: terms regarding Aspect decomposition, Object decomposition, Documentation, Team roles, Management and organization, and Coordination. The percentages are shown in Figure 10 alongside the distribution of all keywords used as clustering input and all keywords that appear in the top 20 of any cluster.

We briefly discuss each cluster in turn.

- *Leaders*: Six of the seven individuals in this group had responsibilities described as a combination of being team lead, managing work, organizing tasks or schedules, or leading subsystem design work. Note also the high proportion of management terms relative to peer clusters shown in Table 10. While only three members of this group were described using the term “team lead”, those three were given this title by their teammates often enough to give an average TF-IDF score in the group of 0.16.
- *Supporters*: This group is small (3 members). While in other clusters it is more common for individuals to refer to themselves simply as a “team member”, in this group it is more common for individuals to refer to themselves as “general engineers”. “Engineer” receives an average TF-IDF score of 0.21 in this group. They emphasized working together, evenly splitting work, and sharing responsibilities. Also emphasized are initial design and brainstorming, typically a collaborative task in teams.

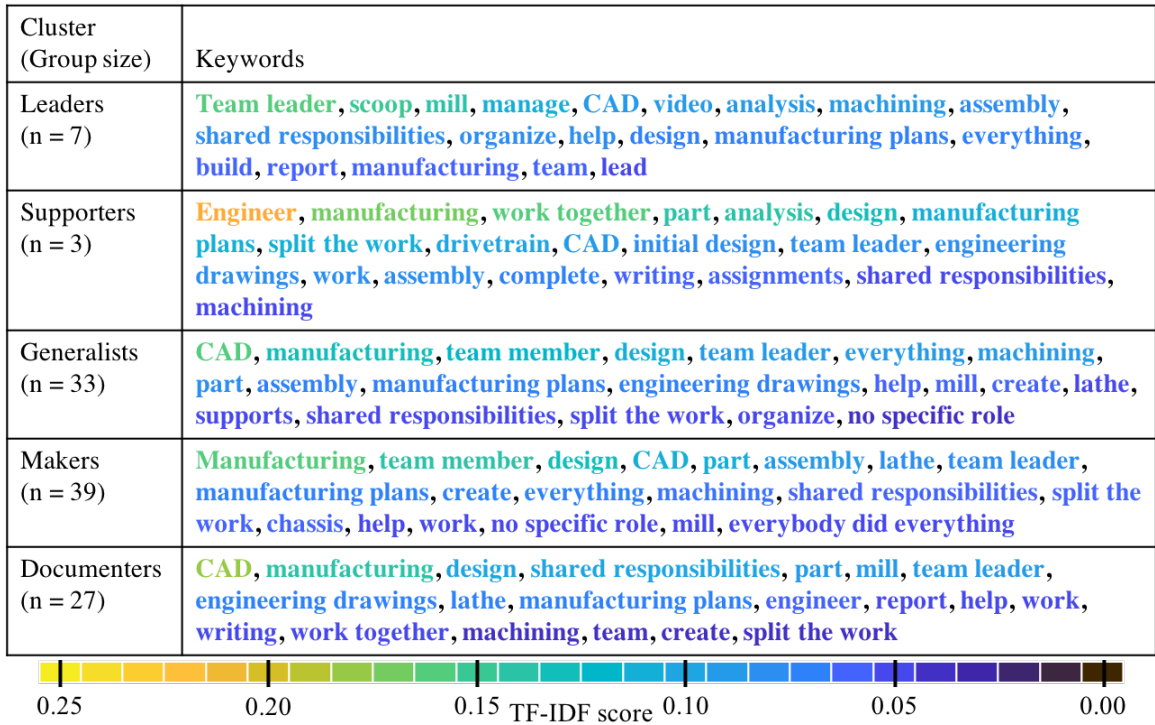


Figure 15: Top 20 keywords in each cluster in order of decreasing average TF-IDF score within cluster. TF-IDF scores are indicated by color.

Table 10: Percent of the top 20 keywords in each cluster in each keyword category, shown alongside the category distribution of the top 50 keywords overall used as clustering input and the 38 unique words in the top 20 keywords of any cluster.

Keyword Group		Percent of Keywords in each Category					
Cluster Name	Terms	Aspect	Object	Document	Team	Manage	Coordinate
All input	50	28%	20%	14%	16%	6%	16%
Top 20 of any cluster	38	34%	13%	16%	16%	5%	16%
Leaders	20	40%	5%	15%	15%	10%	15%
Supporters	20	45%	10%	20%	10%	0%	15%
Generalists	20	40%	10%	10%	15%	5%	20%
Makers	20	45%	5%	5%	15%	0%	30%
Documenters	20	45%	5%	15%	15%	0%	20%

- *Makers*: Almost everyone in this group was described as working on manufacturing, design, CAD, and generic parts. Specific subsystems were separately labelled in the keyword analysis. Chassis is emphasized for six members of this group, enough to bring it into the top 20 keywords in this group with an average TF-IDF score of 0.06.
- *Documenters*: As with other groups, making parts is emphasized. We call them documenters, however, due to the frequent mention that they were working on documentation such as manufacturing plans, engineering drawings, and writing reports. These terms apply to 19 members (70 percent) of this cluster.
- *Generalists*: The top keywords in this group are the same as for the Makers, but there are more diverse keywords in the top 20 of this group. Some individuals are also team leaders and organizers, others are working on documentation such as engineering drawings and manufacturing plans, and several specifically mentioned working on supports to integrate subsystems. 18 members (55 percent) of this group were perceived as working on “everything” by themselves or their peers.

4.4.2 Clusters as Described by Network Measures

We identify typical network measures within each cluster. Select measures calculated from within each cluster along with a representative network diagram are shown in Figure 16. Although these clusters are the same as discussed in the previous section, we describe them here in terms of their local network characteristics. We discuss the labels in combination in Section 4.5.

There are 39 top initiators out of 43 individuals in the groups *Connectors*, *Hubs*, and *Low-density Hubs* due to their maximum out-degree in the Initiator network. The remaining four individuals have high Initiator out-degree. The three clusters are divided by their purpose-directed betweenness centrality values, reflecting the number of other communication edges in the team network. The remaining two clusters of Communicators and



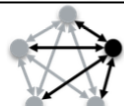

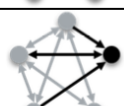
Cluster label (group size)	Select network measures: mean (standard deviation)							Example network, purpose directed network
	Self out-degree	Self in-degree	Peer out-degree	Peer in-degree	Initiator out-degree	Purpose directed total degree	Purpose directed betweenness centrality	
Hubs (n = 7)	3.9 (0.4)	1.0 (0.6)	3.9 (0.4)	0.6 (0.5)	3.9 (0.4)	7.7 (0.8)	3.7 (1.1)	
Low-density Hubs (n = 3)	4 (0)	0.7 (0.6)	4 (0)	0 (0)	4 (0)	8 (0)	10 (1.7)	
Connectors (n = 33)	3.8 (0.6)	2.3 (0.8)	3.9 (0.2)	1.9 (0.7)	3.8 (0.6)	7.8 (0.6)	0.9 (0.8)	
Communicators (n = 39)	1.5 (1.8)	2.5 (0.9)	0.5 (1.2)	2.4 (0.8)	0.1 (0.3)	5.9 (1.0)	0.0 (0.2)	
Receivers (n = 27)	0 (0)	1.4 (0.9)	0 (0)	1.1 (0.8)	0 (0)	3.2 (0.8)	0 (0)	

Figure 16: Individual cluster descriptions with selected network measures and example team network from purpose-directed network with representative node highlighted

Receivers are not initiators, typically having zero out-degree in the Initiator network.

A summary of each cluster is below.

- *Connectors*: These individuals are top initiators in medium to high density teams as indicated by their low purpose-directed betweenness centrality values of 0-3. They tend to be one of multiple connectors on their team.
- *Hubs*: They are top initiators but in contrast to the Connectors, they have low in-degree in the Self and Peer networks. This means that their teammates are communicating with them less often, making Hubs more central to their team.
- *Low-density Hubs*: This group is particularly identified by high purpose-directed betweenness centrality values, ranging from 9-12. They are top initiators in the sparsest of the 22 teams we looked at and consequently they are the main source of commu-

nication in the team at the frequency and usefulness threshold we defined.

- *Communicators*: They are particularly defined by their total degree in the Purpose-Directed network. The average degree of 5.9 indicates that they are well-connected to most of their teammates. Their Self and Peer network in-degree measures are similar to the group of Connectors and their out-degree in these networks is similar to the group of Receivers. This combination suggests these individuals are situationally engaged in discussion with their teammates about either their own work or their peers' work, and rarely both.
- *Receivers*: They are always on the receiving end of communication from their peers, with zero out-degree and typically non-zero in-degree in both Self and Peer networks. Several in this group were completely independent of the rest of their team, meaning no communication with their peers that met our frequency and usefulness threshold for analysis. All measures we calculated were consequently zero for these individuals.

4.5 Discussion

4.5.1 Coordination Roles

Our analysis identified five clusters of individuals, each representing a different combination of project tasks and communication behaviors. Thus far we have considered these roles on their own. Since coordination is a system-level outcome, it is also important to examine how these roles are distributed within teams. Again we seek to characterize what combinations exist rather than evaluate them. Connectors-Generalists tend to be on teams with Communicators-Makers: 59% of the teams in this analysis are comprised of only these two roles. The other 41% of teams have at least one Receiver-Documenter. These individuals did not report frequent and useful communication with their peers, and are

typically in teams with a Hub-Leader or Hub-Supporter. The tasks these Hub roles engaged in are different, but they serve a similar communicative role of keeping the team connected. Their presence alongside Receiver-Documenter roles (and never alongside Communicators-Makers) suggests that team roles are in part a reflection of the team composition. This is consistent with past observations that individuals take on different roles as the situation demands (Senior, 1997; Stewart, Fulmer, and Barrick, 2005; Adams et al., 2009).

The dichotomy between teams that include Communicators-Makers and Connectors-Generalists and those that are some combination of all roles except Connectors-Generalists may also be an indication of how work was partitioned among the team members. The central Hub role, either Supporter or Leader, is in teams with Communicators-Makers and/or Documenters-Receiver. This may indicate highly partitioned work, where the Communicators-Makers and Documenters-Receiver were all working on their own parts. In a hierarchical partitioning strategy, subsystems are divided and their coordination is left to a central overseer, in this case, a Hub role. This is consistent with the observed low communication reported from the remaining team members. In contrast, the teams with Communicators-Makers and Connectors-Generalists tended to form highly connected team networks with frequent communication among all members. This is consistent with non-hierarchical partitioning, where there is no central coordinator and all parties are managing their own interactions.

We also note that while we have a group we call 'Leaders', all clusters included a few team leaders. According to Ehrlich and Cataldo (2014), teams with technical leaders that have high centrality in their team and high communication relative to others in the team are more likely to deliver a higher quality product. Again, we did not evaluate the performance of the teams' output, but our Hubs-Leaders group fits the profile of high-centrality leaders. Shared leadership is also common in student project teams (Novoselich and Knight, 2018), which may look like a relative lack of centrality for those doing leadership tasks (Ehrlich

and Cataldo, 2014). Team leaders not in ‘Hub’ clusters may be an example of instances where leadership tasks were shared by multiple team members. In addition, different leadership styles have different associated communication styles (de Vries, Bakker-Pieper, and Oostenveld, 2010). Therefore, on the whole, these teams may exhibit varied leadership styles including shared leadership among members.

4.5.2 Correlation to Interview Data

We found in Chapter 3 a dichotomy between authority-based or Passive approaches to coordination and empathetic leadership-based or Active approaches to coordination. One distinguishing factor between these methods is the amount of communication each engages in: the Active methods of coordination-facilitation emphasize regular communication with peers to ensure everyone has a common understanding of shared work.

In the results of the survey presented in this chapter, we also see a dichotomy. Of the respondents for whom we had enough data to include in the clustering analysis, about half were “top Initiators”: regularly engaged in communication with all teammates regarding each others’ work. The other half were not top initiators, or were “Non-Initiators”, and instead tended to have higher in-degree than out-degree. This means they were more likely to have teammates initiate communication with them than vice versa. This dichotomy between Initiators and Non-Initiators may be a parallel to the Active and Passive dichotomy shown in Chapter 3. Our finding that Initiators tended to be in teams with Non-Initiators suggests we may expect individuals with Active and Passive coordination-facilitation behaviors to be in equal balance. We explore further the balance between these two behavioral archetypes in Chapter 5.

4.6 Summary

This study is an early step in understanding how coordination is practiced in actual system design. We used a combination of network analysis and text analysis to identify five coordination roles adopted in distributed project work. We see some indication that teams seemed to adopt hierarchical or non-hierarchical partitioning strategies based on the combination of roles within each team, as well as a possible indication of balanced Active and Passive coordination-facilitation behaviors in each team.

Future work could look into additional factors impacting successful teamwork such as those identified by Maier et al. (2008) and Crabtree, Fox, and Baid (1997) to more fully characterize coordination roles. There is also an opportunity to explore correlation between the identified coordination roles and the result of other personality inventories or team role assessments. This includes personality and attitude inventories, such as the Myers-Briggs Type Indicator (Myers and Myers, 1980), Big Five (Goldberg, 1990), Clifton Strengthsfinder (Asplund et al., 2007), and the Belbin Team Inventory (Aritzeta, Swailes, and Senior, 2007). These instruments alone and in combination have been used for team formation and to predict team success (Varvel et al., 2004; Gardner and Martinko, 1996; Mount, Barrick, and Stewart, 1998; Clinebell and Stecher, 2003; Kosti, Feldt, and Angelis, 2014; Acuña and Juristo, 2004). The focus of these instruments is on individual personality and preferences, which may or may not correlate to the roles identified here which focus on coordination behavior within the team. Other instruments that measure attitudes toward engineering and design work such as Greene, Gonzalez, and Papalambros (2019) may also be useful to explore correlation between team roles based on coordination behavior and attitudes toward the tasks and communication that comprise coordination work.

In addition, we acknowledge several limitations that should be addressed in future iterations of this study. The first is that novice designers are not necessarily the best proxy for understanding of organizational processes. Novice and expert designers approach problem-solving differently (Cross, 2004; Smith and Leong, 1998). Novice designers also tend to

focus on subsystem level optimization rather than system optimization (Austin-Breneman, Honda, and Yang, 2012). This suggests professionals may have different coordination approaches than the student teams recruited for this study, indicating validation from industry practice is warranted.

We also acknowledge that self-report surveys suffer from recency bias and reporting bias. Deploying our survey at the end of the design project means end-of-term activities are likely overrepresented in reported project responsibilities. Questions about communication frequency and usefulness may be inaccurate, and tasks reported for each team member may be incomplete. Further, we do not know from our survey data the rationale behind team interactions – why certain individuals reached out frequently and others did not. In subsequent studies, these limitations could be mitigated by additional data sources, such as more frequent surveys, review of documentation, or communication observed via project management software.

CHAPTER 5

Using Agents to Model Coordination

5.1 Introduction

The previous studies presented in Chapters 3 and 4 identified behaviors associated with coordination activity during distributed design work. In Chapter 3, we identified actions and behaviors enabled by authority that emphasize reliance on organizationally-defined lines of communication, common schedules, and common design processes among group members. We also identified actions and behaviors enabled by empathetic leadership or social capital, which emphasize developing and maintaining connections within the organization that go beyond those established via hierarchy and authority. Many of the individuals we interviewed also mentioned that they could use either strategy to facilitate coordination – authority-driven or empathetic leadership-driven – but sometimes preferred one over another. This led us to hypothesize that the two strategies are complementary.

In Chapter 4, we observed that teams of novice designers tended to have a mix of more talkative individuals who engaged with their peers regularly about each others' tasks and quieter individuals who interacted less and worked on their own tasks. This is a similar dichotomy as that between the authority-based actions and empathetic leadership-based actions we identified in Chapter 3.

Combined, these two studies show that individuals have different behaviors they use or prefer when engaging in coordination of distributed work. This finding motivates the

following research question:

How do the identified coordination-facilitation behaviors impact coordination effectiveness in terms of a group's performance on a distributed design task and the interaction costs associated with the coordination strategy used to complete that task?

The objective of the study in this chapter is to address this research question through the development of an agent-based simulation model. In agent-based modeling, individual agents are given simple logic to govern their behaviors, and global or aggregate behavior often emerges through the interaction of agents (Wilensky and Rand, 2015; Railsback and Grimm, 2012). Agent models have been developed to model natural systems, engineered systems, and social systems (Wilensky and Rand, 2015; Railsback and Grimm, 2012; Epstein and Axtell, 1996; Miller and Page, 2007).

Agent models typically focus on a single type of interaction between agents. These are therefore simple models, but illustrate how simple behaviors can propagate throughout an organization to create more complicated outcomes. Agent models have been used to study a number of phenomena, including organizational processes and design processes. Examples include advice-seeking behavior (Levine and Prietula, 2011; Tóth et al., 2018), communication and decision-making (Meluso and Austin-Breneman, 2018; Vermillion and Malak, 2015; Farooqui and Niazi, 2016), cognition (McComb, Cagan, and Kotovsky, 2015; Gero, 2002), and teamwork, collaboration, and design (Soria Zurita et al., 2017; Levitt, 2012; Fernandes et al., 2017; Panchal, 2010). These models have been able to test the impact of observed behaviors on designed outcomes through simple assumptions of how agents interact. Agent modeling is particularly valuable in the context of engineered systems design where agents are heterogeneous and systems are highly networked, making interdependencies difficult to quantify (Heydari and Pennock, 2018).

Agent modeling was selected for this study because of the method's emphasis on connecting individual behavior to global outcomes. This emphasis aligns well with our re-

search question: connecting individuals' actions and behaviors to system-level coordination outcomes. However, for the results of agent models to be interpretable, agent behavior must be simple. Thus the nuances and multiple facets of behavior identified in prior chapters' studies must be pared back. In this third study, we imbue agents with simple versions of the coordination-facilitation behaviors previously identified. This model includes two behavioral preferences. The first is an individuals' preference to proactively initiate communication with peers versus generate new knowledge on their own. The second is a preference to adhere to organizationally defined communication channels versus additionally leveraging a network of local peers. These two behaviors focus especially on the aspects of Active and Passive coordination-facilitation strategies that align with formal and informal organizational behavior.

To study coordination, the agents are tasked with completing a distributed design problem. We prescribe a single design task which is partitioned among all agents. Each agent contributes to the overall task by working on their own local partition. This mimics division of labor within an organization. To quantify the impacts of different coordination-facilitation behaviors, we use the model to evaluate two performance measures: agent's task performance on their local partition, as well as the collective performance on the global task, i.e., the result achieved through coordination of individual tasks. We perform parametric analysis on global variables such as how agents are connected, how the design task is partitioned, i.e., which agents start with what task, and how agent behavior profiles are distributed among the set of agents. We also have the ability to vary the parameters that govern individual agent behaviors. A detailed description of the model follows in Section 5.2. Results from the model and parametric analysis follow in Sections 5.4 and 5.5. The intervening Section 5.3 states the hypotheses motivating the particular exercises of the model.

5.2 Model Description

In this section we describe the design and structure of the agent model. We loosely follow the *ODD Protocol* developed by Grimm et al. (Grimm et al., 2006), which specifies what elements of agent models should be described aiming for reproducibility. The basic elements of the ODD protocol are Overview, Design, and Details, each increasingly detailed. We describe in the following subsections the model objectives, environment, procedural structure, agent behavior, parameters, and model outputs.

5.2.1 Objectives

The purpose of the model is to quantify the impact of agent coordination-facilitation behaviors on organizational and technical performance measures of a distributed design task. For a quantitative assessment, we look at the performance impacts of varied relative concentration of individuals with one of two archetypical behaviors within an organization. We also compare results to a direct solution to the same design task given to the agents. The organization's collective task is to complete a distributed classification task. The agents' aggregate classification accuracy is used as a technical performance measure and the number of peer-to-peer interactions agents engage in is used as an organizational performance measure.

5.2.2 Network Structure

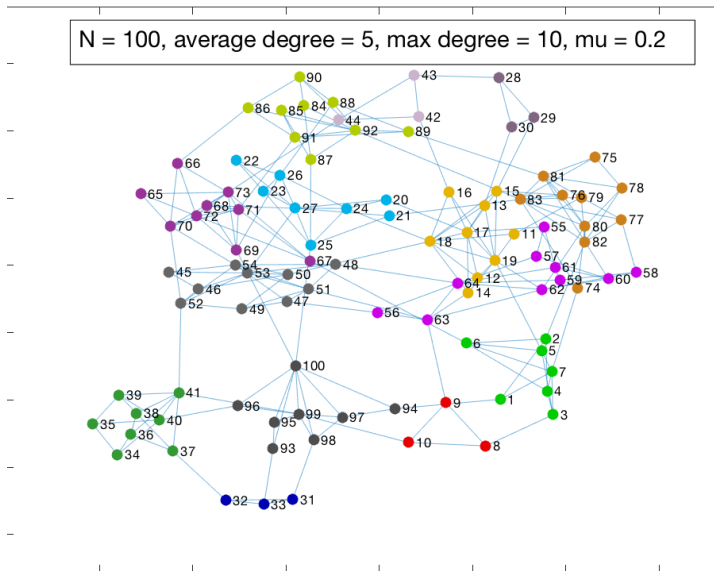
Agents in the model are connected by a network which represents the structure of an organization. Nodes in the network represent individuals and edges represent formal ties between individuals. The network structure is generated using the LFR benchmark model, named after its original authors Lancichinetti, Fortunato, and Radicchi (2008). Their generative model produces networks with inherent community structure. Community structure within networks is akin to modularization: a *community* is a group of nodes that show many

links within the community and few links outside (Newman, 2006). Here, the community structure within the network represents teams within an organization. The LFR generative algorithm was designed to create test cases for community-detection algorithms, with non-uniform distributions of both node degree and community size adding realism (Lancichinetti, Fortunato, and Radicchi, 2008). The degree distribution and community size distribution are fit to power law distributions, but in smaller networks like ours (100 nodes and typically less than 20 communities), the observed distributions do not follow the power law exactly.

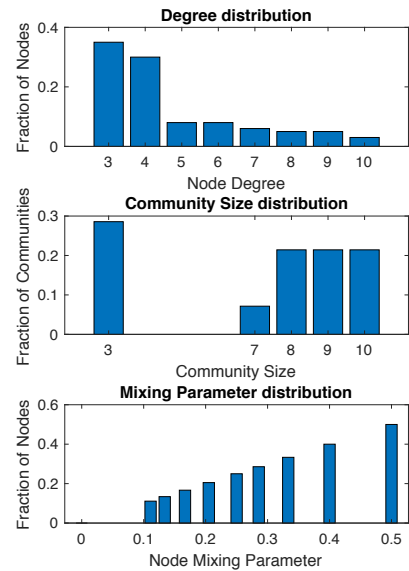
Table 11: Parameters used as input for generated LFR networks

Parameter	Symbol	Value
User-defined Parameters		
Number nodes	N	100
Average links per node	k	5
Maximum node degree	maxk	10
Average node mixing parameter	μ	0.2
Automatically Calculated Parameters		
Community size range		[3, 10]
Default Parameters		
Degree distribution exponent	λ_1	2
Community size distribution exponent	λ_2	1

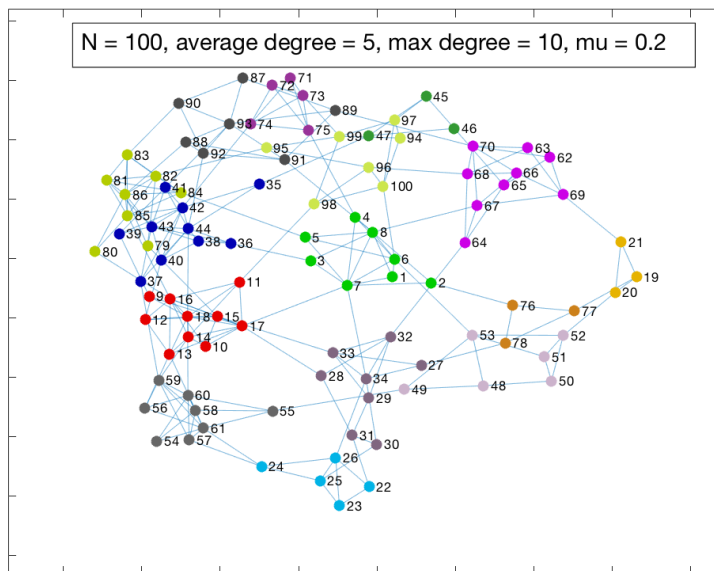
Table 11 gives the parameters used as inputs to the generative network model and the values selected for our use. Several parameters, such as the average degree and average mixing parameter, are average values for the nodes in the network. Therefore a single set of parameters can yield multiple different networks. Two such networks with the same parameters are shown in Figure 17. The selected parameters result in a network of 100 agents, each with an average of five linked neighbors and a maximum of ten. Each agent is assigned membership to a single community; community sizes range from three members to ten. The mixing parameter μ is the proportion of any agent’s connections that are outside their own community. It is specified at the network level as an average of the mixing parameter for all individual nodes. A value of $\mu = 0$ means that, on average, agents have



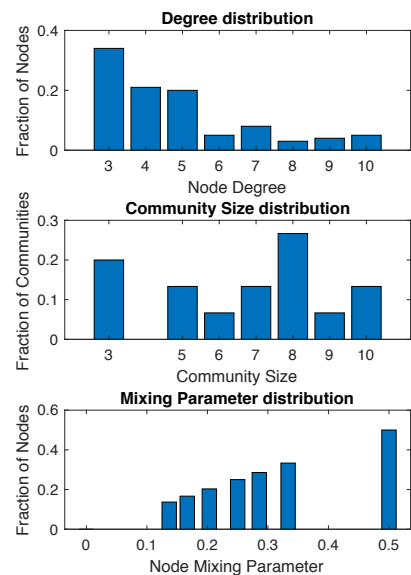
(a) Network 1



(b) Distribution of Measures for Network 1



(c) Network 2



(d) Distribution of Measures for Network 2

Figure 17: Two networks generated using the LFR algorithm using the same parameters. Panels (a) and (c) depict the network structure for each network, where nodes are numbered in order by community and colored according to community membership. Panels (b) and (d) depict distributions of node degree, community size, and node mixing parameter within the respective networks.

no connections outside their own community. The result is a disconnected network where each community is isolated. A value of $\mu = 1$ means that, on average, all of an agent's connections are with agents outside their own group; the result is a dense network with no obvious community structure. The degree distribution and community size distribution are fit to the form given in Equations 5.1 and 5.2:

$$P(k) \sim k^{-\lambda_1} = k^{-2} \quad \text{where } k \text{ is node degree} \quad (5.1)$$

$$P(s) \sim s^{-\lambda_2} = s^{-1} \quad \text{where } s \text{ is community size} \quad (5.2)$$

The power law degree distribution means that few nodes have high degree and many nodes have low degree. Networks with this property are considered scale-free, and are common in natural and social systems (Barabási, 2016). With an average mixing parameter of $\mu = 0.2$, most edges in the generated networks are within-community links, rather than between-community links. Generated networks with communities labeled by color are shown in Figure 17. Multiple networks generated from the same input parameters as given in Table 11 differ primarily in their community size distribution. In addition, numerical community labels are randomly assigned. Both the number of agents in each community and the community label impact how data is partitioned, described more in the following section. However, as we know partitioning and coordination are related, we expect the network structure to be a significant factor impacting results. Therefore we look at multiple different networks, or cases, in our analysis and draw conclusions from across all cases.

5.2.3 Decision-Making Task

In this model, agents complete a distributed classification task. Distributed classification was selected because it represents a distributed decision-making problem: each agent works independently and becomes an independent classifier. Individual agent classification results are aggregated to give the global classification result. This task was selected firstly because

performance for both individual agents and their aggregate is easy to calculate: classification results are either right or wrong. We are also able to leverage an inherent feature of classification: in general, accuracy improves with more data. A distributed classification task means each agent has less data to work with, and coordination through efficient collection of data is therefore necessary to improve performance.

In a classification problem, the objective is to correctly categorize or label unknown data given a prior set of known data and correct labels for that data. Unknown data is referred to as *test* data, whereas prior known data is referred to as *training* data. Data is a composition of *variables* which describe the data and *labels* which indicate the correct categorization. A classifier's accuracy is determined by comparing the classifier's guess at the correct labels for test data with the true labels for test data.

Agents in our model are tasked with the correct classification of data representing letters of the English alphabet. The dataset we use was originally developed by Frey and Slate (Frey and Slate, 1991). They generated 20,000 unique letters of the alphabet based on existing fonts, including all letters from A to Z. Each data item consists of 16 measurements (variable values) of each letter image paired with the letter label. The full dataset therefore consists of 20,000 items: each a matched pair of 16 variables and one alphabetical label. The dataset is available from the UCI Machine Learning Repository as "Letter Recognition Data Set" (Dua and Graff, 2017).

The Letter Recognition dataset includes between 734 and 813 instances of each letter in the alphabet. A confusion matrix of the dataset is given in Figure 18, showing for each letter in the dataset, what is the most similar item out of all other (19,999) items. The label of the most similar item is the predicted label. All letters are classified correctly over 90% of the time, when all possible training data is available. When misclassified, there are several pairs of commonly confused letters. Examples include (I, J), (F, P), (H, K), (K, X), (T, Y), (O, Q), (B, R), and (B, V). Others are asymmetrical: for example, O is often misclassified as D, but not vice versa. This characterization of our data shows that there are more and

less challenging classification tasks given different partitioning, i.e., combinations of letters in test and training data. For example, an agent seeking to distinguish I’s and J’s or O’s and Q’s in their test and training data is more likely to make incorrect classifications than an agent with A’s and B’s. The natural variation of difficulty is akin to the real scenario of different design groups having more and less difficult tasks.

		Predicted Label																										count	
		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z		
Actual Label	A	786	1								1																1	789	
	B		717		3	4	1	1	6		1	1		1						12	4			14				1	766
	C			717		6		6					1			2						1		2					736
	D		1		779	1		1	8						2	5			6	2									805
	E		1	4		728	1	10	1			6	3				1	1		2			1	4			5	768	
	F		1			726		1	2				2			26	1	1	1	10			2	1	1			775	
	G		2	7	5	7		740				2	1			3		1	1		1			2	2			773	
	H		8	1	13	1	1	3	662		1	19	1	2	1	5	2		10	1		2				1		734	
	I					1				731	21						1										1	755	
	J		1		1		2		1	23	714		1		2								2					747	
	K		2			7		1	23		679									11						16		739	
	L					3		2	1		2	2	748					1	2									761	
	M		3					1						777	1	1	1						5	3				792	
	N	1	1		4				2		1			2	757	3			9				2	1				783	
	O			3	10			1								731		7						1				753	
	P				3	2	30		1				2		1		759	2	1		1						1	803	
	Q				2			3								10	4	759					1				1	803	
	R		17				1	1	10			10	1		8			1	708					1				758	
	S		6			3					1								3	732	1	1						748	
	T	1	1	4	1		1		1	1												773				2	10	796	
	U	1	1					6				2		1									800	1			1	813	
	V		11	1			1	1						2	2	1	1						739	1		4		764	
	W							1						2	1	3				1				2	3	739		752	
	X		2		2	5						16							1	1	1	2					756	1	787
	Y	1	1																			10	2	3	1	1	767		786
	Z					1																						725	734
count	790	777	737	823	768	765	771	724	757	743	737	758	787	777	764	795	781	765	745	798	810	774	750	783	785	736			

Figure 18: Confusion matrix of all 20,000 items in Letter Recognition Data Set (Dua and Graff, 2017).

5.2.3.1 Data Partitioning

To set up the distributed classification task, we allocate the 20,000 items from the Letter Recognition Data Set into test and train data at the local (agent) and global levels. A diagram depicting this multi-level partitioning process is shown in Figure 19. First, we set aside the first 200 (1%) items as global test data. This dataset includes both the variables describing each letter and the character labels themselves. Then the remaining 19,800 items are divided among communities of agents. Each community receives training data

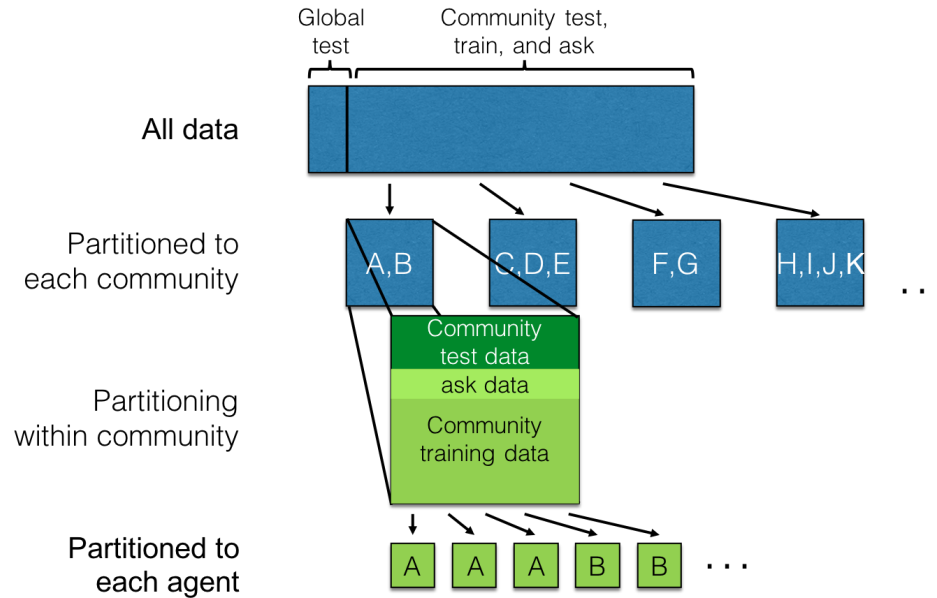


Figure 19: Illustration of how classification data is partitioned into global test, local test, local ask, and local training data. Letters shown in each community partition are examples. Global data is used by all agents, while local data is accessible to agents within a single community.

proportional to the number of agents in the community. Data can be partitioned to each community randomly, sorted alphabetically, or reverse alphabetically. In the random assignment, each community's partition contains information about most or all 26 letters. Alphabetical partitioning assigns data to each community in order according to the data label: e.g., Community 1 receives A, B, and C; Community 2 receives D and E; and so on. Reverse-alphabetical partitioning follows the same procedure, but assigns data in reverse order from Z to A. The number of letters each community receives depends on the group size, and adjacent groups may share some training data for the same letter.

A community's data is then divided into test, training, and what we call *ask* data pools. The local training and test data allows agents to develop local classifier models prior to applying their models to the global classification task. The ask pools are a reserved set of data, which agents can draw from to mimic the creation of new data. Each community's data is divided by default as follows: 10% local test data, 5% local ask data, and 85% local

train data. The local test and ask data is shared by all community members, and the local train data is then divided equally among all agents in the community.

5.2.3.2 Distributed Classification

The model aims to simulate coordination behavior used while solving a distributed design problem. Agents begin with distinct training data and access to their own within-community test data. The letters represented in the community test data reflect the letters agents in the community have in their training data, as described above. Ultimately, each agent will be tested on the same set of global test data, and will be asked to contribute a vote as to what they believe to be the correct classification of those 200 items. Agents only know data in their individual training set, but should also be able to determine whether a letter they are attempting to classify does not look like any of the letters they know. Therefore agents, acting as independent classifiers, should be able to report:

- Whether a test item is one of their items (Does this look like an item I have in my training dataset?)
- If the test item is one of their items, which item is it? (Which item of mine does this look most like?)

Classification

The general structure of the distributed classification task is illustrated in Figure 20. Agents complete this classification task using a k -nearest neighbors or KNN classification algorithm. This algorithm provides a classification label for unknown test data by identifying the points in the agent's training dataset that are closest in variable-space to the test data. The KNN algorithm uses the nearest k of these points in the agent's training dataset, and takes the majority label corresponding to those k points as the classification guess for unknown test data. For the selected letter recognition task, we find that $k = 1$ provides the best results over other choices of k with any amount of training data. This is consistent with

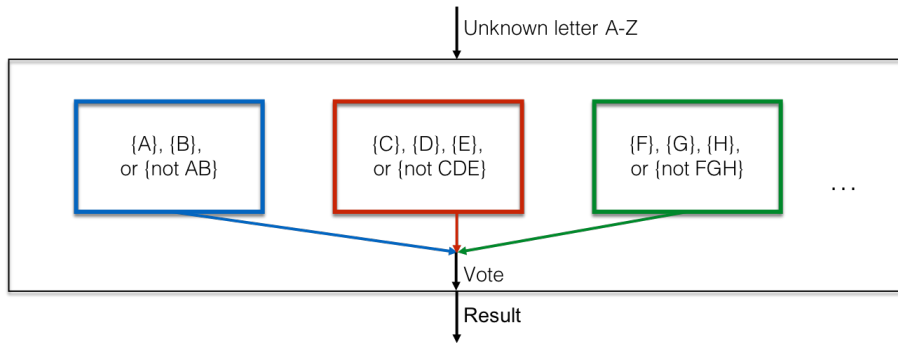


Figure 20: Diagram illustrating distributed classification task as performed by agents. Each box represents a community, and letters in each box represent unique items in the community’s initial test dataset. Letters shown are examples; community test data varies with the community size.

others’ findings in literature (Fogarty, 1992). Therefore all agents use $k = 1$ for their KNN classifier. This classification algorithm allows agents to answer the second question above: given an unknown test item, what item is it?

Confidence Threshold

To answer the first question above, whether an unknown item in their expertise area, agents apply their learned knowledge of classification successes and failures. Agents use a *confidence threshold* that indicates whether the agent trusts the result of their classification. The threshold is based on the Euclidean distance between a test item and an agents’ closest training datapoint, i.e., the result of KNN classification. We observed that there is often a distinct separation between the typical distance between test and train data for correct and incorrect classifications.

The threshold value is calculated from the distribution of distances between test and train items for an agent’s correct and incorrect classifications. The separation between the means and medians of these distributions are distinct, but the distributions themselves overlap. Three thresholds are defined within the model:

- Most incorrect matches above: the 25th percentile of the distribution of distances between test and train data for incorrect matches is used as a threshold

- Average of medians: the average of the medians of the two distributions for incorrect matches and correct matches is used as a threshold
- Most correct matches below: the 75th percentile of the distribution of distances between test and train data for correct matches is used as a threshold

The thresholds act as low-pass filters on the classification result. Items above the threshold are considered untrusted: if the test item is very far away from any of an agent’s known training data, the agent negates the result of that classification. Matches at or below the threshold distance are considered trusted: the classification result is taken as is. The result is a binary filter.

The thresholded accuracy combines the raw classification result and the application of the binary threshold filter. The correct matches below the threshold (close) and the many incorrect matches above the threshold (far) are *positive identifications* and count towards the thresholded accuracy. Incorrect matches below the threshold (close) and correct matches above the threshold (far) are *negative identifications* and count against the thresholded accuracy. The thresholded accuracy is higher than the accuracy without a threshold in most cases. An example of the thresholding process and thresholded accuracy determination is given in Table 12.

Table 12: Example of threshold application and calculation of thresholded accuracy. This agent is tasked with the correct identification of the letter ‘A’.

Classification result	‘A’	‘A’	‘B’	‘B’
Threshold result	Near	Far	Near	Far
Combined result	‘A’	‘not A’	‘B’	‘not B’
Thresholded accuracy	Correct	Incorrect	Incorrect	Correct

Each threshold is a balance between maximizing the number of correct classification results and minimizing the number of incorrect classification results. Because the distributions of distance between test and train data for correct and incorrect matches overlap, no threshold guarantees 100% accuracy. There are both correct matches that happen to be

far apart, and incorrect matches that happen to be very close. This is due in part to the representation of letters using the sixteen dimensions chosen by the creators of the dataset, and the fact that different letters may have similar representations in these dimensions. The agents also use an unweighted Euclidean distance between items as the distance measure in their KNN classifier. It is possible that other distance metrics or weightings may have more favorable results for comparison between letters. It is these inherent features of the classification data and classifier itself, however, that make this a challenging and interesting classification task when a limited amount of training data is available.

Example

The three different thresholds are illustrated for a single agent in Figure 21. First, we look at general trends over the course of twenty model iterations. Each iteration of the model, agents collect new training data to build their classifier. More training data means agents are more likely to have a correct match to the data in the community test pool. Agents also add a portion of newly received data to the common test pool, growing the expertise of the entire community. Thus agents test their classifiers on an increasingly large set of local community test data.

The median distance between test items and training items for correct matches stays roughly constant at around 3, suggesting correct matches stay correct. The median distance between test and train items falls from about 7.5 to 5.5 over the course of 20 model iterations for incorrect matches. This is evidence that agents gain new training data is that is closer to all test items. However, there continue to be incorrect matches.

Next, we step through the classification results for the example agent shown in Figure 21. This agent begins at $T = 0$ with a training dataset consisting of multiple examples of a single letter, A. Their community test dataset has two letters, A and B. Without a threshold, this agent will classify every letter in the test dataset as an A, since that is all they know. This is illustrated in Figure 5.21(a). As the model progresses through 20 iterations this example agent gathers information about seven different letters. The community

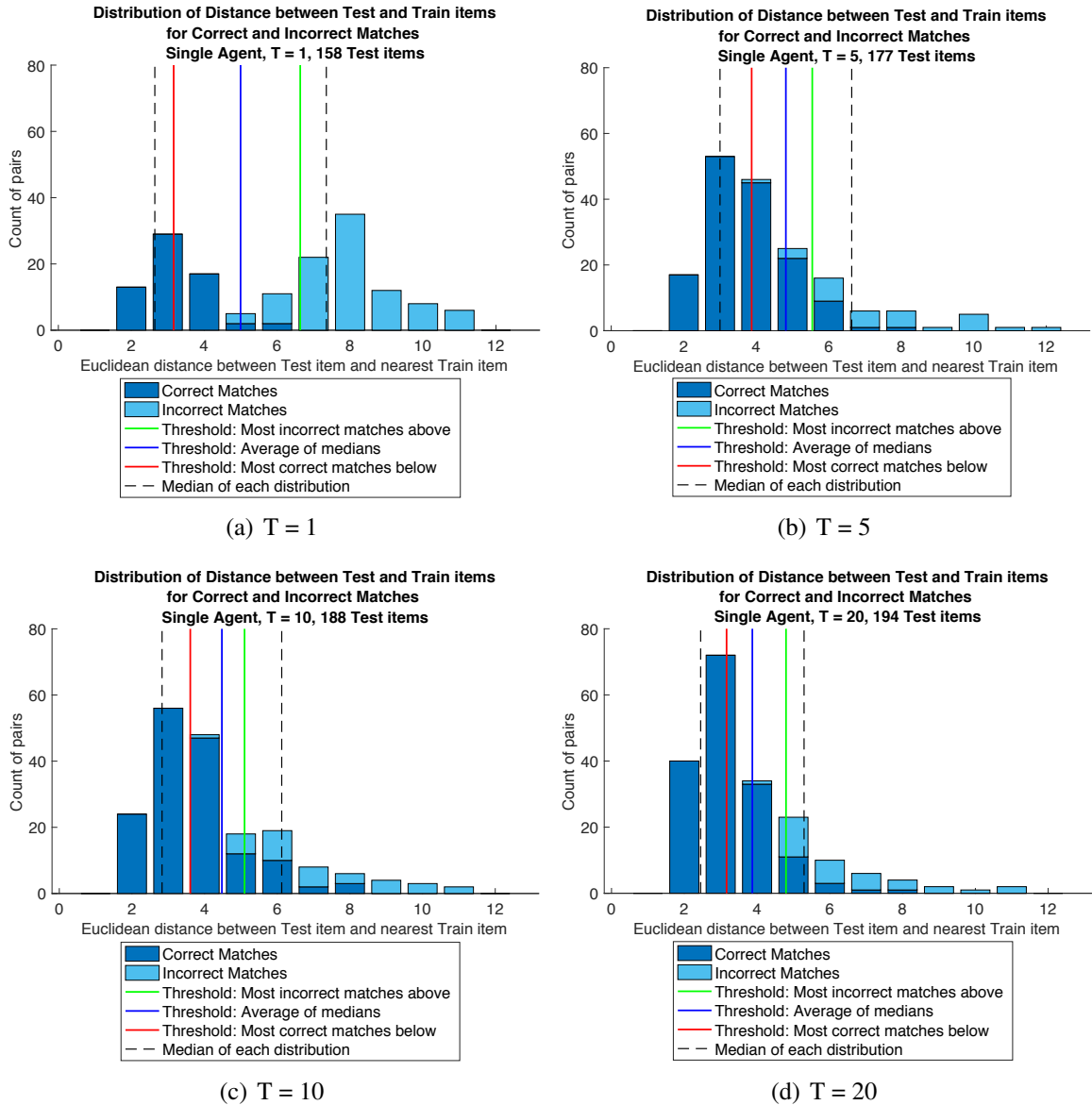


Figure 21: Illustration of overlapping distributions of the distance between test and train items when the test and train items match (correct classification), and when they are different (incorrect classification).

test dataset contains 13 unique items after 20 iterations, so this agent will always make some incorrect classifications: they can still only guess one of the seven labels that are in their own training set. Even so, this agent achieves an accuracy of 83% on their local community test without a threshold applied. With the *most incorrect matches above* threshold (green line in Figure 21) applied, their local task accuracy improves to 92%.

More on how the community test dataset evolves and how agents gain new training data is in the following subsections.

5.2.4 Coordination Problem

In each iteration of the model, agents develop a classification model using their current training data and evaluate it on their local community test data. Additional training data improves the likelihood of having a correct match to a test item. This improvement is in both the classification task accuracy as well as defining an effective confidence threshold. The agents' final objective is accurate contribution to the global classification task through these two tasks: distinguishing whether an item is one they know (have in their training dataset), and correctly distinguishing among labels they have in their dataset. Agents do not receive feedback on their global classification task performance, but they develop their approach through refining their performance on the local classification task.

To improve their classification accuracy on both tasks, agents collect information at each iteration of the model from either a peer or the community ask pool. The act of collecting data therefore serves as coordination: its aim is to improve the agent's contribution to common global task performance. The result of effective data collection is agreement among agents on a correct classification, through positive identification and lack of negative identification. How agents collect data, or their coordination-facilitation behavior, differs by the type of agent, discussed more in the following section.

If each node had all 19,800 training items in their own training dataset, they would have achieved approximately 95% accuracy on the global test. This result is again consistent with other results from the literature using the entire dataset as training data for classification (Fogarty, 1992). However, each agent starts out with just 168 items, 0.8% of the overall dataset. With this small training set, agents aim to collect data efficiently in order to improve their classification task ability.

5.2.5 Agent Behavior and Interaction

Two different types of agents are built into the model, each modeled after the Active and Passive archetypes identified in the studies presented in previous chapters. The agent types behave largely the same way, but have two points of difference governed by different initial parameters: peer agent set definition and probability to interact with peers. A *peer* is an agent considered as a source of information. Passive agents embody elements of the Passive strategy for facilitating coordination identified in Chapter 3. These agents define their peers they can contact as those with whom they have formally-defined ties. In this model, these formal ties are indicated by links in the network structure that connect agents. Passive agents also have a lower probability to interact with peers in order to seek new information, instead preferring to gather new information from the community's ask pool. Receiving data from the ask pool represents independently coming up with new information, rather than seeking similar information from peers.

Active agents in turn embody elements of the Active strategy for facilitating coordination we identified previously. Active agents define their peers as both those they are formally connected to (network links) and "informal" connections: all other members of their community, regardless of whether there is a network link to those members. This difference means Active agents have more choices of peer agents to seek data from as compared to Passive agents. This difference means there is more information readily accessible to Active agents, increasing their ability to act as a source of information for their neighboring agents. Active agents also have a preference for seeking new information from peers rather than generating new data alone, i.e., seeking data from the local community ask pool.

5.2.5.1 Interaction Probability

The choice of data source is governed by a probability called p_{interact} : the probability an agent chooses to interact with a peer. The probability of instead selecting data from the ask pool is $p_{\text{ask}} = 1 - p_{\text{interact}}$. The initial values of p_{interact} vary by agent type as given in

Equations 5.3 and 5.4.

$$\text{Active: } p_{\text{interact}} = 1 \quad (5.3)$$

$$\text{Passive: } p_{\text{interact}} = 0 \quad (5.4)$$

The initial interaction probabilities represent our depiction of archetypical behavior. With the interaction probabilities set to 1 and 0 respectively, an Active agent will initially have the preference to always interact with peers, and a Passive agent will initially prefer to never interact with peers. The interaction probabilities can change, however, as agents engage in successful and unsuccessful interactions. This learning process is described more below.

If an agent chooses to interact with a peer agent in a given iteration of the model, they must also choose which peer to interact with. The choice is governed by the *connection strength*. The connection strength is a directed relationship between two agents, which we denote *self* and *peer*. The strength itself represents the value of a given *peer* as a source of information for the *self* agent.

A single agent has multiple *peers*, as described previously: an agent's peer set include all other agents connected by a network edge. Active agents also include their community members within their set of peers. A *self* agent determines the probability of selecting each of their peers by comparing the connection strength values for each (*self*, *peer*) combination. The raw connection strength values from each pair of agents are normalized across all of the *self* agent's peers: the normalized connection strength values sum to one. The normalized connection strength for a pair of agents is then used as the probability the *self* agent interacts with a given *peer* agent. If all (raw or normalized) connection strength values are equal, the *self* agent is equally likely to interact with any of their peers.

The raw connection strength values start at 1, and are updated after each interaction with a peer agent. A successful interaction, determined by improved classifier accuracy with

the received information, increases the raw connection strength (self, peer) by a parameter p_{change} , initially set to 0.5. The raw connection strength values are restricted to be non-negative so that probabilities can be calculated, so if an entry falls below zero through this process it is reset to zero.

The same updating logic also applies to the value of p_{interact} : if classification accuracy is reduced as a result of receiving data from any source, then the probability to choose that same source again is decreased by the same amount p_{change} . If an agent chooses to interact with a peer rather than the ask pool, and it was successful, they are then more likely by an increment of p_{change} to seek a peer again rather than the ask pool at the next model iteration. Similarly, if an agent chooses to select data from the ask pool and it was successful, the value of p_{interact} is decreased by an increment p_{change} . This increases the likelihood of selecting from the ask pool again. Updating interaction probabilities as the model progresses represents agents' learning about the value of different information sources, and adapting future behavior accordingly.

5.2.5.2 Information Exchange

After an agent has selected an information source, they then receive information from that source in a one-way exchange. Two parameters, *interact_add* and *ask_add*, govern how much data an agent receives from a peer or their community's local ask pool, respectively. Initially, *interact_add* = 10 items and *ask_add* = 2 items. A difference between these values can be interpreted as the relative effort required to get information from each source. The given values indicate that asking a peer is five times more efficient to receive new information than working on one's own to come up with new information. Using other ratios are also plausible scenarios. There is a practical threshold for the *ask_add* value, however, constrained by the finite amount of data available in the ask pool. The utility of the ask pool as a resource is diminished if agents receive all data available there in one or two iterations. The *interact_add* value is constrained only by the amount of data in each

agent’s training set, initially 168 items.

Agents copy data from their chosen source (peer agent or ask pool) randomly without replacement. When selecting from the ask pool, agents track which items they have taken so they can select (randomly) from the remaining set. However, when selecting *interact_add* items from a peer’s current training dataset, there is no guarantee that the items selected are not duplicates. Agents may also receive duplicate entries if data is transmitted among multiple paths to the same recipient agent. There are also 1,315 (6.6%) duplicate entries in the 19,800 items partitioned to agents’ training data, community ask pools, and community test datasets. These are artifacts from compressing character images into sixteen measurements. Whether receiving a new data item that happens to be a duplicate entry or receiving the same item twice on separate occasions, agents discard duplicate information.

If any of an agent’s newly collected items are new to the community (not in the community test pool), the agent will transfer some of their newly acquired data to the community test pool. The effect for the community is a growth of expertise, or test data labels they use to refine their classifiers and thresholds. With new test data, community members able to evaluate their ability to correctly classify the new letter(s). This requires the new label be both in the agent’s training data (known to the agent) and in the community test data (unknown to the agent). The mechanisms of exchange and sharing new labels data with the community mean that both the community test data and each agent’s training data grow over subsequent model iterations.

5.2.5.3 Agent Distribution

Finally, two major parameters that govern the model are the proportion of each type of agent (Passive or Active) included in the model and where those agents are located within the organizational network. Varying these parameters allow us to answer our primary research question by comparing the performance of different fractions of agent types. We look at

proportions of Active and Passive agents that range from all Active agents to all Passive agents, and 25% increments between. In this study, agents are randomly located throughout the network. For each network, results are given for the typical behavior of agents in a single network location.

5.2.6 Summary: Model Parameters

There are three levels of hierarchy in this model: the agents themselves, the communities of agents, and the global set of all agents. Table 13 summarizes all parameters that impact the model at all three levels. All parameters listed are described in previous sections; the corresponding sections are indicated in the table.

Table 13: Local, community, and global parameters that govern agent model

Parameter	Level	Values: Default value listed first
Network Parameters	Global	As given in Table 11, Section 5.2.2
Data Partitioning: Section 5.2.3.1		
Size of Global test	Global	1% (200 items)
Partitioning Order	Global	Alphabetical by Node, Reverse Alphabetical by Node, Random
Community test partition size	Community	10% (60-200 items per community)
Community ask partition size	Community	5% (30-100 items per community)
Agent training data partition size	Community	85% (168 items per node)
Agent Distribution: Section 5.2.5.3		
Ratio of number Active to number Passive agents	Global	0.5, 0, 0.25, 0.75, 1
Location of Active/Passive agents	Global	Random, Highest or lowest centrality network positions
Passive Agent Properties: Section 5.2.5		
Peer definition	Agent	Network links only
Probability to interact with peer over ask, $p_{interact}$	Agent	0, 0.25, 0.3, 0.5

Continued on next page

Table 13 – *Continued from previous page*

Parameter	Level	Values: Default value listed first
Active Agent Properties: Section 5.2.5		
Peer definition	Agent	Network links and community members
Probability to interact with peer over ask, $p_{interact}$	Agent	1, 0.75, 0.7, 0.5
Interaction and Classification Properties: Sections 5.2.5.1, 5.2.3.2		
Items received from peer, interact_add	Global	10
Items received from ask pool, ask_add	Global	2
Number neighbors in KNN classifier	Agent	1
Confidence threshold definition	Agent	Most incorrect above, Most correct below, Average

5.2.7 Model Process

We have alluded to various steps of the model; here we describe the full model execution process. A single run of the model includes multiple iterations. Each iteration, agents collect new data and refine their classifiers. The structure of each iteration is fixed, but how each agent navigates the structure depends on their individual behavioral parameters. A flowchart depicting the structure of each iteration is shown in Figure 22. At every iteration of the model, each agent proceeds through the steps below in sequential order. We divide the process into seven distinct steps:

1. Interaction: Select a peer agent or choose to gather data from the community’s ask pool. This selection is governed by the probability $p_{interact}$.
2. Receive data: Obtain data from selected source. If data (label) is new to the community, half of the new data is added to the community’s test dataset. The remaining data is added to the agent’s own training dataset, excluding any duplicate data.

3. Build a classifier model and test it on the local community test data: Agents use their basic KNN classifier model with $k = 1$ to classify the community test data.
4. Calculate confidence threshold: Agents evaluate their own classification performance on their local community test data, and use the typical distances between test and train data items to develop a confidence threshold.
5. Apply threshold: The confidence threshold is applied to the local community test results, giving a binary classification output. The result for a given item is either a vote of confidence in the classifier output, or a vote of no confidence, which reports 'not' the classifier output.
6. Adaptation: Agents evaluate their classifier performance relative to the previous model iteration. If classifier accuracy increased, then the probability to choose the same source of information again is increased; if the results decreased, then the probability to choose the same source of information again is decreased. Both p_{interact} and the connection strength used to select a peer (if applicable) are adjusted by a fixed probability p_{change} .
7. Aggregation: After all agents have developed their classification model and calculated a confidence threshold, they apply their training data to attempt classification of the global dataset. Each agent gets a vote, and votes are counted as the sum of votes for a given letter plus the number of votes not against a given letter. For example, say, the letter 'T' is the true global test item, and the letters 'B', 'L', and 'T' are offered by the agents as votes with high confidence. The votes for each of 'B', 'L', and 'T' are counted as well as all of the votes of low confidence for other letters. If a vote of low confidence is 'not B', it would count towards a vote for 'L' and for 'T', but not towards a vote for 'B'. The majority vote is selected as the aggregate vote. In the case of ties, the tiebreaker is the number of votes for a letter classification (i.e., a vote with high confidence), and if still tied a winner is picked at random.

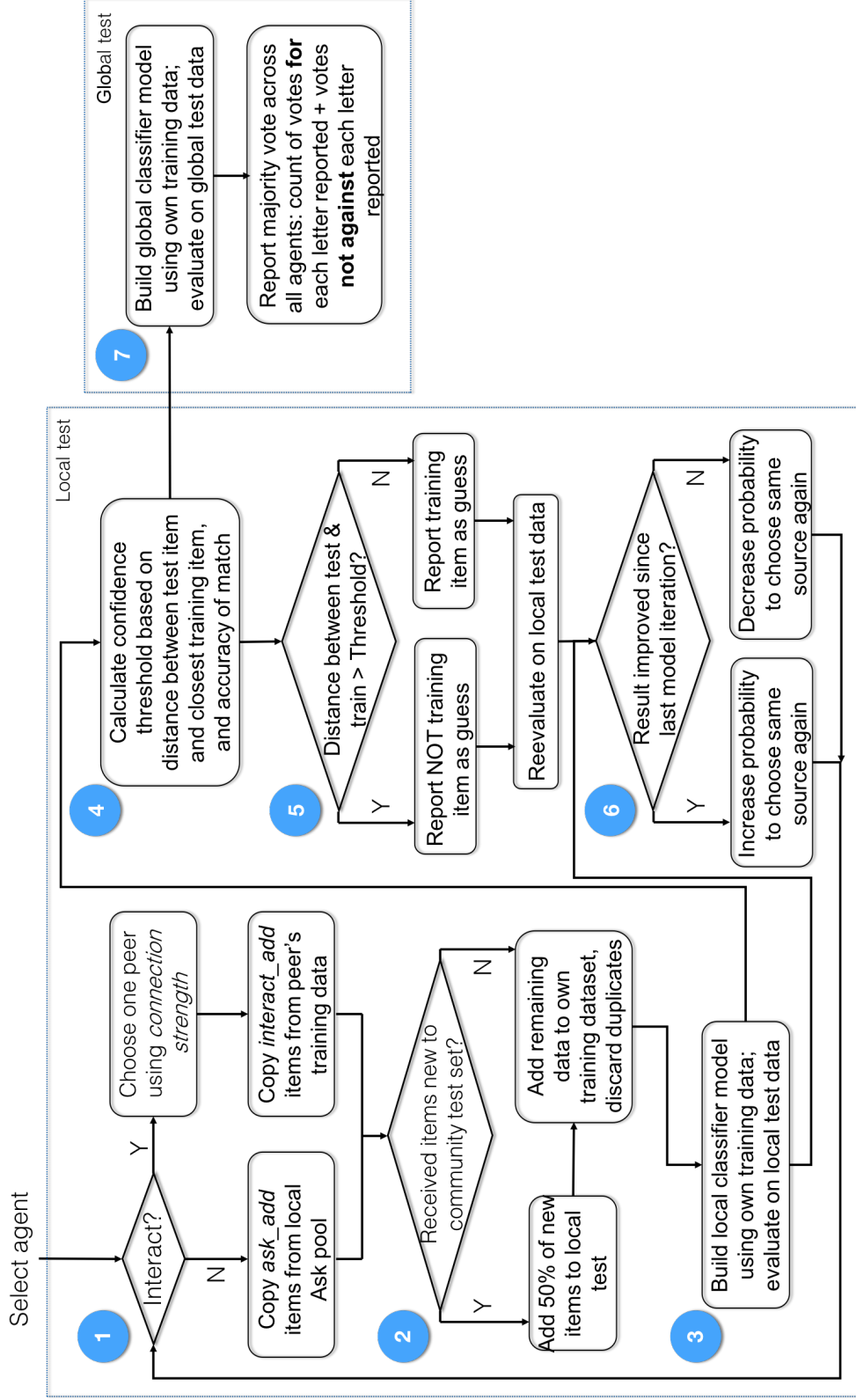


Figure 22: Flowchart of agent model process depicting decisions and actions each iteration of model

Most of the steps in this process are deterministic. The choices that agents make in steps 1 and 2 (Interaction and Receive data), however, are stochastic. The probabilities that govern the stochasticity are updated based on each agent's actions and the results of those actions. These probabilistic choices serve to represent an agent's behavioral tendencies, and the updating of probabilities serves to represent the agents' learning from experience. Adaptation modifies agents' behavioral tendencies.

5.2.8 Model Outputs: Performance and Costs

5.2.8.1 Performance

We measured accuracy of agents' classifier performance after each iteration of the model, on both their local community test as well as the common global test. Agents do not adapt based on the results of the global test; only their results on the local community test are visible to them. This is analogous to actual design tasks, where we can estimate but not always see the projected system-level design. Our model calculates this intermediate performance to visualize the progress of the agents' actions. We calculate from agents' accuracy the average performance within communities, as well as the aggregate performance on the global test through the procedure outlined in Section 3.3 above. For a given set of parameters, the model runs for 20 iterations; one simulation run. Repeated runs with the same parameters shows the typical performance for agents in a given network position, where results are due to the combination of global parameters such as network structure, data partitioning, and agent interaction probabilities.

The global classification task results are aggregated for each iteration of the model, so the results we show are typical trends of the system as a whole across 20 iterations, or from $T = 0$ to $T = 20$. The local classification task results are reported for every agent, but we show results as average performance within a community as community members share the same local classification task. All results presented are the result of Monte Carlo simulation, meaning results are averaged across 100 simulation runs of 20 iterations each.

5.2.8.2 Cost

In addition to looking at the agents' classification accuracy, we also quantified the differences between Active and Passive agents to achieve that performance. At each iteration, each agent in the model has an opportunity to interact with a peer to collect data, or to collect data from their local community ask pool. In the model, the cost of accessing data from each source is implicit through the parameters that govern how much data is received from each source in a single interaction. Any social cost to maintain connections with peers, for example, does not influence agent behavior in this model. However, we recognize that the actions to seek information from a peer and develop one's own information (e.g., collect data from the ask pool) are different. Therefore we report on two measures: the typical number of interactions agents engage in over one 20-iteration simulation, and the typical number of different peers agents interacted with over one 20-iteration simulation. As with the performance measures above, results are given as the result of Monte Carlo simulation: averages across 100 simulation runs of 20 iterations each.

5.3 Hypotheses

We used this agent model to explore two hypotheses. The first is based on the different information-seeking behavior encoded into Active and Passive agent types. Because Passive agents focus on building up information about the data types (letters) initially partitioned to each community, we expect them to outperform Active agents on the local community test. Similarly, because Active agents focus on collecting information from peers, some of whom are external to their own community, they are more likely to gain information about letters outside their community's initially partitioned dataset. Therefore we expect Active agents to outperform Passive agents on the global test. We also expect to see that results depend on the relative concentration of agent types. Specifically, we expect to see a tradeoff between the measures of performance and cost that depends on the

concentration of Passive and Active agents within the network.

To summarize, our hypotheses are:

1. Active agents contribute higher accuracy to the global test than the local test.
2. Passive agents contribute higher accuracy to the local test than the global test.
3. Classification accuracy (performance) and typical number of interactions (cost) of all agents vary with the distribution of agent types within the system.

5.4 Model Behavior

5.4.1 Direct Solution

Before discussing the agents' results on the distributed classification task, we present the all-at-once, or non-distributed (Cramer et al., 1994), results. This test is on uses the same global test data that the agents use: the first 200 items taken from the list of 20,000 test items. Table 14 shows the results of KNN classification using the remaining 19,800 items as training data. A single 1-NN classifier using all 19,800 training items to classify 200 test items achieves 96% accuracy. We also compare this result to the typical accuracy of multiple KNN classifiers. Each individual classifier uses a distinct, randomly selected fraction of the 19,800 items as training data to classify the same 200 test items.

The number of partitions in this test of KNN classifier accuracy represent the number of agents. We chose to use 100 agents in our model, meaning each agent has up to 198 items, or 1 percent of the available training data. With one percent of (randomly allocated) training data, we expect a typical agent to achieve 50% accuracy on the global test. Because the community ask and test data is set aside out of this 198 item allocation, each agent actually starts with 168 training items. We therefore expect performance slightly below 50% for each agent on the global test dataset according to this result. If each agent had all 19,800 training items available, we would expect them to achieve 96% accuracy on the global test.

Table 14: K-NN classification results using fixed set of 200 test items, varying the training data partition size and the value of k nearest neighbors used to determine classification.

partitions	part size	pct. of total	1-NN	2-NN	5-NN	10-NN
1	19,800	100%	0.96	0.93	0.95	0.93
2	9,900	50%	0.95	0.90	0.92	0.90
6	3,300	16.7%	0.87	0.84	0.84	0.81
10	1,980	10%	0.84	0.76	0.78	0.74
20	990	5 %	0.76	0.68	0.67	0.62
60	330	1.7%	0.59	0.50	0.47	0.42
100	198	1%	0.50	0.42	0.38	0.34
200	99	0.5%	0.38	0.32	0.27	0.23
600	33	0.2%	0.23	0.18	0.14	0.11

The classification accuracy for partitions of intermediate sizes follow a roughly linear trend. It is important to note that the directly calculated KNN results shown in Table 14 are the result of random partitions, meaning partitions of any size are likely to include more letters than is achieved from the default alphabetical by node partitioning used in our model.

We compare the initial classification accuracy of model agents with different partitioning schemes in Figure 23. The initial classification results depend only on the partitioning scheme: agents have not yet interacted. Using random partitioning, agents have a typical initial accuracy on the global classification task of 47%, with a standard deviation of 3%. This accuracy is consistent with that predicted above. With the alphabetical or reverse alphabetical partitioning schemes, agents’ initial classification accuracy drops to 5%, with a comparable standard deviation of 2%. The initial classification results for the agents in our model are shown in Figure 23 for both the global classification task and the within-community local classification task. While the alphabetical by node partitioning strategies yield low performance on the global test – as expected due to agents’ few distinct letters – the average performance of these nodes on the local classification task is equal to that of agents that receive randomly partitioned data. As agents interact and refine their classifiers, their accuracy improves: these results are described in the following sections.

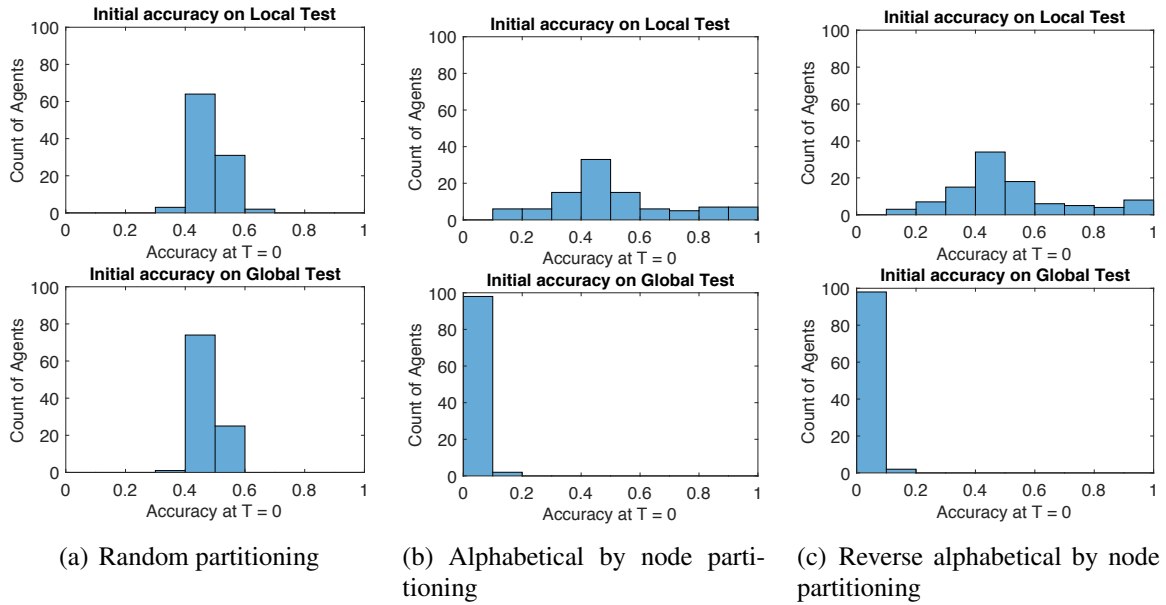


Figure 23: Initial T=0 accuracy results for agents on local community classification task and global classification task. Three training data partitionings are shown: random partitioning, alphabetical by node partitioning, and reverse alphabetical by node partitioning.

5.4.2 Performance

5.4.2.1 Local Classification Accuracy

First, we look at agents' performance on the local classification task. Training data was partitioned to agents alphabetically by node. We look at results across 20 unique networks, each generated with the same parameters. Each network creates a different problem through unique community structure and partitioning. Each community has a different local classification task defined by the data in the community test pool. Despite the differences, we are able to draw some conclusions from the trends across all cases.

We illustrate the typical local classification result using a single network as example. The network has fourteen communities, which are populated with either all Passive agents, or all Active agents. The results of agents' behavior (average result from 100 Monte Carlo simulations), both with and without threshold application, are shown in Figures 24 and 25. The plots show the average percent accuracy within each community on the local

classification task. Accuracy is evaluated after each of twenty model iterations. These results are depicted for a single network, but are illustrative of the typical model results for all twenty networks tested. A summary of results for all twenty networks is in Table 15; this example network is Network 3 in the table.

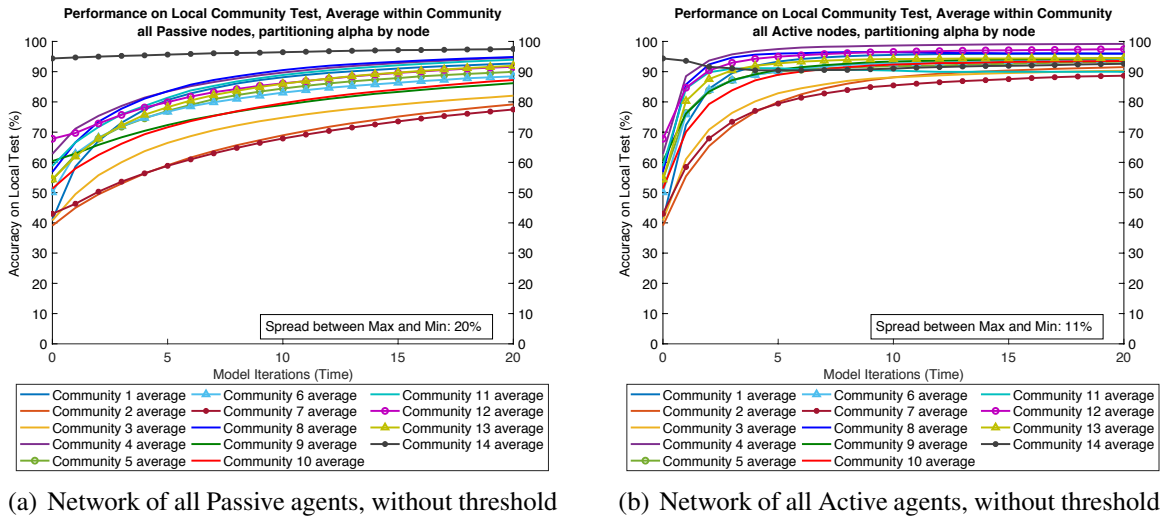


Figure 24: Example local task accuracy, without threshold applied. Results are show for a single network (Network 3 in Table 15), comparing all Passive agents and all Active agents.

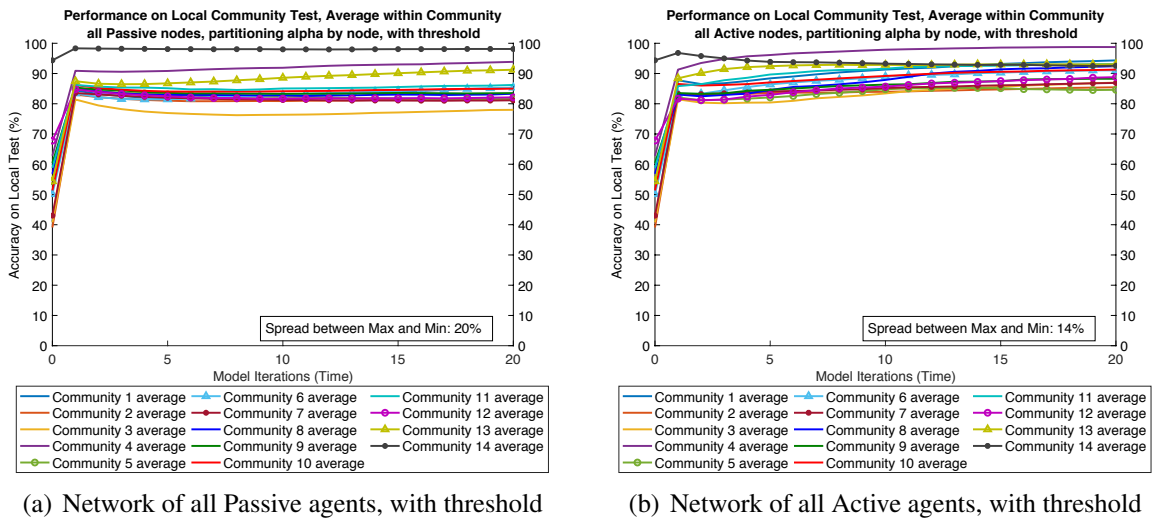


Figure 25: Example local task accuracy, with threshold applied. Results are show for a single network (Network 3 in Table 15), comparing all Passive agents and all Active agents.

First, we compare results without a threshold applied for communities of all Passive and all Active agents, shown in Figure 24. Comparing the two plots, the communities of Passive agents have a slower rate of increase in performance compared to communities of Active agents. The differences are likely due to two factors: The first is that interactions with peers and interactions with the community ask pool yield different amounts of data. The second is that data from the community ask pool is guaranteed to be consistent with the community initial expertise area (partitioned letters), whereas data from peers could be sourced from any peer, including those outside the community. Thus Passive agents, typically querying the ask pool rather than peers, receive less data but it is focused to their classification task. Passive agents, typically querying peers rather than the ask pool, receive more data but its content is higher variance. These results suggest that both quantity and specificity of data lead to improvement. Rapid improvement is gained from quantity, but both data collection strategies yield 80% - 100% accuracy after 20 model iterations.

With the threshold applied, as in Figure 25, all agents are able to apply the threshold to their advantage early on. With the threshold, applied after the first model iteration, the typical classification accuracy in each community jumps up significantly and holds steady for the remaining iterations. With a threshold applied, networks of all Passive agents and all Active agents show approximately equal performance on local classification tasks. The partitioning scheme used here is alphabetical by node, so most agents have only one letter in their initial training dataset. Without a threshold, these agents have no choice but to incorrectly identify any other letters that are in their test dataset. The threshold allows agents to correctly identify that some of their incorrect matches are incorrect, which results in an increased accuracy.

Key characteristics of the plots in Figures 24 and 24 include the highest accuracy achieved by any community, the lowest accuracy achieved by any community, and the spread between highest and lowest performing communities. These three values are tabulated for each of 20 unique networks, with threshold applied, in Table 15. For each network

and corresponding community structure, results are compared for a case where all agents are Passive, and all agents are Active. Our hypothesis is that Passive agents outperform Active agents on the local task due to Passive agents' relative specialization and Active agents' relative generalization. Therefore differences calculated in the table give the accuracy gain or loss for networks of all Passive agents compared to networks of all Active agents.

The trends for thresholded accuracy in Figures 5.25(a) and 5.25(b) appear similar. The values in Table 15 show that neither set of agents consistently outperforms the other:

Table 15: Local classification results for networks of all Passive and all Active agents. Maximum and minimum refer to the highest and lowest within-community average classification accuracy, with threshold applied. Spread refers to the accuracy difference between highest and lowest community average. All values are percentages.

Network	all Passive agents			all Active agents			Difference (Passive - Active)		
	Max	Min	Spread	Max	Min	Spread	Max	Min	Spread
1	95.1	62.1	32.9	99.3	76.7	22.6	-4.2	-14.6	10.3
2	93.3	78.2	15.2	98.2	85.4	12.8	-4.9	-7.2	2.4
3	98.1	78.0	20.1	98.7	84.6	14.2	-0.6	-6.6	5.9
4	100	77.7	22.3	95.1	78.4	16.7	4.9	-0.7	5.6
5	95.8	76.2	19.6	96.2	81.2	15.0	-0.4	-5.0	4.6
6	93.8	76.5	17.3	96.2	82.1	14.1	-2.4	-5.6	3.2
7	93.7	63.2	30.6	97.7	78.5	19.1	-4.0	-15.3	11.4
8	100	76.2	23.8	96.1	81.5	14.6	3.9	-5.3	9.2
9	93.7	75.8	17.9	98.2	82.6	15.6	-4.5	-6.8	2.3
10	100	67.7	32.3	93.3	78.4	14.9	6.7	-10.7	17.4
11	100	74.2	25.8	98.7	80.1	18.6	1.3	-5.9	7.2
12	100	76.5	23.5	99.3	77.4	22	0.7	-0.9	1.5
13	100	74.5	25.4	97.1	75.7	21.3	2.9	-1.2	4.1
14	100	71.9	28.1	99.9	77.2	22.7	0.1	-5.3	5.4
15	95.4	71.0	24.4	95.4	78.0	17.4	0.0	-7.0	7.0
16	100	75.4	24.6	98.3	81.5	16.8	1.7	-6.1	7.8
17	95.9	78.4	17.5	96.6	83.1	13.6	-0.7	-4.7	3.9
18	100	74.9	25.1	84.8	82.7	12.1	5.2	-7.8	13.0
19	100	70.7	29.3	98.0	76.2	21.8	2.0	-5.5	7.5
20	94.3	77.8	16.5	96.5	82.1	14.4	-2.2	-4.3	2.1

the difference in maximum community accuracy is sometimes positive (Passive agents outperform Active) and sometimes negative (Active agents outperform Passive). However, the lowest-performing community of Passive agents is always worse than the lowest-performing community of Active agents. The spread between highest and lowest performing community is also always greater for networks of Passive agents in these 20 cases. Thus networks of all Active agents show more consistent local task performance across all communities. These results do not support our first hypothesis: we are not able to conclusively say that Passive agents outperform Active agents on the local classification task. In terms of highest performance, Passive agents often outperform Active agents. It is also Passive agents that offer some of the lowest local classification accuracies.

Communities across networks differ in terms of their size, members' interconnections and connections to other communities, and the set of letters contained in their respective test datasets. Some communities show particularly divergent results due to the expertise area assigned through partitioning, creating a difficult classification task. In these cases, a community's classification performance decreases each iteration, even with a threshold

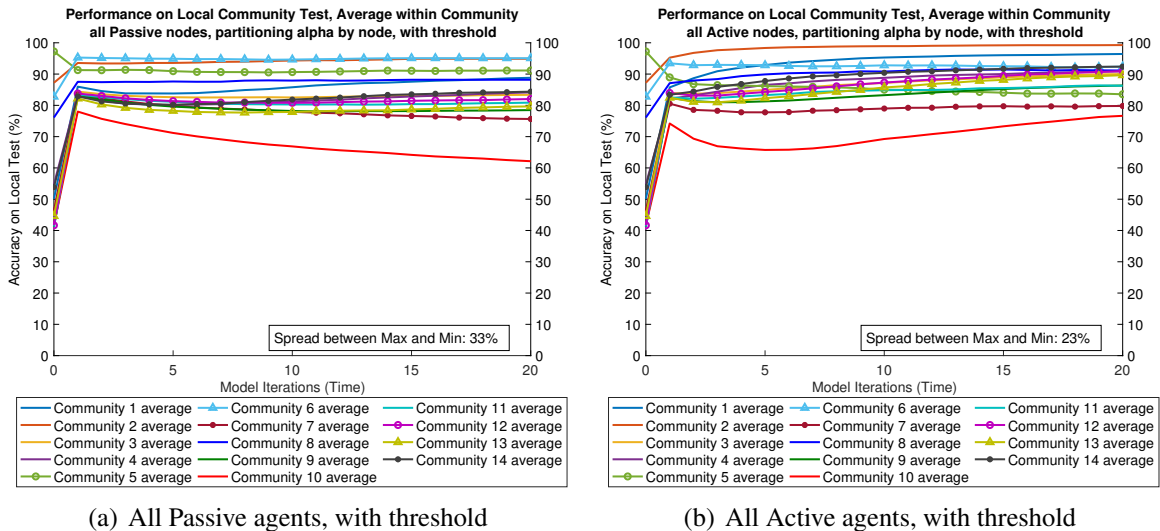


Figure 26: Example of challenging local classification task, highlighting Community 10's (solid red line) atypical performance below all peer communities. Results are shown with a threshold, for all Passive and all Active agents on a single network (Network 1 in Table 15).

applied. When observed, this is particularly for Passive agents. An example is Community 10, shown in red in Figure 26.

Community 10 illustrates a case of an inherently challenging design task. This group of eight agents is tasked with distinguishing the letters 'O', 'P', and 'Q'. For these agents, collecting more data about any of these letters, as Passive agents do by selecting from their local ask pool, increases the frequency of misclassification within their own data. We see this through the Passive agents' decreasing classification accuracy. This includes both 'O' misclassified as 'Q' and vice versa. Because the letters are similar, these misclassifications are not caught by the thresholding. In this case, the agents' preference to pull data from the shared ask pool means all agents end up with similarly confusing training data in their training data that hampers ability to correctly classify their shared test data. The network of all Active agents begins to improve slowly through collection of data directly from other peers' training data, both within and without their community. Community 10 illustrates one example of natural variance we see in our model: some groups of agents, due to their size and location within the network, may receive particularly challenging tasks compared to their peers.

5.4.2.2 Global Classification Accuracy

Next we look at the agents' aggregate accuracy on the global classification task. When tested against the global data, each agent's classifier returns (including thresholding) a vote for a letter, or a vote against a letter. All agents' votes are aggregated according to majority vote: the sum of all votes for each letter plus the number of votes not against that letter. The global accuracy is the percentage of true global test labels that match the majority vote.

The threshold is an important element of aggregation. As with the local classification task, a threshold applied to the global task allows agents to contribute a vote towards a letter not in their training dataset. This is akin to the agent acknowledging the test item is something they do not know enough about in order to make a good judgment, and applying

that uncertainty in the outcome to moderate their classifier’s output. Without the threshold, agents are forced to always contribute a vote for a letter in their training dataset. In aggregation, this means many different letters will receive votes, and a majority may be driven by a larger community’s incorrect classifications rather than a smaller community’s correct classifications. The application of a threshold helps address this inconsistency: agents have some awareness of their expertise areas.

As with the local classification task, we compare the global classification task results for networks of all Passive and all Active agents. An example result for a single network is shown in Figure 27. This is the same network used as example previously, network 3 out of the 20 cases included in analysis.

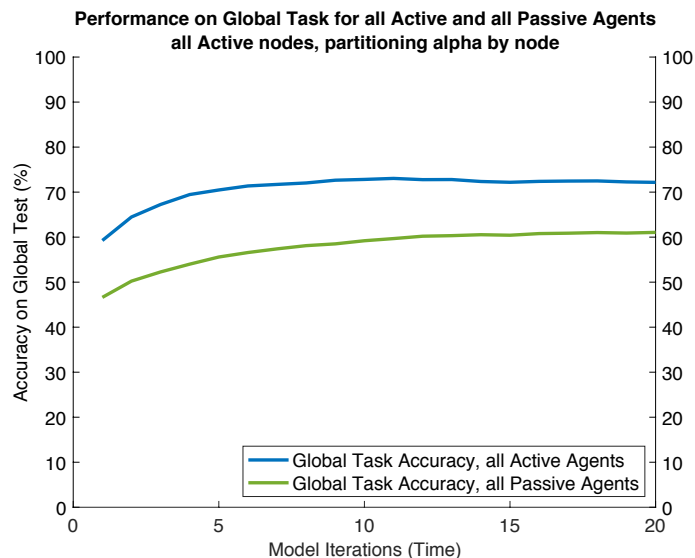


Figure 27: Example global task accuracy, with threshold applied. Results are shown for a single network (Network 3 in Table 16), comparing all Passive agents (lower line) and all Active agents (upper line).

From comparing the results of Passive and Active agents on this network, we observe that the results of all Active and all Passive agents follow a similar trend over the course of twenty model iterations. However, the network of all Active agents (blue; top line) has a higher initial global task accuracy after the first iteration. The Active agents then remain consistently above the Passive agents (green; lower line) given the same task.

The accuracy of all Passive agents and all Active agents increase rapidly at first, and begins to level off after about 3 model iterations. The trend in aggregate accuracy is similar to the local task accuracy, suggesting that as agents improve on their local test, they also improve on the global test. A group's expertise is determined by the letters they have in their community test dataset; the size of the expertise area is proportional to the group size. The local task results indicate agents quickly become proficient in their group's expertise area.

The observed flat curve may be result of the aggregation procedure. aggregating votes for multiple letters that each have similar numbers of votes. As agents improve their classifiers, there may be some fluctuation in support for the competing choices. This may not translate to significant increases in accuracy overall.

The leveling-off of the aggregate result may also be a result of the thresholding process and unequal community sizes. A test item that is in the expertise of a group should be identified correctly with high confidence by these group members. Thresholding helps keep larger groups with incorrect answers from swamping smaller groups. However, a test item could still be similar enough to an element of a larger group's training data that the threshold does not exclude their mistaken classifications. The large community would then still drown out the correct signal coming from small groups. This is one potential downside of the deterministic thresholding process implemented in the model, rather than one that may take into account specific areas of weakness.

The trend of the global classification accuracy curve with respect to increasing model iterations is similar for all networks tested. The final global classification task accuracy is compared for all twenty networks in Table 16. Our second hypothesis is that Active agents outperform Passive agents on the global task. This is expected due to Active agents' interaction actions giving them access to a wider variety of information than Passive agents, thus making them generalists. We expect generalists to contribute more correct classifications to the global task, which includes letters from the entire alphabet. Table 16 shows that

networks of Active agents achieve roughly 60-75% accuracy on the global task. Passive agents are also expected to do well on this task, since the threshold allows correct contributions to the global aggregation even with a narrow specialization. Networks of Passive agents achieve roughly 50-65% accuracy after 20 model iterations. The ranges of accuracy overlap for Active and Passive agents, however we see that the difference between the two results ranges from nearly zero to just over 18 percentage points. In all but the near-zero case, the delta is in favor of the Active agents, lending support for our second hypothesis.

Table 16: Global classification results for networks of all Active and all Passive agents. Results are the aggregate global accuracy after twenty model iterations, with threshold applied. All values are percentages.

Network	all Active	all Passive	Difference (Active - Passive)
1	67.5	65.8	1.7
2	63.2	63.3	-0.1
3	72.2	61.1	11.1
4	69.2	55.1	14.1
5	69.6	59.1	10.5
6	72.5	66.7	5.8
7	68.3	61.0	7.3
8	75.9	57.7	18.2
9	68.7	64.7	4.0
10	74.9	59.3	15.6
11	68.9	54.4	14.5
12	59.8	52.4	7.4
13	67.1	55.9	11.2
14	61.0	48.0	13.0
15	73.1	65.3	7.8
16	70.4	57.4	13.0
17	67.1	60.1	7.0
18	70.5	55.9	14.6
19	68.7	53.5	15.2
20	63.8	58.0	5.8

5.4.3 Active vs. Passive Agent Accuracy

From the above results, we see some evidence for the hypothesis that Active and Passive agents contribute differently to local and global classification accuracy. The local classification results indicate that communities of all Passive and all Active agents can achieve the same performance on their task, and neither agent type outperforms the other consistently. The lowest performing Passive communities are always worse than the lowest performing Active communities, indicating that there is more variance among communities of Passive agents. We also saw that Passive agents are particularly negatively affected by challenging classification tasks (i.e., distinguishing very similar letters). There are differences in how Active and Passive agents contribute to the local classification task; mostly in the variance among communities in a given network. However, this does not support our hypothesis that Passive agents outperform Active agents on the local task.

The global classification results do show evidence that Active agents outperform Passive agents. In all twenty networks we compared, the final aggregate accuracy of Active agents reached a global task accuracy between 0 and 20 percent greater than Passive agents. This lends support for the hypothesis that Active agents outperform Passive agents on the global task.

Finally, we saw that thresholding typically improves classification accuracy. Thresholding may also contribute to the appearance that agents cannot improve the global classification result further after the first few model iterations. This indicates examining the result of thresholding by Active and Passive agents is worth exploring further.

5.4.4 Cost

Having seen how the different combinations of agents perform, we also compare the costs we introduced in Section 5.2.8. The typical number of interactions each agent engages in throughout 20 model iterations differs significantly between networks of all Passive and all Active agents. Agents have a different likelihood to interact with peers, but this can

be updated if agents find that information received decreases their classification accuracy. Across the twenty networks considered, Active agents interact with peers 18 times out of 20 iterations on average. Passive agents on average interact with peers 0 times out of 20. While agents can update their interaction probability, it is not updated enough to shift behavior of the agents from the initial probabilities. Agents look for any improvement in results to continue selecting data from the same source (peer or ask pool), which means that both sources of data are valuable for improving classification accuracy on the local task.

We see that Active agents can achieve a similar local task accuracy as Passive agents, though they expend more resources (interactions) to do so. From the global task results, we see that the additional interactions yield higher accuracy. We explore further this tradeoff between classification accuracy and number of agent interactions in the following section through parametric analysis.

5.5 Parametric Analysis: Agent Concentration

The focus of parametric analysis is varying the relative concentration of each agent type. Whereas in the previous section results were presented for networks of agents all the same type, here we look at networks of mixed agent types. We look at the intermediate ratios of Passive to Active agents of 25/75, 50/50, and 75/25 to complement the all Passive and all Active cases presented above.

We varied the relative proportion of each Agent type within each of the twenty networks we have been working with to this point. Agent types are randomly distributed throughout the network; this distribution is held constant through Monte Carlo simulations. Results for each network are therefore the result of an agent's typical behavior in a fixed network location.

The local task accuracy for each combination is interpolates between trend seen for all Passive and all Active agents. The slope of local task accuracy with respect to model it-

erations is steeper the higher concentration of Active agents within the network. The final accuracy after 20 iterations also interpolates between the two extremes of agent composition. While the maximum within-community average accuracy remains roughly constant, the minimum within-community average increases with the concentration of Active agents.

For the global classification task, both performance and costs are reported for each combination of agents. The global classification accuracy for each combination of Passive and Active agents is given in Table 17. The average number of interactions across all agents in the network is given in Table 18.

Table 17: Global classification task (aggregate) accuracy, reported for varied combinations of Passive and Active agents. All values are percentages.

Network	all Passive	75% P 25% A	50% P 50% A	25% P 75% A	all Active	Maximum value at:	Minimum value at:
1	65.8	68.0	69.5	67.5	67.5	50P/50A	all Passive
2	63.3	65.4	64.9	63.7	63.2	75P/25A	all Active
3	61.1	66.5	66.8	68.7	72.2	all Active	all Passive
4	55.1	60.7	62.6	62.2	69.2	all Active	all Passive
5	59.1	63.2	62.5	64.1	69.6	all Active	all Passive
6	66.7	69.0	71.3	70.8	72.5	all Active	all Passive
7	61.0	59.1	61.2	58.2	68.3	all Active	25P/75A
8	57.7	72.0	67.7	73.6	75.9	all Active	all Passive
9	64.7	67.3	70.2	71.2	68.7	25P/75A	all Passive
10	59.3	62.4	70.4	68.9	74.9	all Active	all Passive
11	54.4	58.9	60.6	67.7	68.9	all Active	all Passive
12	52.4	61.0	60.0	65.6	59.9	25P/75A	all Passive
13	55.9	61.0	66.7	73.1	67.1	25P/75A	all Passive
14	48.0	55.5	57.4	62.6	61.0	25P/75A	all Passive
15	65.3	67.0	71.1	71.1	73.1	all Active	all Passive
16	57.4	65.0	65.2	69.6	70.4	all Active	all Passive
17	60.1	63.0	62.4	63.9	67.1	all Active	all Passive
18	55.9	61.1	60.5	68.2	70.5	25P/75A	all Passive
19	53.5	57.4	65.2	67.4	68.7	all Active	all Passive
20	58.0	59.1	59.0	60.0	63.8	all Active	all Passive

Table 18: Average number of interactions across all agents, reported for varied combinations of Passive and Active agents.

Network	all Passive	75% P 25% A	50% P 50% A	25% P 75% A	all Active
1	0.0	5.1	10.1	14.7	18.6
2	0.2	6.2	9.7	13.1	17.9
3	0.3	5.2	10.1	13.5	18.3
4	0.0	5.8	9.9	14.0	17.9
5	0.0	5.4	9.7	14.3	17.1
6	0.2	5.8	9.4	13.9	18.3
7	0.0	5.1	9.2	14.2	18.3
8	0.2	5.2	9.6	13.8	17.7
9	0.4	5.3	10.0	13.7	17.9
10	0.5	5.7	10.0	13.7	18.2
11	0.2	5.4	9.4	13.3	17.7
12	0.3	6.5	9.7	14.4	16.8
13	0.1	5.6	10.4	13.7	16.5
14	0.2	6.3	9.5	13.5	17.4
15	0.3	5.3	10.4	13.8	18.6
16	0.2	5.8	9.0	14.4	17.3
17	0.0	5.0	9.8	14.6	18.0
18	0.0	6.0	10.3	14.1	17.8
19	0.0	5.2	9.3	14.4	18.4
20	0.1	5.8	10.2	15.1	17.0

First, the typical number of agent interactions increases linearly with the fraction of Active agents within the network. In a network of all Active agents, the average agent interacts with peers 18 iterations out of 20. With 75% Active agents throughout the network, the typical interactions drop to 14 iterations out of 20. Half Passive and half Active yields 10 interactions per agent on average, and 75% Passive agents yield 6 interactions per agent on average. A network of all Passive agents gives on average no interactions throughout the twenty model iterations. The results are consistent with Passive and Active agent behavior remaining constant regardless of the agent type concentration within the network: Active agents interact nearly every model iteration, and Passive agents interact nearly none. This

means that the majority of interaction costs attributed to the entire network can be attributed to Active agents alone.

Next we consider the global task performance. There is not one combination of Passive and Active agents that consistently achieves the highest performance across all networks. In thirteen of the 20 networks studied, the combination of all Active agents achieves the highest global task accuracy. In the remaining networks, the highest-performing combination is an intermediate mix of Passive and Active agents.

A network of all Passive agents typically does not outperform other combinations of agents for any network. Rather, Passive agents achieve the lowest global task accuracy across all agent combinations 18 times out of 20. For the remaining two networks, all Active agents and 75% Active agents are the lowest-performing agent combinations.

Taking the interactions costs and global task performance together, we can ask how the two objectives trade off. Thus far, results suggest that more Active agents increases both global task performance and the interaction costs for the network. This simple linear trend is observed for 13 (65%) of networks. A linear trend indicates that higher performance is simply the result of more interaction among peers.

The remaining seven networks (35%) show a quadratic trend, with a maximum for one of the intermediate combinations of Passive and Active agents. A concave-down trend gives a different interpretation: that more interactions have diminishing returns, and do not always yield higher performance. From the networks sampled, we do not have enough information to say which trend is correct. However, we can say that a trend exists, giving evidence in support of our third hypothesis.

Though the two trends observed give different insights, there is a consistent finding: in all twenty cases there is an indication that Active agent behavior is beneficial. A combination of Active and Passive agents, or all Active agents, always outperforms a network of all Passive agents.

5.6 Discussion

The agent model is based on studies in previous chapters. We hypothesized from Studies 1 and 2 that Active and Passive coordination strategies are complementary. The Passive strategy is enabled by the use of authority, and actions and behaviors map to formal organizational processes. Passive agent behaviors include agents' peer set definition restricted to network links only, and a preference to work independently. The Active strategy is enabled by the use of empathetic leadership, and actions and behaviors map to informal organizational processes. In particular the Active strategy emphasizes proactive communication with peers. Active agent behaviors include a peer set definition of all network links and all community members, and a preference to seek information from peers rather than work independently.

In the previous sections, we explored the results from the model including simple variation of one parameter. The results from this parametric study are partially conclusive. We found that Active and Passive agents achieve similar accuracy on local classification tasks. Active agents also typically outperform Passive agents by a clear margin on the global classification task. We find that the application of a threshold increases task accuracy for all agents, though it may benefit Passive agents more by moderating the drawbacks of slower information acquisition. Finally, we see evidence of a trend between the costs expended by Active agents over Passive agents, i.e., the number of peer interactions Active agents typically engage in, and the accuracy of the collection of agents on the global classification task. Passive agents engage in the fewest interactions, therefore minimize the interaction costs. However, a mix of Active and Passive agents or a set of all Active agents always outperforms a set of all Passive agents on the global classification task.

Our analysis identified both a linear trend, which suggests more interactions always yield higher performance, and a quadratic trend, which indicates that interactions have a diminishing return. Either trend offered by our analysis suggests that some number of Active agents is better than none for improved performance. Whether an intermediate number

of Active agents (balanced by a corresponding number of Passive agents) is better or worse than a set of all Active agents is however unclear. Two major factors that likely influence this result are the network structure, which we have seen can have a significant impact on task accuracy, and the location of agent types within the network. The classification problem agents are tasked with is a result of partitioning. How the solution is reached and the solution quality are impacted by each agent's behavior and the mix of agent types within each community.

We have seen in our results that specific combinations of network structure, community size, and partitioning order can negatively impact classification ability. The example we highlighted previously is a community that is tasked with classifying the similarly-shaped letters of 'O', 'P', and 'Q' – and the fact that this is the community's task is an artifact of the network structure and the size of communities in that network. Thus the findings presented here would be augmented by testing additional networks, and networks with different structures. The LFR generative model used in this study creates scale-free networks (at large scale) like those common in natural systems. Scale-free networks tend to have large hubs, or nodes of high degree. High degree has been found to correlate with increased failures (Zimmermann and Nagappan, 2008). While there is currently no penalty for agents engaging in multiple interactions, errors are a practical consideration for such hub agents. Network structures that follow non-power law degree distributions have been found to be particularly robust to defects: one example is a bimodal degree distribution (Mirzakhali et al., 2017). This is just one example of alternative network structures worth testing in this model.

The results presented here are intended to be a proof of concept. We demonstrated the utility of this agent-based model platform by quantifying the performance outcomes of agents' coordination behaviors. We have shown here just one parametric test that is possible to perform: the agent type distribution. Beyond the network structure, the location of different agent types is another parameter that may be particularly impactful, especially

in combination with the network structure and resultant problem partitioning. For example, agents in network positions of high centrality have access to more diverse information, and thus may particularly influence their community result. This centrality effect should benefit both Passive and Active agents, though perhaps Active agents are better able to reap the benefits of such a position. Similarly, Passive agents may be at a disadvantage in positions of low centrality. A next test for this model is to quantify the impacts of placing Passive and Active agents in positions of high and low network centrality. Beyond network structure, agent type, and agent position, each of the parameters in Table 13 would also be suitable for additional analysis. The number of parameters is a limitation of the model, and the value of future parametric analysis is to discern effects of individual parameters on the agent model outcomes.

Finally, while the structure and inputs for the model we present here are based on our previous results given in Chapters 3 and 4, we acknowledge that the output of the model is not validated. We provide a description of the data to be collected to conduct such a validation study in the following section.

5.7 Validation plan

We describe in this section the data we believe necessary to plan a model validation study.

One of the major parameters of this model's operation is the network structure, which dictates community size, an agent's connections within their team and with other teams, and the size of the initial problem the team receives. The network as we define it is based on formal organizational links between individuals. This is obtainable from an organizational chart. While our network appears to be flat, we intend that its structure can include individuals from multiple levels of hierarchy, and we believe it is important to include both management and engineering or design personnel in such a network structure. This is similar in approach to other studies that have examined the structure of collaboration networks

within organizations, including a variety of different positions within the same network (Parraguez, Eppinger, and Maier, 2016).

Next, we define our agents as Active or Passive, distinguished by their likelihood to seek information from peers versus finding it themselves, as well as who of their connections they would consider as a source of information. These are extreme archetypes. It is possible however to conduct a survey of individuals to assess whether their behaviors are more in line with one archetype or the other. Questions might include asking who they are likely to seek information from within their organization, and their preferences for asking for help from peers versus working problems out on their own. Similar approaches have been used to build information networks of organizations (Levine and Prietula, 2011), models of social exchange (Agneessens and Wittek, 2012), and models of inter-office behavior (Langevin, Wen, and Gurian, 2015).

Finally we suggest ways to measure task performance and costs. Counting interactions between individuals is not an easy task as there are multiple ways to count interactions: physical movement, individuals working on a common file, and common meeting attendance are examples. Examples of methods to collect interaction data are those used to calculate socio-technical congruence, as by Cataldo et al. (2006).

Overall system performance is not usually measurable as a project progresses. Not only is it difficult to enumerate all interactions between subsystems, the design itself evolves throughout the design process (Bloebaum and McGowan, 2012). However, estimates of individual or group performance as well as their expected contribution to a system aggregate may be obtainable from project management personnel. The type of performance we measure is simple task performance, and our system performance is a relatively simple aggregation of individual performance on a common task. Validation data obtained from an organization should be commensurate with the type of work the organization is engaged in.

5.8 Summary

In this chapter, we described and demonstrated an agent-based model that simulates the completion of a distributed design task. We showed the utility of this model as a platform for conducting a contingency analysis of coordination-facilitation behaviors, where results are a function of network structure, agent behaviors, and the distribution of agents with differing behaviors throughout the network. We demonstrated that there is a correlation between the aggregate performance of agents and the behaviors those agents embody. However we noted there are limitations to this study, particularly the limited parametric analysis and the lack of deeper model validation. We have provided recommendations for these tasks.

There are several next steps to extend this work. First is to consider more realistic agent behavior: this includes less than perfect execution of coordination behaviors or custom strategies that incorporate elements of both Active and Passive strategies. Another area is to focus on the interaction exchanges as transactions. The model presented here includes only one-way exchanges, where the agent seeking information is always able to request and receive something from their selected source. This is not necessarily realistic: often transactions also include information flowing the other direction, creating reciprocity. This could be a basis for developing the concept of social capital within the model. Another aspect is that agents always receive exact information, though it may be duplicate or not useful to them. Miscommunication is possible, which includes uncertainty or ambiguity in an agent's interpretation of received data, as well as the errors transmitted by the source agent (Meluso and Austin-Breneman, 2018). Third, the agent model developed as part of Study 3 is based primarily on a network-based system of exchange. Other models exist, including markets (based on incentives and rewards) and hierarchies (based on authority and control) (Powell, 1990; Jung and Lake, 2011). The structures are complementary; incentives and rewards can influence behavior and for this model, shape perceptions of what information is more and less valuable. Authority is central to the Passive coordination strat-

egy we identified, and explicit mechanisms of hierarchy would be a logical next inclusion to improve the realism of the model. Finally, the distributed classification given to agents in this model is just one example problem, and similar models may be created around other problem types. Different tasks also permit exploring the impact of problem size on results.

This agent model complements the exploratory studies presented in the previous chapters. We are able to translate the qualitative and quantitative findings from study of managers, engineers, and designers working on distributed design tasks into the development of an agent-based model. This model has yielded interesting preliminary results and is a promising platform for extensive parametric analysis.

CHAPTER 6

Conclusions

6.1 Review of Dissertation

In this dissertation, we introduced the problem of coordinating distributed design work effectively to achieve a desired system-level outcome. While the literature on coordination and effective approaches to facilitate coordination emphasizes top-down standardized approaches, in this dissertation we emphasized the importance of considering individual actions and behaviors as impacting coordination.

In Chapter 3 we presented a qualitative study of engineering and management personnel, experts in their respective roles and all with extensive experience working on large-scale decomposition-based design work. We interviewed twenty individuals, and thematic analysis of their responses yielded our understanding of coordination strategies and behaviors used in industry practice. We identified two strategies, based on either authority, permitting top-down enforcement of procedures and plans, or empathetic leadership, used to develop tailored interactions with peers throughout the organization and facilitate a common understanding of everyone's contribution to the overall goal. We named these archetypical strategies respectively *Passive* and *Active*, referring to the relative proactivity required – not the amount of work required – for either strategy.

In Chapter 4 we presented a quantitative study of novice designers working on small but complex distributed design task. This study used self-report surveys to identify coordina-

tion patterns through communication and task assignment within teams of five. The survey responses were analyzed using text analysis, network analysis, and clustering analysis. The result was identification of coordination roles adopted by members of the teams. Each role is a combination of typical tasks as well as typical communication patterns. We found some evidence of both nonhierarchical and hierarchical team coordination approaches, as well as a mix of more communicative and less communicative team members within each team. Drawing a parallel to our finding of Active (more communicative) and Passive (less communicative) coordination methods from Chapter 3, these results suggest teams include a balance of each archetype.

In Chapter 5, we presented an agent-based model built on the findings from previous chapters. This model allowed us to explore the balance of Active and Passive coordination-facilitation actions and behaviors within an organization. Organization structure is represented by a network, nodes representing individuals and edges formal links between individuals. The networks we used as input were randomly generated and feature inherent community or team structure. We learned from our analysis of the simulation's output that while Active and Passive agents perform similarly on local, team-level tasks, there is a difference in their contribution to a common system-level task. Although we were able to show that there is a dependence on global task performance with respect to the distribution of Active and Passive agent behaviors, we have not conclusively found whether a mix of Active and Passive behaviors (requiring fewer peer interactions per agent) performs better than solely agents with Active behaviors (requiring the maximum peer interactions per agent). Further analysis on additional network structures is needed to determine which mix of behaviors yields a desirable balance between task performance and interaction costs.

6.2 Dissertation Contributions

We found evidence from each of our studies that some mix of Active and Passive behaviors is beneficial for system-level coordination of design tasks. This is an important finding as it shifts the discussion of effective coordination methods from looking only at top-down prescriptive methods to studying the bottom-up effects of different individual human behaviors in communication and interaction during the system design process. These findings also emphasize the importance of individuals who embody elements of the Active archetype, methods of coordination facilitation that are based on empathetic leadership. Empathy is recognized as important by practitioners, but there is little literature to support the concept and a quantitative understanding of its value in systems design practice.

In summary, as noted in Chapter 1, the main contributions of this dissertation are:

1. Contribution to the development of systems engineering theory and best practice by identifying theory relevant to coordination practice in multiple disciplines.
2. Development of quantitative approaches to describe and evaluate coordination practice, including identification of coordination roles within teams and a simulation model to evaluate the impact of communication behavior on design task performance.
3. Inclusion of coordinator and systems engineering roles and behavior in models of coordination effectiveness, offering a new representation of organizational coordination processes.

In conclusion, this dissertation illustrates the existence and importance of a balance between proactive, empathetic-leadership-based and more passive, authority-based processes and behaviors for the effective coordination of decomposition-based design work.

6.3 Future Work and Extensions

This work has several limitations and future extensions.

6.3.1 Limitations

The first two studies are limited by the scope of data collection: qualitative results would be further validated with rigorous observations of organizational processes to complement interviewee responses.

Our survey methodology in Chapter 4 is not robust to reporting bias and hindsight bias, which a future study should seek to mitigate with additional longitudinal and observational data collection. This is difficult in a classroom setting as it involves additional overhead for students focused on their coursework. For example, project management software could be used to collect information about who is working on and communicating about what tasks, and when. However the value of the software as a reporting tool relies on students actively using the tool as they work. One way to do this is require use of the software as part of coursework. A carefully designed study using such project management tools could be worked into a similar project to minimize the potential burden for study participants while allowing collection of representative data.

Finally, the agent model presented in Chapter 5 is not fully validated, and collection of data to support selection of both inputs and compare outputs is needed for a complete proof of concept. An outline for a validation plan is included in Chapter 5, as are recommendations for additional parametric analysis to refine the conclusions offered from the model. Of particular interest for parametric analysis are the input network structure, problem partitioning strategy, and agent type location throughout the network. Additional parameters that likely impact results are the initial probabilities to interact, and the amount of data that is received through each interaction. The initial interaction probabilities are set to extremes in this study. Exploring other initial values would give an idea of the stability of behavior throughout multiple model iterations.

6.3.2 Extensions

There are three areas where we see this work could have impacts. First, systems engineering as a discipline and practice benefits from rigorous research into best practices. Systems engineering is a nascent area of research and will benefit from the identification of concepts in other disciplines such as positive leadership, empathy, and social capital that may be leveraged to advance systems engineering theory. The research presented in this dissertation suggests such a link is beneficial to understanding best practice. Future studies in this area may focus on mapping social capital within an organization or testing the impact of positive leadership training on organizational and technical outcomes.

Second, modular systems design was mentioned briefly in our review of related literature in Chapter 2. The coordination question in system design is often to identify the best partitioning strategy based on assumed coordination costs for certain architectures. While there is merit in the existing approaches, the research presented here could be used to augment those approaches with an improved understanding of how people in fact work across disciplines. Thus partitioning approaches could be tailored to an organization and the working behaviors preferred by those individuals. An organization too may be able to seek out specific individual behaviors to promote effective design of a system.

Finally we mention a connection to our study of novice designers. This study may present new ways to teach students about project management and coordination approaches that go beyond setting team roles, schedules, and design processes. This research emphasizes the importance of interpersonal communication and the willingness to be both proactive and empathetic in those interactions, which are skills that could be introduced into engineering design curriculum. This research does not have evidence to support that any of the coordination approaches adopted by student teams in our study are better or worse than the others. However, the teams we observed adopted similar compositions of more and less communicative members, suggesting a mix is valuable.

APPENDIX A

Interview Protocol

The interview protocol used to conduct interviews described and analyzed in Chapter 3 is reproduced here.

Please consider your first-hand experiences with designing and maintaining a large-scale, complex engineered system.

1. Please describe a specific project in which you participated in the design and management of a complex system. Can you sketch the technical system? Using this sketch, can you tell me which groups work on which parts of the technical system? How many engineers were involved in the project? How many engineers were in each group you drew in the sketch? Where was each group physically located?
2. Where do you work? What is your formal title? How many years of work experience do you have? For this project, can you describe your typical work day? In a typical week, what are the top 3-5 tasks you spend/spent the most time on? What would you say has contributed most to your ability to do these tasks effectively? Formal education, on the job experience, mentoring, or something else? Could you elaborate? (Who is your mentor? When do you switch?)
3. What was your role in the project? Did your responsibilities change throughout the design of this system? Can you indicate which subsystem(s) you worked on, and their relationship to the other systems in the project? How would you characterize

your work on these subsystems (design, interface management, analysis, something else)? Which of the groups you indicated did you feel you “belonged” to?

4. How did you arrive at this partitioning (in the sketch)? Is there a typical partitioning common to your organization, or is each project broken down differently? Can you describe it? Is this partitioning reflected in the structure of your organization? Who (what title) is responsible for deciding how the work gets done/what the subsystems are? Can you describe how these decisions are made? How are the design teams selected? Are you directly involved in making these decisions?
 - a. (If participant makes partitioning decisions): Generally speaking, what heuristics or processes did you use to decide how to distribute the technical work for this project? At what stage did you finalize this breakdown? Is this consistent with the original work breakdown structure for the project? What information did you have available to you at the time you were making decisions about how to distribute the technical work for this project? (Within groups and within organization?) Did you use all of the information available to you when making decisions about how best to distribute the work?
 - b. (If participant does not make partitioning decisions): How was this work breakdown structure presented or communicated to you? Who delivered it to you? Is this consistent with the original work breakdown structure for the project? Do you feel that this was the best way to break down the project/distribute the work? To communicate the project structure? Why or why not? Would you have broken things down differently? How? Why? Would you have communicated this information differently?
5. Going back to your sketch, which groups you indicated often found your work relevant to theirs? Which groups in the sketches rarely found your work relevant? How did you know? Did you work with these groups frequently (e.g., several times per

week)? Infrequently (e.g., a few times per month)? How would you characterize your interactions with these groups, e.g. requesting information, or providing information? Would you characterize these interactions as primarily formal or informal? Did you routinely anticipate any requests for information from other groups or the need to provide information to other groups? How do you work this into your personal process?

6. What methods of communication does your organization use (email, meetings/face-to-face/documents/coffee breaks/other)? What roles do each of these communication methods have? Can you compare them? How could these communication channels be improved? Is there a common technology or software that you use to keep track of documentation or host meetings? Do you feel that this technology or software is useful/ effective? Why/why not? Is there something you would do differently, or another tool that you would use?
7. How do/did subsystems interface throughout the design process (early conceptual stages through to final design)? Was this different during different stages of design? In what way? Is there a single person or group with which all subsystems regularly communicated? What is the role of direct communication between groups as compared to communication with this central individual/group? Do these interactions tend to be through scheduled meetings or informal conversation? Something else?
8. At what stage in the design process do systems engineers attempt to coordinate the design of the subsystems? Can you describe how this was done in this project? Who was involved? Did the coordination change over time? What information was available to the systems engineer in each of these cases? How was this information presented to the systems engineer? Was this information presented to design groups?
9. At any stage in system design or subsystem coordination, were you uncertain about either the reliability or the relevance of the information that you had available? Can

you elaborate? At any stage, were you uncertain about the appropriateness of the decisions you made based on this information? How did you handle this situation?

10. Was there any stage during the system design process in which you found it difficult to process and integrate the information available? Describe precisely the nature of the situation.
11. Were you reminded of similar experiences/projects at any point during your work on this project? Were you at any point reminded of different experiences/projects? Were you at any point reminded of a project that succeeded? Were you at any point reminded of a project that failed? Did these experiences affect the decisions you made or actions that you took? How? Who refers relevant lessons learned?
12. Do you think that you could develop a rule, based on your experience, which could assist another person to make the same design decisions successfully? Why/why not? What advice would you give to someone new to the role you had on this project?
13. Is there anything we might have missed? Do you have any other thoughts about systems design that you'd like to share?

APPENDIX B

Team Coordination Survey Protocol

The survey used to collect data for the analysis presented in Chapter 4 is reproduced here. The formatting is modified slightly from the original but the text is unchanged.

Please complete this survey on your own without discussion with your team or squad.

1. Please provide your **survey ID** from the attached page so that we can identify your responses: _____
2. Describe in one or two sentences your RMP's strategy and its features.
3. What were your role(s)?
 Within your team?:

 Within your squad?:
4. What were your responsibilities regarding the design of your RMP? Include part-specific tasks.
 - a.
 - b.
 - c.

5. Describe the roles and responsibilities of your teammates:

Survey ID	Teammates' role(s):	Teammates' responsibilities:	Describe any overlap with your own role/responsibility:

Please continue on reverse

The following questions will ask you more about the roles and responsibilities you mentioned on the previous page. Focus on interactions among your team that pertain to the responsibilities you identified above.

6. How often did you communicate with other members of your team where the topic was **your** design responsibilities? Include communications in class, out of class, in person, and virtual (e.g., email).

Survey ID	How often did you communicate? Circle one					How useful were these interactions? Circle one			
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential

7. How often did you communicate with other members of your team where the topic was *their* design responsibilities? Include communications in class, out of class, in person, and virtual (e.g., email).

Survey ID	How often did you communicate? Circle one					How useful were these interactions? Circle one			
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential

Please continue on the following page

8. How often did you communicate with other members of your **squad** where the topic was *your* design responsibilities?

Survey ID	How often did you communicate? Circle one					How useful were these interactions? Circle one			
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential
	4+ days/week	2-3 days/week	<2 days/week	Monthly	Never	N/A	Not Needed	Helpful	Essential

9. Anything else?

Thank you for your participation in this survey!

APPENDIX C

Agent Model Code

The MATLAB code for the agent model described in Chapter 5 is reproduced here.

```
% Distributed classification by agents to test
% coordination methods
% Written by: Arianne Collopy
%
% MATLAB Version: 9.4.0.949201 (R2018a)
% Update 6
% Requires MATLAB statistics and machine
% learning toolbox

%% GLOBAL PARAMETERS %%%%%%%%%%%
% number of model iterations:
T = 20;

% initialize random seed:
rng(1)

% threshold options:
% threshold defined to exclude most incorrect results
```

```

thresh_option = 'incorrectabove';
% threshold defined to include most correct results
% thresh_option = 'correctbelow';
% threshold average of medians of incorrect & correct dist.
% thresh_option = 'avgthresh';

%% INTERACTION PARAMETERS %%%%%%%%%%%
interact_add = 10;
ask_add = 2;
fprintf('interaction parameters:
    interact_add = %d, ask_add = %d\n', ...
    interact_add, ask_add)

%% PARTITIONING SELECTION %%%%%%%%%%%
% uniform distribution, random:
% partitioning = 'random';

% group in alphabetically-ordered bins by node,
% agents' data assigned in order by community:
partitioning = 'alphabynode';

% reverse alpha order, assignment in node order by comm.:
% partitioning = 'revalphabynode';

%% NETWORK STRUCTURE %%%%%%%%%%%
global num_nodes net GroupMembers
num_nodes = 100;

```

```

% index of generated network structure, algorithm provided
% by [189]. load edge list and community members list
neti = 1;

EndNodes = table2array(readtable(sprintf(
    './networks/communitynetwork%d_edgetable.csv',neti)));
EdgeTable = table(EndNodes);

GroupMembers = table2array(readtable(sprintf(
    './networks/communitynetwork%d_membership.csv', neti)));

% calculate community sizes:
commsize = zeros(max(GroupMembers(:,2)),1);
for ki = 1:max(GroupMembers(:,2))
    commsize(ki) = sum(GroupMembers(:,2) == ki);
end

net = graph(table(EndNodes));

%% AGENT DISTRIBUTION %%%%%%%%%%%%%%%
% agent mix, or assign percentages
num_active = 0.5*num_nodes; % type 1
num_passive = 0.5*num_nodes; % type 2

if num_active == num_nodes
    agent_type_rpt = 'all active';

```

```

elseif num_passive == num_nodes
    agent_type_rpt = 'all passive';
else
    agent_type_rpt = 'mixed';
end

% ensure total is correct
if num_active + num_passive ~= num_nodes
    fprintf('error: check number of agents \n')
end

% assign agent types, random assignment based on above
global typeassign
types = [ones(num_active, 1); repmat(2, num_passive, 1)];
agentdist = [randperm(length(types))', types];
typeassign = sortrows(agentdist,1);
typeassign = typeassign(:,2);

% connection matrix: start at ones for all peers
global connectstrength
connectstrength = ones(num_nodes) - eye(num_nodes);

%% AGENT BEHAVIOR PARAMETERS %%%%%%%%%%%%%%%
% initial probability to interact:
% random value below this threshold means interact,
% above means no interact (ask)
active.p_interact = 1;

```



```

passive.p_interact = 0;

% change p_interact based on performance increase/decrease
pchange = 0.5;

p_interact = zeros(num_nodes, T);
for k = 1:num_nodes
    if typeassign(k) == 1 % active
        p_interact(k, 1) = active.p_interact;
    elseif typeassign(k) == 2 % passive
        p_interact(k, 1) = passive.p_interact;
    end
end

% INITIALIZE CLASSIFICATION PROBLEM %%%%%%%%%%%%%%
% load classification data, obtained from
% UCI Machine Learning Repository [191]
data = readtable('character-data.txt');

datavars = table2array(data(:,2:end));
datalabels = table2cell(data(:,1));

% set aside global test data:
holdoutsizesize = 200;
testvars = datavars(1:holdoutsizesize,:);
testlabels = datalabels(1:holdoutsizesize,:);

```

```

% remaining training data to distribute:
trainvars = datavars(holdoutsizes+1:end,:);
alltrainlabels = datalabels(holdoutsizes+1:end, :);

% possible reference letters:
alphabet = unique(alltrainlabels);
partitionlabels = cell(max(GroupMembers(:,2)), 5);

% identify duplicates in initial data:
[uniques, uidx] = unique(table(trainvars(:, :),
    alltrainlabels), 'rows');
nonuniques = setdiff(1:length(trainvars), uidx);
sortrows(table(trainvars(nonuniques, :),
    alltrainlabels(nonuniques)), 2);

% INITIALIZE MULTIPLE RUNS %%%%%%%%%%%
runnumber = 100;

% individual performance by node on local task, global task
store_indperf_local_multi = zeros(num_nodes, T, runnumber);
store_indperf_local_binary = zeros(num_nodes, T, runnumber);

% global accuracy, 3 ways
store_aggregateaccuracy = zeros(T, runnumber);
store_aggregateaccuracy_binary = zeros(T, runnumber);
store_aggregateaccuracy_binary_bycomm = zeros(T, runnumber);

```

```

% global test predictions by node
store_currentglobalprediction_multi = cell(runnumber);
store_currentglobalprediction_binary = cell(runnumber);

% connectionstrength matrix and history of interactions
store_connectstrength =
    zeros(num_nodes, num_nodes, runnumber);
store_eventlog = cell(runnumber);

% start iteration
for run = 1:runnumber
%% PARTITION DATA %%%%%%%%%%%%%%%
% partition data so that each agent gets approximately
% equal size portion

% percent of each community for test and ask
comm_testpct = 0.1;
comm_askpct = 0.05;

% for weighted allocation to community by number nodes
fullpartsize = floor(size(trainvars,1)./num_nodes);

% training data size for each agent
partsize = floor(0.85*fullpartsize); %round down

% contribution from each agent's max possible allocation to

```

```

% community ask and test
a_asksize = ceil(comm_askpct*fullpartsize); % round up.
a_commtestsize = ceil(comm_testpct*fullpartsize); %round up.

if partsize + a_asksize + a_commtestsize ~= fullpartsize
    fprintf('rounding error\n')
end

% alert if partitioning uneven among agents
if mod(length(trainvars),num_nodes) ~= 0
    fprintf('agents have unequal partitions\n')
end

% storage vectors
trainsamples =
    zeros(partsize, size(trainvars,2), num_nodes);
trainlabels = cell(partsize, 1, num_nodes);

max_asksize = a_asksize*max(commsize);
max_commtestsize = a_commtestsize*max(commsize);

communityaskvars =
    zeros(max_asksize, size(trainvars,2),
        max(GroupMembers(:,2)));
communityasklabels =
    cell(max_asksize, 1, max(GroupMembers(:,2)));
communitytestvars =

```

```

        zeros(max_commtestsize, size(trainvars,2),
        max(GroupMembers(:,2)));
communitytestlabels =
        cell(max_commtestsize, 1, max(GroupMembers(:,2)));

% RANDOM PARTITIONS
if strcmp(partitioning, 'random')
    fprintf('partitioning using random order of data\n')

    % randomization index:
    index = randperm(size(trainvars,1))';

    % add random index to data and sort
    sortvars = sortrows([index, trainvars],1);
    sortlabels = sortrows(table(index, alltrainlabels));

    % extract from table; remove column of indices
    sortvars = sortvars(:,2:end);
    sortlabel = table2cell(sortlabels(:,2));

    % assign data to each ask and internal train datasets;
    % remainder to nodes within community
    nodeidx = 1;
    for ki = 1:max(GroupMembers(:,2))
        loc_alldatasize = commsize(ki) *fullpartsize;
        loc_asksize = commsize(ki) * a_asksize;
        loc_testsize = commsize(ki) * a_commtestsize;
    end
end

```

```

% indices of held-out data
heldoutindices = randsample(loc_alldatasize,
    loc_asksize + loc_testsize );

% flag 1 for held-out, 0 for included in agents'
% training data.
indices = [1:loc_alldatasize]';
holdout = ismember(indices,heldoutindices);

% community data
communityvars =
    sortvars((nodeidx-1)*fullpartsize + 1 :
    (nodeidx + commsize(ki) - 1)*fullpartsize, :);
communitylabels =
    sortlabel((nodeidx-1)*fullpartsize + 1 :
    (nodeidx + commsize(ki) - 1)*fullpartsize, :);
partitionlabels(ki, 1:length(
    unique(communitylabels))) =
    unique(communitylabels)';

% sort community data by holdout vs not;
% held out set at bottom
communitytable =
    sortrows(table(communityvars,
    communitylabels, holdout), 3);

```

```

% iterate over community members and
% assign training data
for k = 1:commsize(ki)
    trainsamples(:, :, nodeidx) =
        communitytable{(k-1)*partsize + 1 :
            k*partsize, 1};
    trainlabels(:, :, nodeidx) =
        communitytable{(k-1)*partsize + 1 :
            k*partsize, 2};

    % increment counter: count through members
    nodeidx = nodeidx + 1;
end

% assign leftover data to internal test and ask data
communityaskvars(1 : loc_asksize,
    1:size(trainvars,2), ki) =
    communitytable{ commsize(ki)*partsize+1 :
        commsize(ki)*partsize + loc_asksize, 1};
communityasklabels(1 : loc_asksize, 1, ki) =
    communitytable{ commsize(ki)*partsize+1 :
        commsize(ki)*partsize + loc_asksize, end-1};
communitytestvars(1 : loc_testsize,
    1 : size(trainvars, 2), ki) =
    communitytable{commsize(ki)*partsize +
        loc_asksize + 1 : commsize(ki)*partsize +
        loc_asksize + loc_testsize, 1};

```

```

        communitytestlabels(1 : loc_testsize, 1, ki) =
            communitytable{commsize(ki)*partsize +
                loc_asksize + 1 : commsize(ki)*partsize +
                loc_asksize + loc_testsize, end-1};
    end

% ALPHA by NODE
elseif strcmp(partitioning, 'alphabynode')
    fprintf('partitioning using alphabetical sort
    by node\n')

    % sort paired data by data label
    tempalphasort =
        sortrows(table(trainvars, alltrainlabels), 2);

    % extract from table
    sortvars = tempalphasort{:, 1 : end-1};
    sortlabel = tempalphasort{:, end};

    % assign data to each ask and internal train datasets;
    % remainder to nodes within community
    nodeidx = 1;
    % iterate over groups
    for ki = 1:max(GroupMembers(:,2))
        loc_alldatasize = commsize(ki) *fullpartsize;
        loc_asksize = commsize(ki) * a_asksize;
        loc_testsize = commsize(ki) * a_commtestsize;
    end

```



```

% indices of held-out data: allocate ask and test
heldoutindices =
    randsample(loc_alldatasize, loc_asksize +
        loc_testsize );

% flag 1 for held-out, 0 for included in agents'
% training data.
indices = [1:loc_alldatasize]';
holdout = ismember(indices,heldoutindices);

% community data
communityvars =
    sortvars((nodeidx-1)*fullpartsize + 1 :
        (nodeidx + commsize(ki) - 1)*fullpartsize, :);
communitylabels =
    sortlabel((nodeidx-1)*fullpartsize + 1 :
        (nodeidx + commsize(ki) - 1)*fullpartsize, :);
partitionlabels(ki,
    1:length(unique(communitylabels))) =
    unique(communitylabels)';

% sort community data by holdout vs not;
% held out set at bottom
communitytable =
    sortrows(table(communityvars, communitylabels,
        holdout), 3);

```

```

% iterate over community members and assign
% training data
for k = 1:commsize(ki)
    trainsamples(:, :, nodeidx) =
        communitytable{(k-1)*partsize + 1 :
            k*partsize, 1};
    trainlabels(:, :, nodeidx) =
        communitytable{(k-1)*partsize + 1 :
            k*partsize, 2};

    % increment counter: count through members
    nodeidx = nodeidx + 1;
end

% leftover data is internal ask and test data
asktesttable =
    communitytable(commsize(ki)*partsize+1:end, :);

% randomize within ask and test
rerandidx = randperm(loc_asksize + loc_testsize);
asktesttable2 =
    sortrows(addvars(asktesttable, rerandidx',
        'before', 'communityvars'), 1);

% assign leftover data (set aside earlier) to
% internal test and ask data

```

```

communityaskvars(1:loc_asksize,
    1:size(trainvars,2), ki) =
    asktesttable2{1 : loc_asksize, 2};
communityasklabels(1:loc_asksize, 1, ki) =
    asktesttable2{1: loc_asksize, end-1};
communitytestvars(1:loc_testsize,
    1:size(trainvars,2), ki) =
    asktesttable2{loc_asksize + 1 :
    loc_asksize + loc_testsize, 2};
communitytestlabels(1:loc_testsize, 1, ki) =
    asktesttable2{loc_asksize + 1 :
    loc_asksize + loc_testsize, end-1};
end

```

```

% REVERSE ALPHA by NODE
elseif strcmp(partitioning, 'revalphabynode')
    fprintf('partitioning using reverse alphabetical
        sort by node\n')

    % sort labels in reverse order
    tempalphasort =
        sortrows(table(trainvars, alltrainlabels), 2,
            'descend');

    % extract from table
    sortvars = tempalphasort{:,1:end-1};

```

```

sortlabel = tempalphasort(:,end);

% assign data to each ask and internal train datasets;
% remainder to nodes within community
nodeidx = 1;

% iterate over communities
for ki = 1:max(GroupMembers(:,2))
    loc_alldatasize = commsize(ki) *fullpartsize;
    loc_asksize = commsize(ki) * a_asksize;
    loc_testsize = commsize(ki) *a_commtestsize;

    % indices of held-out data: ask and test
    heldoutindices =
        randsample(loc_alldatasize, loc_asksize +
            loc_testsize );

    % flag 1 for held-out, 0 for included in agents'
    % training data.
    indices = [1:loc_alldatasize]';
    holdout = ismember(indices,heldoutindices);

    % community data
    communityvars =
        sortvars((nodeidx-1)*fullpartsize + 1 :
            (nodeidx + commsize(ki) - 1)*fullpartsize, :);
    communitylabels =

```

```

        sortlabel((nodeidx-1)*fullpartsize + 1 :
        (nodeidx + commsize(ki) - 1)*fullpartsize, :);
partitionlabels(ki,
        1:length(unique(communitylabels))) =
        unique(communitylabels)';

% sort community data by holdout vs not;
% held out set at bottom
communitytable =
        sortrows(table(communityvars, communitylabels,
        holdout), 3);

for k = 1:commsize(ki)
        trainsamples(:, :, nodeidx) =
                communitytable{(k-1)*partsize + 1 :
                k*partsize, 1};
        trainlabels(:, :, nodeidx) =
                communitytable{(k-1)*partsize + 1 :
                k*partsize, 2};

        % increment counter: iterate through members
        nodeidx = nodeidx + 1;
end

% leftover data is internal ask and test data
asktesttable =
        communitytable(commsize(ki)*partsize+1:end, :);

```

```

% randomize within ask and test
rerandidx = randperm(loc_asksize + loc_testsize);
asktesttable2 =
    sortrows(addvars(asktesttable, rerandidx',
        'before', 'communityvars'), 1);

% assign leftover data (set aside earlier) to
% internal test and ask data
communityaskvars(1:loc_asksize,
    1:size(trainvars,2), ki) =
    asktesttable2{1 : loc_asksize, 2};
communityasklabels(1:loc_asksize, 1, ki) =
    asktesttable2{1: loc_asksize, end-1};
communitytestvars(1:loc_testsize,
    1:size(trainvars,2), ki) =
    asktesttable2{loc_asksize + 1 :
    loc_asksize + loc_testsize, 2};
communitytestlabels(1:loc_testsize, 1, ki) =
    asktesttable2{loc_asksize + 1 :
    loc_asksize + loc_testsize, end-1};

    end
end

% common to all partitioning schemes:
% create storage matrix (of max size equal to total training
% dataset size) for current working set each agent's

```

```

% classifier is built from
mytraindata =
    zeros(size(trainvars,1), size(trainvars,2), num_nodes);
mytrainlabels =
    cell(size(trainlabels,1), size(trainlabels, 2),
        num_nodes);
for k = 1:num_nodes
    mytraindata(1:partsize,:,k) = trainsamples(:, :, k);
    mytrainlabels(1:partsize,:,k) = trainlabels(:, :, k);
end

%% INITIALIZE MODEL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% k-nn model parameter
nneighbors = ones(num_nodes,1);

% log agents' action each tick:
eventlog = cell(num_nodes,1,T);
choice_weights_history =
    zeros(num_nodes, max(degree(net)), T);

% compare model results to global test; local
% initialized later
compare_g = zeros(length(testvars),1);

% accuracy on global and local test data from each agent:
% multiclass (w/o thresh)
global_accuracy_multi = zeros(num_nodes, T);

```

```

local_accuracy_multi = zeros(num_nodes, T);

% store distance and y/n accuracy of global and local test
distaccuracy_g = zeros(length(testvars), 2, num_nodes, T);
% may have empty rows:
distaccuracy_l = zeros(max_commtestsize, 2, num_nodes, T);

% store threshold calculations from each agent at each T
% 3rd quartile correct
thresh_corrbelow = zeros(num_nodes, T);
% 1st quartile incorrect
thresh_incorrabove = zeros(num_nodes, T);
% average of above two measures
thresh_avg = zeros(num_nodes, T);
% average of medians of correct/incorr dist
thresh_med = zeros(num_nodes, T);

% accuracy on global and local test data from each agent:
% binary (w/thresh);
% after aggregation from votes for/against:
global_accuracy_binary = zeros(num_nodes, T);
local_accuracy_binary = zeros(num_nodes, T);

% actual vote, test, vote, correct/incorrect for local
% and global test
localvote_binary = cell(max_commtestsize, 4, num_nodes, T);
globalvote_binary = cell(length(testvars), 4, num_nodes, T);

```



```

% log of each agent's prediction at each timestep,
% multiclass and binary
currentglobalprediction_multi =
    cell(length(testvars), num_nodes, T);
currentglobalprediction_binary =
    cell(length(testvars), num_nodes, T);

% store comparison of each model to test: logical value
storedcompare = zeros(length(testvars), num_nodes, T);
storedcompare_l = zeros(max_commtestsize, num_nodes, T);

% aggregate of agents' prediction at each step --
% currently calculated as mode of all agents' guesses
aggregateprediction = cell(length(testvars), 1, T);

% compare aggregate result to test labels
% global test, aggregation by node without threshold
aggregatecompare = zeros(length(testvars), T);
aggregateaccuracy = zeros(T, 1);

% global test, aggregation by node with threshold
aggregatecompare_binary = zeros(length(testvars), T);
aggregateaccuracy_binary = zeros(T, 1);

% global test, aggregation by community with threshold
commvotefor = cell(length(testvars),

```

```

        max(GroupMembers(:,2)), T);
commvoteagainst = cell(length(testvars),
        max(GroupMembers(:,2)), T);
aggregatecommvote = cell(length(testvars), T);
incomm_aggcompare_binary = zeros(T, 1);
aggregateaccuracy_binary_bycomm = zeros(T, 1);

% track items added from ask pool
askselected = zeros(num_nodes, ask_add, T);

%% CALCULATE T = 0 ACCURACY %%%%%%%%%%%%%%%
initaccuracy_g = zeros(num_nodes, 1);
initaccuracy_l = zeros(num_nodes, 1);

% initial agent performance on global dataset
for k = 1:num_nodes
    self = k;
    mycomm = GroupMembers(self, 2);
    commtestsize = a_commtestsize * commsize(mycomm);

    % create classifier from own data
    train =
        mytraindata(any(mytraindata(:, :, self), 2),
            :, self);
    label = mytrainlabels(1:size(train,1), :, self);

    knnfit = fitcknn(train, label, 'NumNeighbors',

```

```

        nneighbors(self));
knnpred_g = predict(knnfit, testvars);
knnpred_l = predict(knnfit,
        communitytestvars(1:commtestsize, :, mycomm));

% compare results to global test data
for j = 1:length(knnpred_g)
        compare_g(j) = strcmp(knnpred_g(j), testlabels(j));
end

% compare model to community test data
compare_l = zeros(
        size(communitytestlabels(1:commtestsize, :, mycomm)));
for j = 1:length(knnpred_l)
        compare_l(j) = strcmp(knnpred_l (j),
                communitytestlabels(j, :, mycomm));
end

% store initial accuracy as percent correct
initaccuracy_g(self) =
        sum(compare_g)/length(testlabels);
initaccuracy_l(self) = sum(compare_l)/commtestsize;
end

%% BEGIN MODEL ITERATIONS %%%%%%%%%%%
for tick = 1:T

```

```

% output log:
if mod(tick,10) == 0
    fprintf('tick %d, run %d\n', tick, run)
end

% iterate over agents
for k = 1:num_nodes
    self = k;
    mycomm = GroupMembers(self, 2);
    mycommmasksize = sum(any(communityaskvars(:, :, mycomm), 2));

% cases for each agent type
r = rand();
if r < p_interact(self, tick)
    interact = 'peer'; % interact
else
    interact = 'ask'; % don't interact
end

if strcmp(interact, 'peer')
%% CHOOSE PEER %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % choose peer to interact
    [peer, w] = choose_peer(self);
    choice_weights_history(
        self, 1:length(w), tick) = w';

    eventlog{k, 1, tick} = ...

```

```

        sprintf('interact with node %d,
                exchange %d item(s)',
                peer, interact_add);

%% EXCHANGE DATA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% peer data available
peerdatasize = sum(any(mytraindata(:, :, peer), 2));
sample = randsample(peerdatasize, interact_add);

datatransf = mytraindata(sample, :, peer);
labeltransf = mytrainlabels(sample, :, peer);

% current size of agent's own training data:
currentsize =
    sum(any(mytraindata(:, :, self), 2));
currentttest =
    sum(any(communitytestvars(:, :, mycomm), 2));

% characterize new categories:
% any new to community?
[incommtest, ~, loc_test] =
    unique(communitytestlabels(
        any(communitytestvars(:, :, mycomm), 2), :, mycomm));
[inask, ~, loc_ask] = unique(labeltransf);

% in received data, but not in community test:
if ~isempty(setdiff(inask, incommtest))

```

```

% how many labels are new? cell array, uniques
new = setdiff(inask, incommtest);

% count instances, accounting for more than one
% new data type
countnew = 0;
newtotest = ;
for newlett = 1:length(new)
    % find indices of new category data
    newtotest = [newtotest; find(
        strcmp(labeltransf, new{newlett}))];
    % count number of indices
    countnew = countnew + length(find(
        strcmp(labeltransf, new{newlett})));
end

% index of items to add: add 50% of new labels
% to community test and remainder to own train
addtotest =
    newtotest(randsample(length(newtotest),
        ceil(countnew/2)));
addtotrain = setdiff([1:length(labeltransf)]',
    addtotest);

% add if new. check for duplicates
dupecheck = zeros(length(addtotrain),1);
for idx = 1:length(addtotrain)

```

```

        if ismember(datatransf(addtotrain(idx),:),
            mytraindata(
                1:currentsize,:,self), 'rows')
            && ismember(
                labeltransf(addtotrain(idx)),
                mytrainlabels(1:currentsize,:,self))

            dupecheck(idx) = 0; % don't keep
        else
            dupecheck(idx) = 1; % keep
        end
    end

    % store as logical value
    dupecheck = logical(dupecheck);

    % add non-dupes to current train data
    if sum(dupecheck) > 0
        mytraindata(currentsize+1 :
            currentsize+sum(dupecheck),:,self) =
            datatransf(addtotrain(dupecheck),:);
        mytrainlabels(currentsize+1 :
            currentsize+sum(dupecheck),:,self) =
            labeltransf(addtotrain(dupecheck));
    end

    % add data to test
    communitytestvars(currenttest+1 :

```

```

        currenttest+length(addtotest),:,mycomm) =
        datatransf(addtotest,:);
communitytestlabels(currenttest+1 :
        currenttest+length(addtotest), :, mycomm) =
        labeltransf(addtotest);

else % labels not new to community test
    % add if new. check for duplicates
    dupecheck = zeros(length(labeltransf),1);
    for idx = 1:length(labeltransf)
        if ismember(datatransf(idx,:),
            mytraindata(
                1:currentsize,:,self), 'rows')
            && ismember(labeltransf(idx),
                mytrainlabels(1:currentsize,:,self))
            dupecheck(idx) = 0; % don't keep

        else
            dupecheck(idx) = 1; % keep
        end
    end
end

% store as logical
dupecheck = logical(dupecheck);

if sum(dupecheck) > 0
    mytraindata(currentsize+1 :

```



```

        currentsize+sum(dupecheck),:,self) =
        datatransf(dupecheck,:);
    mytrainlabels(currentsize+1 :
        currentsize+sum(dupecheck),:,self) =
        labeltransf(dupecheck);
    end
end
else
%% DRAW from ASK %%%%%%%%%%%%%%%
% agents draw ask_add items from in-community `ask' pool

% ask pool data available
sample = 1:mycommasksize;
% retain only previously unselected options
sample =
    sample(~ismember(sample,askselected(self,:, :)));

if any(sample) % any choices left
    % choose ask_add number of datapoints to add
    askselected(self, :, tick) =
        randsample(sample, ask_add);

% find corresponding data in community ask pool
datatransf = communityaskvars(
    askselected(self, :, tick), :, mycomm);
labeltransf = communityasklabels(
    askselected(self, :, tick), :, mycomm);

```

```

% current size of agent's own training data:
currentsize =
    sum(any(mytraindata(:, :, self), 2));
currenttest =
    sum(any(communitytestvars(:, :, mycomm), 2));

% characterize new categories: any new to community?
[incommtest, ~, loc_test] =
    unique(communitytestlabels(
        any(communitytestvars(:, :, mycomm), 2),
        :, mycomm));
[inask, ~, loc_ask] = unique(labeltransf);

% in ask, but not in commtest
if ~isempty(setdiff(inask, incommtest))
    % how many labels are new? cell array, uniques
    new = setdiff(inask, incommtest);

    % count instances, accounting for more than one
    % new data type
    countnew = 0;
    newtotest = ;
    for newlett = 1:length(new)
        % find indices of new category data
        newtotest = [newtotest;
            find(strcmp(

```

```

        labeltransf, new{newlett}));
% count number of indices
countnew = countnew + length(
    find(strcmp(
        labeltransf, new{newlett})));
end

% index of items to add: add 50% of
% new labels to community test
addtotest =
    newtotest(randsample(length(newtotest),
        ceil(countnew/2)));
addtotrain =
    setdiff(1:length(labeltransf), addtotest);

% add if new. check for duplicates
dupecheck = zeros(length(addtotrain),1);
for idx = 1:length(addtotrain)
    if ismember(datatransf(addtotrain(idx),:),
        mytraindata(
            1:currentsize,:,self), 'rows')
        && ismember(
            labeltransf(addtotrain(idx)),
            mytrainlabels(
                1:currentsize,:,self))

        dupecheck(idx) = 0; % don't keep
    end
end

```

```

else
    dupecheck(idx) = 1; % keep
end
end

end

% store as logical
dupecheck = logical(dupecheck);

% add non-dupes to train
if sum(dupecheck) > 0
    mytraindata(currentsize+1 :
        currentsize+sum(dupecheck), :, self) =
        datatransf(addtotrain(dupecheck), :);
    mytrainlabels(currentsize+1 :
        currentsize+sum(dupecheck), :, self) =
        labeltransf(addtotrain(dupecheck));
end

% add data to test
communitytestvars(currenttest+1 :
    currenttest+length(addtotest), :, mycomm) =
    datatransf(addtotest, :);
communitytestlabels(currenttest+1 :
    currenttest+length(addtotest), :, mycomm) =
    labeltransf(addtotest);

eventlog{self, 1, tick} = ...

```

```

        sprintf('queried ask pool; added new data
                to train and comm. test');

else % labels not new to community test
    % add if new. check for duplicates
    dupecheck = zeros(length(labeltransf),1);
    for idx = 1:length(labeltransf)
        if ismember(datatransf(idx,:),
                    mytraindata(
                        1:currentsize,:,self), 'rows')
            && ismember(labeltransf(idx),
                        mytrainlabels(
                            1:currentsize,:,self))

            dupecheck(idx) = 0; % don't keep
        else
            dupecheck(idx) = 1; % keep
        end
    end
end

% store as logical value
dupecheck = logical(dupecheck);

if sum(dupecheck) > 0
    mytraindata(currentsize+1 :
                currentsize+sum(dupecheck), :, self) =
        datatransf(dupecheck, :);
end

```

```

        mytrainlabels(currentsize+1 :
            currentsize+sum(dupecheck),:,self) =
            labeltransf(dupecheck);
    end
    eventlog{self,1,tick} = ...
        sprintf('queried ask pool; added new data
            to train');
    end

else
    eventlog{self,1,tick} =
        sprintf('queried ask pool; no data available');
end

end

%% CREATE AND EVALUATE MODEL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% use just nonzero rows of mytraindata to train model
train =
    mytraindata(any(mytraindata(:, :, self), 2), :, self);
label =
    mytrainlabels(1:size(train,1), :, self);

%% MULTICLASS CLASSIFIER %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
currenttest =
    sum(any(communitytestvars(:, :, mycomm), 2));

```

```

[idx_g, dist_g] = knnsearch(train, testvars);
knnpred_g = label(idx_g);

[idx_l, dist_l] = knnsearch(train, communitytestvars(
    1:currenttest, :, mycomm));
knnpred_l = label(idx_l);

% compare results to test data
for j = 1:length(knnpred_g)
    compare_g(j) = strcmp(knnpred_g(j), testlabels(j));
    storedcompare(j,self,tick) = strcmp(
        knnpred_g(j), testlabels(j));
end

% compare model to community test data
compare_l = zeros(size(communitytestlabels(
    1:currenttest, :, mycomm)));
for j = 1:length(knnpred_l)
    compare_l(j) = strcmp(knnpred_l(j),
        communitytestlabels(j, :, mycomm));
    storedcompare_l(j,self,tick) = strcmp(
        knnpred_l(j),
        communitytestlabels(j, :, mycomm));
end

% log distance to match and correct or no
distaccuracy_g(:, :, self, tick) = [dist_g, compare_g];

```

```

distaccuracy_l(1:currenttest, :, self, tick) =
    [dist_l, compare_l];

% results = table(knnpred_g, testlabels, compare);

% store multiclass model results
global_multi_results =
    table(knnpred_g, testlabels, compare_g);
global_accuracy_multi(self,tick) =
    sum(compare_g)/length(testlabels);
local_accuracy_multi(self,tick) =
    sum(compare_l)/currenttest;

currentglobalprediction_multi(:,self,tick) = knnpred_g;

%% THRESHOLDS FROM LOCAL TEST RESULTS %%%%%%%%%%%
% find indices for correct and incorrect results:
dtemp = [dist_l, compare_l];
findia = find(dtemp(:, 2) == 0);
finda = find(dtemp(:, 2) == 1);

if any(finda)
    dist_correct = dtemp(finda, 1);
else
    dist_correct = NaN;
end

if any(findia)

```



```

        dist_incorrect = dtemp(findia, 1);
else
        dist_incorrect = NaN;
end

% correct below:
% find third quartile of correct matches
thresh_corrbelow(self, tick) =
    quantile(dist_correct, 0.75);
% incorrect above:
% find first quartile for incorrect matches
thresh_incorrabove(self, tick) =
    quantile(dist_incorrect, 0.25);
% average of two thresholds
thresh_avg(self, tick) = nanmean(
    [thresh_corrbelow(self,tick),
    thresh_incorrabove(self,tick)]);
% average of medians
thresh_med(self, tick) = nanmean(
    [median(dist_correct), median(dist_incorrect)]);

%% BINARY CLASSIFIER WITH THRESHOLD %%%%%%%%%%%
% labels assigned to community, according to what agent
% received in initial partition
initlabels = unique(partitionlabels(mycomm, find(
    ~cellfun('isempty', partitionlabels(mycomm, :)))));

```

```

% any new labels agent may have acquired through
% interaction with peers + initial labels
newlabels = unique(communitytestlabels(any(
    communitytestvars(:, :, mycomm), 2), 1, mycomm));

% sort out only new labels
newlabels = setdiff(newlabels, initlabels);

% translate multiclass results (knnpred_1) to binary
% result: flip if above threshold, keep same if below
if strcmp(thresh_option, 'correctbelow')
    thresh = thresh_corrbelow;
elseif strcmp(thresh_option, 'incorrectabove')
    thresh = thresh_incorrabove;
elseif strcmp(thresh_option, 'avgthresh')
    thresh = thresh_med;
end

% use threshold to flag untrusted results, and compare
% to test data:
% 1 if not flipped (keep), 0 is flip (disregard):
binaryflip = zeros(length(knnpred_1), 1);
% 1 if correct, 0 if incorrect:
compare_bin = zeros(length(knnpred_1), 1);

for j=1:length(knnpred_1)
    % temporary storage: as char

```

```

truetest = communitytestlabels{j, :, mycomm};
guess = knnpred_l{j};

% determine confidence from threshold
if dist_l(j) >= thresh(self, tick)
    binaryflip(j) = 0;
    % check for disagreement
    if truetest ~= guess
        compare_bin(j) = 1; % correct
    else
        compare_bin(j) = 0; % incorrect
    end
else
    binaryflip(j) = 1;
    % enter value as is and evaluate as is:
    % check for agreement
    if truetest == guess
        compare_bin(j) = 1; % correct
    else
        compare_bin(j) = 0; % incorrect
    end
end

end

end

% store results
binresulttable = table(communitytestlabels(1 :

```

```

        length(knnpred_l), :, mycomm),
        knnpred_l, binaryflip, compare_bin);
localvote_binary(1:length(knnpred_l), :, self, tick) =
        table2cell(binresulttable);

% calculate accuracy
local_accuracy_binary(self, tick) =
        sum(compare_bin) ./ length(compare_bin);

%% BINARY TEST - GLOBAL TEST %%%%%%%%%%%%%%%
% store flip: 1 if not flipped (keep),
% 0 if flipped (disregard)
binaryflip_g = zeros(length(knnpred_g), 1);
% store result: 1 if correct, 0 if incorrect
compare_bin_g = zeros(length(knnpred_g), 1);

for j=1:length(knnpred_g)
    truetest = testlabels{j, :}; % char
    guess = knnpred_g{j}; % char

    % determine confidence from threshold
    if dist_g(j) >= thresh(self, tick)
        binaryflip_g(j) = 0;
        % check for disagreement
        if truetest ~= guess
            compare_bin_g(j) = 1; % correct
        else

```

```

        compare_bin_g(j) = 0; % incorrect
    end

else
    binaryflip_g(j) = 1;
    % enter value as is and evaluate as is:
    % check for agreement
    if truetest == guess
        compare_bin_g(j) = 1; % correct
    else
        compare_bin_g(j) = 0; % incorrect
    end

end

end

end

% store results
binresulttable_g = table(
    testlabels, knnpred_g, binaryflip_g, compare_bin_g);
globalvote_binary(:, :, self, tick) =
    table2cell(binresulttable_g);

% calculate accuracy: agent's score on all test items
global_accuracy_binary(self, tick) =
    sum(compare_bin_g) ./ length(compare_bin_g);
currentglobalprediction_binary(:, self, tick) = knnpred_g;

```

```

%% UPDATE LINK STRENGTH %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% update p(choice) given increase or decrease in
% accuracy from last iteration

% ensure interaction probability at least constant from
% this iteration to the next:
if tick < T
    p_interact(self, tick+1) = p_interact(self, tick);
end

if tick > 1 && tick < 20
    % adjust probability based on observed increase or
    % decrease in accuracy
    if local_accuracy_multi(self, tick) >
        local_accuracy_multi(self, tick-1)
        || local_accuracy_binary(self, tick) >
            local_accuracy_binary(self, tick-1)

        if strcmp(interact, 'peer')
            % increase p(interact)
            p_interact(self, tick+1) =
                p_interact(self, tick+1) + pchange;

            % increase peer connection strength
            connectstrength(self,peer) =
                connectstrength(self, peer) + pchange;
        end
    end
end

```

```

elseif strcmp(interact, 'ask')
    % if drew from ask? increase p(ask) by
    % decreasing p_interact
    p_interact(self, tick+1) =
        p_interact(self, tick+1)-pchange;
end

elseif local_accuracy_multi(self, tick) <
    local_accuracy_multi(self, tick-1)
    || local_accuracy_binary(self, tick) <
    local_accuracy_binary(self, tick-1)

if strcmp(interact, 'peer')
    % decrease p(interact)
    p_interact(self, tick+1) =
        p_interact(self, tick+1)-pchange;

    % increase peer connection strength
    connectstrength(self,peer) =
        connectstrength(self, peer)-pchange;

    % preserve caps at 0%, 100%
    if p_interact(self, tick+1) > 1
        p_interact(self, tick+1) = 1;
    end

    if p_interact(self, tick+1) < 0

```



```

for item = 1:length(testvars)
    % initialize counter:
    aggregatecount = zeros(length(alphabet), 1);

    % iterate through alphabet, count each instance
    for iy = 1:length(alphabet)
        aggregatecount(iy) =
            length(find(strcmp(alphabet{iy},
                currentglobalprediction_multi(item,:,tick))));
    end

    % find first mode across all agents;
    % what most agents voted for given same test item
    [~, itemp] = max(aggregatecount);

    % flag tied majority vote
    if length(itemp) > 2
        sprintf(
            'more options: tick %d, letters %c',
            tick, itemp)
    end

    % store result:
    aggregateprediction{item, :, tick} =
        alphabet{itemp};

    % compare to true label:

```

```

        aggregatecompare(:,tick) = strcmp(
            aggregateprediction(:, :, tick), testlabels(:));
end

aggregateaccuracy(tick) =
    sum(aggregatecompare(:,tick))/length(testlabels);

%% calculate aggregate performance from binary
% classifier result
for item = 1:length(testvars)
    % each agent has stored: test data, vote, flip
    % (confidence vote), and accuracy at end in
    % globalvote_binary. Want to count yeses and nos
    % separately: only collide if yes and no for the
    % same letter.

    globaltrue = testlabels{item}; % char

    % find votes and confidence
    confvote_temp = squeeze(cell2mat(
        globalvote_binary(item, 3, :, tick)));
    vote_temp = char(squeeze(
        globalvote_binary(item, 2, :, tick))); % char

    % votes with confidence 1 (noflip)
    votesforidx = find(confvote_temp == 1);
    votesagainstidx = find(confvote_temp == 0);

```

```

votesfor = vote_temp(votesforidx);
votesagainst = vote_temp(votesagainstdx);

% find unique votes for
uniquefor = unique(votesfor);
votes = zeros(length(uniquefor),1);

for uniquevote = 1:length(uniquefor)
    % number votes for + votes not against
    votes(uniquevote) =
        length(find(votesfor ==
            uniquefor(uniquevote))) +
        (length(votesagainst) -
            length(find(votesagainst ==
                uniquefor(uniquevote))));
end

maxvote = find(votes == max(votes));
finalvote = uniquefor(maxvote);
% how to deal with ties?
if finalvote == globaltrue
    aggregatecompare_binary(item, tick) = 1;
else
    aggregatecompare_binary(item, tick) = 0;
end

```

```

end

aggregateaccuracy_binary(tick) = sum(
    aggregatecompare_binary(:, tick)) /
    length(testlabels);

%% calculate aggregate accuracy - aggregate by
% community from binary (thresholded) test
for comm = 1:max(GroupMembers(:,2))
    members = find(GroupMembers(:,2) == comm);
    for item = 1:length(testvars)
        % each agent has stored: test data, vote, flip
        % (confidence vote), and accuracy at end in
        % globalvote_binary. Count yeses and nos
        % separately: only collide if yes and no for the
        % same letter.

        globaltrue = testlabels{item}; % char

        % find votes and confidence: pull out of cell
        vote_temp = char(squeeze(
            globalvote_binary(item, 2, members, tick)));
        confvote_temp = squeeze(cell2mat(
            globalvote_binary(item, 3, members, tick)));

        % votes with confidence 1 (noflip)

```

```

votesforidx = find(confvote_temp == 1);
votesagainstidx = find(confvote_temp == 0);

votesfor = vote_temp(votesforidx);
votesagainst = vote_temp(votesagainstidx);

% find unique votes for
uniquefor = unique(votesfor);
votes = zeros(length(uniquefor),1);
uniqueagainst = unique(votesagainst);
votesag = zeros(length(uniqueagainst), 1);

if ~isempty(uniquefor)
    for uniquevote = 1:length(uniquefor)
        % number votes for + votes not against
        votes(uniquevote) =
            length(find(votesfor ==
                uniquefor(uniquevote))) +
            (length(votesagainst) -
                length(find(votesagainst ==
                    uniquefor(uniquevote)))));
    end

% find item with most votes within community
maxcommvote = find(votes == max(votes));
if length(maxcommvote) > 1
    % tiebreak = char with more entries in

```

```

    % votesfor
    tiebreak = zeros(length(maxcommvote),1);
    for tied = 1:length(maxcommvote)
        tiebreak(tied) = length(find(
            votesfor == uniquefor(
                maxcommvote(tied))));
    end
    tiebreakidx = find(max(tiebreak));
    commvotefor{item, comm, tick} =
        uniquefor(maxcommvote(tiebreakidx));
else
    commvotefor{item, comm, tick} =
        uniquefor(maxcommvote);
end
else
    % what if no votes for?
    for uniquevoteag = 1:length(uniqueagainst)
        % number votes against each item
        votesag(uniquevoteag) =
            length(find(votesagainst ==
                uniqueagainst(uniquevoteag)));
    end
    maxcommanti = find(votesag == max(votesag));
    % can have multiples in this list:
    commvoteagainst{item, comm, tick} =
        uniqueagainst(maxcommanti);
end
end

```

```

        end
    end

    % aggregate: iterate over items
    for item = 1:length(testvars)
        globaltrue = testlabels{item};
        % choose max vote from commvotes
        communityvotes = [commvotefor{item, :, tick}];

        % nobody voted for an item?
        if ~isempty(communityvotes)
            uniquecommvotes = unique(communityvotes);
            aggregatevotes = zeros(
                length(uniquecommvotes), 1);

            for commvoteidx = 1:length(uniquecommvotes)
                aggregatevotes(commvoteidx) =
                    length(find(communityvotes ==
                        uniquecommvotes(commvoteidx)));
            end

            maxaggcommvote = find(
                aggregatevotes == max(aggregatevotes));
            if length(maxaggcommvote) > 1
                % random tiebreak
                tiebreak = randi(length(
                    maxaggcommvote));
            end
        end
    end

```

```

        aggregatecommvote{item, tick} =
            uniquecommvotes(maxaggcommvote(
                tiebreak));
    else
        aggregatecommvote{item, tick} =
            uniquecommvotes(maxaggcommvote);
    end

    if aggregatecommvote{item,tick} == globaltrue
        incomm_aggcompare_binary(item, tick) = 1;
    else
        incomm_aggcompare_binary(item, tick) = 0;
    end

    else
        aggregatecommvote{item, tick} = '0';
        incomm_aggcompare_binary(item, tick) = 0;
    end

end

% store binary (thresholded) accuracy as pct
aggregateaccuracy_binary_bycomm(tick) =
    sum(incomm_aggcompare_binary(:, tick)) /
        length(testlabels);

end

%% STORE DATA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```



```

% individual performance by node on local task, 2 ways
store_indperf_local_multi(:, :, run) =
    local_accuracy_multi(:, :);
store_indperf_local_binary(:, :, run) =
    local_accuracy_binary(:, :);

% global accuracy, 3 ways
store_aggregateaccuracy(:, run) = aggregateaccuracy;
store_aggregateaccuracy_binary(:, run) =
    aggregateaccuracy_binary;
store_aggregateaccuracy_binary_bycomm(:, run) =
    aggregateaccuracy_binary_bycomm;

% global test votes
store_currentglobalprediction_multi{run} =
    currentglobalprediction_multi;
store_currentglobalprediction_binary{run} =
    currentglobalprediction_binary;

% connectionstrength matrix and history of interactions
store_connectstrength(:, :, run) = connectstrength;
store_eventlog{run} = eventlog;

end

%% SAVE RESULTS TO FILE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
save(sprintf('LFR%d_allpassive_indperf', neti),

```

```

    'store_indperf_local_multi',
    'store_indperf_local_binary');
save(sprintf('LFR%d_allpassive_globperf', neti),
    'store_aggregateaccuracy',
    'store_aggregateaccuracy_binary',
    'store_aggregateaccuracy_binary_bycomm');
save(sprintf('LFR%d_allpassive_eventlog', neti),
    'store_eventlog');
save(sprintf('LFR%d_allpassive_connectstrength', neti),
    'store_connectstrength');
save(sprintf('LFR%d_allpassive_typeassign', neti),
    'typeassign');

% FUNCTION: CHOOSE PEER %%%%%%%%%%%%%%%
function[chosenpeer, weights] = choose_peer(agent_no)
global connectstrength net GroupMembers typeassign
% identify immediate neighbors in network
[~,n] = outedges(net, agent_no);

% identify community members
selfcomm = GroupMembers(agent_no,2);
n2 = find(GroupMembers(:,2) == selfcomm);
% combined set of neighbors, except self
comb = setdiff(union(n, n2), agent_no);

% if active, assign larger set as neighbors
if typeassign(agent_no) == 1

```

```

        n = comb;
    end

    % calculate weights based on connection matrix
    weights = zeros(length(n),1);
    for k = 1:length(n)
        weights(k) = connectstrength(agent_no, n(k));
    end

    % check if peer sources exhausted (connectstrength = 0
    % for all): replace zeros with ones to calculate weights.
    % note this does not update connectstrength values,
    % just weights to choose peer.
    if ~any(weights)
        weights = ones(length(n),1);
    end

    % normalize weights
    weights = weights./sum(weights);

    % select index based on weights
    choiceidx = randsample(length(n),1,true,weights);
    chosenpeer = n(choiceidx);
end

```

BIBLIOGRAPHY

- Acuña, S. T. and N. Juristo (2004). “Assigning people to roles in software projects”. In: *Software: Practice and Experience* 34.7, pp. 675–696. DOI: [10.1002/spe.586](https://doi.org/10.1002/spe.586).
- Adams, R. et al. (2009). “Exploring the Boundaries: Language, Roles, and Structures in Cross-Disciplinary Design Teams”. In: *About: Designing - Analysing Design Meetings*. Ed. by J. McDonnell and P. Lloyd. London, UK: CRC Press, pp. 339–358.
- Adler, P. S. and S.-W. Kwon (2002). “Social Capital: Prospects for a New Concept”. In: *The Academy of Management Review* 27.1, p. 17. DOI: [10.2307/4134367](https://doi.org/10.2307/4134367).
- Agneessens, F. and R. Wittek (2012). “Where do intra-organizational advice relations come from? The role of informal status and social capital in social exchange”. In: *Social Networks. Dynamics of Social Networks* (2) 34.3, pp. 333–345. DOI: [10.1016/j.socnet.2011.04.002](https://doi.org/10.1016/j.socnet.2011.04.002).
- Alexandrov, N. M. and R. M. Lewis (2002). “Analytical and Computational Aspects of Collaborative Optimization for Multidisciplinary Design”. In: *AIAA Journal* 40.2, pp. 301–309. DOI: [10.2514/2.1646](https://doi.org/10.2514/2.1646).
- Allison, J. T. (2008). “Optimal Partitioning and Coordination Decisions in Decomposition-based Design Optimization”. Ph.D. Dissertation. Ann Arbor, MI: University of Michigan.
- Allison, J. et al. (2005). “On the use of analytical target cascading and collaborative optimization for complex system design”. In: *Proceedings of 6th World Congress on Structural and Multidisciplinary Optimization*.
- Alyaqout, S. F. et al. (2011). “Generalized Coupling Management in Complex Engineering Systems Optimization”. In: *Journal of Mechanical Design* 133.9, pp. 091005–091005. DOI: [10.1115/1.4004541](https://doi.org/10.1115/1.4004541).
- Aritzeta, A., S. Swailes, and B. Senior (2007). “Belbin’s Team Role Model: Development, Validity and Applications for Team Building*”. In: *Journal of Management Studies* 44.1, pp. 96–118. DOI: [10.1111/j.1467-6486.2007.00666.x](https://doi.org/10.1111/j.1467-6486.2007.00666.x).

- Asikoglu, O. and T. Simpson (2012). “A New Method for Evaluating Design Dependencies in Product Architectures”. In: *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics.
- Asplund, J. et al. (2007). *The Clifton StrengthsFinder® 2.0 Technical Report: Development and Validation*. Tech. rep. Princeton, NJ: The Gallup Organization.
- Austin-Breneman, J., T. Honda, and M. C. Yang (2012). “A Study of Student Design Team Behaviors in Complex System Design”. In: *Journal of Mechanical Design* 134.12, p. 124504. DOI: [10.1115/1.4007840](https://doi.org/10.1115/1.4007840).
- Austin-Breneman, J., B. Y. Yu, and M. C. Yang (2015). “Biased Information Passing Between Subsystems Over Time in Complex System Design”. In: *Journal of Mechanical Design* 138.1, pp. 011101–011101. DOI: [10.1115/1.4031745](https://doi.org/10.1115/1.4031745).
- Baker, W. E., R. Cross, and M. Wooten (2003). “Positive Organizational Network Analysis and Energizing Relationships”. In: *Positive Organizational Scholarship: Foundations of a New Discipline*. Ed. by K. S. Cameron, J. E. Dutton, and R. E. Quinn. San Francisco, CA, USA: Barrett-Koehler Publishers, Inc., pp. 328–342.
- Baldwin, C. and K. B. Clark (2000). *Design Rules: the Power of Modularity*. Vol. 1. Cambridge, MA: MIT Press.
- Barabási, A.-L. (2016). *Network Science*. Cambridge, UK: Cambridge University Press.
- Barnard, C. I. (1964). *The Functions of the Executive*. Cambridge, MA: Harvard University Press.
- Bayrak, A. E. et al. (2018). “Multiobjective optimization of modular design concepts for a collection of interacting systems”. In: *Structural and Multidisciplinary Optimization* 57.1, pp. 83–94. DOI: [10.1007/s00158-017-1872-4](https://doi.org/10.1007/s00158-017-1872-4).
- Bhatia, G., B. Mesmer, and K. Weger (2018). “Mathematical Representation of Stakeholder Preferences for the SPORT Small Satellite Project”. In: *2018 AIAA Aerospace Sciences Meeting*. Kissimmee, Florida: American Institute of Aeronautics and Astronautics. DOI: [10.2514/6.2018-0708](https://doi.org/10.2514/6.2018-0708).
- Blanchard, B. S. and W. J. Fabrycky (2011). *Systems Engineering and Analysis*. 5th. Upper Saddle River, NJ: Pearson.
- Blau, P. (1974). *On the Nature of Organizations*. New York, NY: Wiley.
- Bloebaum, C. L. and A. R. McGowan (2012). “The Design of Large-Scale Complex Engineered Systems: Present Challenges and Future Promise”. In: *12th AIAA Aviation*

- Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. Indianapolis, IN: American Institute of Aeronautics and Astronautics.
- Bloebaum, C., P. Collopy, and G. Hazelrigg (2012). “NSF/NASA Workshop on the Design of Large-Scale Complex Engineered Systems - From Research to Product Realization”. In: *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. American Institute of Aeronautics and Astronautics.
- Borgatti, S. P. and P. C. Foster (2003). “The Network Paradigm in Organizational Research: A Review and Typology”. In: *Journal of Management* 29.6, pp. 991–1013. DOI: [10.1016/S0149-2063\(03\)00087-4](https://doi.org/10.1016/S0149-2063(03)00087-4).
- Borjesson, F. and K. Hölttä-Otto (2013). “A module generation algorithm for product architecture based on component interactions and strategic drivers”. In: *Research in Engineering Design* 25.1, pp. 31–51. DOI: [10.1007/s00163-013-0164-2](https://doi.org/10.1007/s00163-013-0164-2).
- Bosch, J. and P. Bosch-Sijtsema (2010). “From integration to composition: On the impact of software product lines, global development and ecosystems”. In: *Journal of Systems and Software* 83.1, pp. 67–76. DOI: [10.1016/j.jss.2009.06.051](https://doi.org/10.1016/j.jss.2009.06.051).
- Braun, V. and V. Clarke (2006). “Using thematic analysis in psychology”. In: *Qualitative Research in Psychology* 3.2, pp. 77–101.
- Brooks, F. P. (1995). *The Mythical Man-Month: Essays on Software Engineering*. 2nd. Addison-Wesley.
- Browning, T. R. (2001). “Applying the design structure matrix to system decomposition and integration problems: a review and new directions”. In: *IEEE Transactions on Engineering Management* 48.3, pp. 292–306.
- Bryant, C. R. et al. (2005). “A Computational Technique for Concept Generation”. In: pp. 267–276. DOI: [10.1115/DETC2005-85323](https://doi.org/10.1115/DETC2005-85323).
- Burt, R. S. (1992). *Structural Holes: The Social Structure of Competition*. United States: First Harvard University Press.
- Burt, R. S. and J. Merluzzi (2014). “Embedded Brokerage: Hubs Versus Locals”. In: *Research in the Sociology of Organizations*. Ed. by D. J. Brass et al. Vol. 40. Emerald Group Publishing Limited, pp. 161–177. DOI: [10.1108/S0733-558X\(2014\)0000040008](https://doi.org/10.1108/S0733-558X(2014)0000040008).
- Cataldo, M. and J. D. Herbsleb (2008). “Communication Patterns in Geographically Distributed Software Development and Engineers’ Contributions to the Development Ef-

- fort”. In: *Proceedings of the 2008 International Workshop on Cooperative and Human Aspects of Software Engineering*. CHASE '08. New York, NY, USA: ACM, pp. 25–28. DOI: [10.1145/1370114.1370121](https://doi.org/10.1145/1370114.1370121).
- Cataldo, M., J. D. Herbsleb, and K. M. Carley (2008). “Socio-technical congruence: a framework for assessing the impact of technical and work dependencies on software development productivity”. In: *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*. ACM, pp. 2–11.
- Cataldo, M. et al. (2006). “Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools”. In: *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work*. CSCW '06. New York, NY, USA: ACM, pp. 353–362. DOI: [10.1145/1180875.1180929](https://doi.org/10.1145/1180875.1180929).
- Checkland, P. (2000). “Soft systems methodology: a thirty year retrospective”. In: *Syst. Res.* P. 48.
- Chen, W. et al. (2018). “Network-based Modeling and Analysis in Design”. In: *Design Science* 4. DOI: [10.1017/dsj.2018.8](https://doi.org/10.1017/dsj.2018.8).
- Clinebell, S. and M. Stecher (2003). “Teaching Teams to be Teams: An Exercise Using the Myers-Briggs® Type Indicator and the Five-Factor Personality Traits”. In: *Journal of Management Education* 27.3, pp. 362–383.
- Colfer, L. J. and C. Y. Baldwin (2016). “The mirroring hypothesis: theory, evidence, and exceptions”. In: *Industrial and Corporate Change*, dtw027. DOI: [10.1093/icc/dtw027](https://doi.org/10.1093/icc/dtw027).
- Collopy, A. X. et al. (2017). “Estimating the Impact of Systems Engineers on Systems Design Processes”. In: *Proceedings of the 21st International Conference on Engineering Design, ICED17*. Vol. Vol. 3: Product, Services and Systems Design. Vancouver, Canada.
- Conway, M. E. (1968). “How do Committees Invent?” In: *Datamation* 14.
- Crabtree, R. A., M. S. Fox, and N. K. Baid (1997). “Case studies of coordination activities and problems in collaborative design”. In: *Research in Engineering Design* 9.2, pp. 70–84. DOI: [10.1007/BF01596483](https://doi.org/10.1007/BF01596483).
- Cramer, E. J. et al. (1994). “Problem formulation for multidisciplinary optimization”. In: *SIAM Journal on Optimization* 4.4, pp. 754–776.
- Cross, N. (2004). “Expertise in design: an overview”. In: *Design Studies*. Expertise in Design 25.5, pp. 427–441. DOI: [10.1016/j.destud.2004.06.002](https://doi.org/10.1016/j.destud.2004.06.002).

- Cross, R., S. P. Borgatti, and A. Parker (2002). “Making Invisible Work Visible: Using Social Network Analysis to Support Strategic Collaboration”. In: *California Management Review* 44.2, pp. 25–46. DOI: [10.2307/41166121](https://doi.org/10.2307/41166121).
- Cumming, M. (2002). “Flexible and distributed coordination models for collaborative design”. In: *Connecting the Real and the Virtual - design e-education: 20th eCAADe Conference Proceedings*. Warsaw, Poland.
- Dahmus, J. B., J. P. Gonzalez-Zugasti, and K. N. Otto (2001). “Modular product architecture”. In: *Design Studies* 22.5, pp. 409–424. DOI: [10.1016/S0142-694X\(01\)00004-7](https://doi.org/10.1016/S0142-694X(01)00004-7).
- de Vries, R. E., A. Bakker-Pieper, and W. Oostenveld (2010). “Leadership = Communication? The Relations of Leaders’ Communication Styles with Leadership Styles, Knowledge Sharing and Leadership Outcomes”. In: *Journal of Business and Psychology* 25.3, pp. 367–380. DOI: [10.1007/s10869-009-9140-2](https://doi.org/10.1007/s10869-009-9140-2).
- de Weck, O. L., D. Roos, and C. L. Magee (2011). *Engineering Systems*. Cambridge, MA: MIT Press.
- Developers, S.-L. (2018). *Feature Extraction*. URL: https://scikit-learn.org/stable/modules/feature_extraction.html (visited on 04/18/2018).
- Doran, T. (2006). “IEEE 1220: for practical systems engineering”. In: *Computer* 39.5, pp. 92–94. DOI: [10.1109/MC.2006.164](https://doi.org/10.1109/MC.2006.164).
- Dossick, C. S. and G. Neff (2010). “Organizational Divisions in BIM-Enabled Commercial Construction”. In: *Journal of Construction Engineering and Management* 136.4, pp. 459–467. DOI: [10.1061/\(ASCE\)CO.1943-7862.0000109](https://doi.org/10.1061/(ASCE)CO.1943-7862.0000109).
- Dua, D. and C. Graff (2017). *UCI Machine Learning Repository*. URL: <http://archive.ics.uci.edu/ml> (visited on 01/26/2019).
- Easley, D. and J. M. Kleinberg (2010). *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge, UK: Cambridge University Press.
- Ehrlich, K. and M. Cataldo (2014). “The Communication Patterns of Technical Leaders: Impact on Product Development Team Performance”. In: *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*. CSCW ’14. New York, NY, USA: ACM, pp. 733–744. DOI: [10.1145/2531602.2531671](https://doi.org/10.1145/2531602.2531671).
- Ehrlich, K. et al. (2008). *An Analysis of Congruence Gaps and Their Effect on Distributed Software Development*.

- Epstein, J. M. and R. Axtell (1996). *Growing Artificial Societies: Social Science from the Bottom Up*. Washington, D.C.: Brookings Institution Press and the MIT Press.
- Espinosa, J. A., F. Armour, and W. F. Boh (2010). “Coordination in Enterprise Architecting: An Interview Study”. In: *2010 43rd Hawaii International Conference on System Sciences*, pp. 1–10. DOI: [10.1109/HICSS.2010.450](https://doi.org/10.1109/HICSS.2010.450).
- Esser, H. (2008). “The Two Meanings of Social Capital”. In: *The Handbook of Social Capital*. Ed. by D. Castiglione, J. W. Van Deth, and G. Wolleb. New York, NY, USA: Oxford University Press, pp. 22–49.
- Farooqui, A. D. and M. A. Niazi (2016). “Game theory models for communication between agents: a review”. In: *Complex Adaptive Systems Modeling* 4.1. DOI: [10.1186/s40294-016-0026-7](https://doi.org/10.1186/s40294-016-0026-7).
- Fernandes, J. V. et al. (2017). “Modelling the dynamics of complex early design processes: an agent-based approach”. In: *Design Science* 3. DOI: [10.1017/dsj.2017.17](https://doi.org/10.1017/dsj.2017.17).
- Fogarty, T. C. (1992). “First nearest neighbor classification on Frey and Slate’s letter recognition problem”. In: *Machine Learning* 9.4, pp. 387–388. DOI: [10.1007/BF00994113](https://doi.org/10.1007/BF00994113).
- Frank, M. (2012). “Engineering Systems Thinking: Cognitive Competencies of Successful Systems Engineers”. In: *Procedia Computer Science* 8, pp. 273–278. DOI: [10.1016/j.procs.2012.01.057](https://doi.org/10.1016/j.procs.2012.01.057).
- Frey, P. W. and D. J. Slate (1991). “Letter recognition using Holland-style adaptive classifiers”. In: *Machine Learning* 6.2, pp. 161–182. DOI: [10.1007/BF00114162](https://doi.org/10.1007/BF00114162).
- Galbraith, J. R. (1974). “Organization design: An information processing view”. In: *Interfaces* 4.3, pp. 28–36.
- Galviņa, Z. and D. Šmite (2012). “Low Degree of Separation Does Not Guarantee Easy Coordination”. In: *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*. Cesme, Izmir, Turkey: IEEE, pp. 345–348. DOI: [10.1109/SEAA.2012.79](https://doi.org/10.1109/SEAA.2012.79).
- Gao, J., B. Barzel, and A.-L. Barabási (2016). “Universal resilience patterns in complex networks”. In: *Nature* 530.7590, pp. 307–312. DOI: [10.1038/nature16948](https://doi.org/10.1038/nature16948).
- Gardner, W. L. and M. J. Martinko (1996). “Using the Myers-Briggs Type Indicator to Study Managers: A Literature Review and Research Agenda”. In: *Journal of Management* 22.1, pp. 45–83.

- Gero, J. S. (2002). “Computational models of creative designing based on situated cognition”. In: *Proceedings of the fourth conference on Creativity & cognition - C&C '02*. Loughborough, UK: ACM Press, pp. 3–10. DOI: [10.1145/581710.581712](https://doi.org/10.1145/581710.581712).
- Given, L., ed. (2008). *The SAGE Encyclopedia of Qualitative Research Methods*. Thousand Oaks, California, United States: SAGE Publications, Inc.
- Goldberg, L. R. (1990). “An Alternative "Description of Personality": The Big-Five Factor Structure”. In: *Journal of Personality and Social Psychology* 59.6, pp. 1216–1229.
- Granovetter, M. S. (1973). “The Strength of Weak Ties”. In: *American Journal of Sociology* 78.6, pp. 1360–1380.
- Greene, M. T., R. Gonzalez, and P. Y. Papalambros (2019). “Measuring Systems Engineering and Design Thinking Attitudes”. In: *Proceedings of the 22nd International Conference on Engineering Design (ICED)*. Delft, the Netherlands: Cambridge University Press.
- Greene, M. T., P. Y. Papalambros, and A.-M. R. McGowan (2016). “Position Paper: Designing Complex Systems to Support Interdisciplinary Cognitive Work”. In: *Proceedings of the DESIGN 2016 14th International Design Conference*. Dubrovnik, Croatia.
- Griffin, M. D. (2010). “How do we fix system engineering?” In: *61st Annual International Congress, Prague, Czech Republic*. Vol. 27.
- Grimm, V. et al. (2006). “A standard protocol for describing individual-based and agent-based models”. In: *Ecological Modelling* 198.1, pp. 115–126. DOI: [10.1016/j.ecolmodel.2006.04.023](https://doi.org/10.1016/j.ecolmodel.2006.04.023).
- Grogan, P. T. et al. (2018). “Multi-Actor Value Modeling for Federated Systems”. In: *IEEE Systems Journal* 12.2, pp. 1193–1202. DOI: [10.1109/JSYST.2016.2626981](https://doi.org/10.1109/JSYST.2016.2626981).
- Grogan, P. T. and O. L. de Weck (2016). “Collaboration and complexity: an experiment on the effect of multi-actor coupled design”. In: *Research in Engineering Design* 27.3, pp. 221–235. DOI: [10.1007/s00163-016-0214-7](https://doi.org/10.1007/s00163-016-0214-7).
- Grubb, A. M. and A. Begel (2012). “On the perceived interdependence and information sharing inhibitions of enterprise software engineers”. In: *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. ACM, pp. 1337–1346.
- Gulati, R. and H. Singh (1998). “The Architecture of Cooperation: Managing Coordination Costs and Appropriation Concerns in Strategic Alliances”. In: *Administrative Science Quarterly* 43.4, p. 781. DOI: [10.2307/2393616](https://doi.org/10.2307/2393616).

- Hajela, P., C. L. Bloebaum, and J. Sobieszczanski-Sobieski (1990). “Application of global sensitivity equations in multidisciplinary aircraft synthesis”. In: *Journal of Aircraft* 27.12, pp. 1002–1010. DOI: [10.2514/3.45974](https://doi.org/10.2514/3.45974).
- Hazelrigg, G. A. (1996). *Systems Engineering: An Approach to Information-Based Design*. Upper Saddle River, NJ: Prentice-Hall.
- Herbsleb, J. D. and R. E. Grinter (1999). “Architectures, coordination, and distance: Conway’s law and beyond”. In: *IEEE Software* 16.5.
- Herbsleb, J. D. and A. Mockus (2003). “An empirical study of speed and communication in globally distributed software development”. In: *IEEE Transactions on Software Engineering* 29.6, pp. 481–494. DOI: [10.1109/TSE.2003.1205177](https://doi.org/10.1109/TSE.2003.1205177).
- Herbsleb, J. D. (2007). “Global software engineering: The future of socio-technical coordination”. In: *2007 Future of Software Engineering*. IEEE Computer Society, pp. 188–198.
- Heydari, B. and M. J. Pennock (2018). “Guiding the behavior of sociotechnical systems: The role of agent-based modeling”. In: *Systems Engineering* 21.3, pp. 210–226. DOI: [10.1002/sys.21435](https://doi.org/10.1002/sys.21435).
- Honda, T. et al. (2015). “Comparison of Information Passing Strategies in System-Level Modeling”. In: *AIAA Journal* 53.5, pp. 1121–1133. DOI: [10.2514/1.J052568](https://doi.org/10.2514/1.J052568).
- Hutchison, N., D. Henry, and A. Pyster (2016). “Atlas : Understanding What Makes Systems Engineers Effective in the U.S. Defense Community”. In: *Systems Engineering* 19.6, pp. 510–521. DOI: [10.1002/sys.21372](https://doi.org/10.1002/sys.21372).
- International Council on Systems Engineering (INCOSE) (2004). *Systems Engineering Handbook*.
- Johnson, S. B. (1997). “Three Approaches to Big Technology: Operations Research, Systems Engineering, and Project Management”. In: *Technology and Culture* 38.4, pp. 891–919. DOI: [10.2307/3106953](https://doi.org/10.2307/3106953).
- (2002). *The United States Air Force and the Culture of Innovation, 1945-1965*. Washington, DC, USA: Air Force History and Museums Center.
- Jung, D. F. and D. A. Lake (2011). “Markets, Hierarchies, and Networks: An Agent-Based Organizational Ecology: Markets, Hierarchies, and Networks”. In: *American Journal of Political Science* 55.4, pp. 972–990. DOI: [10.1111/j.1540-5907.2011.00536.x](https://doi.org/10.1111/j.1540-5907.2011.00536.x).

- Kannan, H., B. L. Mesmer, and C. L. Bloebaum (2017). “Increased System Consistency through Incorporation of Coupling in Value-Based Systems Engineering”. In: *Systems Engineering* 20.1, pp. 21–44. DOI: [10.1002/sys.21377](https://doi.org/10.1002/sys.21377).
- Kim, H. M. et al. (2003). “Target Cascading in Optimal System Design”. In: *Journal of Mechanical Design* 125.3, p. 474. DOI: [10.1115/1.1582501](https://doi.org/10.1115/1.1582501).
- Kleinbaum, A. M., T. Stuart, and M. Tushman (2008). *Communication (and coordination?) in a modern, complex organization*.
- Kosti, M. V., R. Feldt, and L. Angelis (2014). “Personality, emotional intelligence and work preferences in software engineering: An empirical study”. In: *Information and Software Technology* 56.8, pp. 973–990. DOI: [10.1016/j.infsof.2014.03.004](https://doi.org/10.1016/j.infsof.2014.03.004).
- Kraut, R. E. and L. A. Streeter (1995). “Coordination in Software Development”. In: *Commun. ACM* 38.3, pp. 69–81. DOI: [10.1145/203330.203345](https://doi.org/10.1145/203330.203345).
- Krebs, V. (2000). *InFlow Survey*. URL: <http://web-beta.archive.org/web/20070418102219/http://www.orgnet.com/INSNA/survey.html> (visited on 04/04/2017).
- Kress, G. L. and M. Shar (2012). “Teamology - The Art and Science of Design Team Formation”. In: *Design Thinking Research: Studying Co-Creation in Practice*. Ed. by H. Plattner, C. Meinel, and L. Leifer. Understanding Innovation. Springer, pp. 189–209.
- Krishnamachari, R. (1996). “A decomposition synthesis methodology for optimal systems design”. Ph.D. Dissertation. Ann Arbor, MI: University of Michigan.
- Kroo, I. and V. Manning (2000). “Collaborative optimization: status and directions”. In: *Presented at the 8th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. Vol. 6, p. 8.
- Kullback, S. and R. A. Leibler (1951). “On Information and Sufficiency”. In: *The Annals of Mathematical Statistics* 22.1, pp. 79–86. DOI: [10.1214/aoms/1177729694](https://doi.org/10.1214/aoms/1177729694).
- Kwan, I., A. Schroter, and D. Damian (2011). “Does Socio-Technical Congruence Have an Effect on Software Build Success? A Study of Coordination in a Software Project”. In: *IEEE Transactions on Software Engineering* 37.3, pp. 307–324. DOI: [10.1109/TSE.2011.29](https://doi.org/10.1109/TSE.2011.29).
- Lancichinetti, A., S. Fortunato, and F. Radicchi (2008). “Benchmark graphs for testing community detection algorithms”. In: *Physical Review E* 78.4. DOI: [10.1103/PhysRevE.78.046110](https://doi.org/10.1103/PhysRevE.78.046110).

- Langevin, J., J. Wen, and P. L. Gurian (2015). "Simulating the human-building interaction: Development and validation of an agent-based model of office occupant behaviors". In: *Building and Environment* 88, pp. 27–45. DOI: [10.1016/j.buildenv.2014.11.037](https://doi.org/10.1016/j.buildenv.2014.11.037).
- Larsson, A. (2007). "Banking on social capital: towards social connectedness in distributed engineering design teams". In: *Design Studies* 28, pp. 605–622.
- Lawrence, P. R. and J. W. Lorsch (1967). *Organization and Environment: Managing Differentiation and Integration*. Homewood, Illinois: Richard D. Irwin, Inc.
- Le, Q. and J. H. Panchal (2012). "Analysis of the interdependent co-evolution of product structures and community structures using dependency modelling techniques". In: *Journal of Engineering Design* 23.10-11, pp. 807–828. DOI: [10.1080/09544828.2012.695014](https://doi.org/10.1080/09544828.2012.695014).
- Levine, S. S. and M. J. Prietula (2011). "How Knowledge Transfer Impacts Performance: A Multilevel Model of Benefits and Liabilities". In: *Organization Science* 23.6, pp. 1748–1766. DOI: [10.1287/orsc.1110.0697](https://doi.org/10.1287/orsc.1110.0697).
- Levitt, R. E. (2012). "The Virtual Design Team: Designing Project Organizations as Engineers Design Bridges". In: *Journal of Organization Design* 1.2, pp. 14–41. DOI: [10.7146/jod.6345](https://doi.org/10.7146/jod.6345).
- MacCormack, A., C. Baldwin, and J. Rusnak (2012). "Exploring the duality between product and organizational architectures: A test of the "mirroring" hypothesis". In: *Research Policy* 41.8, pp. 1309–1324. DOI: [10.1016/j.respol.2012.04.011](https://doi.org/10.1016/j.respol.2012.04.011).
- Mackay, D. and M. Zundel (2017). "Recovering the Divide: A Review of Strategy and Tactics in Business and Management". In: *International Journal of Management Reviews* 19.2, pp. 175–194. DOI: [10.1111/ijmr.12091](https://doi.org/10.1111/ijmr.12091).
- Madni, A. M. and M. Sievers (2014). "Systems Integration: Key Perspectives, Experiences, and Challenges". In: *Systems Engineering* 17.1, pp. 37–51. DOI: [10.1002/sys.21249](https://doi.org/10.1002/sys.21249).
- Maier, A. M. et al. (2008). "Exploration of Correlations between Factors Influencing Communication in Complex Product Development". In: *Concurrent Engineering* 16.1, pp. 37–59. DOI: [10.1177/1063293X07084638](https://doi.org/10.1177/1063293X07084638).
- Maier, M. W. and E. Rechtin (2009). *The Art of System Architecting*. 3rd ed. Boca Raton, FL: CRC Press.
- Malone, T. W. (1987). "Modeling Coordination in Organizations and Markets". In: *Management Science* 33.10, pp. 1317–1332. DOI: [10.1287/mnsc.33.10.1317](https://doi.org/10.1287/mnsc.33.10.1317).

- Malone, T. W. and K. Crowston (1994). “The interdisciplinary study of coordination”. In: *ACM Computing Surveys* 26.1, pp. 87–119. DOI: [10.1145/174666.174668](https://doi.org/10.1145/174666.174668).
- Manning, C. D., P. Raghavan, and H. Schütze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.
- March, J. G. and H. A. Simon (1958). *Organizations*. United States: John Wiley & Sons, Inc.
- Martins, J. R. R. A. and A. B. Lambe (2013). “Multidisciplinary Design Optimization: A Survey of Architectures”. In: *AIAA Journal* 51.9, pp. 2049–2075. DOI: [10.2514/1.J051895](https://doi.org/10.2514/1.J051895).
- McComb, C., J. Cagan, and K. Kotovsky (2015). “Lifting the Veil: Drawing insights about design teams from a cognitively-inspired computational model”. In: *Design Studies* 40, pp. 119–142. DOI: [10.1016/j.destud.2015.06.005](https://doi.org/10.1016/j.destud.2015.06.005).
- McCord, K. R. and S. Eppinger (1993). “Managing the integration problem in concurrent engineering”. WP# 3594-93-MSA.
- McEvily, B., G. Soda, and M. Tortoriello (2014). “More Formally: Rediscovering the Missing Link between Formal Organization and Informal Social Structure”. In: *The Academy of Management Annals* 8.1, pp. 299–345. DOI: [10.1080/19416520.2014.885252](https://doi.org/10.1080/19416520.2014.885252).
- McGowan, A.-M. R. (2014). “Interdisciplinary Interactions During R&D and Early Design of Large Engineered Systems”. PhD thesis. Ann Arbor, MI: University of Michigan.
- Meluso, J. and J. Austin-Breneman (2018). “Gaming the System: An Agent-Based Model of Estimation Strategies and their Effects on System Performance”. In: *Journal of Mechanical Design* 140.12, p. 9. DOI: [10.1115/1.4039494](https://doi.org/10.1115/1.4039494).
- Metzger, L. S. and L. R. Bender (2007). *MITRE Systems Engineering (SE) Competency Model*. Tech. rep. The MITRE Corporation.
- Miller, J. H. and S. E. Page (2007). *Complex Adaptive Systems: An Introduction to Computational Models of Social Life*. Princeton Studies in Complexity. Princeton, NJ: Princeton University Press.
- Mirzakhilili, E. et al. (2017). “Synaptic Impairment and Robustness of Excitatory Neuronal Networks with Different Topologies”. In: *Frontiers in Neural Circuits* 11, p. 38. DOI: [10.3389/fncir.2017.00038](https://doi.org/10.3389/fncir.2017.00038).
- Mosleh, M., P. Ludlow, and B. Heydari (2016). “Resource allocation through network architecture in systems of systems: A complex networks framework”. In: *2016 Annual*

- IEEE Systems Conference (SysCon)*, pp. 1–5. DOI: [10 . 1109 / SYSCON . 2016 . 7490629](https://doi.org/10.1109/SYSCON.2016.7490629).
- Mount, M. K., M. R. Barrick, and G. L. Stewart (1998). “Five-Factor Model of personality and Performance in Jobs Involving Interpersonal Interactions”. In: *Human Performance* 11.2-3, pp. 145–165. DOI: [10 . 1080 / 08959285 . 1998 . 9668029](https://doi.org/10.1080/08959285.1998.9668029).
- Myers, I. B. and P. B. Myers (1980). *Gifts Differing: Understanding Personality Type*. 1st. Mountain View, CA: Davies-Black Publishing.
- National Aeronautical and Space Administration (2007). *NASA Systems Engineering Handbook*.
- National Science Board (2018). *Science and Engineering Indicators 2018*. Tech. rep. NSB-2018-1. Alexandria, VA: National Science Foundation.
- Newman, M. (2003). “The Structure and Function of Complex Networks”. In: *SIAM Review* 45.2, pp. 167–256. DOI: [10 . 1137 / S003614450342480](https://doi.org/10.1137/S003614450342480).
- Newman, M. E. J. (2006). “Modularity and community structure in networks”. In: *Proceedings of the National Academy of Sciences* 103.23, pp. 8577–8582. DOI: [10 . 1073 / pnas . 0601602103](https://doi.org/10.1073/pnas.0601602103).
- Newman, M. E. J. (2018). *Networks: An Introduction*. 2nd. New York, NY: Oxford University Press.
- Novoselich, B. J. and D. B. Knight (2018). “Shared Leadership in Capstone Design Teams: Social Network Analysis”. In: *Journal of Professional Issues in Engineering Education and Practice* 144.4, p. 04018006. DOI: [10 . 1061 / \(ASCE \) EI . 1943 - 5541 . 0000376](https://doi.org/10.1061/(ASCE)EI.1943-5541.0000376).
- Nowell, L. S. et al. (2017). “Thematic Analysis: Striving to Meet the Trustworthiness Criteria”. In: *International Journal of Qualitative Methods* 16.1, p. 1609406917733847. DOI: [10 . 1177 / 1609406917733847](https://doi.org/10.1177/1609406917733847).
- Obstfeld, D., S. P. Borgatti, and J. Davis (2014). “Brokerage as a Process: Decoupling Third Party Action from Social Network Structure”. In: *Research in the Sociology of Organizations*. Ed. by D. J. Brass et al. Vol. 40. Emerald Group Publishing Limited, pp. 135–159. DOI: [10 . 1108 / S0733 - 558X \(2014 \) 0000040007](https://doi.org/10.1108/S0733-558X(2014)0000040007).
- Ogata, H. et al. (2001). “Computer Supported Social Networking For Augmenting Cooperation”. In: *Computer Supported Cooperative Work (CSCW)* 10.2, pp. 189–209. DOI: [10 . 1023 / A : 1011216431296](https://doi.org/10.1023/A:1011216431296).

- Olson, G. M. and J. S. Olson (2000). “Distance matters”. In: *Human-computer interaction* 15.2, pp. 139–178.
- Page, S. E. (2015). “What Sociologists Should Know About Complexity”. In: *Annual Review of Sociology* 41.1, pp. 21–41. DOI: [10 . 1146 / annurev - soc - 073014 - 112230](https://doi.org/10.1146/annurev-soc-073014-112230).
- Panchal, J. H. (2010). “Coordination in Collective Product Innovation”. In: pp. 333–346. DOI: [10 . 1115 / IMECE2010 - 37116](https://doi.org/10.1115/IMECE2010-37116).
- Panjer, L. D., D. Damian, and M.-A. Storey (2008). “Cooperation and Coordination Concerns in a Distributed Software Development Project”. In: *Proceedings of the 2008 International Workshop on Cooperative and Human Aspects of Software Engineering*. CHASE '08. New York, NY, USA: ACM, pp. 77–80. DOI: [10 . 1145 / 1370114 . 1370134](https://doi.org/10.1145/1370114.1370134).
- Papalambros, P. Y. and D. J. Wilde (2017). *Principles of Optimal Design: Modeling and Computation*. 3rd ed. Cambridge University Press.
- Papalambros, P. Y. (2015). “Design Science in Design Education”. In: *XI Mudd Design Workshop on Design Thinking in Design Education*. Claremont, CA, pp. 28–30.
- Parnas, D. L. (1972). “On the criteria to be used in decomposing systems into modules”. In: *Communications of the ACM* 15.12, pp. 1053–1058.
- Parraguez, P., S. Eppinger, and A. Maier (2016). “Characterizing Design Process Interfaces as Organization Networks: Insights for Engineering Systems Management”. In: *Systems Engineering* 19.2, pp. 158–173. DOI: [10 . 1002 / sys . 21345](https://doi.org/10.1002/sys.21345).
- Parraguez, P. and A. Maier (2016). “Using Network Science to Support Design Research: From Counting to Connecting”. In: *Experimental Design Research*. Ed. by P. Cash, T. Stanković, and M. Štorga. Springer International Publishing, pp. 153–172. DOI: [10 . 1007 / 978 - 3 - 319 - 33781 - 4 _ 9](https://doi.org/10.1007/978-3-319-33781-4_9).
- Patton, M. Q. (2015). *Qualitative Research and Evaluation Methods*. 4th edition. United States: Sage Publications, Inc.
- Pennock, M. J. and W. B. Rouse (2016). “The Epistemology of Enterprises”. In: *Systems Engineering*. DOI: [10 . 1002 / sys . 21335](https://doi.org/10.1002/sys.21335).
- Piccolo, S. A., S. Lehmann, and A. Maier (2018). “Design process robustness: a bipartite network analysis reveals the central importance of people”. In: *Design Science* 4. DOI: [10 . 1017 / dsj . 2017 . 32](https://doi.org/10.1017/dsj.2017.32).

- Pietrzyk, V. J. and H. A. H. Handley (2016). “Outcome-based competency model for systems engineering training”. In: *2016 IEEE International Symposium on Systems Engineering (ISSE)*. Edinburgh, United Kingdom: IEEE, pp. 1–7. DOI: [10.1109/SysEng.2016.7753168](https://doi.org/10.1109/SysEng.2016.7753168).
- Pimmler, T. U. and S. D. Eppinger (1994). “Integration Analysis of Product Decompositions”. In: Minneapolis, MN: ASME.
- Poleacovschi, C. and A. Javernick-Will (2016). “Spanning Information and Knowledge across Subgroups and Its Effects on Individual Performance”. In: *Journal of Management in Engineering* 32.4, p. 04016006. DOI: [10.1061/\(ASCE\)ME.1943-5479.0000423](https://doi.org/10.1061/(ASCE)ME.1943-5479.0000423).
- Powell, W. W. (1990). “Neither Market nor Hierarchy: Network Forms of Organization”. In: *Research in Organizational Behavior* 12, pp. 295–336.
- Pugh, D. S. and D. J. Hickson (2007). “The Organization in Its Environment”. In: *Writers on Organizations*. 6th. London: Sage, p. 55.
- Railsback, S. F. and V. Grimm (2012). *Agent-based and Individual-based Modeling: A Practical Introduction*. 1st. Princeton, NJ: Princeton University Press.
- Reyer, J. A. and P. Y. Papalambros (2002). “Combined Optimal Design and Control With Application to an Electric DC Motor”. In: *Journal of Mechanical Design* 124.2, p. 183. DOI: [10.1115/1.1460904](https://doi.org/10.1115/1.1460904).
- Ryschkewitsch, M., D. Schaible, and W. Larson (2009). “The art and science of systems engineering”. In: *Systems Research Forum* 3.2, pp. 81–100.
- Sage, A. P. and C. L. Lynch (1998). “Systems integration and architecting: An overview of principles, practices, and perspectives”. In: *Systems Engineering* 1.3, pp. 176–227. DOI: [10.1002/\(SICI\)1520-6858\(1998\)1:3<176::AID-SYS3>3.0.CO;2-L](https://doi.org/10.1002/(SICI)1520-6858(1998)1:3<176::AID-SYS3>3.0.CO;2-L).
- “Cohen’s Kappa” (2010). In: *Encyclopedia of Research Design*. Ed. by N. Salkind. Thousand Oaks, California, United States: SAGE Publications, Inc.
- Salton, G. and M. J. McGill (1986). *Introduction to Modern Information Retrieval*. New York, NY, USA: McGraw-Hill, Inc.
- Salzberg, S. and M. Watkins (2016). “Managing information for concurrent engineering: Challenges and barriers”. In: *Research in Engineering Design* 2.1, pp. 35–52. DOI: [10.1007/BF02029820](https://doi.org/10.1007/BF02029820).

- Schmidt, L. C. and J. Cagan (1998). “Optimal Configuration Design: An Integrated Approach Using Grammars”. In: *Journal of Mechanical Design* 120.1, p. 2. DOI: [10.1115/1.2826672](https://doi.org/10.1115/1.2826672).
- Scott, W. R. and G. F. Davis (2006). *Organizations and Organizing: Rational, Natural and Open Systems Perspectives*. 1 edition. Upper Saddle River, N.J: Routledge.
- Senior, B. (1997). “Team roles and team performance: Is there ‘really’ a link?” In: *Journal of Occupational and Organizational Psychology* 70.3, pp. 241–258. DOI: [10.1111/j.2044-8325.1997.tb00646.x](https://doi.org/10.1111/j.2044-8325.1997.tb00646.x).
- Sheard, S. A. (1996). “Twelve systems engineering roles”. In: *INCOSE International Symposium*. Vol. 6. Wiley Online Library, pp. 478–485.
- Sheard, S. et al. (2015). *A Complexity Primer for Systems Engineers*.
- Simon, H. A. (1955). “A Behavioral Model of Rational Choice”. In: *The Quarterly Journal of Economics* 69.1, pp. 99–118. DOI: [10.2307/1884852](https://doi.org/10.2307/1884852).
- (1973). “Applying Information Technology to Organization Design”. In: *Public Administration Review* 33.3, p. 268. DOI: [10.2307/974804](https://doi.org/10.2307/974804).
- Simpson, T. W. and J. R. Martins (2011). “Multidisciplinary design optimization for complex engineered systems: report from a national science foundation workshop”. In: *Journal of Mechanical Design* 133.10, p. 101002.
- Smith, R. P. and A. Leong (1998). “An Observational Study of Design Team Process: A Comparison of Student and Professional Engineers”. In: *Journal of Mechanical Design* 120.4, p. 636. DOI: [10.1115/1.2829326](https://doi.org/10.1115/1.2829326).
- Sobieszcanski-Sobieski, J. (1995). “Multidisciplinary design optimization: an emerging new engineering discipline”. In: *Advances in Structural Optimization*. Springer, pp. 483–496.
- Sonnenwald, D. H. (1996). “Communication roles that support collaboration during the design process”. In: *Design Studies* 17.3, pp. 277–301. DOI: [10.1016/0142-694X\(96\)00002-6](https://doi.org/10.1016/0142-694X(96)00002-6).
- Soria Zurita, N. F. et al. (2017). “Design of Complex Engineered Systems Using Multi-Agent Coordination”. In: *Journal of Computing and Information Science in Engineering* 18.1, pp. 011003–011003–13. DOI: [10.1115/1.4038158](https://doi.org/10.1115/1.4038158).
- Sosa, M. (2007). “Aligning process, product, and organizational architectures in software development”. In: *International Conference on Engineering Design, ICED*. Vol. 7, pp. 28–31.

- Sosa, M. E., S. D. Eppinger, and C. M. Rowles (2003). “Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions”. In: *Journal of Mechanical Design* 125.2, pp. 240–252. DOI: [10.1115/1.1564074](https://doi.org/10.1115/1.1564074).
- (2004). “The Misalignment of Product Architecture and Organizational Structure in Complex Product Development”. In: *Management Science* 50.12, pp. 1674–1689. DOI: [10.1287/mnsc.1040.0289](https://doi.org/10.1287/mnsc.1040.0289).
- Souza, C. R. de et al. (2004). “How a good software practice thwarts collaboration: the multiple roles of APIs in software development”. In: *ACM SIGSOFT Software Engineering Notes* 29.6, pp. 221–230.
- Stevenson, A. and C. A. Lindberg, eds. (2011). *New Oxford American Dictionary*. 3rd. Oxford University Press.
- Steward, D. V. (1981). “The design structure system: A method for managing the design of complex systems”. In: *IEEE Transactions on Engineering Management* EM-28.3, pp. 71–74. DOI: [10.1109/TEM.1981.6448589](https://doi.org/10.1109/TEM.1981.6448589).
- Stewart, G. L., I. S. Fulmer, and M. R. Barrick (2005). “An Exploration of Member Roles as a Multilevel Linking Mechanism for Individual Traits and Team Outcomes”. In: *Personnel Psychology* 58.2, pp. 343–365. DOI: [10.1111/j.1744-6570.2005.00480.x](https://doi.org/10.1111/j.1744-6570.2005.00480.x).
- Strode, D. E. et al. (2012). “Coordination in co-located agile software development projects”. In: *Journal of Systems and Software*. Special Issue: Agile Development 85.6, pp. 1222–1238. DOI: [10.1016/j.jss.2012.02.017](https://doi.org/10.1016/j.jss.2012.02.017).
- Strogatz, S. H. (2001). “Exploring complex networks”. In: *Nature* 410.6825, pp. 268–276. DOI: [10.1038/35065725](https://doi.org/10.1038/35065725).
- Temdee, P. and L. Korba (2001). “Of networks, interactions and agents: an approach for social network analysis”. In: *Proceedings of the Sixth International Conference on Computer Supported Cooperative Work in Design (IEEE Cat. No.01EX472)*. London, Ont., Canada: NRC Res. Press, pp. 324–329. DOI: [10.1109/CSCWD.2001.942280](https://doi.org/10.1109/CSCWD.2001.942280).
- Terrasi, E. M. (2015). “Leaders who care: Exploring empathy as an essential trait in 21st century corporate leadership”. MA thesis. Ypsilanti, MI: Eastern Michigan University.
- Thompson, J. D. (1967). *Organizations in Action: Social Science Bases of Administrative Theory*. New York, NY, USA: McGraw-Hill, Inc.
- Thorndike, R. L. (1953). “Who belongs in the family?” In: *Psychometrika* 18.4, pp. 267–276. DOI: [10.1007/BF02289263](https://doi.org/10.1007/BF02289263).

- Tichy, N. M., M. L. Tushman, and C. Fombrun (1979). “Social Network Analysis for Organizations”. In: *Academy of Management Review* 4.4. DOI: <https://doi.org/10.5465/amr.1979.4498309>.
- Tóth, B. J. et al. (2018). “Emergence of Leader-Follower Hierarchy Among Players in an On-Line Experiment”. In: *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 1184–1190. DOI: [10.1109/ASONAM.2018.8508278](https://doi.org/10.1109/ASONAM.2018.8508278).
- Triantis, K. P. and P. D. Collopy (2014). “A comprehensive basis for systems engineering theory”. In: *2014 IEEE International Systems Conference Proceedings*, pp. 97–102. DOI: [10.1109/SysCon.2014.6819242](https://doi.org/10.1109/SysCon.2014.6819242).
- Tushman, M. L. (1979). “Work Characteristics and Subunit Communication Structure: A Contingency Analysis”. In: *Administrative Science Quarterly* 24.1, p. 82. DOI: [10.2307/2989877](https://doi.org/10.2307/2989877).
- United States Department of Defense (2017). “Systems Engineering”. In: *Defense Acquisition Guidebook*. Washington, DC: US Department of Defense.
- Valetto, G. et al. (2007). “Using Software Repositories to Investigate Socio-technical Congruence in Development Projects”. In: *Proceedings of the Fourth International Workshop on Mining Software Repositories. MSR '07*. Washington, DC, USA: IEEE Computer Society, p. 25. DOI: [10.1109/MSR.2007.33](https://doi.org/10.1109/MSR.2007.33).
- Van De Ven, A. H., A. L. Delbecq, and R. Koenig (1976). “Determinants of Coordination Modes within Organizations”. In: *American Sociological Review* 41.2, p. 322. DOI: [10.2307/2094477](https://doi.org/10.2307/2094477).
- Van Deth, J. W. (2008). “Measuring Social Capital”. In: *The Handbook of Social Capital*. Ed. by D. Castiglione, J. W. Van Deth, and G. Wolleb. New York, NY, USA: Oxford University Press, pp. 150–176.
- Varvel, T. et al. (2004). “Team Effectiveness and Individual Myers-Briggs Personality Dimensions”. In: *Journal of Management in Engineering* 20.4, pp. 141–146. DOI: [10.1061/\(ASCE\)0742-597X\(2004\)20:4\(141\)](https://doi.org/10.1061/(ASCE)0742-597X(2004)20:4(141)).
- Vermillion, S. D. and R. J. Malak (2015). “Using a Principal-Agent Model to Investigate Delegation in Systems Engineering”. In: *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, V01BT02A046–V01BT02A046.
- Vroljik, A. and Z. Szajnarfarber (2015). “When Policy Structures Technology: Balancing upfront decomposition and in-process coordination in Europe’s decentralized space

- technology ecosystem”. In: *Acta Astronautica* 106, pp. 33–46. DOI: [10.1016/j.actaastro.2014.10.017](https://doi.org/10.1016/j.actaastro.2014.10.017).
- Wagner, T. and P. Papalambros (1993). “A general framework for decomposition analysis in optimal design”. In: *Advances in Design Automation*. Vol. DE 65-2. Albuquerque, NM: American Society of Mechanical Engineers.
- Wasserman, S. and K. Faust (1994). *Social Network Analysis: Methods and Applications*. Cambridge ; New York: Cambridge University Press.
- Watts, D. J. and S. H. Strogatz (1998). “Collective dynamics of ‘small-world’ networks”. In: *Nature* 393, pp. 440–442.
- Weick, K. E., K. M. Sutcliffe, and D. Obstfeld (2005). “Organizing and the Process of Sensemaking”. In: *Organization Science* 16.4, pp. 409–421. DOI: [10.4337/9781849807630.00024](https://doi.org/10.4337/9781849807630.00024).
- Wilensky, U. and W. Rand (2015). *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo*. United States: MIT Press.
- Williams, C. and M.-E. Derro (2008). *NASA Systems Engineering Behavior Study*. Tech. rep. National Aeronautics and Space Administration.
- Woodcock, H., ed. (2010). *Systems Engineering Competency Framework*. INCOSE UK.
- Yao, W. et al. (2011). “Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles”. In: *Progress in Aerospace Sciences* 47.6, pp. 450–479. DOI: [10.1016/j.paerosci.2011.05.001](https://doi.org/10.1016/j.paerosci.2011.05.001).
- Zimmermann, T. and N. Nagappan (2008). “Predicting defects using network analysis on dependency graphs”. In: *Proceedings of the 30th international conference on Software Engineering*. ACM, pp. 531–540.