Multilayer Network Modeling of Change Propagation for Engineering Change Management

by

Michael C. Pasqual

B.S.E., Electrical Engineering
Princeton University, 2005

Submitted to the Engineering Systems Division and the Department of Aeronautics and Astronautics in Partial Fulfillment of the Requirements for the Degrees of

Master of Science in Technology and Policy
and
Master of Science in Aeronautics and Astronautics

at the
Massachusetts Institute of Technology
June 2010

Signature of Author ....
Engineering Systems Division
May 7, 2010

Certified by ....
Olivier L. de Weck
Associate Professor of Aeronautics and Astronautics and Engineering Systems
Associate Director, Engineering Systems Division
Thesis Supervisor

Accepted by ....
Eytan H. Modiano
Associate Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Accepted by ....
Dava J. Newman
Professor of Aeronautics and Astronautics and Engineering Systems
Director, Technology and Policy Program

This page has been intentionally left blank.

Disclaimer

This work is sponsored by the Department of the Air Force under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

This page has been intentionally left blank.

Multilayer Network Modeling of Change Propagation
for Engineering Change Management

by

Michael C. Pasqual

Submitted to the Engineering Systems Division and the
Department of Aeronautics and Astronautics on May 7, 2010
in Partial Fulfillment of the Requirements for the Degrees of

Master of Science in Technology and Policy
and
Master of Science in Aeronautics and Astronautics

ABSTRACT

Engineering change management is a critical and challenging process within product development. One pervasive source of difficulty for this process is the phenomenon of change propagation, by which a change to one part or element of a design requires additional changes throughout the product. Research efforts to understand and manage change propagation have largely drawn on network analysis. This thesis builds upon past research by introducing a multilayer network model that incorporates three proposed layers, or domains, that contribute to change propagation: namely, the product layer, change layer, and social layer. Each layer contains a distinct network of nodes and intra-layer edges, but also connects to the other two layers through inter-layer dependencies. The model facilitates extensive quantitative analysis of change propagation using a repository of single-layer, double-layer, and triple-layer tools and metrics. A case study of a large technical program, which managed over 41,000 change requests in eight years, is employed to demonstrate the practical utility of the model. Most significantly, the case study explores the program's social layer and discovers a real-world correspondence between an engineer's organizational role and the propagation effects of his or her work, as measured by the newly proposed Engineer Change Propagation Index (Engineer-CPI). The case study also reveals that parent-child propagation often spanned more than one, but never more than three, system interfaces, thus confirming the possibility of indirect propagation. Finally, the study finds that propagation always stopped after five, and rarely more than four, generations of descendants. In all, the multilayer network model's holistic approach has significant policy implications for engineering change management in industry.

Thesis Supervisor: Olivier L. de Weck
Title: Associate Professor of Aeronautics and Astronautics and Engineering Systems

This page has been intentionally left blank.

Acknowledgements

My time at MIT would not have been nearly as successful, exciting, or enjoyable without the help and support of the following people.

First, I would like to thank Oli (de Weck) for accepting me into the Strategic Engineering Research Group and handing me a research topic that has fascinated me from the very beginning. Oli's guidance, enthusiasm, and encouragement were invaluable throughout this research endeavor.

I would also like to thank Monica Giffin and Gergana Bounova for their willingness to share their expertise and experience with me.

I must additionally thank the Group 38 Office at MIT Lincoln Laboratory for their financial, intellectual, and technical support.

I would also like to thank my beautiful and loving fiancée Marianne for supporting me in every way throughout my MIT career. I owe the same love and gratitude to her mother, father, and sister, also known as my Boston family.

Last but not least, I would like to thank my mother, father, and two brothers for their loving support throughout graduate school and my entire life before MIT. I owe all of my life's successes, past, present, and future, to their constant love, guidance, and encouragement.

This page has been intentionally left blank.

Table of Contents

List of Figures

List of Tables

This page has been intentionally left blank.

List of Acronyms

CAD	Computer-aided Design
CAI	Change Acceptance Index
CCB	Configuration Control Board
CPI	Change Propagation Index
CPFM	Change Propagation Frequency Matrix
CPM	Change Prediction Method
CR	Change request
CRI	Change Request Index
DSM	Design Structure Matrix
DMM	Domain Mapping Matrix
ECM	Engineering Change Management
ESM	Engineering Systems Matrix
IPT	Integrated Program Team
IT	Information Technology
NASA	National Aeronautics and Space Administration
PAR	Proposal Acceptance Rate
PD	Propagation Directness
PDM	Product Data Management
SAP	System Adjustable Parameter

This page has been intentionally left blank.

1. Introduction

The design of a complex product is rarely, if ever, straightforward or permanent. In fact, an organization is bound to make design changes throughout the development and subsequent operation of almost any product. Some changes are required to fix problems that are discovered with a baseline design. Others are necessary when a product must evolve to satisfy new requirements levied by the customer or market [13]. As such, changes theoretically present opportunities for an organization to innovate and please customers. However, in practice, changes can consume considerable time, money, and resources [18]. Although industry has come to accept change as a reality of product development, the management of change is hardly trivial, especially for large and complex products or systems. Indeed, engineering change management must balance the costs, benefits, and risks of making design changes, in light of their implications for schedule, budget, and product quality [16, 18].

One pervasive source of difficulty for engineering change management is the phenomenon of change propagation, by which a change to one part or element of a design spreads, or propagates, throughout the product [17, 32]. For instance, when designing a dining room table, an increase in the tabletop length may require a stronger type of wood to withstand the increased load; in other words, the first change (i.e., a longer tabletop) may give rise to an additional change (i.e., a different type of wood). Likewise, when designing an Earth-observing satellite, an increase in orbital altitude may require an increase in aperture diameter to maintain resolution, which may require a longer focal length or smaller pixels to maintain image quality [33]. Finally, when designing a software system, a new algorithm in one function may require a new input from a second function, which may require additional memory allocation for a new calculation.

Change propagation, as in these examples, occurs by virtue of the complex dependencies among the parts of modern products and systems. As a result, an organization must account for possible propagation effects when evaluating and implementing a change to a single part of a product. The topic of change propagation has received considerable research attention over the last decade. The highlights of the literature include qualitative and quantitative efforts to characterize [13, 14, 16, 17, 32], predict [6], and prevent [31] change propagation, primarily through the use of network analysis.

Building on these efforts, this thesis develops a novel *multilayer network model* of change propagation, and seeks to determine and demonstrate its utility. Multilayer network models have been employed in broader research on product development and project management. However, to the author's knowledge, no previous research on change propagation has, at least explicitly, taken a multilayer network approach.

The multilayer network model developed by this thesis is composed of three layers, or domains, that contribute to the phenomenon of change propagation: namely, the product layer, change layer, and social layer. An illustration of the model appears in Figure 1. Each layer contains a distinct network of *nodes* and *intra-layer edges*. The *product layer* consists of components connected through technical interfaces, depending on how the components interact in the product. The *change layer* consists of change requests connected through change propagation relationships, depending on how some changes give rise to others. Finally, the *social layer* consists of engineers connected through communication links, depending on the intended or actual communication between engineers. As represented by the *inter-layer* edges, engineers in the social layer work on changes in the change layer which affect components in the product layer.

Figure 1 shows a generic illustration of the multilayer network model. This thesis additionally develops a baseline repository of tools and metrics to help make sense of the model's complexity. As such, the multilayer network model lends itself to extensive quantitative analysis of change propagation and engineering change management for practical use in both academia and industry.

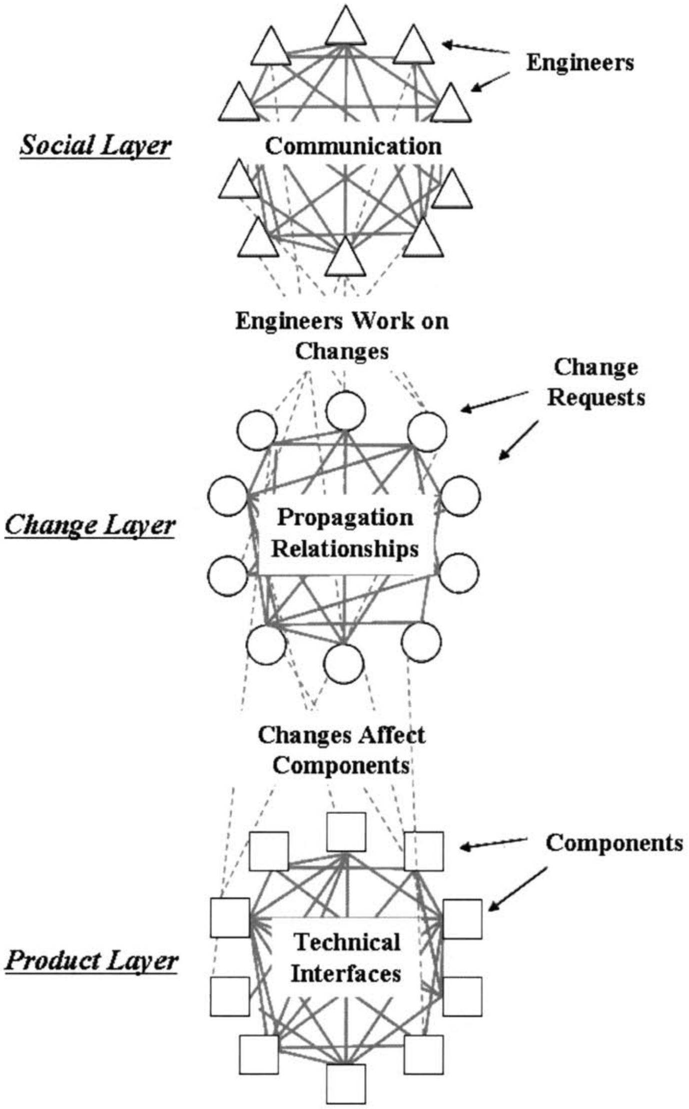


Figure 1. Multilayer Network Model of Change Propagation

1.1. Research Framework

The research framework for this thesis was constructed according to the guidelines outlined by de Neufville [9] and Kirchain [20]. In all, this thesis develops a multilayer network model of change propagation and investigates its practical utility through a case study of a real-world engineering project. Although the research framework is presented quite neatly here, the actual research undertaking was both iterative and recursive en route to a feasible research question, hypothesis, method, and logic. Indeed, this thesis is the result of repeated literature review, model development, data collection, and trial and error analysis.

1.1.1. Research Question

The topic of change propagation presents a range of worthwhile research questions to investigate. Among these are questions regarding the characterization, prediction, prevention, and control of change propagation. For instance, are there fundamental characteristics of change propagation than an organization can exploit? How can an organization predict change propagation in various types of products and systems? Moreover, how can an organization prevent or control change propagation through modular or flexible designs? These types of questions have been addressed in previous research, which has produced promising, but limited, answers. This thesis does not attempt to provide complete answers to these questions either; the development of robust methods for dealing with change propagation will require repeated experimentation in real-world situations. Instead, this thesis takes a step back to consider a fresh perspective with which to view the research field: namely, a multilayer network model of change propagation. In kind, this thesis investigates the following primary research question:

- *What insights can be gained from a multilayer network model of change propagation?*

Two secondary questions follow (or propagate) from this primary one:

- *What are potential tools and metrics for analyzing a multilayer network model of change propagation?*
- *How can these tools and metrics inform future engineering change management policy in terms of design strategy, change request evaluation, and human resource management?*

Overall, the answers to these questions will help determine the value of multilayer network modeling for understanding change propagation and informing engineering change management.

1.1.2. Hypothesis

The overarching hypothesis of this thesis is that a multilayer network model facilitates extensive quantitative analysis of change propagation using a repository of tools and metrics. These tools and metrics can be single-layer, double-layer, or triple-layer in origin. Furthermore, the model unites previous research on change propagation in a comprehensive paradigm. Finally, the model uniquely allows quantitative analysis of the previously overlooked, but significant, social layer of change propagation. Overall, a multilayer network model provides holistic framework with significant policy implications for engineering change management's handling of change propagation.

1.1.3. Research Method

Similar to other literature on change propagation [6, 13, 32], this thesis employs a case study to answer the research questions and test the corresponding hypotheses. More specifically, the case study here serves as a demonstration [9] of the practical utility of multilayer network modeling.

The case under investigation is a technical program whose purpose was to develop a large-scale sensor system, ultimately composed of legacy hardware and newly written software subsystems. The author received a dataset extracted from the program's configuration management records that contained detailed information about more than 41,000 changes proposed over a period of eight years. Additionally, one of the program's lead systems engineers was interviewed to put the results, analysis, and interpretation in context. Giffin et al. investigated the same program in [16] and [17], and her results and conclusions are cited throughout the case study and thesis.

To demonstrate the multilayer network model's practical utility, the case study applies the model to investigate two topics of significant interest to academia and industry. The first and most pioneering topic revolves around the social layer's effects on change propagation. Some newly developed tools and metrics reveal critical aspects of an engineer's performance in the proposal and implementation of changes. The other major topic of investigation involves the general characterization of change propagation. Specifically, the issues of indirect propagation, propagation extent, and component centrality are scrutinized closely with an array of multilayer tools and metrics. The results of these quantitative analyses confirm and offer counterexamples to many qualitative conclusions about change propagation in previous literature.

1.1.4. Logic

The choice of a case study as a research method is justified given Yin's [35] well known analysis of case studies as a research strategy. Following Yin's dissection of the appropriate setting for case study research, this thesis largely performs an *exploratory* case study. The case study here is exploratory in that it primarily attempts to answer "what" questions: namely, "what insights can be gained from a multilayer network model?" The study seeks to explore the multilayer network model's utility by elucidating interesting issues concerning change propagation. Nevertheless, in demonstrating the utility of the model, the case study also touches on a few "why" questions regarding change propagation as a phenomenon. Consequently, the case study sometimes borders on being *explanatory* as well.

Furthermore, the case here is what Yin calls a *revelatory case*, because the type of data acquired for this case study is rarely available to general researchers in the field. As explained later, research on change propagation has traditionally suffered from a lack of substantive data to quantitatively analyze. By contrast, the dataset here is unique in its richness. For example, the dataset indicates which engineer proposed, evaluated, or implemented each change request, which, for the first time, allows the construction of a social layer and subsequent quantitative analysis.

Because the case study here is of a single case, the generalization of results might be considered tenuous. However, Yin [35] explains that case studies rely on analytical generalization, rather than statistical generalization. As such, a single case can be used to advance a theory, similarly to how scientific experiments are touted to do the same. The "theory" asserted by this thesis is that a multilayer network model provides valuable insights into change propagation through the use of various tools and metrics. The case study here substantiates that theory. Moreover, the findings are compared to previous research for additional corroboration. Of course, more case studies in the future could strengthen the arguments here even further. In kind, one thrust of this thesis is to assess the data collection and data mining necessary for future researchers and industry to construct and utilize the multilayer network model.

1.2. Document Overview

The remaining chapters of this document are structured as follows:

- *Chapter 2 (Background and Motivation)* gives essential background information on several relevant topics, including engineering change management, change propagation, and network analysis. The chapter is basically a motivational literature review.

- *Chapter 3 (A Multilayer Network Model of Change Propagation)* develops a novel multilayer network model of change propagation.

- *Chapter 4 (Tool and Metric Development)* presents a baseline repository of tools and metrics for use with the multilayer network model. The repository consists of single-layer, double-layer, and triple-layer tools and metrics, including a few newly developed methods for analyzing the social layer. A hypothetical application called Project X is carried throughout the discussion to illustrate the repository's theoretical utility.

- *Chapter 5 (Case Study)* demonstrates the practical utility of the multilayer network model through a case study of a real-world engineering project. Specifically, the model is applied to investigate general characteristics of change propagation, as well as the social layer's effects on propagation phenomena.

- *Chapter 6 (Management Policy Implications)* discusses the policy implications of the multilayer network model and general change propagation research for engineering change management.

- *Chapter 7 (Conclusion)* closes the thesis with a summary of the research findings and recommendations for future work.

2. Background and Motivation

This chapter presents background material on the subject matter at the heart of this thesis, and provides motivation for the ensuing model development and case study. The two overlapping topics discussed here are engineering change management and network analysis. Engineering change management is shown to be a challenging element of product development and project management, particularly because of the phenomenon of change propagation. Change propagation contributes significantly to the time, money, and resources required to make changes to a complex design. Research efforts to quantitatively investigate change propagation have chiefly employed network analysis. However, no study of change propagation has deliberately explored the utility of a *multilayer network model* to capture the dependencies within and among multiple layers (or domains) of an engineer project. The use of multilayer networks models in similar contexts suggests that such an approach is viable for the study of change propagation.

This thesis builds upon past research by developing a novel multilayer network model that integrates the product, change, and social layers of change propagation. In addition to providing a comprehensive paradigm for investigating change propagation and unifying past research, this new model uniquely illuminates the impact of the previously overlooked social layer of change propagation. As such, this thesis fills a significant gap in the published literature.

2.1. Engineering Change Management

Engineering change management (ECM) has been defined as the process by which an organization proposes, evaluates, implements, and audits changes to the design of a product (or technical system) [18]. In other words, ECM deals with the evolution of a design, which is routinely changed to satisfy dynamic customer requirements, correct design problems, and converge on a good engineering solution. ECM is a critical component of the more cited discipline of configuration management, which more generally includes the monitoring, verification, and accurate documentation of the design process (NASA/SP-2007-6105 R1) [23]. However, this thesis will concentrate on ECM because of its specific emphasis on the change process. It's also important to distinguish ECM from business change management, by which a company changes the structure and operation of its teams and employees. Indeed, ECM is more concerned with technical changes, as opposed to organizational ones (though certainly some underlying principles likely apply to both).

Engineering change management is undoubtedly a critical element of product development and project management. Any organization involved in the design or manufacture of complex products must perform ECM. Moreover, the literature on ECM reports that the vast majority of engineering firms have a formal review and decision making process in place for handling potential changes to their designs [18]. For example, the National Aeronautics and Space Administration (NASA) performs ECM through a Configuration Control Board (CCB), composed of organization and team representatives who collectively make decisions regarding change proposals. Furthermore, as a testament to the importance of ECM, NASA considers ECM (within configuration management) to be one of the crosscutting technical management processes at the core of its systems engineering engine [23]. Overall, the aerospace, automobile, electronics, and software industries agree that ECM is necessary to deal with the inevitable desire and need for design changes [6, 18, 32, 34].

Changes are basically inevitable during product development [25]. Giffin et al. [17] and Suh and de Weck [31] emphasize that most products are adaptations of predecessor products, such that redesign efforts are inherent in the design process. Still, even de novo (or clean sheet) designs are bound to undergo changes. In either scenario, an organization might want or need to change some aspect of a design for a variety of reasons. Change requests may originate *externally* or *internally* (Eckert et al. [14] would call these initiating and emergent changes, respectively). External changes requests arise from outside the design process, as from users, customers, contractors, suppliers, and other stakeholders. The classic external change is a change in product requirements often levied by the customer or derived from business and marketing decisions. Meanwhile, internal change requests arise from within the design process, as from the designers themselves. Internal changes include rework and fixes that are expected as part of any iterative and recursive design process [12, 23].

Furthermore, change requests might arise at any stage in the engineering project lifecycle. During the design stage, many changes naturally occur as a design matures from conceptual to detailed Later during manufacturing, testing, and integration, more changes might become necessary if problems are discovered with a baseline design. Likewise, during operation of the product by the customer, additional changes may prove desirable as users develop new needs and explore unanticipated operational environments [16]. Moreover, throughout the project lifecycle, customer requirements and expectations are bound to evolve and continually press the developing organization to change the purpose and performance of the product. As a result, the baseline design of a product may persistently and progressively change in order to satisfy stakeholders and successfully complete product development.

Change is both a blessing and a curse in the engineering industry. This dichotomy makes ECM an especially challenging task. On the one hand, change drives innovation and allows an organization to improve its products and please its customers. Without change, an organization would not be able to stay competitive in its market [18, 34]. On the other hand, changes are not free. In fact, they require considerable time, money, and resources. Terwiesch and Loch [32] report that changes can absorb nearly half of a firm's engineering capacity and lead to similar proportions of manufacturing costs. In aggregate, the ECM process can consume weeks, months, and perhaps a year within the product development timeframe. In the same vein, Clarkson et al. [6] found that in the case of helicopter design, a single change can cost up to US$80,000. Consequently, organizations often view changes as evil and annoying, often resulting in the pessimistic attitude that changes should always be avoided where possible. Nevertheless, changes are expected, and an organization must balance the benefits, costs, and risks of making changes through effective ECM.

Researchers have investigated why changes can be so abundant, costly, and challenging, and hence why effective ECM is so necessary. Terwiesch and Loch [32] summarize some of the core reasons identified in the literature. Some researchers blame administrative inefficiencies in the processing of change requests. Others conclude that many changes are actually unnecessary if only organizations would realize a better initial design. Unfortunately, many engineers make little effort up front in their design decisions because they know their first designs are likely not permanent. In effect, the inevitability of change can become a self-fulfilling prophesy. These problems are exacerbated and nuanced by the workload capacity and varying skill and experience of individual engineers. Almost all researchers and interviewees from industry emphasize that changes made late in the development cycle are especially costly because of the increased amount of rework, documentation, retooling (for manufacturing), and renegotiating (with customers) that ensues. Finally, a significant and pervasive reason for the high cost of change is the occurrence of change propagation [6, 19, 32], which is the topic of the following section and the focus of this thesis.

2.2. Change Propagation

Change propagation is a phenomenon that contributes significantly to the negative impact of change during the design process. Borrowing language from Giffin et al. [17], change propagation is the "process by which a change to one part or element of an existing system [or product] configuration or design results in one or more additional changes to the system, when those changes would not have otherwise been required." In other words, change propagation occurs when making a single change ultimately requires the implementation of multiple changes, in order to achieve the objective of the overall redesign. When a change to one component (or subsystem) triggers another change in another component, that change is said to have propagated from one component to the other.

Change propagation can turn a single change into multiple changes through repeated and recursive acts of *parent-child propagation*. Parent-child propagation refers to the act of one change (the parent) yielding an immediate descendant change (the child). Figure 2 illustrates an initiating change that led to four generations of descendants through recursive parent-child propagation. Propagation over this many generations has been reported in the literature [6] and will be demonstrated again in Chapter 5's case study. In Figure 2, a parent-child relationship is represented by an arrow going from the parent change to the child change. The bidirectional arrows indicate sibling relationships between child changes of the same parent change. Later in this thesis, these parent-child and sibling-sibling arrows will become the intra-layer edges in the change layer of a multilayer network model.

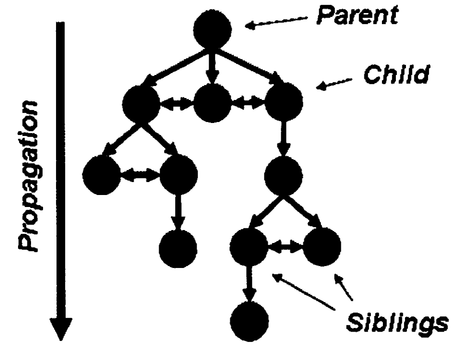


Figure 2. Illustration of Change Propagation

The snowball effect [32] of change can cause a seemingly simple change to cost much more than originally expected or, more drastically, to avalanche beyond control or feasibility. Consequently, over the last decade, much research has focused on characterizing, predicting, and controlling change propagation as a central ingredient to effective ECM.

The consensus among researchers is that change propagation occurs primarily because of the complexity of modern products and systems [6, 11, 13, 31, 32]. Complex systems, such as automobiles, airplanes, computers, and electronic devices, consist of many (sometime tens of thousands) interconnected and interdependent electro-mechanical, software, and human components and subsystems. One can easily imagine how the alteration of one component could propagate changes to others. After all, a system is a collection of interacting parts whose integrated value is greater than the sum of the values of the individual parts [23]. As such, the parts of a system are highly interdependent and the potential for change propagation is inherent. For example, Clarkson et al. [6] explains that a change made to the rotor of a helicopter would necessitate significant redesign of the entire aircraft, because of the functional dependence of the rest of the aircraft on the rotor. Because of the threat of change propagation like this, an organization's ECM process must account for possible propagation effects when evaluating and implementing a change to a single part of a product.

Eckert et al. [14] explained that different parts of a design exhibit different propagation behavior at different times. That is, components can act as absorbers, carriers, multipliers, or constants. *Absorbers* tend to internalize changes without causing many changes to other components, while *multipliers* give rise to more changes than they absorb. Meanwhile, *carriers* absorb and cause a roughly equal number of changes. Finally, *constants* do not contribute to any propagation effects; that is, they are only affected by isolated changes. According to Eckert et al., the propagation behavior of a component depends on the status of its tolerance margin. An initial design usually gives each component a certain margin for change so that it may absorb changes in the future. However, as theses margin are consumed by future changes, components transform from absorbers, into carriers, and eventually into multipliers. The conclusion is that ECM must be aware of these diminishing tolerance margins to predict the propagation behavior of components at any time in the design process.

Most of this discussion of change propagation has been qualitative. Indeed, research on change propagation has suffered from a lack of available data to corroborate hypotheses more quantitatively. Giffin [16] explains the reasons for this misfortune. One is the failure of organizations to resurrect ECM records once a project is complete, both by habit and for fear of embarrassment. Furthermore, ECM records, especially for large or long engineering projects, may be too cumbersome to efficiently extract the data needed for analysis. As a substitute for full data records, several researchers [6, 12, 13, 18, 32] have conducted interviews to acquire details about an organization's experiences with change propagation. Limited progress has been achieved using the interview approach combined with some data mining. By contrast, the quantitative thrust of this thesis enjoyed a uniquely rich dataset to be explored in Chapter 5's case study.

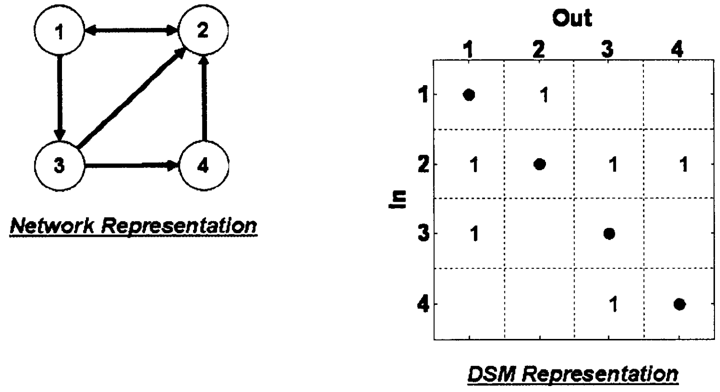
Despite having limited data, the literature still provides some notable quantitative progress in the field. As discussed in the next section, efforts to quantitatively characterize, predict, and control change propagation have mostly drawn on network analysis. This thesis builds on previous network approaches in a novel way through a multilayer network model of change propagation.

2.3. Network Analysis of Change Propagation

A network, or graph, is a set of nodes connected by edges. Literature on product development and project management has revolved around three broad categories of networks: products, processes, and social networks (similar to the network categorizations in [4, 24]). These categories are obvious candidates for network modeling and analysis. After all, a product is a network of hardware and software components that interact to perform a higher level function. Meanwhile process networks are composed of individual tasks and activities that depend on each other in some way. Finally, social networks consist of people who are connected via their relationships and communication with one another as leaders, subordinates, teammates, and friends. As reviewed by Newman [24], the study of networks has been prolific in a variety of research settings, including physics, engineering, biology, and the social sciences. Work in these areas has produced a tremendous bank of tools and metrics for understanding network behavior. Many of the tools and metrics discussed in this section will be further illustrated in Chapter 4's development of a tool and metric repository for the multilayer network model.

One of the primary network analysis tools created for product development and project management is the *Design Structure Matrix* (DSM) [48]. A DSM is essentially a matrix representation of a directed network, sometimes referred to as an *adjacency matrix* in graph theory [5], or an *N-squared* (N^2) *diagram* in systems engineering [23]. A DSM is a square matrix with a row (*m*) and column (*n*) for each node of the network. Element (*m*, *n*) indicates the number of a directed interfaces going from component *n* to component *m*. In graph theory, the number of interfaces incident upon (i.e., touching) a particular node is called the node's *degree* [5]. Since the DSM represents a directed network, one can further specify a node's *in-degree* and *out-degree*, according to how many interfaces are directed in and out of the node, respectively. For undirected networks (i.e., all interfaces go in both directions), the DSM would be symmetric, and each node's in-degree and out-degree would be equal.

Figure 3 displays a hypothetical network composed of four nodes, along with its DSM representation. Black dots appear along the diagonal of the DSM because each node is not considered to be connected to itself (though self-loops can have meaning in broader graph theory). The off-diagonal elements of the DSM count the directed edges starting in the "out" nodes and ending in the "in" nodes. In this example, node #2 has an in-degree of three, with interfaces coming from nodes #1, #3, and #4, and an out-degree of one, with an interface going to node #1. The reader should note that the convention for which matrix axes are "in" and "out" has alternated in the literature; Figure 3 and the rest of this thesis uses the rows for "in" nodes and the columns for "out" nodes.



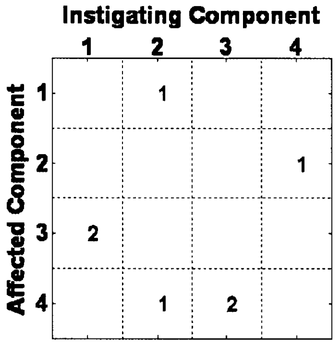
In \ Out	1	2	3	4
1	●	1		
2	1	●	1	1
3	1		●	
4			1	●

Figure 3. Example DSM

Larger and more complex networks (compared to Figure 3) are naturally more challenging to develop and understand. Fortunately, the DSM provides engineers and project managers a succinct way to represent and analyze the structure of otherwise overwhelmingly complex networks. As outlined by Browning [4], the DSM has been used to investigate both static networks (e.g., products and organizations) and time-based networks (e.g., processes). For static networks, clustering algorithms can order the rows and columns of the DSM to identify clusters of tightly coupled nodes, otherwise known as modules or subsystems in product design. Meanwhile, for time-based networks, sequencing algorithms can order the nodes (e.g., tasks) to minimize feedback and find opportunities for parallel processing. Overall, the DSM, together with its adaptations [4, 10], has become a cornerstone of product development and project management in both industry and academia.

Research on change propagation has benefited from network analysis. The DSM of a product design has obvious implications for the prediction of change propagation. After all, the interfaces captured by the DSM indicate how individual components depend on each other, and consequently how a change to one component might affect other components. Keller, et al. [21] and Clarkson et al. [6] developed the *Change Prediction Method* (CPM), which uses the DSM as a basis for predicting the occurrence of change propagation. The crux of CPM is that changes are assumed to propagate from one component to another along the dependencies shown in the DSM. By tracing all possible propagation paths, CPM creates a matrix showing the likelihood and impact of propagation between all components. Clarkson et al. achieved notable success with CPM in predicting change propagation in a few real-world scenarios at Westland Helicopters (a UK company). Chapter 6 will discuss the ability to predict propagation as an essential element of ECM policy for handling change propagation.

A related network analysis tool, called the *Propagation DSM*, has been developed to further characterize change propagation. Giffin et al. [17] actually names this tool the "Change DSM," but this thesis substitutes the word "propagation" for "change" to help distinguish it from the DSM of a change network. The Propagation DSM is a matrix that tallies instances of change propagation from one component to another over some time period in the design process. The matrix is square with a row (*m*) and column (*n*) for each product component. Element (*m*, *n*) of the Propagation DSM counts the number of times a parent change in the *instigating* component *n* spawned a child change in the *affected* component *m*. Figure 4 shows a hypothetical example of Propagation DSM. The matrix indicates, for example, that one change propagated from component #4 to component #2 and two changes propagated from component #3 to component #4 (but not necessarily from the same parent change in node #3). Overall, the Propagation DSM provides a succinct way to represent and analyze the location of propagation within a product design.



Affected Component \ Instigating Component	1	2	3	4
1		1		
2				1
3	2			
4		1	2	

Figure 4. Example Propagation DSM

In fact, a significant metric from the literature, known as the *Change Propagation Index* (CPI), is readily calculated from the Propagation DSM. As developed by Suh and de Weck [31] and refined by Giffin et al. [17], the CPI describes a single component's propagation behavior by comparing the numbers of changes that propagated in and out of that component. Most significantly, the normalized CPI spectrum (-1 to +1), quantifies the propagation behavior of a component as a multiplier (CPI > 0), carrier (CPI = 0), or absorber (CPI < 0). The exact calculation and interpretation of the CPI will be presented later in this thesis. Moreover, the CPI concept will be extended to assess the performance of engineers who implement changes to a product design.

One final example of network analysis in the literature is the change motif analysis conducted by Giffin et al. [17]. This study investigated the occurrence of certain motifs, or building blocks, within large change networks from a real-world engineering project (actually, the same program from Chapter 5's case study). The analysis suggested that the distribution of motifs could help explain propagation patterns in a complex system design.

Despite the promising findings about change propagation through network analysis to date, other research suggests that something is lacking in the usual approach toward change propagation. That is, the literature on product development and project management has stressed the need to consider multiple network *layers*, or domains, of an engineering project, including a product, process, and social layer (identical to the network categories described earlier). To date, change propagation research has not, at least explicitly, taken a multilayer network approach. To be fair, one should acknowledge that tools and metrics like the Propagation DSM and CPI are essentially double-layer approaches, since they do integrate both the product layer and change (i.e., process) layer. Still, other contributions like Clarkson's CPM and Giffin's change motif analysis only stem from single-layer models of the design and change layers, respectively. Furthermore, a glaring weakness in the change propagation literature is the failure to substantially consider the social layer in any quantitative way. This shortcoming is surprising since many researches have emphasized the importance of teamwork, individual skills, and system awareness in the ECM process [6, 13, 18].

This thesis proposes a novel multilayer network model of change propagation, which not only explores the social layer for the first time, but also unites past single-, double-, and triple-layer analyses in a new and comprehensive paradigm. But before developing the model in Chapter 3, it's insightful to take a cue from other multilayer network approaches in the related literature.

2.4. Multilayer Network Approaches

The concept that an engineering project involves multiple layers, or domains, has been proposed and utilized in the literature on product development and project management. This thesis prefers to use the term "layer" instead of "domain" because it is convenient to draw a multi-domain network with the domains in a vertically layered fashion. Chapter 3 will further discuss the visualization of the multilayer network model in this regard. Nevertheless, the premise of a multilayer approach is that a project's performance depends significantly on the interactions within and between layers. The literature review below will temporarily use the term "domain" to be consistent with the original authors.

Bartolomei [1] offers one of the clearest justifications for a multi-domain approach in his development of the Engineering Systems Matrix (ESM). The ESM augments the single-domain DSM to incorporate nodes from multiple domains. The ESM's domains include technical, functional, process, social, and environmental domains. The result is an adjacency matrix containing edges between nodes of the same domain, as well as edges connecting nodes of different domains. Figure 5 shows the generic layout of an ESM. Intra-domain edges would appear in the black boxes and inter-domain edges in the off-diagonal white boxes.

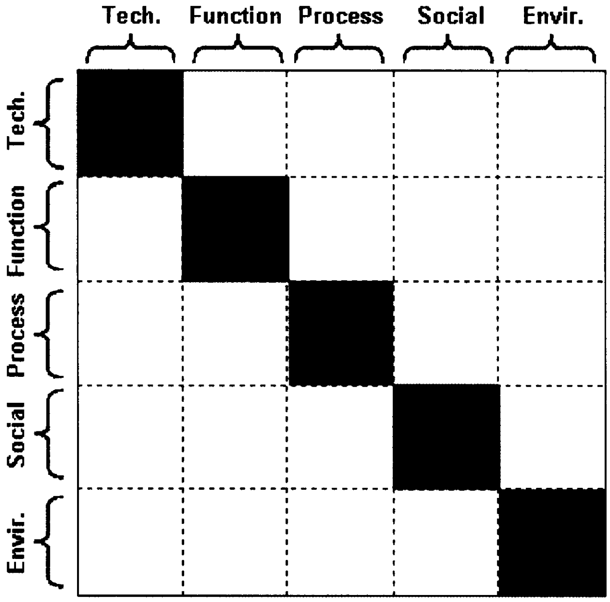
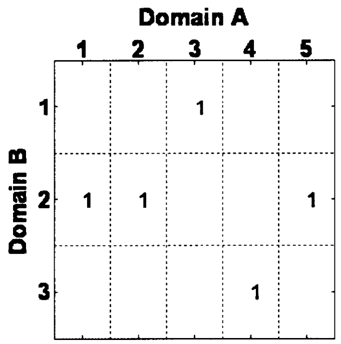


Figure 5. Generic ESM Layout

Bartolomei argues that the ESM is the only framework in the literature that facilitates quantitative analysis of multiple domains and their interactions (including variations over time). Furthermore, he suggests that a combination of single- and multi-domain analysis can provide considerable insight into the management and performance of engineering projects. Because his framework is relatively new, Bartolomei urges the research community to develop appropriate tools and metrics, especially inter-domain ones, to fully reap the benefits of multi-domain network models. This thesis takes on Bartolomei's challenge in the context of change propagation, a research subject which has lacked such a holistic framework.

While perhaps not achieving the comprehensiveness promoted by Bartolomei, others have conducted multilayer network analysis in notable ways. For example, Danilovic and Browning [8] develop yet another variation of the DSM known as a *Domain Mapping Matrix* (DMM). A DMM, as the name implies, maps different domains onto one another with respect to inter-domain dependencies. A DMM is generally rectangular with *M* rows for one domain and *N* columns for the other. Non-zero elements in the DMM count the dependencies between the nodes of the two domains. Figure 6 shows a hypothetical example of a DMM between Domains A and B, which have five and three nodes, respectively. The DMM indicates that node #2 in Domain B depends on nodes #1, #2, and #5 in Domain A. Danilovic and Browning argue that organizations can use the DMM to capture, analyze, and act on the relationships and constraints among the different domains of product development.



Domain B \ Domain A	1	2	3	4	5
1			1		
2	1	1			1
3				1	

Figure 6. Example DMM

Finally, several studies by Eppinger, Sosa, and Morelli have championed a multi-domain model most analogous to the one proposed in this thesis. Eppinger [12] applies a multi-domain model, including product, process, and organization domains, to the general challenge of product development. Eppinger primarily investigated whether interactions within each domain tend to follow a common, predictable pattern. For example, he and Sosa et al. [27] focused on potential patterns between the product domain and the organization domain. In an industry case study, they found that interfaces between product components were likely to correspond with communication between the teams designing those connected components. In other words, product interfaces were usually matched by communication interfaces. Likewise, in a case study with Morelli et al. [22], Eppinger found similar patterns between the process domain and the organization domain. Namely, dependencies between tasks tended to predict communication between the individuals performing those interdependent tasks. Eppinger et al.'s work suggests that project managers can improve performance by addressing the mismatches among the patterns of interactions within each domain.

Thus, the stage is set for further analysis of engineering endeavors using a multi-domain, or multilayer, network approach. This thesis continues this effort in the context of change propagation.

2.5. Filling the Gap

To summarize this chapter, the Venn diagram in Figure 7 shows where most of the authors cited above might be placed in the relevant research landscape. The diagram's two overlapping circles contain research on engineering change management (on the left) and network analysis (on the right). The ECM circle is further decomposed (by the gray dotted line) into general research and change propagation research. Meanwhile, the network analysis circle is divided into single-layer models (e.g., the DSM) and multilayer models. This thesis, labeled "Pasqual (2010)," sits at the intersection of research on change propagation and multilayer network analysis. As such, a gap in the literature is addressed by this research.

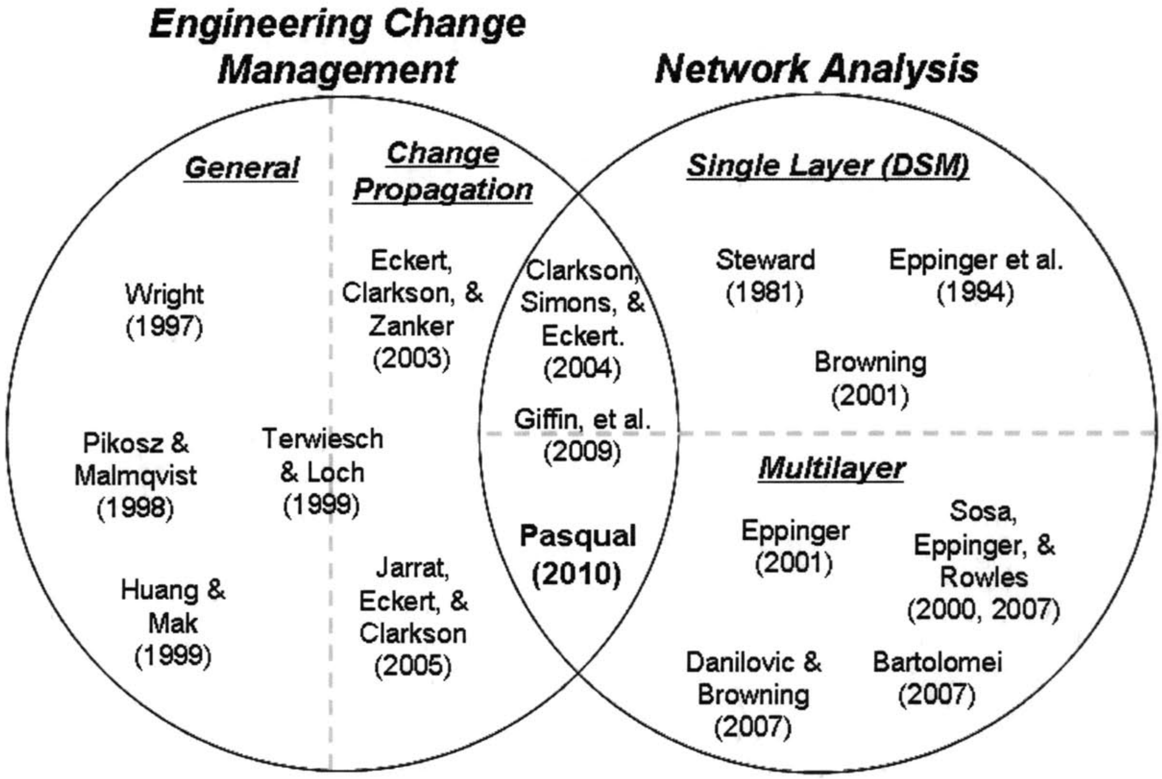


Figure 7. Research Venn Diagram

Here's what the research landscape looks like in Figure 7. During the 1990s, general research on engineering change management, such as by Wright [34], Pikosz and Malmqvist [25], and Huang and Mak [18], focused on how ECM was performed and why it was so challenging. Eventually, the phenomenon of change propagation was pinpointed, most notably by Terwiesch and Loch [32] in 1999, as a major reason for the pervasiveness and high cost of design changes. From there, qualitative studies via industry interviews (e.g., Eckert et al. [14] and Jarratt et al. [19]) explored real-world organizations' experiences with propagation effects. Clarkson et al. [6] and Giffin [16] conducted more quantitative analysis to predict and characterize change propagation through the use of network analysis throughout the 2000s. These network analyses ware largely single-layer (and some double-layer) in origin, borrowing from DSM research by Steward [30], Eppinger et al. [11], and Browning [4]. Meanwhile, work by Sosa et al. [27, 28], Eppinger [12], Danilovic and Browning [8], and Bartolomei [1] recognized the need for a multilayer approach to product development, and achieved notable success. However, change propagation research has never shared in that progress.

There in lies a void in the literature. To the author's knowledge, researchers of change propagation have never, at least explicitly, taken a multilayer network approach. Thus, this thesis (Pasqual, 2010) fills a significant gap in the literature on change propagation and ECM. To this end, the next chapter develops a novel multilayer network model, followed by the development of a baseline repository of tools and metrics for subsequent analysis.

3. A Multilayer Network Model of Change Propagation

This chapter develops a novel multilayer network model of change propagation. The model captures the interactions within and between three proposed layers of change propagation, which include a product layer, change layer, and social layer. Intra-layer edges connect (a) components within the product layer, (b) change requests within the change layer, and (c) engineers within the social layer. Meanwhile, inter-layer edges connect nodes across layers to represent how (a) engineers work on changes, (b) changes affect components, and (c) engineers are in charge of designing components. The multilayer network model lends itself to extensive analysis using an array of tools and metrics.

3.1. Model Setup

In general, a *multilayer network model* of change propagation acknowledges the existence of multiple layers, or domains, contributing to propagation phenomena. The specific model proposed here is composed of *three* primary layers: namely the *product layer*, *change layer*, and *social layer*. If desirable, additional layers, such as the environmental and functional domains suggested by Bartolomei [1], could also be incorporated into this model in the future. Nevertheless, the product, change, and social layers introduced here are analogous to previous multilayer (or multi-domain) approaches [12], as outlined in Table 1. These three layers should provide a good stepping stone for further analysis.

Table 1. Layer Nomenclature

Name of Layer Used in this Thesis	Analogous Names Used in the Literature
Product	Product [8, 12]; technical [1]
Change	Change [17]; process [1, 8, 12]
Social	Social [1]; organization [1, 8, 12]

The multilayer network model captures the interactions within and across the three layers of change propagation. Each layer forms an individual network composed of nodes of the same type that are connected by *intra-layer edges*. Meanwhile, the layers themselves are linked by *inter-layer edges* that connect nodes from different layers. As such, the model bears some resemblance to the concept of a *multipartite* (or *r-partite*) network in graph theory [10]. Strictly speaking, a multipartite network is a network of multiple types of nodes with edges that only run between different node types. The multilayer network model here deviates from the multipartite definition by also including intra-layer edges (i.e., edges running between identical node types). Thus, the multilayer network model has multipartite roots, but does not ignore intra-layer dependencies.

Figure 8 illustrates the generic multilayer network model in a *triangle formation*. The triangle formation, similar to that in [12], displays all the intra-layer and inter-layer edges at once.

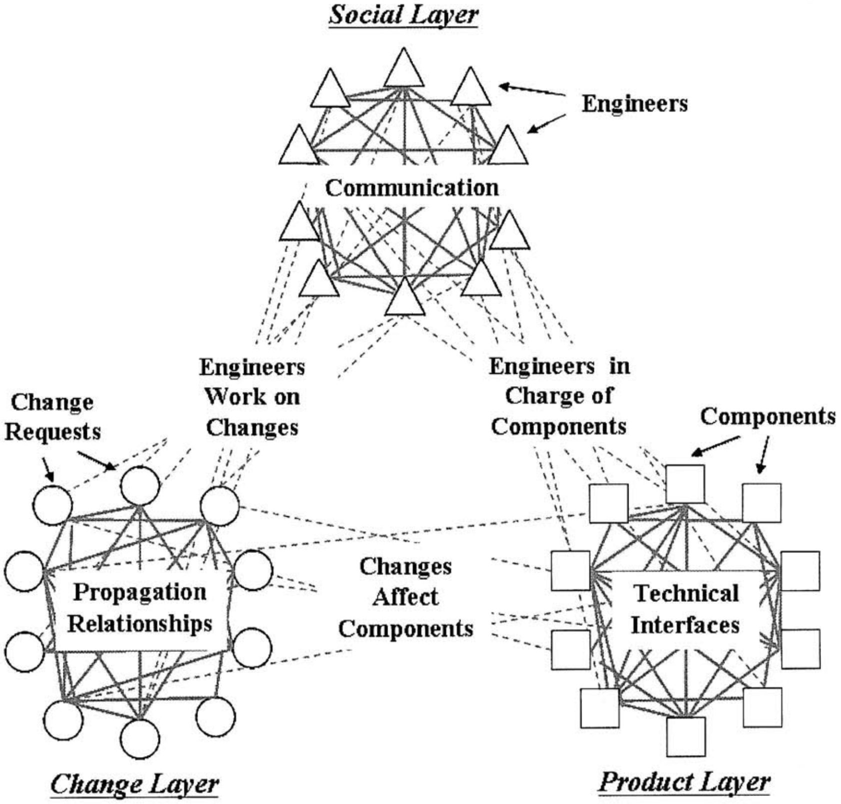

Social Layer
Engineers
Communication
Engineers Work on Changes
Engineers in Charge of Components
Change Requests
Components
Changes Affect Components
Propagation Relationships
Technical Interfaces
Change Layer
Product Layer

Figure 8. Triangle Formation of the Multilayer Network Model

By contrast, an alternative *linear formation* of the multilayer network model is shown in Figure 9.

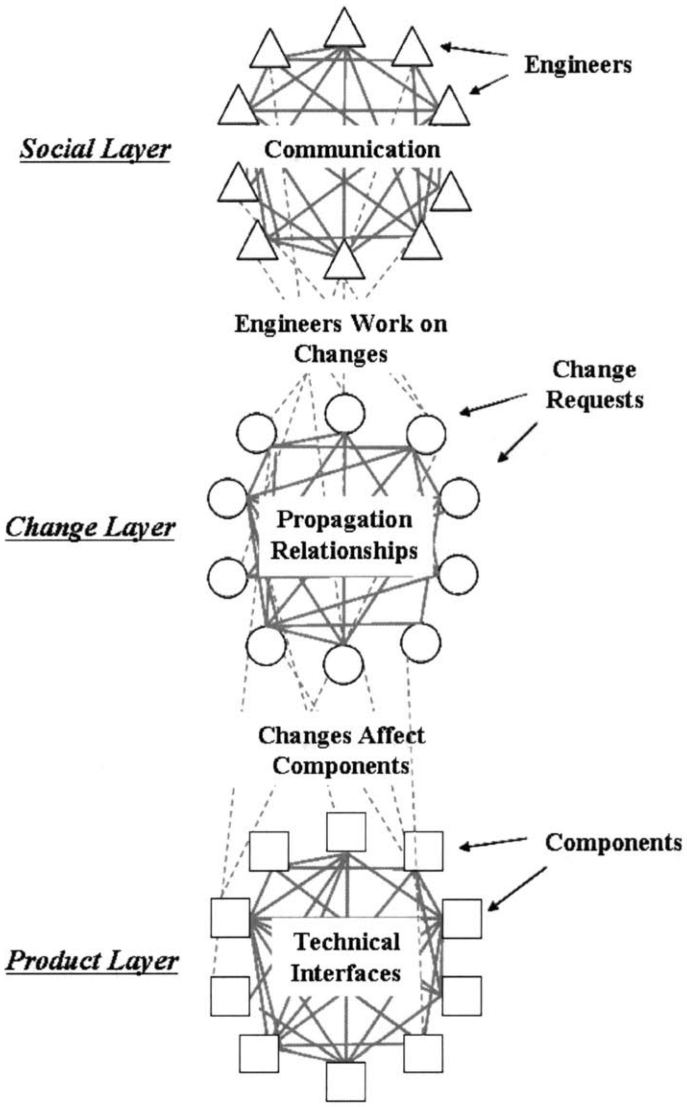


Figure 9. Linear Formation of the Multilayer Network Model

The linear formation purposely removes the product-to-social edges and aligns the layers vertically. While the linear formation is not a complete visual representation, it still has a few advantages. Firstly, it highlights the "layered" nature of the model, which is the original reason why the term "multilayer" was chosen over "multi-domain." Moreover, the linear formation gives a more intuitive impression of the engineering change management process. That is, *engineers* in the *social layer* act on *components* in the *product layer* via *changes* in the *change layer.* One last advantage of the linear formation is that it provides a less jumbled visualization for larger numbers of components, changes, and engineers. The linear formation will be used to visualize real-world data in Chapter 5's case study. Nevertheless, other formations and adaptations might prove useful for different purposes. The development of effective visualization techniques for the model is a great area for future work. The interpretation of all the nodes and edges in the multilayer network model, as labeled in Figure 8 and Figure 9, will now be described in detail.

3.2. Intra-layer Edges

Each layer of the multilayer network model forms a distinct, directed network composed of nodes connected by *intra-layer edges.*

- The product layer's network consists of nodes representing hardware components, software components, and possibly documentation too. At a higher level of abstraction, the nodes might also represent modules or subsystems. The intra-layer edges of the network represent various types of technical interfaces among the components and subsystems. Some interfaces are physical connections (e.g., bolted or welded together). Others might be channels for the flow of energy (e.g., electrical power and heat), mass (e.g., fuel and coolant), and information (e.g., software inputs and outputs) [31]. More generally, from a design perspective, an interface could also represent an important functional dependency or technical constraint, such as physical laws that relate design variables to a desired performance level (e.g., electrical current depends on resistance and voltage through Ohm's law, V = IR). In general, physical connections will correspond to bidirectional edges, whereas flow channels and technical constraints might be unidirectional. For example, the product layer for an automobile would contain nodes representing the fuel line and the fuel tank, among tens of thousands of other parts. These two nodes in particular would be connected by a bidirectional edge representing their physical connection and a unidirectional edge representing the flow of mass (i.e., fuel) out of the fuel tank and into the fuel line.

- The change layer's network is an example of a process network. Each node here represents a single change. More accurately, each node is a change request (CR), or a proposal for a single change, which emphasizes the reality that each "request" could be accepted or rejected. Still, for simplicity, the remainder of this document will use the term "change" to denote the nodes of the change layer.

  The intra-layer edges of the network represent propagation relationships between changes. A parent-child relationship appears as a unidirectional edge going from the parent change to the child change. By contrast, a sibling relationship looks like a bidirectional edge that connects children of the same parent change, or two changes related in another significant way. For example, Figure 10 shows a simple example of a change layer with one parent, two children, and their associated directed edges.

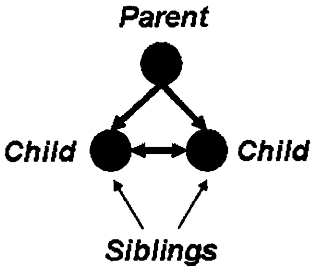


Figure 10. Example Change Layer

  It's important to note that the change layer is not time-based like most other process networks in which the nodes have sequential dependencies. However, an organization could schedule the implementation of change activities using a traditional process network, once sequential dependencies are determined.

- The social layer's network consists of nodes representing people. Each node could be an organizations, team, sub-team, or even more granularly, an individual engineer or employee. The intra-layer edges can represent various relationships between people, and can be unidirectional or bidirectional. For example, the edges might be based on an organization's hierarchical structure, such that they correspond to some chain of command. Another approach is to have the edges represent theoretical or actual communication between groups and individuals. Morelli et al. [22] describes three types of communication that can occur between people involved in product development: coordination-type, knowledge-type, and inspiration-type. Coordination-type communication describes information transfer performed by people to complete a task. Meanwhile, knowledge-type communication involves cooperative learning that does not necessarily relate to a specific task. Finally, inspiration-type communication encompasses motivational interactions, usually with a manager. For any type of edge in the social layer, the direction of communication might be significant for certain analyses. Indeed, a distinction between the provider and receiver of information could make a difference.

A possible augmentation to these intra-layer edge definitions is a measure of edge strength. After all, a strong connection between nodes might exhibit different behavior than a weaker connection. For example, Sosa et al. [27] uses a five-point scale to denote the criticality of interactions among product components, according to whether the interaction is required (+2), desired (+1), indifferent (0), undesired (-1), or detrimental (-2) to the functionality of the product. Their case study of the development of a commercial aircraft engine found that strong technical interfaces in the engine, compared to weaker ones, more often elicited communication in the corresponding organization domain. Consequently, edge strength could be a reasonable nuance to consider, if an objective and consistent quantification scheme is employed.

3.3. Inter-layer Edges

The other half of the multilayer network model consists of the *inter-layer edges* that essentially link the layers together. Unlike the intra-layer edges, the inter-layer edges are nominally undirected (or essentially, bidirectional). However, the concept of a directed inter-layer edge might have meaning in another context. Overall, inter-layer edges represent the critical dependencies between the layers of the model.

- A *product-to-change edge* connects a component (or subsystem) in the product layer with a change in the change layer. The edge identifies the component affected by that change. For example, if change #1 involves a redesign of component #6, then an edge would appear between node #1 in the change layer and node #6 in the product layer.

- A *product-to-social edge* connects a component (or subsystem) in the product layer with an engineer (or team) in the social layer. The edge identifies the engineer who is in charge of designing, redesigning, or sourcing that component. For example, if engineer #3 is assigned to component #2, then an edge would appear between node #3 in the social layer and node #2 in the product layer. In some scenarios, such as the cases studied by Eppinger [12], a one-to-one mapping exists between the product and social layers. In other words, engineer *n* focuses on component *n*. Such a mapping facilitates easier analysis and interpretation. However, other scenarios might have engineers focusing on various overlapping areas of the product at once.

- A *change-to-social edge* connects a change in the change layer with an engineer (or team) in the social layer. The edge identifies the engineer who worked on the change, by proposing, evaluating, or implementing the change request. For example, if engineer #6 implemented change #9, then an edge would appear between node #6 in the social layer and node #9 in the change layer. Again, sometimes there can be a one-to-one mapping between the changes (or tasks) and the engineers. That is, engineer *n* works on all changes that affect component *n*. However, the program in Chapter 5's case study notably had individual engineers working on multiple changes involving multiple areas of the system.

To summarize the layer definitions presented above, Table 2 shows the meanings of all the edges in the multilayer network model. Each box in Table 2 describes the edges occurring between the nodes of the layers in the corresponding row and column. The boxes along the diagonal hold the intra-layer edges, while the off-diagonal boxes hold the inter-layer edges. The language in the off-diagonal boxes is meant to reflect the symmetry (or undirected nature) of the inter-layer edges. By contrast, the intra-layer edges generally have an associated direction.

Table 2. Intra/Inter-edges of the Multilayer Network Model

	Product	**Change**	**Social**
Product	Technical interfaces (physical, energy, mass, information, constraints)	Changes affect components	Engineers assigned to components
Change	Components affected by changes	Propagation relationships (parents, children, and siblings)	Engineers work on changes
Social	Components assigned to engineers	Changes worked on by engineers	Communication or hierarchical structure

Most significantly, the multilayer network model provides a platform for extensive development of tools and metrics for analyzing change propagation. A baseline repository of tools and metrics is the subject of the next chapter.

4. Tool and Metric Development

This chapter presents a baseline repository of tools and metrics applicable to the multilayer network model of change propagation. Such tools and metrics can be neatly categorized as being single-layer, double-layer, or triple-layer in origin. Interestingly, as discussed in Chapter 2, a number of relevant tools and metrics have already been developed in the literature, although they were not always explicitly classified in a multilayer context. The multilayer network model unites these previous methods in a comprehensive paradigm for investigating change propagation. Still, further tool and metric development is needed. In fact, this chapter introduces several new tools and metrics, particularly for analyzing the model's social layer. To help illustrate the repository of tools and metrics presented here, a hypothetical application is carried throughout this discussion. The reader may occasionally wish to consult Table 19 (page 92) for a summary of the repository. Additionally, Table 20 (page 93) outlines the data required to exercise each tool and metric in practice.

4.1. A Comprehensive Paradigm

The multilayer network model creates a platform for an array of potential tools and metrics for investigating change propagation and engineering change management. *Tools* here refer to specific methods for analyzing or visualizing the nodes and edges of the model. Meanwhile, *metrics* are qualitative and quantitative measures for characterizing the nodes and edges, individually or together. Since any use of the multilayer network model will focus on one layer or multiple layers simultaneously, it's useful to categorize each tool or metric as being single-layer, double-layer, or triple-layer in origin.

Fortunately, the development of these tools and metrics has actually already begun in the literature. After all, although an explicit multilayer network model of change propagation is new, the network analyses from past research on change propagation (and beyond) are easily incorporated into the model. Indeed, it's possible to cast all the tools and metrics cited in Chapter 2 as being single-layer, double-layer, or triple-layer in origin. Some of these tools and metrics, such as Clarkson et al.'s CPM [6] and Suh and de Weck's CPI [31], were deliberately created for analyzing change propagation. Others are grounded in general product development and project management, where the overall research goal is to improve efficiency and product quality. Of course, engineering change management is a primary facet of efficiency and quality in industry. As such, generic product development tools, such as Danilovic and Browning's DMM, have important implications for engineering change management and the handling of change propagation. Consequently, the multilayer network model serves as a comprehensive paradigm that puts past research in a common context.

Still, there are notable gaps in the repository, particularly for quantitative analysis of the social layer. This thesis cultivates several new tools and metrics for this very purpose. Among these are the Engineer-CPI and Proposal Acceptance Rate, which quantify an engineer's performance as implementers and proposes of change, respectively.

The following sections develop and critically evaluate a baseline repository of tools and metrics, both old and new. The discussion is divided into three separate sections focusing on single-layer, double-layer, and triple-layer analyses. As previewed in Table 3, each type of analysis enjoys a useful set of tools and metrics. The tools and metrics that are marked with a "*" are being *proposed for the first time* by this thesis.

Each tool or metric will be assessed in light of its implications for change propagation and engineering change management. To help illustrate the theoretical value of the repository, a hypothetical application called Project X is introduced shortly and carried throughout the remainder of the discussion. Additionally, occasional examples from the literature and Chapter 5's case study are cited for practical corroboration.

Table 3. Preview of Repository of Multilayer Network Tools and Metrics

	Tools and Metrics	Section
Single-layer	**Tools**	
	• DSM [4, 11, 30]	4.3.1.1
	• CPM [6, 21]	4.3.1.2
	• Change Motifs [17]	4.3.1.3
	Metrics	
	• Graph properties [5, 10, 24]	4.3.2.1
	• Node attributes [17]	4.3.2.2
Double-layer	**Tools**	
	• DMM [8]	4.4.1.1
	• Propagation DSM [17]	4.4.1.2
	• CPFM [17]	4.4.1.3
	• Product/Propagation DSM Overlay [17]	4.4.1.4
	• Alignment Matrix [28]	4.4.1.5
	• Engineer Propagation DSM*	4.4.1.6
	Metrics	
	• Propagation Directness*	4.4.2.1
	• Component-CPI [17, 31]	4.4.2.2
	• Engineer-CPI*	4.4.2.3
	• CAI/CRI [17]	4.4.2.4
	• Proposal Acceptance Rate [16]	4.4.2.5
Triple-layer	**Tools**	
	• ESM [1]	4.5.1.1
	• Product/Propagation/Social DSM Overlay*	4.5.1.2
	Metrics	
	• Graph properties [1, 5, 10, 24]	4.5.2.1

* Proposed first by this thesis

4.2. Project X: A Hypothetical Application

To guide this discussion, a simple hypothetical application of the multilayer network model is introduced now. The imagined project, generically named Project X, involves six (6) engineers who worked on eight (8) changes to a product of six (6) components. For simplicity, a one-to-one mapping is assumed between the product layer and social layer, such that engineer *n* is in charge of component *n*, and works on all changes affecting component *n*. Figure 11 draws the corresponding multilayer network of Project

X in a linear formation. The linear formation unfortunately does not show the product-to-social edges, but the one-to-mapping between the product and social layers makes these missing edges easy to ascertain mentally, given the node labels.

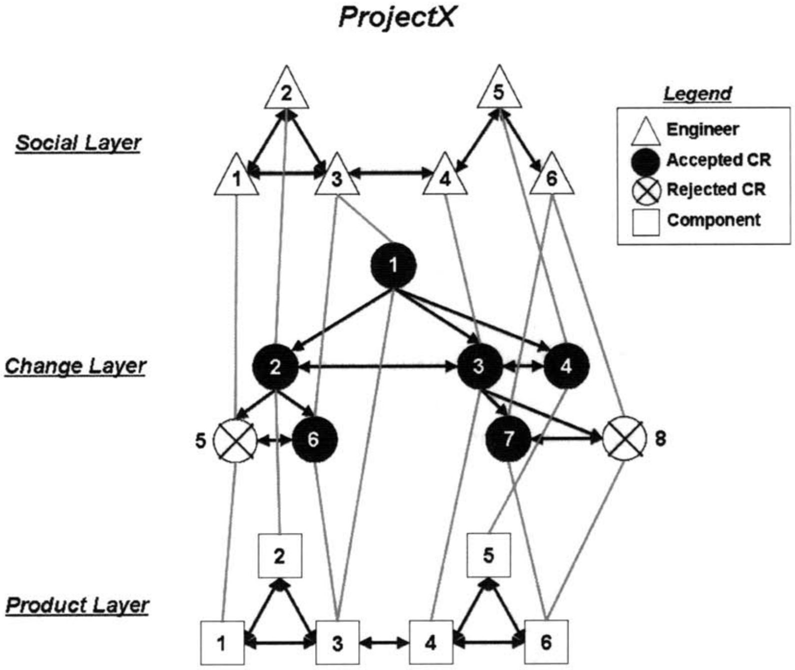


Figure 11. Multilayer Network Drawing of Project X

There are a few other features to note about Project X in Figure 11. The nodes of the product, change, and social layers represent components (squares), changes (circles), and engineers (triangles), respectively, and the intra-layer edges represent technical interfaces, propagation relationships, and communication, respectively. The inter-layer edges represent how changes affect components and engineers work on changes. A change node is colored black if the change was accepted, and has an "X" if it was rejected. For simplicity, the product layer and social layer have bidirectional intra-layer edges; that is, technical interfaces (between components) and communication (between engineers) are all two-way. The inter-layer edges are drawn without arrows because they are nominally undirected (i.e., bidirectional).

The Project X example was fabricated to exhibit an array of behaviors that can be analyzed using the repository presented next. In fact, all the tools and metrics presented here were developed by this thesis or in the literature to help investigate these exact behaviors as discovered in real-world case studies. Project X combines all these behaviors in a single hypothetical example. The example is simple enough that the reader should mentally be able to do all the calculations by referring back to Figure 11's drawing. Thus, the application is for theoretical demonstration only. The case study in Chapter 5 will demonstrate the practical utility of the multilayer network model in a real-world engineering project.

4.3. Single-layer Analysis

Single-layer analysis considers each layer of the multilayer network model by itself. Although the inter-layer dependencies are ignored, single-layer tools and metrics can highlight intra-layer characteristics of great significance for engineering change management.

4.3.1. Single Layer Tools

Single-layer tools will be discussed first. Table 4 lists some notable single-layer tools with their respective layers of focus and literature references.

Table 4. Single-layer Tools

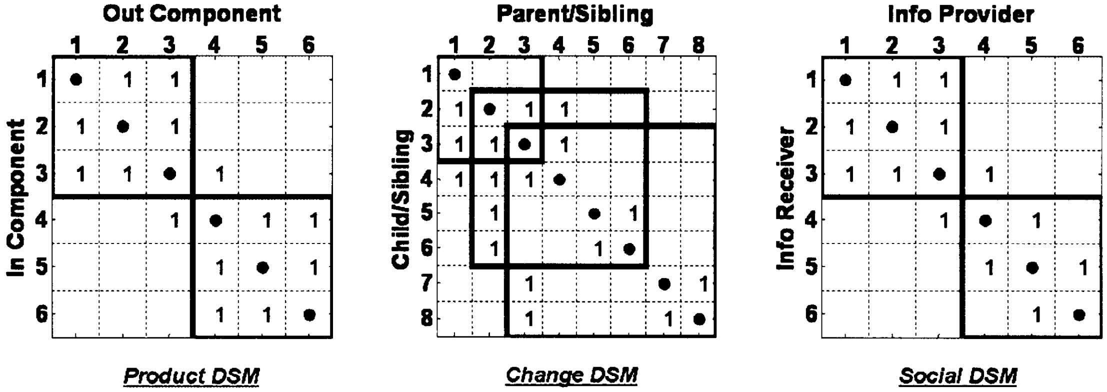
Tool	Layers	Reference
Design Structure Matrix (DSM)	Any	4, 11, 30
Change Prediction Model (CPM)	Product	6, 21
Change Motif Analysis	Change	17

All of these tools were introduced in Chapter 2. The DSM is broadly applicable to any layer of the multilayer network model, while CPM and change motif analysis are tailored to the product and change layers, respectively.

4.3.1.1. Design Structure Matrix

The most recognized single-layer tool is the *Design Structure Matrix* (DSM), a fundamental tool for product development and project management in both industry and academia [11, 12, 30]. As explained in Chapter 2, clustering and sequencing algorithms exist to manipulate the rows and columns of the DSM to inform better engineering and management decisions. These decisions can help minimize unnecessary future changes and stem change propagation.

To illustrate this point, Figure 12 displays the Product DSM, Change DSM, and Social DSM of Project X. The DSMs in Figure 12 immediately reveal the overall structure of each layer in terms of clusters, as outlined in the solid boxes.



Out Component / In Component

	1	2	3	4	5	6
1	●	1	1			
2	1	●	1			
3	1	1	●	1		
4			1	●	1	1
5				1	●	1
6				1	1	●

Product DSM

Parent/Sibling / Child/Sibling

	1	2	3	4	5	6	7	8
1	●							
2	1	●	1	1				
3	1	1	●	1				
4	1	1	1	●				
5		1			●	1		
6		1			1	●		
7			1				●	1
8			1				1	●

Change DSM

Info Provider / Info Receiver

	1	2	3	4	5	6
1	●	1	1			
2	1	●	1			
3	1	1	●	1		
4			1	●	1	
5				1	●	1
6					1	●

Social DSM

Figure 12. DSMs for Project X

For example, the product layer clearly has two subsystems, one composed of components #1, #2, and #3, and the other composed of components #4, #5, and #6. The subsystems are connected via an interface between components #3 and #4. An organization could use the product DSM to improve the product architecture in anticipation of the challenges of testing, building, integrating, and evolving a product. Forward-thinking in the design process can reduce the amount of redesign and retooling needed in the future. For example, an organization could create a platform for a line of products (e.g., automobiles). The platform would be impervious to change (i.e., a reflector), thus minimizing changes and change propagation during future product variations and customization [31].

The change layer contains three clusters that correspond to distinct families of changes. One family was parented by change #1, another by change #2, and the last by change #3. The clusters overlap because all three families are part of the same propagation chain, or extended family, initiated by change #1. It is also important to note that the change layer's DSM is asymmetric for parent-child relationships (e.g., changes #1 and #2) and symmetric for sibling relationships (e.g., changes #2 and #3).

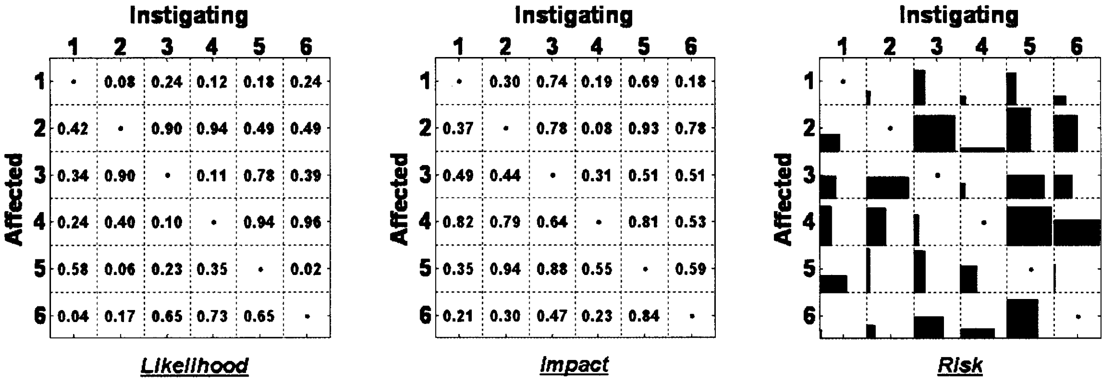
Finally, the social layer in Figure 12 has two distinct clusters of communication. A project manager might use the existence of communication clusters as a justification for reorganizing teams or co-locating engineers who must interact frequently. These strategies enable and improve communication that is vital to engineering change management. Eckert et al. [14] suggests that insufficient communication is a primary cause of redesigns throughout product development.

In all, the DSMs convey a lot of information in a relatively compact form. Project X was only a small example of the DSM's utility in the multilayer network model. The power of the DSM and clustering algorithms can be even more critical in larger projects involving hundreds or thousands of intricately connected components, changes, and engineers.

4.3.1.2. Change Prediction Model

Keller, et al. [21] and Clarkson et al. [6] developed another single-layer tool called the *Change Prediction Model* (CPM). This tool focuses specifically on the product layer alone, although the change layer obviously provides motivation. CPM uses the DSM to identify propagation paths between components, under the assumption that changes propagate along the technical interfaces of a product. The tool uses combinatorics to calculate the likelihood ($L_{b,a}$) that a change in component a will lead to a change in component b, derived from all possible propagation paths between a and b. CPM then calculates, in reverse order, the impact ($I_{a,b}$) and risk ($R_{a,b}$, the product of $L_{a,b}$ and $I_{a,b}$) of such an event occurring. The final product of the prediction tool is a risk matrix with rectangles of width $L_{a,b}$ and height $I_{a,b}$ in each cell of the DSM, representing the risk of propagation from one component to another (after all paths are considered). As such, tall and fat rectangles imply more risk than short and skinny rectangles.

Continuing with the Project X example, Figure 13 shows a fabricated set of likelihood, impact, and risk matrices. These matrices are analogous to the ones presented in [6]. The matrices illustrate that a change in one component could ultimately lead to a change in a nonadjacent component, because of progressive parent-child propagation along some path in the underlying DSM. This is clear from the fact that every off-diagonal cell of the risk matrix contains a black rectangle, even though interfaces do not exist between every pair of components. Figure 13 also highlights one shortcoming of the CPM tool, which is that self-propagation, or propagation from component *a* back to component *a* through some path, is not allowed; this explains why the diagonals are all empty. However, the tool could probably be augmented to account for this possibility.



Likelihood

Affected \ Instigating	1	2	3	4	5	6
1	•	0.08	0.24	0.12	0.18	0.24
2	0.42	•	0.90	0.94	0.49	0.49
3	0.34	0.90	•	0.11	0.78	0.39
4	0.24	0.40	0.10	•	0.94	0.96
5	0.58	0.06	0.23	0.35	•	0.02
6	0.04	0.17	0.65	0.73	0.65	•

Impact

Affected \ Instigating	1	2	3	4	5	6
1	•	0.30	0.74	0.19	0.69	0.18
2	0.37	•	0.78	0.08	0.93	0.78
3	0.49	0.44	•	0.31	0.51	0.51
4	0.82	0.79	0.64	•	0.81	0.53
5	0.35	0.94	0.88	0.55	•	0.59
6	0.21	0.30	0.47	0.23	0.84	•

Figure 13. Likelihood, Impact, and Risk Matrices for Project X

Another weakness, though subtle and imperceptible here, is that CPM appears to only allow for *direct propagation.* That is, CPM assumes that parent-child propagation can only occur between adjacent components. However, as the case study in Chapter 5 will demonstrate, changes *can* propagate to a child component with no direct interface with the parent component. This unexpected phenomenon might be called *indirect propagation.* The terminology here can be confusing, because CPM does explicitly account for what Clarkson et al. [6] calls "indirect dependencies," by which changes eventually affect non-adjacent components, but only through recursive parent-child propagation between adjacent components. Thus, Clarkson et al.'s indirect dependencies actually constitute grandparent-grandchild (great-grandparent-great-grandchild, etc.) relationships between non-adjacent components. This clearly differs from the new notion of indirect propagation suggested now by which non-adjacent components have parent-child relationships.

Beside risk matrices, another significant feature of the full CPM tool is a set of visualization techniques for viewing potential propagation paths [21]. Admittedly, the DSM and the risk matrices do not give a clear picture of the possible propagation paths between components in the product layer. CPM's visualization techniques overcome this weakness by drawing the product layer's network more explicitly. These techniques generally place a focal component at the center of a plot, and then position the other components at radial distances according to some network relationship with the focal component. Organizations can use CPM techniques, as complements to the traditional DSM approach, to better visualize how a change to one component might affect the rest of the product or system. One variation of the CPM technique is shown in Figure 14 for Project X.

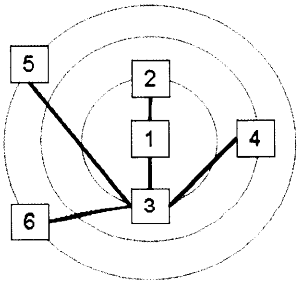


Figure 14. CPM Visualization for Project X

In Figure 14, the focal component is component #1. All other components are positioned on concentric circles corresponding to their minimum paths from component #1 in the product layer. For example, component #4 is on the second innermost circle because a minimum of two interfaces separate components #4 and #1 in the product. The black lines trace the actual paths taken from component #1 to all other components. The reader should refer to [21] for more sophisticated examples of CPM visualization techniques.

4.3.1.3. Change Motifs

While CPM focused on the product layer, Giffin et al. [17] performed *motif* analysis on the change layer. The premise here is that change networks can be decomposed into motifs, or distinct building blocks of nodes (changes) and edges (propagation relationships). Different classes of motifs involve different numbers of changes and different kinds of propagation relationships. One class of motifs, called a 1-motif, contains a single change that has an approval status of accepted, rejected, or pending; these three 1-motifs are designated the symbols *C11*, *C10*, or *C00*, respectively. Meanwhile, 2-motifs contain two changes that form a parent-child or sibling-sibling relationship, where each change still has an individual approval status. The nine possible parent-child 2-motifs are designated the symbol *PXY*, where *P* indicates that it is a parent-child relationship, *X* denotes the approval status of the parent change with 0 (pending), 1 (rejected), or 2 (approved), and *Y* denotes the approval status of the child change in the same way. Similarly, the six possible sibling-sibling 2-motifs are designated the symbol *SXY*, where *S* indicates that it is a sibling-sibling relationship, *X* denotes the approval status of one sibling, and *Y* denotes the approval status of the other. Only six sibling-sibling 2-motifs are recognized because of the symmetry of the relationship; that is, S10 and S01 are considered the same motif, and likewise for S02 and S20, and S12 and S21. In all, the change layer's network is essentially a collection of 1-motifs and 2-motifs; Giffin et al. also considers the presence of 3-motifs.

Applying this concept to Project X, Table 5 shows the frequency distribution of 1-motifs, and Table 6 shows the frequency distributions of parent-child 2-motifs and sibling-sibling 2-motifs.

Table 5. 1-Motif Distribution for Project X

1-Motif	Count
C00	0
C10	2 (25%)
C11	6 (75%)

Table 6. 2-Motif Distributions for Project X

Parent-Child 2	Count
P00	0
P01	0
P02	0
P10	0
P11	0
P12	0
P20	0
P21	2 (29%)
P22	5 (71%)

Sibling-Sibling 2	Count
S00	0
S01 (and S10)	0
S02 (and S20)	0
S11	0
S12 (and S21)	3 (60%)
S22	2 (40%)

Motif distributions reveal what types of propagation patterns are dominant in a project. Furthermore, the real-world explanation behind the patterns can teach an organization more about its engineering change management process. For example, Project X's most popular 1-motif was an accepted change (C11), meaning the organization accepted 75% of all change requests. The most prevalent parent-child 2-motif was the scenario where the parent and child changes were both accepted (P22). That is, when the organization identified the need for propagation, it was justified 71% of the time. The other 29% of child changes were ultimately rejected. Finally, the sibling-sibling 2-motifs shed light on the breadth of propagation; each motif represents an instance of one parent spawning two children. Giffin [16] supposes that the "one sibling accepted, one sibling rejected" motif (S12 or S21) might be a case of substitution. In other words, the organization rejects one sibling in favor of another, in order to achieve the overall objective of their mutual parent change. For example, Project X might have rejected change #8 in favor of its sibling, change #7. The exact explanation lies in the lower-level details of the change records. In all, a better understanding of propagation patterns might require broad motif analysis followed by more meticulous investigation of specific instances of propagation.

In summary, the set of single-layer tools just illustrated enables engineers and project managers to analyze the individual layers of the multilayer network model The DSM, in particular, is applicable to all layers, as clustering algorithms can identify subsystems in the product layer, propagation families (and larger chains) in the change layer, and groups of interactions in the social layer. Meanwhile, CPM and change motif analysis were tailored specifically to the product layer and change layer, respectively. Complementing these single-layer tools is an array of single-layer metrics described next.

4.3.2. Single-layer Metrics

Some potential single-layer metrics are listed in Table 7.

Table 7. Single-layer Metrics

Metric	Layers	Reference
Graph properties	Any	5, 10, 24
Node attributes	Any	New

These metrics have been employed in the literature on change propagation and engineering change management, but without any formal development. This discussion hopes to officially establish their utility for future research.

4.3.2.1. Graph Properties

Graph theory provides a number of properties generally applicable to any layer of the multilayer network model. Graph properties are graph invariant, such that their calculations depend only on the structure of the graph, or network, and not the specific drawing, representation, or context of that structure [10]. Nevertheless, the following graph properties can have interesting implications within each layer:

- A *path* between two nodes in a network is a sequence of nodes that can be traversed to get from one node to the other. The shortest path between two nodes is called a *geodesic path*. The length of the longest geodesic path between any two nodes in a network is called the network's *diameter* [10].

  The concept of paths has significant meaning for the different layers of the multilayer network model. As emphasized by CPM, paths in the product layer reveal how a change to one component might propagate to another component. In the social layer, paths relate to the degrees of freedom among engineers in their communication patterns or the chain of command. Finally, paths in the change layer reflect the amount of propagation that occurred. For instance, the geodesic path length between an initiating change and a descendent change equals the number of generations between them. The number of generations propagated by an initiating change can be referred to as *propagation extent.* The literature reports that propagation extent usually peaks at four generations; interestingly, the program in Chapter 5's case study experienced a few situations with five generations of extent.

- A network's *clustering coefficient* is the fraction of triples in the network that are also triangles (defined one way, [24]). A *triple* is a set of three nodes in which at least one node is connected to both other nodes. Meanwhile, a *triangle* is a triple in which all three nodes are connected to both other nodes. The clustering coefficient is a measure of how much a network's nodes tend to cluster together. In the product layer, the clustering coefficient roughly reflects a product's modularity. Integrative products, which have relatively high clustering coefficients, may be more susceptible to change propagation, since their components are more interdependent [31].

- *Centrality* is a gauge of the importance of a node in a network. One measure of a node's centrality is its degree, or the number of edges incident upon it. Another centrality measure is *betweenness*, which is the number of times a node appears in all the geodesic paths between all the other nodes [24]. Centrality has implications for the behavior characteristics of nodes in the social and product layers. For instance, in the social layer, centrality reflects the significance or role of an individual engineer. A manager, team lead, or liaison will likely have a higher centrality than a specialist. In the product layer, a component's centrality may reflect its potential propagation behavior. Namely, components with higher centrality might be more involved in change propagation, as parents or children. In fact, Chapter 5's case study demonstrates this correlation quantitatively.

A few of these graph properties are calculated for the product and social layers of Project X in Table 8. The centralities are the most interesting. For example, component #3 has the highest centrality in terms of degree (6) and betweenness (12). Not surprisingly, component #3 was also involved in the most propagation, as a parent three times and a child once. Similarly, engineer #3 has the highest centrality and was also involved in the most propagation as the implementer of a change spawning three child changes. Judging by their relative centralities, component #3 and engineer #3 sit at influential positions in their respective layers.

Table 8. Graph Properties for Project X

Layer	Diameter	Clustering Coeff.	Node# → Total Degree	Node# → Betweenness
Product	3	0.33	1 → 4 2 → 4 3 → 6 4 → 6 5 → 4 6 → 4	1 → 0 7 → 0 7 → 12 7 → 12 7 → 0 7 → 0
Social	4	0.2	1 → 4 2 → 4 3 → 6 4 → 4 5 → 4 6 → 2	1 → 0 2 → 0 3 → 12 4 → 12 5 → 0 6 → 0

The list of graph properties does not end here, and the author is certainly not privy to them all. Newman [24] gives a comprehensive review of graph theory and graph properties in the context of real-world network applications. Many of the reviewed concepts, including cliques, network resilience, and mixing patterns, are likely also applicable to the multilayer network model.

4.3.2.2. Node Attributes

Single-layer analysis could also explore the effects of different node attributes. "Node attributes" refer to qualitative or quantitative measures of a node, other than nodal graph properties. The attributes of a node might influence its contributions to change propagation phenomena.

A major node attribute in the product layer is *component class*. Classes of components include hardware, software, and documentation. Depending on the situation, different component classes might exhibit different change propagation behavior. For example, in Chapter 5's case study, the requirements document was naturally a strong multiplier, because this component essentially recorded changes to system requirements, which (almost) always led to redesigns among the various technical parts of the system. By contrast, certain software algorithms behaved as constants, because altering these algorithms was cost and time prohibitive. Similarly, the hardware segment of the system was mostly held constant, due to the insertion of a buffer component that shielded the hardware from significant change. In all, the class of a component can affect its change propagation behavior.

Likewise, the change layer's nodes also possess attributes worth investigating. For instance, the *magnitude* of a change in terms of time and resources consumed, obviously might affect propagation behavior. The program in Chapter 5's case study measured change magnitude on a scale of 0 to 5, depending on, for example, the number of lines of code being added or altered. Over the course of the program, change magnitude seems to have influenced the amount of ensuing propagation, in terms of the number of children and the total number of descendants (i.e., children, grandchildren, great-grandchildren, etc.). Figure 15 plots the average number of both quantities as a function of change magnitude. The curves indicate a direct relationship between change magnitude and the number of resulting children or descendants. The averages plotted by the solid curves only consider change requests (CRs) that had at least one child. Meanwhile, the dotted curves consider all CRs, whether they had children or not. The values for the dotted curve are quite small (<1), because 87% of all changes had zero children (and, hence, zero descendants). Interestingly, Figure 15 also shows a spike in the curves when change magnitude is zero (0). A closer look at the data (tapping into the product layer momentarily) reveals that nearly half (44%) of the 0-magnitude changes were requirements changes. As mentioned previously, the requirements document behaved as a strong multiplier, which apparently skewed the averages for 0-magnitude changes upward (both by factors of 2, relative to the averages without them).

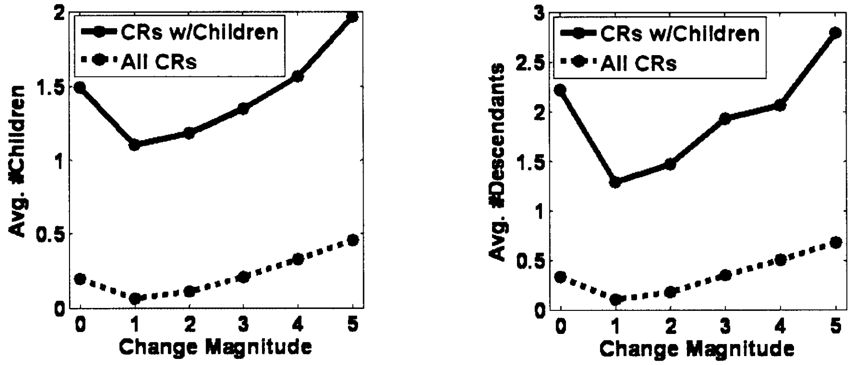


Figure 15. Change Magnitude and the Amount of Propagation

Another node attribute in the change layer is the *approval status* of each change. The approval status refers to whether a change request is accepted, rejected, or pending. Giffin et al. [17] found that approval status had a relationship with change magnitude. More specifically, half the 0-magnitude changes were ultimately rejected, but the majority of all 1- to 5-magitude changes were accepted. Furthermore, Giffin found that lower magnitude changes were more common, regardless of their ultimate approval status. Thus, these example analyses demonstrate the significance of two node attributes in the change layer: magnitude and approval status. Others attributes of interest would be process time and total cost per change.

Finally, investigations of the social layer might consider an engineer's *organizational role* as an important node attribute. The role of an engineer (e.g., specialists, team lead, systems engineer, or manager) will likely impact his or her responsibilities in the engineering change management process. The case study in Chapter 5 will quantitatively elaborate on this interesting relationship further.

In summary, several tools and metrics facilitate single-layer analysis of the multilayer network model. Some of these tools and metrics are applicable to any layer, such as the DSM and graph properties. Others are tailored to specific layers, including CPM (product layer), change motif analysis (change layer), and node attributes (layer-specific). Many of these single-layer methods also factor into double-layer analyses, as described next.

4.4. Double-layer Analysis

Double-layer analysis considers two layers simultaneously by taking account of the inter-layer edges in the multilayer network model. Past research on change propagation has produced some extremely promising double-layer tools and metrics. Additionally, general research on product development and project management has developed a few others with specific implications for engineering change management.

4.4.1. Double-layer Tools

Double-layer tools generally attempt to capture or visualize the interactions between two layers. Table 9 outlines the double-layer tools discussed next.

Table 9. Double-layer Tools

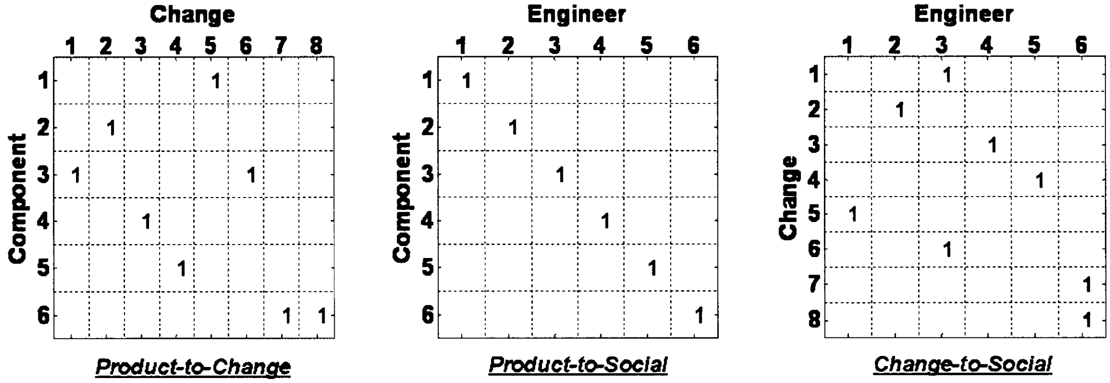
Tool	Layers	Reference
Domain Mapping Matrix (DMM)	Any pair	8
Propagation DSM	Product & Change	17
Change Propagation Frequency Matrix (CPFM)	Product & Change	17
Product/Propagation DSM Overlay	Product & Change	17
Alignment Matrix	Product & Social	28
Engineer Propagation DSM	Social & Change	New

Many of these double-layer tools, such as the DMM, Propagation DSM, and Alignment Matrix, were introduced in Chapter 2. Others, like the CPFM and Product/Propagation DSM overlay, have been previously utilized, but not formally developed, in the literature.

The last double-layer tool listed in Table 9, named the Engineer Propagation DSM, is a tool proposed for the first time by this thesis to help explore the social layer involved in change propagation.

4.4.1.1. Domain Mapping Matrix

The *Domain Mapping Matrix* (DMM) is a variation of the DSM that represents inter-layer edges, as opposed to the intra-layer edges in the traditional DSM. The DMM is a rectangular (but possibly square) matrix that counts the number of dependencies between nodes of two different layers. Danilovic and Browning [8] argue that the DMM can help an organization make better decisions in light of these inter-layer dependencies. Returning to the Project X example, Figure 16 shows the three DMMs that map the product, change, and social layers in pairs of two.



Change

Component	1	2	3	4	5	6	7	8
1					1			
2		1						
3	1					1		
4			1					
5				1				
6							1	1

Product-to-Change

Engineer

Component	1	2	3	4	5	6
1	1					
2		1				
3			1			
4				1		
5					1	
6						1

Product-to-Social

Engineer

Change	1	2	3	4	5	6
1			1			
2		1				
3				1		
4					1	
5	1					
6			1			
7						1
8						1

Change-to-Social

Figure 16. DMMs for Project X

The DMMs for Project X are not terribly interesting, because each inter-layer edge is relatively one-to-one. This is most clear in the product-to-social DMM, which essentially translates to "engineer *n* is in charge of component *n*." Meanwhile, in the product-to-change DMM, each individual change only affects one component. Finally, in the change-to-social DMM, each change is implemented by only one engineer. Consequently, these DMMs are not good candidates for meaningful clustering.

However, in more complex cases, the DMM might reveal more coupling among layers, and hence, opportunities to restructure an organization's teams or operation. For example, Danilovic and Browning explain how a multi-project business might cluster a project-to-organization DMM to identify ways to coordinate its projects with its organization's technical competencies. Likewise, the program in Chapter 5's case study restructured its organization based on similar logic. In the middle of system development, the program created integrated program teams (IPTs), each of which united the designers, testers, and integrators for a particular software segment. Before this restructuring, these people were disadvantageously dispersed in the organization. Interestingly, this strategic move led to an onslaught of change requests, because the multidisciplinary IPTs fostered better communication between people dealing with the same parts of the system. The IPTs unsurprisingly discovered a large number of problems with initial design decisions. Thus, DMM-type strategies can have significant implications for engineering change management.

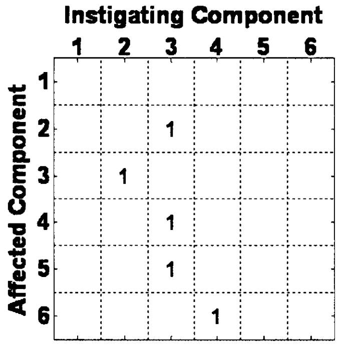
4.4.1.2. Propagation DSM

The next double-layer tool, the *Propagation DSM*, is yet another augmentation of the DSM for representing propagation relationships among product components. It's important not to confuse the Propagation DSM (formerly called the "Change DSM" in [17]) with the change layer's DSM, since the axes and underlying concepts are completely different. A Propagation DSM is a matrix that counts instances of parent-child propagation from one component to another over some time period in the design process. As such, it taps into both the product layer and change layer.

The Propagation DSM is a square matrix with a row (m) and column (n) for each product component. Element (m, n) of the Propagation DSM counts the number of times a parent change in the *instigating* component n spawned a child change in the *affected* component m. In the original definition by Giffin et al. [17], only *accepted* child changes are counted in the Propagation DSM, regardless of the approval status of the parent change.

The Propagation DSM is especially useful in calculating a double-layer metric called the Component Change Propagation Index (CPI), described as a double-layer metric soon.

Figure 17 displays the Propagation DSM for Project X. The matrix indicates, for instance, that a change originating in component #3 caused one accepted child change each in components #2, #4, and #5.

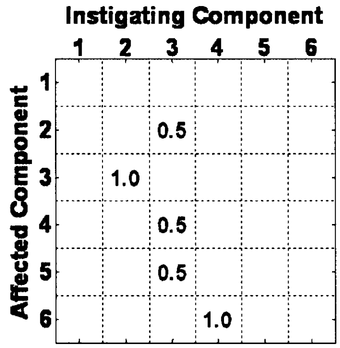


Affected Component \ Instigating Component	1	2	3	4	5	6
1						
2			1			
3		1				
4			1			
5			1			
6				1		

Figure 17. Propagation DSM for Project X

4.4.1.3. Change Propagation Frequency Matrix

A useful derivative of the Propagation DSM is another tool called the *Change Propagation Frequency Matrix* (CPFM) [17]. The CPFM divides each column (n) of the Propagation DSM by the total number of changes that occurred in component n. As a result, element (m, n) of the CPFM gives the frequency (0 to 1) with which a parent change in component n led to a child change in component m. The CPFM for Project X has been generated in Figure 18. The CPFM shows, for example, that half of all changes to component #3 caused changes to components #2, #4, and #5 as well.



Affected Component \ Instigating Component	1	2	3	4	5	6
1						
2			0.5			
3		1.0				
4			0.5			
5			0.5			
6				1.0		

Figure 18. Change Propagation Frequency Matrix for Project X

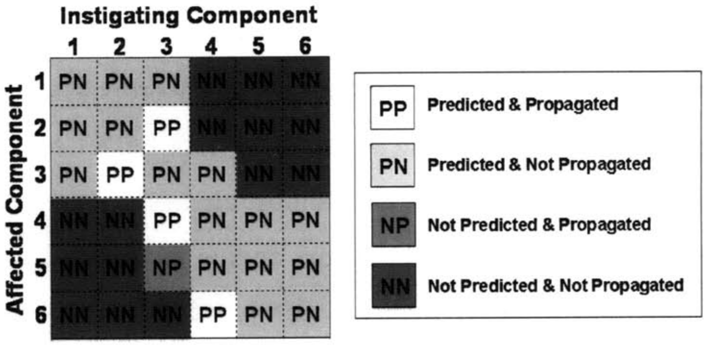
The CPFM might give some indication of the strength of dependencies among product components. Mechanical systems, for example, frequently propagate changes because of the strong interdependence of their physical parts. Indeed, Eckert et al. [14] reports that in a helicopter design, a change to the engine almost always causes a change to the bare fuselage, the transmission, the avionics, and the engine auxiliaries, among others. By contrast, modular software systems may be less prone to change propagation. For example, the software-dominated system in Chapter 5's case study usually exhibited a low propagation frequency of less than 10% between all subsystems [17].

The frequency of change propagation might also increase over time. If the CPFM was tracked over the course of product development, an organization might notice the effects of diminishing tolerance margins. Eckert et al. argues that each product component has some margin for change. However, once that that margin is depleted, the component is subject to propagating changes elsewhere. As such, a time-stamped CPFM would reflect this margin depletion through time, if other factors were held constant.

4.4.1.4. Product/Propagation DSM Overlay

Another useful perspective comes from *overlaying* the Propagation DSM with the Product DSM (i.e., the DSM of the product layer). Such an overlay reveals where propagation was predicted versus where it actually occurred. The reasoning here is that the Product DSM captures all the connections among the components of a product. Consequently, the Product DSM predicts where parent-child propagation could occur, assuming it can only occur between adjacent components. Meanwhile, the Propagation DSM shows where parent-child propagation actually occurred. Thus, the overlay of these matrices compares *theory* with *practice*. Giffin et al. performed an equivalent overlay in [17], but did not formalize the tool in any detail.

As an example, Figure 19 performs the overlay for Project X. Although the Product DSM, strictly speaking, does not denote self-connections (i.e., ones on the diagonal), the overlay here assumes self-propagation is predictable.



	Instigating Component					
Affected Component	1	2	3	4	5	6
1	PN	PN	PN	NN	NN	NN
2	PN	PN	PP	NN	NN	NN
3	PN	PP	PN	PN	NN	NN
4	NN	NN	PP	PN	PN	PN
5	NN	NN	NP	PN	PN	PN
6	NN	NN	NN	PP	PN	PN

PP	Predicted & Propagated
PN	Predicted & Not Propagated
NP	Not Predicted & Propagated
NN	Not Predicted & Not Propagated

Figure 19. Product/Propagation DSM Overlay for Project X

The overlay exposes four types of behavior, each with several possible explanations:

- *Predicted and Propagated* (PP) means that the Product DSM predicted propagation by virtue of the components' connectivity, and that propagation did actually occur as predicted. This behavior, called *direct propagation*, is relatively tolerable, because propagation, while still non-ideal, occurred as expected. Project X experienced this situation in 16 out of 36 pairs of components.

- *Predicted and Not Propagated* (PN*)* means that the Product DSM predicted propagation, but that propagation did not occur. This behavior is advantageous, because somehow direct propagation was avoided despite component adjacencies. Project X enjoyed this situation in four out of 36 component pairs.

  This thesis suggests that explanations for the avoidance of direct propagation might come from all three layers. Some reasoning could lie in the product layer. For instance, the product might have been designed cleverly in the areas that did not propagate. Eckert et al. [14] would argue that these areas simply had ample tolerance margins relative to the amount of change they experienced. Furthermore, the change layer may reveal that propagation did not occur because the changes were low magnitude and therefore less likely to propagate. Most interestingly, the social layer may offer some explanation in terms of an engineer's competence and communication with others. Namely, maybe the engineers working on these non-propagating changes were uniquely talented or highly aware of neighboring component dependencies. As a result, they were able to find clever redesign solutions that avoided propagation. Similarly, perhaps good communication between interdependent engineers yielded solutions that absorbed these changes rather than multiplying them.

- *Not Predicted and Propagated* (NP) means that the product DSM did not predict propagation, yet propagation still occurred. This behavior, called *indirect propagation*, contradicts the conventional belief that parent-child propagation can only occur between adjacent components. Project X experienced indirect propagation once between component #3 and component #5. One explanation for this behavior is that the Product DSM is incomplete (i.e., missing technical interfaces), such that the indirect propagation is actually just direct propagation in disguise. The program in Chapter 5's case study experienced an alarming amount of indirect propagation, despite having an accurate DSM. The case study explores this phenomenon further and finds some context-based explanations.

- *Not Predicted and Not Propagated* (NN) means that the product DSM did not predict propagation and propagation did not occur. This behavior is expected and the least interesting. Project X experienced this situation in 15 out of 36 pairs of components.

Given any of these behavior types (PP, PN, NP, and NN), an organization can benefit from investigating their causes in more depth. When propagation did occur, whether predicted or not (i.e., PP or NP), the organization might find ways to improve its operation to avoid propagation in the future. When propagation did not occur (i.e., PN or NN), the organization should evaluate the reasons for its success, and formally adopt or encourage any good practices. Overall, the Product/Propagation DSM overlay can facilitate a useful self-assessment exercise to inform better engineering change management. The concepts of direct and indirect propagation will be illustrated further by the newly proposed double-metric, Propagation Directness.

4.4.1.5. Alignment Matrix

The *Alignment Matrix* is a double-layer tool developed by Sosa et al. [28] that looks for patterns between the product layer and social layer. Inconsistencies between product interfaces and communication links have significant implications for engineering change management.

The Alignment Matrix performs an overlay of the Product DSM and the Social DSM. The premise is that if components *a* and *b* are connected in the Product DSM, then communication should exist between engineers *a* and *b* in the Social DSM. The Alignment Matrix discovers discrepancies between the two DSMs for further analysis. One weakness of the Alignment Matrix is that it is only applicable when there is a one-to-one mapping between the product and the organization. If a one-to-one mapping does not exist, as may be the case for large and complex development projects [27], use of the Alignment Matrix is not as straightforward. However, Eppinger [12] and Morelli et al. [22] have found successful workarounds in similar situations. The Project X example was fabricated to have a one-to-one mapping (i.e., six engineers neatly assigned to six components), and its Alignment Matrix appears in Figure 20.

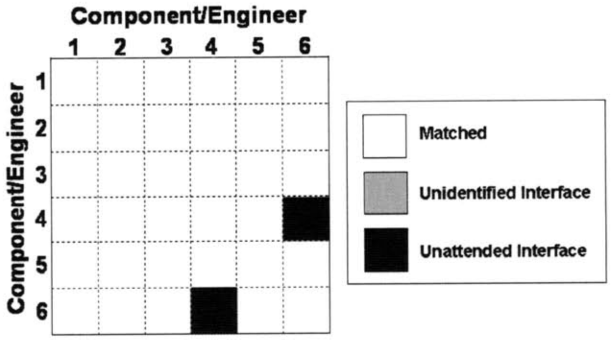


Figure 20. Alignment Matrix for Project X

In general, the Alignment Matrix exposes two types of mismatches, *unidentified interfaces* and *unattended interfaces*, between the Product and Social DSMs [28]. An unidentified interface is a communication link lacking a corresponding product interface. Unidentified interfaces are usually welcome surprises. That is, the fact that engineers communicated unexpectedly might expose an unforeseen, but critical, product interface. Project X had no unidentified interfaces.

By contrast, unattended interfaces can pose problems for engineering change management and product development efforts. An unattended interface is a technical interface in the product layer which is missing a corresponding communication link in the organization layer. Project X had one unattended interface between components #4 and #6, in both directions. Unattended interfaces can be detrimental when critical product interfaces go unnoticed from the lack of communication. The missing communication can lead to poor initial designs that need changing later. Thus, product and social layer inconsistencies can have implications for engineering change management. In fact, Sosa et al. gives the example of a Pratt & Whitney project in which an unattended interface resulted in the company having to disassemble, redesign, and rebuild several test engines. The program in Chapter 5's case study likely had a similar experience when they established integrated program teams (IPTs, referring back to the DMM discussion on page 66). It's reasonable to suppose that the IPTs discovered previously unattended interfaces that now necessitated a large number of design changes.

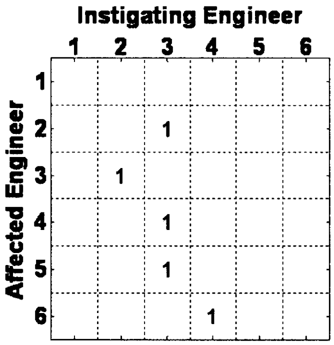
Overall, an organization can use the Alignment Matrix to check the consistency between its product and communication structure, and improve its operation by addressing the mismatches. The benefit for engineering change management is that future changes are prevented through collaborative design decisions.

4.4.1.6. Engineer Propagation DSM

The last double-layer tool, the *Engineer Propagation DSM*, is an invention of this thesis. The Engineer Propagation DSM is an extension of the original Propagation DSM that now taps into the change layer and the social layer, rather than the change layer and product layer.

The Engineer Propagation DSM is a square matrix with a row (*m*) and column (*n*) for each engineer. Element (*m*, *n*) of the Engineer Propagation DSM counts the number of times a parent change implemented by the *instigating* engineer *n* spawned a child change implemented by the *affected* engineer *m*. Only *accepted* child changes are counted in the Engineer Propagation DSM, just as in the Propagation DSM.

Figure 21 shows the Engineer Propagation DSM for Project X. The matrix indicates, for instance, that engineer #3 implemented a parent change that caused three child changes implemented by engineers #2, #4, and #5.



Affected Engineer \ Instigating Engineer	1	2	3	4	5	6
1						
2			1			
3		1				
4			1			
5			1			
6				1		

Figure 21. Engineer Propagation DSM for Project X

The greatest use of the Engineer Propagation DSM is for calculating the Engineering Change Propagation Index (Engineer-CPI), which is a new double-layer metric introduced shortly.

4.4.2. Double-layer Metrics

A number of promising double-layer metrics have been developed and suggested in the literature. All of them are specific to research on change propagation and engineering change management. Table 10 lists these metrics along with the two layers they each draw upon.

Table 10. Double-layer Metrics

Metric	Layers	Reference
Propagation Directness	Product & Change	New
Component Change Propagation Index (Component-CPI)	Product & Change	17, 31
Engineer Change Propagation Index (Engineer-CPI)	Social & Change	New
Change Acceptance/Reflection Index (CAI/CRI)	Product & Change	17
Proposal Acceptance Rate (PAR)	Social & Change	17

Among these double-layer metrics are two new ones introduced by this thesis. The first is Propagation Directness, which reflects whether propagation is direct or indirect. The second is the Engineer-CPI, used to characterize the propagation effects of an engineer's work.

4.4.2.1. Propagation Directness

Propagation Directness is a double-layer metric proposed for the first time by this thesis. Propagation Directness, PD, is defined as the number of product interfaces spanned by an instance of parent-child propagation.

PD can be calculated using the Propagation DSM and geodesic paths (a graph property) in the Product DSM. Specifically, if the Propagation DSM indicates that a change propagated from component n to component m, then the PD of that propagation is equal to the geodesic (shortest) path from component n to m in the Product DSM.

Propagation Directness reflects whether propagation is direct or indirect. Direct propagation implies PD $\leq$ 1, because direct propagation occurs when a child change arises in a component that is adjacent (PD = 1) or identical (PD = 0) to the component affected by the parent change. By contrast, indirect propagation has PD > 1, because a child change arises in a component nonadjacent to the component affected by the parent change. As mentioned earlier (page 69), direct and indirect propagation correspond with the PP and NP behavior types, respectively, that may be exposed when overlaying the Product DSM with the Propagation DSM. The relationship between these concepts is summarized in Table 11.

Table 11. Propagation Behavior and Propagation Directness

Behavior Type	Alternative Name	Propagation Directness (PD)
PP	Direct Propagation	≤ 1
NP	Indirect Propagation	>1

Returning to the Project X application, Table 12 calculates the Propagation Directness of each of the five occurrences of parent-child propagation in which the child change was accepted. Indirect propagation only occurred once from component #3 to component #5; these two components are separated by two interfaces in the product (PD = 2). All other parent-child propagation in Project X was direct (PD = 1).

Table 12. Propagation Directness for Project X

Parent Change	Child Change	Parent Component	Child Component	Propagation Directness (PD)
1	1	3	2	1
1	2	3	4	1
1	3	3	5	2
2	4	2	1	1
3	7	5	6	1

Propagation directness has obvious implications for the successful prediction of change propagation. Conventional wisdom says that Propagation Directness should always be PD ≤ 1; in other words, all propagation should be direct propagation. Accordingly, the CPM suite [6] notably only allows for direct propagation, but emphasizes that recursive direct propagation can form propagation chains spanning several product interfaces. However, the program in Chapter 5's case study experienced a considerable amount of indirect propagation, in which Propagation Directness was usually PD = 2, and occasionally PD = 3.

4.4.2.2. Component Change Propagation Index

The Component Change Propagation Index (Component-CPI, formerly just "CPI") quantifies a product component's change propagation behavior. As defined by Suh and de Weck [31] and refined by Giffin et al. [17], the index is a number between -1 and +1, calculated as follows:

Equation 1

$$CPI(j) = \frac{C_{out}(j) - C_{in}(j)}{C_{out}(j) + C_{in}(j)}$$

In Equation 1, the Component-CPI equation, $C_{in}(j)$ and $C_{out}(j)$ are the numbers of changes that propagated in and out of component *j*, respectively. More simply, $C_{in}(j)$ and $C_{out}(j)$ are the in-degree and out-degree, respectively, of the Propagation DSM, a double-layer tool described earlier.

The Component-CPI's quantitative spectrum corresponds helpfully with the qualitative behavior spectrum proposed by Eckert et al. [14]. That is, a *positive* CPI indicates that a component is a *multiplier,* because it caused more changes to other components than it internalized ($C_{out} > C_{in}$). Meanwhile, a *negative* CPI describes an *absorber* because the component internalized more changes than it caused ($C_{in} > C_{out}$). At the extremes, a CPI of +1 indicates a perfect multiplier, and CPI of -1 indicates a perfect absorber. A CPI of zero represents a carrier, which causes and internalizes an equal number of changes ($C_{in} = C_{out}$). Finally, if a component did not cause or internalize any changes ($C_{in} = C_{out} = 0$), it is a constant, and its CPI is undefined.

Suh and de Weck [31] use the Component-CPI as a basis for embedding flexibility in a design. For instance, they recommend that multipliers (and sometimes carriers) are prime targets for flexibility in anticipation of potentially costly propagation behavior by these components. Flexibility will be addressed further as a prevention strategy in Chapter 6's discussion of the management policy implications of change propagation research.

To illustrate the Component-CPI spectrum, Table 13 classifies each component in Project X according to its CPI. The calculations only consider child changes that were ultimately accepted. The critical components for Project X seem to be component #3 (the only multiplier) and its neighbors, components #2 and #4 (both carriers). By contrast, components #5 and #6 were perfect absorbers, and component #1 was a constant and didn't contribute to any propagation.

Table 13. Component-CPIs for Project X

Component #	C_{out}	C_{in}	CPI	Behavior
1	0	0	-	Constant
2	1	1	0	Carrier
3	3	1	0.5	Multiplier
4	1	1	0	Carrier
5	0	1	-1	Absorber
6	0	1	-1	Absorber

4.4.2.3. Engineer Change Propagation Index

To begin quantifying the social layer's contribution to change propagation, this thesis now proposes a new double-layer metric called the *Engineer Change Propagation Index* (Engineer-CPI). The Engineer-CPI extends the Component-CPI concept to the performance of the engineers who *implement* changes. The Engineer-CPI is a number between -1 and +1, calculated analogously to the Component-CPI:

Equation 2

$$CPI(j) = \frac{E_{out}(j) - E_{in}(j)}{E_{out}(j) + E_{in}(j)}$$

In Equation 2, $E_{out}(j)$ is the number of changes that propagated from changes implemented by engineer j. $E_{in}(j)$ is the number of changes implemented by engineer j that propagated from changes implemented by other engineers. More simply, $E_{in}(j)$ and $E_{out}(j)$ are the in-degree and out-degree, respectively, of the Engineer Propagation DSM, the new double-layer tool introduced earlier. The Engineer-CPI quantifies an engineer's behavior with respect to the propagation effects of his (or her) implementation of changes.

The Engineer-CPI spectrum can be interpreted similarly to the Component-CPI spectrum; namely, positive, negative, zero, and undefined Engineer-CPIs correspond with multipliers, absorbers, carriers, and constants, respectively. This thesis argues further that the Engineer-CPI spectrum should also map onto the spectrum of organizational roles. That is, an engineer's CPI should theoretically correspond with his or her job description. *Managers* and *systems engineers* should be *multipliers* ($E_{out} > E_{in}$) because they initiate high-level changes that potentially require many lower-level changes to be completed. For example, a manager might coordinate with customers and consequently change the requirements for a product to satisfy. Similarly, a systems engineer might recognize a high-level problem (e.g., given unsatisfactory test results) and consequently initiate corrective action that propagates through the product. By contrast, *specialists* should behave like *absorbers* ($E_{in} > E_{out}$), because they perform changes in detailed areas of the product where there is little chance of further propagation. Specialists essentially implement changes at the end of propagation chains. Meanwhile, *team leaders* might correspond with *carriers* ($E_{in} = E_{out}$), since they make some high-level changes but are also involved with low-level changes in the product. Finally, *constants* ($E_{in} = E_{out} = 0$) do not seem to have an obvious corresponding organizational role. If an engineer is a constant, that means he or she only implements isolated changes, which have no parent change and no children changes. An interpretation of this behavior might be a good topic for future research.

To illustrate the Engineer-CPI spectrum, Table 14 classifies each engineer in Project X, according to its Engineer-CPI. The calculations only consider child changes that were ultimately accepted. The engineers of Project X exhibit the full spectrum of performance.

Table 14. Engineer-CPIs for Project X

Engineer #	E_{out}	E_{in}	CPI	Performance	Organizational Role
1	0	0	-	Constant	?
2	1	1	0	Absorber	Team Leader
3	3	1	0.5	Multiplier	Manager/Systems Engineer
4	1	1	0	Carrier	Team Leader
5	0	1	-1	Absorber	Specialist
6	0	1	-1	Absorber	Specialist

The Engineer-CPI is a reflection of several factors contributing to the propagation effects of an engineer's implementation of changes. As described above, one of those factors may be an engineer's organizational role. Other performance factors could include an engineer's technical competence and the context of his assignments. Chapter 5's case study explores these compounding factors by analyzing real-world data.

4.4.2.4. Change Acceptance and Reflection Index

As proposed by Giffin et al. [17], the *Change Acceptance Index* (CAI) is the fraction of proposed changes ultimately accepted by a product component. The CAI of component *j* is calculated as follows:

Equation 3 $$CAI(j) = \frac{\text{Total \# of Changes Accepted by Component } j}{\text{Total \# of Changes Originally Proposed for Component } j}$$

The related *Change Reflection Index* (CRI) of component *j* is calculated similarly:

Equation 4 $$CRI(j) = \frac{\text{Total \# of Changes Rejected by Component } j}{\text{Total \# of Changes Originally Proposed for Component } j}$$

At any given time during product development, the CAI and CRI of a component may not sum to one, because of pending change requests.

The CAI (and CRI) measures a component's openness (and stubbornness) to accommodate change. For example, Table 15 lists the CAIs and CRIs of the components in Project X. Components #2, #3, #4, and #5 are perfect acceptors of change with CAIs = 1, while component #1 is a perfect change reflector with CRI = 1. Meanwhile, component #6 accepts and reflects changes equally (CAI = CRI = 0.5).

Table 15. CAIs and CRIs for Project X

Component #	# Accepted	# Rejected	CAI	CRI
1	0	1	0	1
2	1	0	1	0
3	2	0	1	0
4	1	0	1	0
5	1	0	1	0
6	1	1	0.5	0.5

4.4.2.5. Proposal Acceptance Rate

The Engineer-CPI measures the propagation effects of an engineer's performance as an implementer of change. Another double-layer metric, called the *Proposal Acceptance Rate* (PAR), measures an engineer's performance as a *proposer* of change. Such a metric was suggested by Giffin [16], but not developed in detail.

When an engineer proposes a change request, the request is ultimately accepted or rejected. The PAR is essentially an engineer's rate of acceptance as a proposer of changes. The PAR of engineer *j* can be intuitively calculated as follows:

Equation 5

$$PAR(j) = \frac{\text{Total \# of Changes Proposed by Engineer } j \text{ and Implemented}}{\text{Total \# of Changes Proposed by Engineer } j}$$

An engineer's PAR can reflect his or her skill, attitude, and expertise. A high PAR might mean the engineer is innovative, while a low PAR might imply he or she tends to have bad ideas. However, other rationalizations for the PAR of a particular engineer could exist. For instance, a truly innovative engineer could still have a low PAR if the organization or product is sluggish or stubborn to make changes. Conversely, a less creative engineer could still have a high PAR if the organization or product is especially receptive of change. The case study in Chapter 5 will explore these competing explanations in a real-world scenario.

4.5. Triple-layer Analysis

Triple-layer analysis is the most complex since it considers all three layers of the multilayer network model simultaneously.

4.5.1. Triple-layer Tools

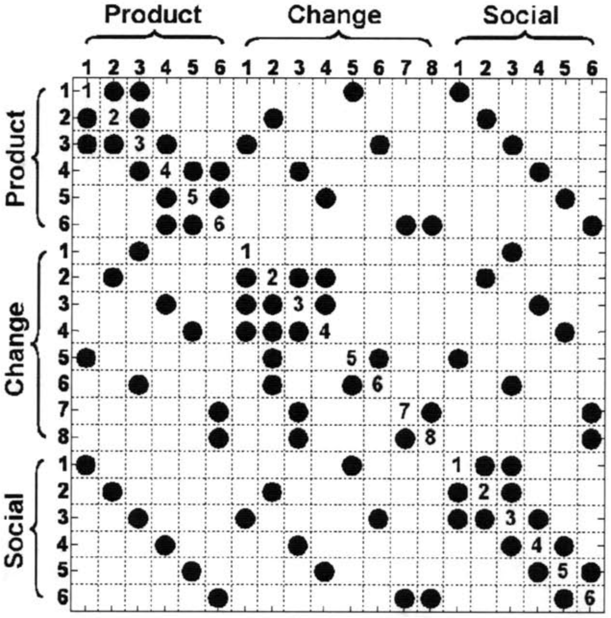
To the author's knowledge, research on engineering change management, product development, and project management has not produced many triple-layer tools or metrics. Bartolomei [1] appears to be an exception in his attempt to analyze three or more layers simultaneously using the ESM. Additionally, this thesis proposes another triple-layer tool that checks the consistency between the Product DSM, Propagation DSM, and Social DSM. For completeness, these two tools are listed in Table 16.

Table 16. Triple-layer Tools

Tool	Layers	Reference
Engineer Systems Matrix (ESM)	All	1
Product/Propagation/Social DSM Overlay	All	New

4.5.1.1. Engineer Systems Matrix

As introduced in Chapter 2, Bartolomei's [1] *Engineering Systems Matrix* (ESM) is a DSM augmented to include nodes from multiple domains and edges within and across those domains. The multilayer network model has three layers, or domains, for the ESM to encompass. For example, the ESM for Project X is shown in Figure 22, where the locations of the product, change, and social layers have been bracketed.



		Product						Change								Social					
		1	2	3	4	5	6	1	2	3	4	5	6	7	8	1	2	3	4	5	6
Product	1	1	●	●								●				●					
	2	●	2	●					●								●				
	3	●	●	3	●			●					●					●			
	4			●	4	●	●			●									●		
	5				●	5	●				●									●	
	6				●	●	6							●	●						●
Change	1			●				1										●			
	2		●					●	2	●	●						●				
	3				●			●	●	3	●								●		
	4					●		●	●	●	4									●	
	5	●							●			5	●			●					
	6			●					●			●	6					●			
	7						●			●				7	●						●
	8						●			●				●	8						●
Social	1	●										●				1	●	●			
	2		●						●							●	2	●			
	3			●				●					●			●	●	3	●		
	4				●					●								●	4	●	
	5					●					●								●	5	●
	6						●							●	●					●	6

Figure 22. ESM for Project X

An obvious but important feature to note is that the ESM can be decomposed into DSMs on the diagonal and DMMs off the diagonal. Figure 23 illustrates this decomposition for Project X.

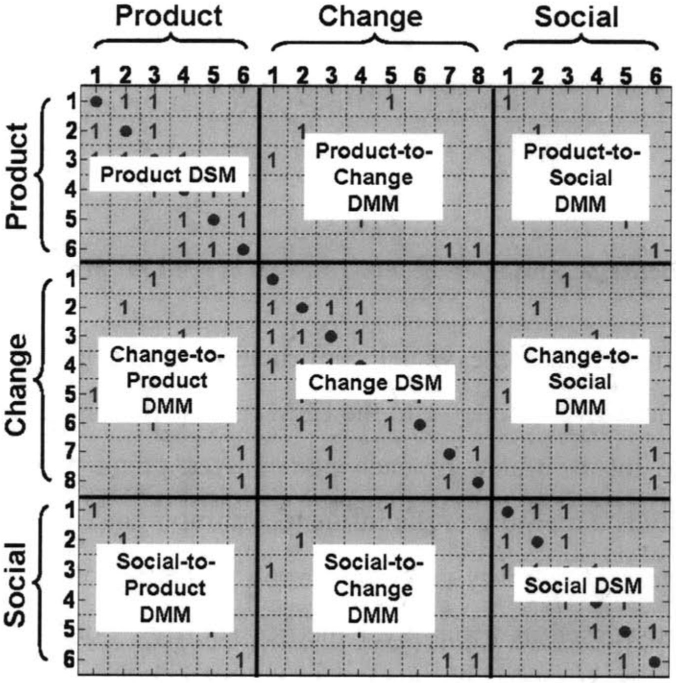


Figure 23. Decomposed ESM for Project X

The ESM representation highlights that the multilayer network essentially forms a single grand network with multiple types of nodes and edges (similar to a multipartite graph). Accordingly, Figure 24 draws the grand network using a spring embedding algorithm that treats all nodes and edge equally, regardless of type. All the edges are drawn without arrows for simplicity, although the change layer's intra-layer edges are, in fact, unidirectional. The spring embedding algorithm [15] works by imagining each directed edge is a spring pushing or pulling on its two incident nodes. The algorithm then positions the nodes such that the network's imaginary potential spring energy is minimized.

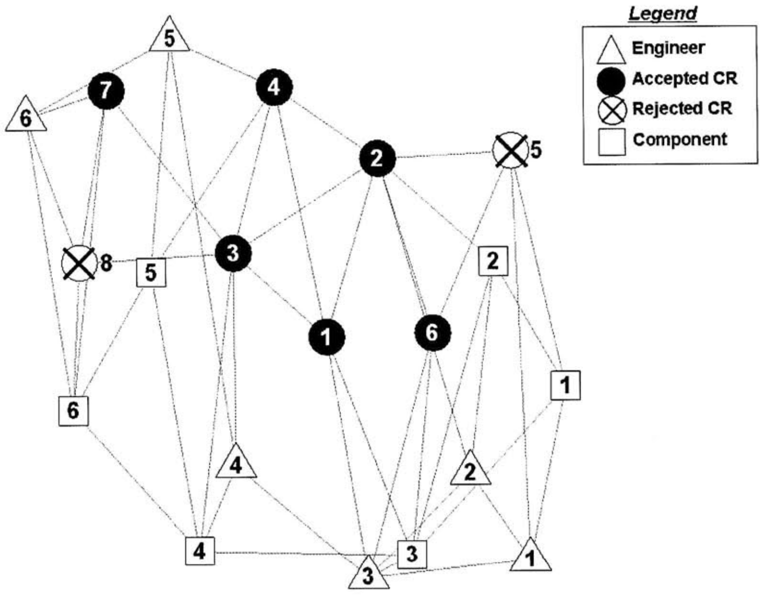


Figure 24. Spring Embedded Drawing of ESM for Project X

The network drawing gives some visual confirmation of the importance of individual nodes in the grand network. Relative to the ESM representation, the drawing makes it is easier to appreciate the centrality of each node in the grand network. As explained previously, centrality is a measure of the influence or importance of a node in terms of its degree or betweenness. Centrality and other graph properties of the grand network will be calculated and discussed as triple-layer metrics shortly. For now, it is qualitatively evident that changes #1 and #3 and component #3 are at least some of the most influential nodes in the grand network. This observation will prove to be quantitatively true as well.

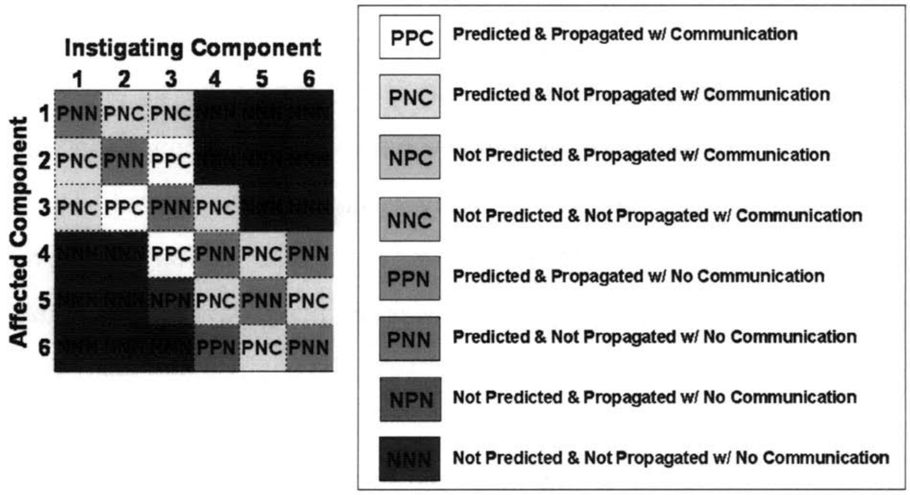
Bartolomei employs similar reasoning to rank the stakeholders of a project by viewing the ESM as a single grand network. Overall, the ESM can remind analysts that all the components, changes, and engineers in the multilayer network are all part of one broader system of interactions.

4.5.1.2. Product/Propagation/Social DSM Overlay

This thesis now introduces a novel tool that checks the consistency among all three layers of the multilayer network model. The tool overlays three DSMs: the Product DSM, the Propagation DSM, and the Social DSM. Earlier it was shown that an organization could overlay the first two of these DSMs to compare where change propagation was predicted against where it actually occurred. That overlay was a double-layer tool. The triple-layer tool proposed here brings in the Social DSM for more clarification. The overlay of all three DSMs should reveal where propagation was predicted, where propagation actually occurred, and where communication existed between the engineers in charge.

The overlay assumes that a one-to-one mapping exist between the product layer, change layer, and social layer. That is, engineer *n* is in charge of component *n*, and makes all the changes affecting component *n*. Project X was fabricated to meet this criterion. However, other engineering endeavors could very well have engineers working on changes outside of their immediate jurisdiction. Such a situation would complicate the applicability of this tool. In these situations, a few workarounds might be possible. One workaround may be to consider a higher level of abstraction (e.g., subsystems instead of components, subsystem changes instead of component changes, and teams instead of engineers) in which the mapping between the Product, Propagation, and Social DSM might be more straightforward. For example, if engineers *m* and *n* each sporadically work on both components *m* and *n*, a one-to-one mapping between engineers and components does not exist. However, engineers *m* and *n* and components *m* and *n* could be viewed as one subsystem and one team, respectively, such that a one-to-one-mapping would exist between subsystem *mn* and team *mn.* Another workaround is to only consider one propagation chain, or set of change descendants, at a time; that way, it is more likely that the people (or teams) involved with the changes are each focusing on only one part of the product. Hence, the one-to-one mapping would be restored for this smaller set of changes, even if certain people were temporarily working outside of their primary jurisdictions.

To illustrate the theoretical power of the overlay, Figure 25 shows its application to Project X.



Affected Component \ Instigating Component	1	2	3	4	5	6
1	PNN	PNC	PNC	NNN	NNN	NNN
2	PNC	PNN	PPC	NNN	NNN	NNN
3	PNC	PPC	PNN	PNC	NNN	NNN
4	NNN	NNN	PPC	PNN	PNC	PNN
5	NNN	NNN	NPN	PNC	PNN	PNC
6	NNN	NNN	NNN	PPN	PNC	PNN

PPC	Predicted & Propagated w/ Communication
PNC	Predicted & Not Propagated w/ Communication
NPC	Not Predicted & Propagated w/ Communication
NNC	Not Predicted & Not Propagated w/ Communication
PPN	Predicted & Propagated w/ No Communication
PNN	Predicted & Not Propagated w/ No Communication
NPN	Not Predicted & Propagated w/ No Communication
NNN	Not Predicted & Not Propagated w/ No Communication

Figure 25. Product/Propagation/Social DSM Overlay for Project X

The overlay exposes eight types of behavior. Each behavior is distinguished by whether propagation was predicted, whether it occurred, and whether communication existed between the engineers involved. It's excessive to list the eight combinations again here, as Figure 25 does this already. The most remarkable element of each behavior is whether or not communication existed (from the Social DSM). If communication did exist and propagation was avoided, then good communication might have contributed to this success. However, if communication existed and propagation still occurred, then the communication might have been ineffective. An organization may want to scrutinize whether better communication could have prevented the propagation, or if the propagation was unavoidable even with mutual consultation. Moreover, the overlay reveals whether propagation was predictable in the first place (from the Product DSM). If propagation was predictable, communication should definitely have occurred to address possible propagation effects. If propagation was not predictable, an organization may want to initiate a new communication link to attend to the unrealized interface. Many of these concepts are reminiscent of the Product/Propagation DSM overlay and Sosa et al.'s Alignment Matrix [28].

4.5.2. Triple-layer Metrics

As listed in Table 17, the only triple-layer metrics to be discussed here are graph properties.

Table 17. Triple-layer Metrics

Metric	Layers	Reference
Graph Properties	All	1, 5, 10, 24

The development of more triple-layer network metrics (double-, triple-, and more) is a great area for future research.

4.5.2.1. Graph Properties

Just as graph properties were applicable to any single-layer, they can also help describe the grand network formed by all three layers. In the context of the grand network, all nodes and edges are treated equally. Consequently, the graph properties of individual nodes take on new meaning in the grand network relative to their properties in their respective single-layers. For example, Bartolomei [1] compared the centrality of stakeholders in the social domain with their centralities in the ESM. He found that some stakeholders who were highly central in the social domain maintained their influence in the ESM context. Meanwhile, one stakeholder actually ascending in the centrality rankings, suggesting that his significance within the grand scheme is greater than in the smaller social domain.

A similar test can be conducted for the grand network of Project X. Table 18 compares the degree ranking of each node in its respective single-layer with its degree ranking in the grand network, as pictured in Figure 24. The ranking of engineers essentially remained the same between the social layer and the grand network. Engineer #6 increased his relative degree, primarily because he implemented two changes and most other engineers only implemented one. Similarly, in the ranking of product components, component #6 (underlined in Table 18) increased its ranking considerably because it was affected by two changes, while most other components were only affected by one.

Table 18. Component and Engineer Degree Ranking Comparison for Project X

Layer	Rank	Node#	Degree in Single Layer
Product	1	3	6
	2	4	6
	3	1	4
	4	2	4
	5	5	4
	6	6	4
Social	1	3	6
	2	1	4
	3	2	4
	4	4	4
	5	5	4
	6	6	2

Layer	Rank	Node#	Degree in Grand Network
Product	1	3	12
	2	4	10
	3	6	10
	4	1	8
	5	2	8
	6	5	8
Social	1	3	12
	2	1	8
	3	2	8
	4	4	8
	5	5	8
	6	6	8

Overall, graph properties of the grand network, such as centrality, can provide useful insights into the relative influence of items in the grand scheme of engineering change management. For instance, an organization could look for components of high centrality in the grand network to find critical spots in the product. A highly central component is likely the subject of extensive change. The organization may consider redesigning or buffering that component so that it does not consume so much time, money, and resources in the future. Similarly, an engineer of high centrality in the grand network is likely a systems engineer, high performer, or go-to man in the organization. By contrast, an engineer of low centrality might be a specialist, an underperformer, or someone who is underutilized. In summary, graph properties can be useful measures in the multilayer network model. The discussion here has focused on centrality, but other graph properties may be equally or more insightful.

4.6. Summary of Repository

This chapter developed a baseline repository of tools and metrics for investigating the multilayer network model of change propagation. The repository enables single-layer, double-layer, and triple-layer analyses. Furthermore, examples from the hypothetical Project X application, previous literature, and observations from Chapter 5's case study illustrated the potential use of each tool and metric.

Table 19 summarizes the repository by clearly categorizing each tool and metric according to the specific layer or layers it targets. The displayed matrix has a row and column for each layer of the model. As such, the items located along the diagonal are single-layer tools and metrics for the corresponding individual layer. The items in the upper-right triangle are double-layer tools and metrics for the corresponding pairs of layers. Finally, the items in the lower left triangle are triple-layer tools and metrics for all three layers at once. The tools and metrics proposed for the first time by this thesis are marked with a "*."

Table 20 complements Table 19 by specifying the data required to exercise any of these tools and metrics in practice. The displayed matrix has a row for each tool or metric and a column for each type of intra-layer and inter-layer edge. Check marks (✓) denote which edge data would be required to use each tool and metric. For example, to construct a DSM for a given layer, an organization would need to know the intra-ledges for that layer. To construct a Propagation DSM, an organization would need to know the intra-ledges of the change layer (i.e., propagation relationships) and the product-to-change inter-layer edges (i.e., which changes affect which components). Intuitively, the reader should note that all the single-layer tools and metrics only require intra-layer edges, because only one layer is targeted at a time. By contrast, the double-layer and triple-layer tools all tap into the inter-layer edges as well, because they focus on multiple layers simultaneously. Finally, the reader should also note that node attributes (page 63) are purposely missing from Table 20 because they do not require data on edges; rather, they require data on the nodes themselves. Overall, Table 20 shows what specific data should be captured by an organization's configuration management system, if the organization (or a researcher) wishes to utilize the multilayer network model and associated tools and metrics.

As highlighted by Table 20, an inherent aspect of the multilayer network model is data collection and mining. Different amounts and types of data are available at different stages of product development. As such, the multilayer network model and the repository of tools and metrics can be employed in three settings: before, during, and after product development. *Before product development*, complete data on the nodes and edges will not exist, because all the components, change requests, engineers, and their relationships will not have manifested themselves yet. In kind, the utility of the multilayer network is limited to prediction and preparatory work. For example, an organization could use a preliminary Product DSM to analyze potential propagation paths. Likewise, an organization may establish teams and communication links (i.e., a Social DSM) in anticipation of the needs of product development and engineering change management. Later, *during product development*, data can be collected through configuration management, which allows the construction of a multilayer network model with whatever fidelity and completeness that the organization wishes. Analysis of the model during product development can be used to guide change impact analysis (e.g., with CPM), organization structuring (e.g., with the DMM and Alignment Matrix), design strategy (e.g., with the DSM, Propagation DSM, Component-CPI, CAI, etc.) and human resource management (e.g., with the Engineer-CPI). Finally, *after product development*, analysis of the multilayer network turns into a lessons-learned effort. At this stage, an organization can use all the data collected over the course of product development to assess its performance in retrospect. Moreover, the data then becomes useful for academic research to further characterize industry's experience with change propagation. Further assessment of the practical applications of the multilayer network model can be found in Chapter 6's discussion of the policy implications for engineering change management.

The Project X application carried throughout this chapter was hypothetical, so the power of the multilayer network model (and tools and metrics) has only been established in theory. The next chapter will demonstrate its practical utility through a case study of a real-world engineering project, specifically the design and construction of a large scale sensor system.

Table 19. Categorized Repository of Multilayer Network Tools and Metrics

<table>
<tr><th></th><th>Product</th><th>Change</th><th>Social</th></tr>
<tr><td><b>Product</b></td><td><b>Tools</b>:
• DSM [4, 11, 30]
• CPM [6, 21]
<b>Metrics</b>:
• Graph properties [5, 10, 24]
• Node attributes, e.g.,
o Component class [17]</td><td><b>Tools</b>:
• DMM [8]
• Propagation DSM [17]
• CPFM [17]
<b>Metrics</b>:
• Component-CPI [17, 31]
• CAI/CRI [17]
• Propagation Directness*</td><td><b>Tools</b>:
• DMM [8]
• Alignment Matrix [28]</td></tr>
<tr><td><b>Change</b></td><td></td><td><b>Tools</b>:
• DSM [30]
• Change motifs [17]
<b>Metrics</b>:
• Graph properties [5, 10, 24]
• Node attributes, e.g.,
o Approval status [17]
o Magnitude [17]</td><td><b>Tools</b>:
• DMM [8]
• Engineer Propagation DSM*
<b>Metrics</b>:
• Engineer-CPI*
• Proposal Acceptance Rate [16]</td></tr>
<tr><td><b>Social</b></td><td colspan="2"><b>Tools</b>
• ESM [1]
• Product/Propagation/Social DSM Overlay*
<b>Metrics</b>
• Graph properties [1, 5, 10, 24]</td><td><b>Tools</b>:
• DSM [30]
<b>Metrics</b>:
• Graph properties [5, 10, 24]
• Node attributes, e.g.,
o Organizational role*</td></tr>
</table>

* Proposed first by this thesis

□ Single-Layer ▨ Double-Layer ▩ Triple-Layer

Table 20. Data Requirements for Multilayer Network Tools and Metrics

			Intra-Layer Edges			Inter-Layer Edges		
			Product	Change	Social	Product-to-Change	Product-to-Social	Change-to-Social
Single-Layer	Tools	DSM*	✓	✓	✓			
		CPM	✓					
		Change Motif		✓ (+ approval status)				
	Metrics	Graph properties*	✓	✓	✓			
Double-Layer	Tools	DMM*				✓	✓	✓
		Propagation DSM		✓		✓		
		CPFM		✓		✓		
		Product/Propagation DSM Overlay	✓	✓		✓		
		Alignment Matrix	✓		✓		✓	
		Engineer Propagation DSM		✓				✓
	Metrics	Propagation Directness	✓	✓		✓		
		Component-CPI		✓		✓		
		CAI / CRI		✓		✓		
		Engineer-CPI		✓				✓
		Proposal Acceptance Rate		(approval status)				✓
Triple-Layer	Tools	ESM	✓	✓	✓	✓	✓	✓
		Product/Propagation/Social DSM Overlay	✓	✓	✓	✓		
	Metrics	Graph properties	✓	✓	✓	✓	✓	✓

* These tools only require one of the checked (✓) edge categories in their respective rows, per implementation. All other tools/metrics require all the checked edge categories in their respective rows, per implementation.

This page has been intentionally left blank.

5. Case Study

This chapter conducts a case study of a real-world engineering project to demonstrate the practical utility of Chapter 3's multilayer network model of change propagation. The case study investigates the development of a large scale sensor system through a rich dataset extracted from the host program's configuration management records. Furthermore, a lead systems engineer from the program was interviewed to help interpret the results of the study.

To demonstrate the multilayer network model's utility, the case study addresses two topics of significant interest to academia and industry. Both topics are investigated using the tools and metrics from Chapter 4's baseline repository. The first and most novel topic revolves around the social layer's effect on change propagation. The Employee-CPI and Proposal Acceptance Rate (PAR) are used to evaluate an engineer's performance as an implementer and proposer of change, respectively. The second topic involves the general characterization of change propagation. Of particular interest here are the issues of indirect propagation, propagation extent, and component centrality. Overall, the results of the case study suggest that the multilayer network model is a valuable paradigm for future applications in academia and industry.

5.1. The Case

The case under investigation here is that of a large technical program whose purpose was to develop a large scale sensor system. The system consisted of globally distributed hardware and software segments. The hardware segments were largely reused from a pre-existing system, but the software segments essentially required full development. The entire endeavor was very complex and involved multiple stakeholders and distributed users and operators.

The system can be decomposed into 46 areas, or coherent segments of software, hardware, and different levels of associated documentation. These "areas" are roughly analogous to subsystems. A network drawing of the system, borrowed from [16] and [17], appears in Figure 26. Figure 27 draws the same network using a spring embedding algorithm, which gives a slightly clearer representation.

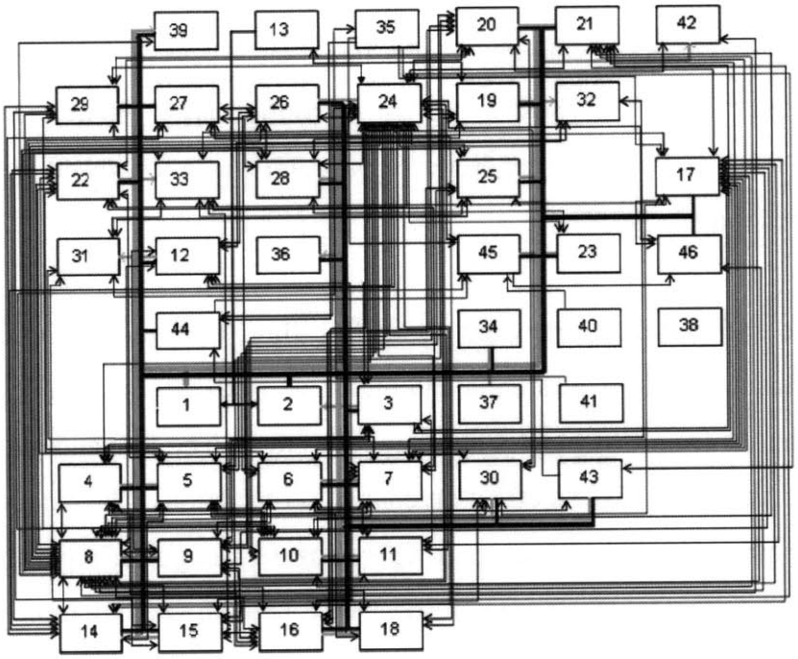


Figure 26. System Network from Case Study (from [16])

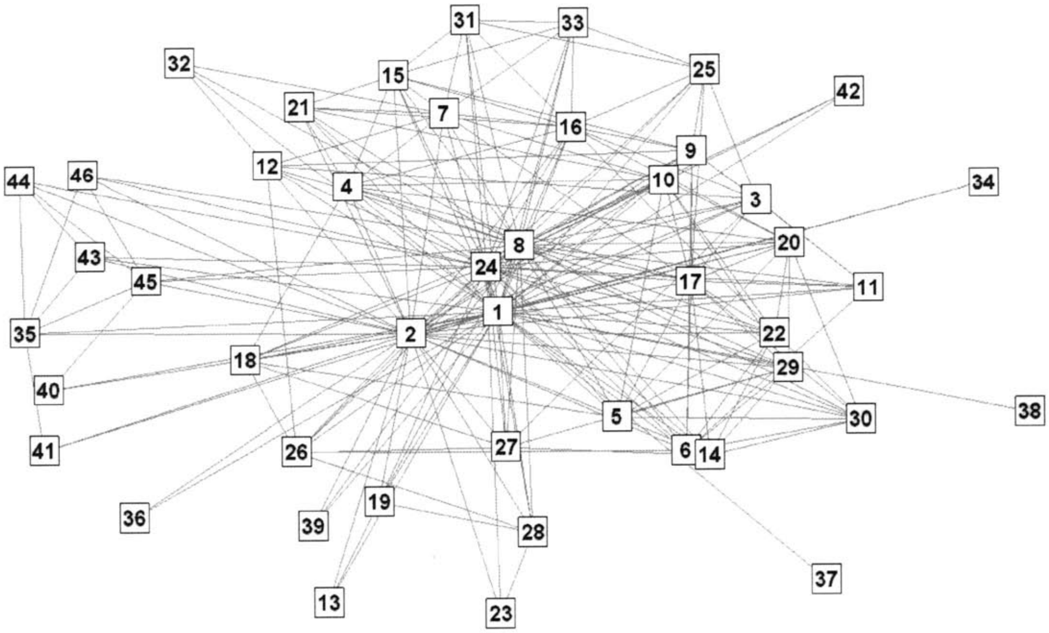


Figure 27. Spring Embedded Drawing of System Network from Case Study

Because the system here was dominated by software areas, most of the interfaces shown in Figure 26 and Figure 27 involve information and data transfer. Unfortunately, the identity of each area and interface is confidential. However, the mere location of the interfaces is far more important for research on change propagation than their detailed nature. Some additional facts about the system were provided through interviews with one of the program's lead systems engineers. Moreover, the author received a rich dataset that facilitated extensive quantitative analysis of the program's experiences with change propagation.

5.2. The Data

The data for this case study was extracted from the program‘s configuration management records. Details about the data extraction methodology can be found in Giffin et al.'s previous analysis of the same program [16, 17]. The full extracted dataset contains detailed information about 41,551 change requests generated by the program over an eight year time period. Each change request (CR) has a separate record, as shown in Table 21.

Table 21. Sample Change Request Record (from [17])

ID Number	12345
Date Created & Last Updated	MAR-Y5, JAN-Y6
Area Affected	19
Change Magnitude	3
Parent ID	8648
Children ID(s)	15678, 16789
Sibling ID(s)	9728
Submitter	eng231
Assignee(s)	eng008 eng231 eng018
Associated Individuals	admin001 eng271
Stage Originated, Defect Reason	[blank], [blank]
Severity	[blank]
Completed?	1

The meaning of each piece of information in Table 21 is as follows:

- *Identification Number* – the CR's unique tracking number assigned in chronological order
- *Date Created* - the month and year that the CR was entered in the change management system
- *Data Last Updated* – the month and year that the CR's record was last updated.
- *Area* - the system area (1 of 46) affected by the CR
- *Change Magnitude* - the expected effort required to evaluate and implement the CR on a scale of 0 to 5 (or -1 if no information was present), based on the number of source lines of code (Total SLOC) affected or Total Hours required
- *Parent ID* – the ID number of the CR's parent change request, if any
- *Children ID(s)* – the ID number of the CR's children change requests, if any
- *Sibling ID(s)* – the ID numbers of the CR's siblings, which include children of the same parent or CRs related in some other significant way.
- *Submitter* – the individual who entered the CR into the change management system.
- *Assignees* – the individual(s) who formally possessed responsibility for the CR at some point, either as an evaluator or implementer
- *Stage Originated, Defect Reason, & Severity* – an indication of whether the CR originated from a documented customer request; this field was often left blank
- *Completed?* – the approval status of the CR, which could be accepted (1), rejected (-1), or still pending (0)

Such a rich dataset is historically rare in related research. Most quantitative investigations of change propagation have largely relied on interviews and surveys of engineers from industry [e.g., 6, 18, 25, 32]. The dataset here offers a unique opportunity for extensive quantitative analysis.

5.3. Model Construction

Hidden in the raw data is a terribly complex multilayer network. In all, the dataset identifies 46 system areas, 41,551 change requests, and 501 engineers and administrators who constitute the nodes of the product layer, change layer, and social layer, respectively. However, knowledge of the intra-layer and inter-layer edges is not complete. Of the intra-layer edges, only those in the product layer and change layer are currently known. The product layer's intra-layers can be gleaned from the network drawing in Figure 26, and the change layer's intra-layer edges can be extracted from the propagation relationships (i.e., parent, children, and siblings) recorded for each change request, as in Table 21. Unfortunately, data were unobtainable on the social layer's intra-layers edges (i.e., communication links between engineers). Previous research has procured data on actual communication links by interviewing people at an organization [22, 28]. Such interviews were infeasible for this case study due to the program's large size and the unavailability of relevant people to interview since the end of system development.

Of the inter-layer edges, only the product-to-change and change-to-social inter-layer edges are known. These edges are ascertained from the area and engineers specified for each change request. Data on the product-to-social inter-layer edges (i.e., which engineers were in charge of which areas) were unavailable; however, the lead systems engineer interviewed for this case study may eventually be able to extract such data from other records. In summary, nearly all the pieces of the multilayer network are available for this case study, as outlined in Table 22.

Table 22. Data Availability for Case Study

	Product	**Change**	**Social**
Product	▨ Interfaces between areas	▨ Changes affect areas	Engineers in charge of areas
Change	▨ Areas affected by changes	▨ Propagation relationships (parents, children, and siblings)	▨ Engineers work on changes
Social	Areas assigned to engineers	▨ Changes worked on by engineers	Communication or organizational structure

▨ Available Edges □ Unavailable Edges

Despite some missing pieces, sufficient data are still available to exercise most of the baseline repository of tools and metrics presented in Chapter 4. According to the data requirements matrix in Table 20 (page 93), the only tools and metrics that cannot be exercised are the Social DSM, the Product-to-Social DMM, the Alignment Matrix, and all triple-layer analyses. Fortunately, that still leaves some of the newest and most promising tools and metrics within the case study's reach.

Before delving into the data further, visualizing the multilayer network might be an instructive preliminary step. Unfortunately, the enormous number of nodes and edges here makes it infeasible to draw the entire multilayer network. However, for illustrative purposes, it is still enlightening to draw the multilayer network for a small subset of change requests within the change layer. After all, the program's 41,551 change requests do not form one big connected layer; in fact, the dataset contains more than 29,000 stand-alone change networks. Table 23 shows the size distribution of these stand-alone networks.

Table 23. Distribution of Stand-alone Change Networks from Case Study

Number of Nodes in Stand-alone Change Network	Count
1	26,125 (88%)
2 to 86	3,458 (12%)
87	1
170	1
445	1
2,566	1

Table 23 reveals that the large majority (88%) of stand-alone networks consists of a single isolation change request. The dataset's four largest stand-alone change networks contain 2,566, 445, 170, and 87 change requests each. It's important to realize that these relatively giant networks do not stem from one initiating change followed by a chain of repeated and recursive parent-child propagation; for example, the network of 2,566 change requests does not consist of one change followed by 2,555 descendants. Rather, such large change networks (as well as many of the smaller ones) were formed by several propagation chains coalescing together through non-traditional sibling relationships in which two changes were related in some way but do not have a mutual parent. Thus, to create these networks, separate chains of descendants were declared to be related, despite the lack of traditional propagation relationships between them.

For example, Figure 28 draws the multilayer network associated with a stand-alone change network called "11-CR," which consists of 11 change requests evaluated and implemented by nine engineers and affecting only three system areas. The layers are drawn in a linear formation, and all the node labels correspond exactly with those in the raw dataset. The same 11-CR network was illustrated by Giffin et al. in [17], but not as part of a multilayer network.

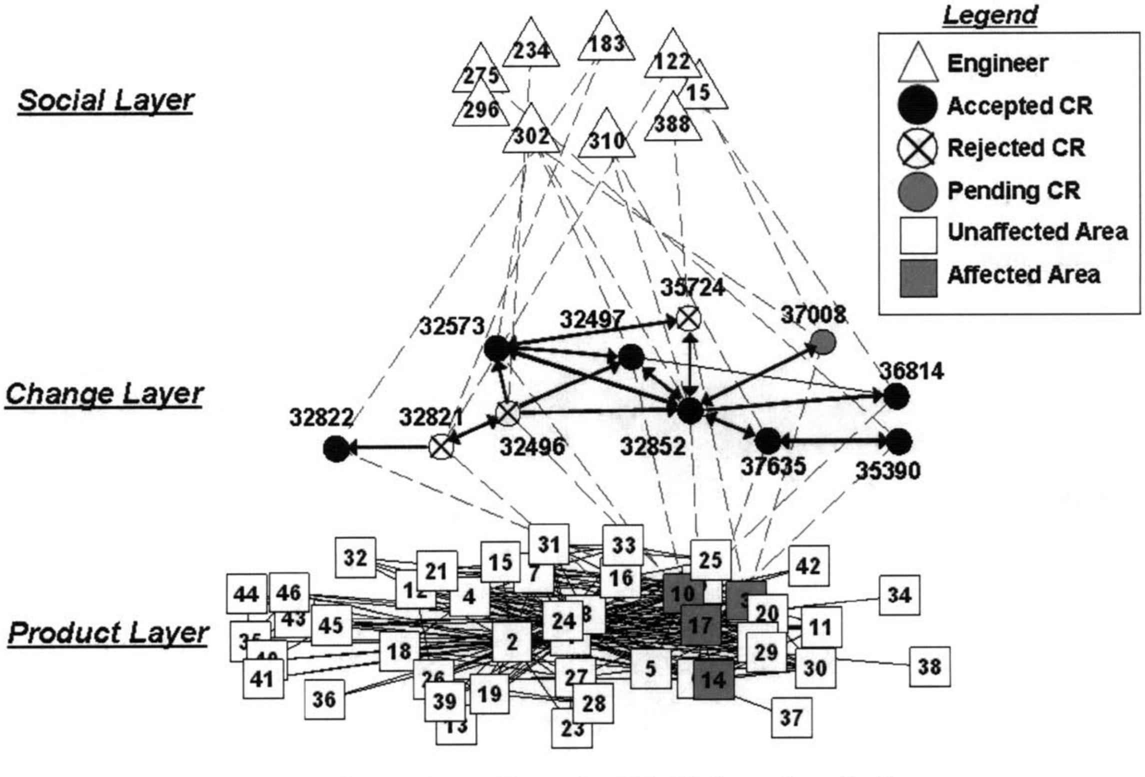


Figure 28. Multilayer Network of 11-CR from Case Study

The multilayer network drawing for 11-CR can be interpreted as follows. The product layer contains 46 areas connected by technical interfaces, but only Areas #3, #10, #14, and #17 were affected by any changes. The change layer contains 11 change requests connected through propagation relationships with arrows drawn to indicate their directions. For instance, change #32496 yielded three child changes (#32497, #32573, #32852). According to the legend, change #32496 was rejected, but all three of its children were accepted. Change #32496 is also a sibling of change #32821, despite them not having a mutual parent; in fact, neither change has a parent. Still, the dataset indicates that they are siblings because they were related in some other way. The social layer contains nine engineers, but no intra-layer edges are shown among them because data on their communication links were not available. Though it may be hard to determine visually, the inter-layer edges indicate, for example, that engineer #302 implemented change #32497 which affected Area #10.

For a larger example, Figure 29 draws the multilayer network for "87-CR," a stand-alone network of 87 change requests evaluated and implemented by 50 engineers and affecting 12 system areas. The node labels and edge arrows have been removed for simplicity.

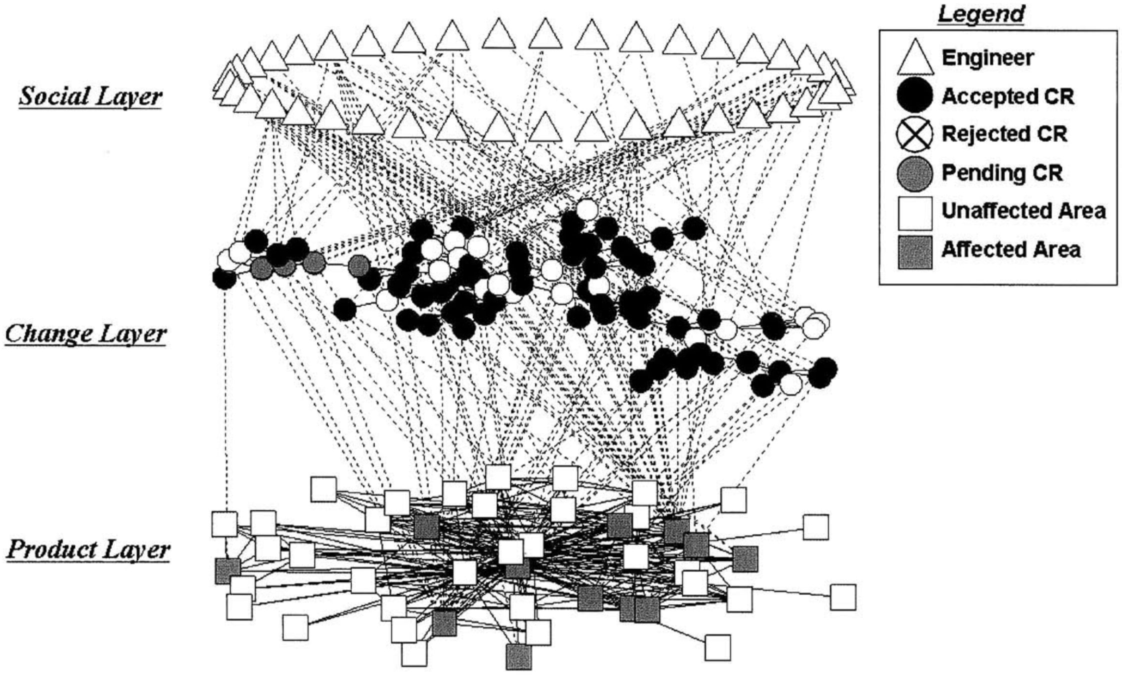


Figure 29. Multilayer Network of 87-CR from Case Study

Figure 28 and Figure 29 give an impression of the complexity of multilayer networks in practice. The raw data for 11-CR and 87-CR (i.e., the record for each change request) can be found in the Appendix. Ideally, better visualization techniques might draw these networks with more clarity. Nevertheless, this shortcoming does not prevent the case study from analyzing the data behind the pictures.

5.4. Analysis and Interpretation

The following sections will investigate two topics of significant interest to academia and industry. The first topic revolves around the social layer's effects on change propagation. The case study employs the Engineer-CPI and the Proposal Acceptance Rate to characterize the performance of engineers as implementers and proposers of change, respectively. The second topic focuses on the general characterization of change propagation in terms of the product layer and change layer. Specifically, the issues of indirect propagation, propagation extent, and component centrality are scrutinized closely with the Product DSM, Change DSM, Propagation DSM, and graph properties.

It's important to remember that this case study is more *exploratory* than *explanatory*. That is, the case study seeks to explore the multilayer network model's utility by elucidating interesting issues in the field of engineering change management. However, in doing so, the author does hope to at least suggest answers to some explanatory questions regarding the phenomenon of change propagation. The case study begins with an investigation of the social layer.

5.4.1. Engineer Performance

Over the eight year time period recorded in the dataset, 501 individual engineers and administrators were identified as implementers and proposers of changes. These employees make up the social layer of the multilayer network. The following investigation focuses on the social layer's effects on change propagation and engineering change management. Past research has lacked any quantitative analysis of the people within the engineering change management process. Indeed, this case study introduces a quantitative treatment of the social layer. Table 26 (page 126) summarizes the results.

The first round of analysis focuses on engineers as *implementers* of change. Specifically, the Engineer-CPI is used to quantify the propagation effects of an engineer's work. The results suggest that an organization can use the Engineer-CPI, in light of an engineer's organizational role and the context of his (or her) assignments, to evaluate employee performance. Secondly, this investigation considers the performance of engineers as *proposers* of changes. Specifically, the Proposal Acceptance Rate is used to quantify an engineer's innovativeness and systems awareness. Overall, the multilayer network model shows promise as a paradigm for analyzing the social layer's influence on change propagation and engineering change management.

5.4.1.1. Engineers as Implementers of Change

One element of an engineer's work is the implementation of changes. To assess an engineer's performance in this regard, this case study uses the Engineer-CPI. The data for this case demonstrate the significance of several factors that contribute to the Engineer-CPI. Specifically, the Engineer-CPI is shown to correspond roughly with an engineer's organizational role. However, deviations from this correspondence suggest that other factors are also at play. For instance, the Engineer-CPI also appears to depend on the Component-CPIs of his assigned areas. Furthermore, based on previous literature and interview input, it is also postulated that the Engineer-CPI is a reflection of an engineer's technical competence and an organization's human resource management.

5.4.1.1.1. Engineer-CPI

The Engineer-CPI is a double-layer metric introduced by this thesis to quantify the propagation effects of an engineer's implementation of changes. As calculated by Equation 2 (page 78), the Engineer-CPI compares the number of changes propagating in and out of an engineer's work to determine whether his behavior multiplies (CPI > 0), absorbs (CPI < 0), or carries (CPI = 0) changes.

Applying the metric to this case study, Figure 30 shows the distribution of Engineer-CPIs calculated for all of the program's 501 engineers. The histogram spans the entire spectrum ranging from perfect absorbers to perfect multipliers (i.e., -1 to +1). The bars do not sum to 501, because nearly half of the engineers (226) actually behaved like constants (i.e., CPI undefined) who were only involved with isolated changes, and hence, did not contribute to any change propagation.

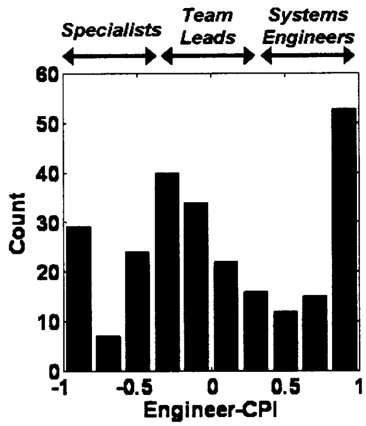


Figure 30. Distribution of Engineer-CPIs from Case Study

5.4.1.1.2. *Effects of Organizational Role*

It was argued in Chapter 4 that the Engineer-CPI should correspond to the organizational role of an engineer. Namely, as indicated in Figure 30, managers and systems engineers should behave like multipliers (CPI > 0), team leads should behave like carriers (CPI = 0), and specialists should behave like absorbers (CPI < 0). The rationale here is that systems engineers implement high level changes that usually propagate to lower level components; hence, they tend to be change multipliers. Meanwhile, specialists implement changes in the small components of a system that are unlikely to propagate any further; hence, these people tend to be change absorbers. The data from this case confirms this intuition.

To determine the effects of an engineer's organizational role on his Engineer-CPI, the engineers in this program were divided into two classes: coders and testers/integrators. Coders were the specialists who actually made changes to lines of code within the system's software areas. By contrast, testers and integrators were more like systems engineers who tested and integrated the system areas together. Data on the exact class of each engineer was regrettably unavailable to the author. However, it was still possible to roughly classify each engineer according to a heuristic recommended by the lead systems engineer interviewed in this study. The heuristic classified an engineer as a "coder" if 60% or more of his work focused on core technology in the system (as opposed to support structure, testing tools, etc.). Otherwise, the engineer was classified as a "tester/integrator."

Figure 31 shows the distribution of Engineer-CPIs for each class of engineer. The distributions offer some evidence that the Engineer-CPI indeed corresponds with an engineer's organizational role. As expected, the coders' distribution is heavy on the absorber end of the spectrum. In fact, 74% of coders had negative CPIs. By contrast, the testers/integrators' distribution is heavy on the multiplier end of the spectrum, with 53% having positive CPIs. The average coder's CPI was -0.16 (weak absorber), while the average tester/integrator's CPI was 0.2 (weak multiplier).

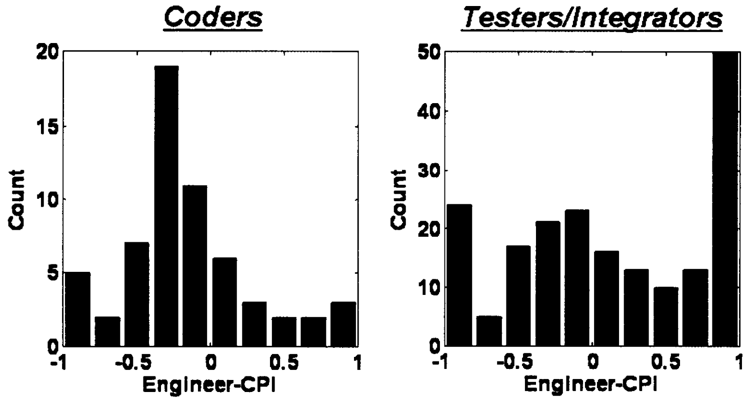


Figure 31. Role-based Distributions of Engineer-CPI from Case Study

Thus, this case study offers some verification of the correspondence between the Engineer-CPI and organizational roles. Namely, the coders (or specialists) tended to be absorbers, while the testers and integrators (or "systems" engineers) tended to be multipliers. Stronger proof of this theory would require data on the exact role of each engineer in the program.

Of course, the organizational roles did not completely dictate the Engineer-CPIs. In fact, Figure 31 reveals that 26% of coders were actually multipliers instead of absorbers, and 47% of testers/integrators were actually absorbers instead of multipliers. These discrepancies suggest that some other factors must be at play. Consequently, this case study also considered the context of an engineer's work, particularly the Component-CPIs of his assigned areas, and its effect on his Engineer-CPI.

5.4.1.1.3. Effects of Component-CPI

The context of an engineer's work may also influence his Engineer-CPI. For instance, an obvious consideration would be the propagation behavior of the areas to which an engineer was assigned to implement changes. The rationale here is that some engineers may be assigned to parts of the product that are inherently multipliers or inherently absorbers, as measured by their Component-CPIs. As a result, these engineers may have little control over the propagation effects of their work. The engineers who work on multipliers (Component-CPI > 0) will likely appear as multipliers themselves (Engineer-CPI > 0). By contrast, the engineers who work on absorbers (Component-CPI < 0) will likely appear as absorbers themselves (Engineer-CPI < 0). Overall, the inherent propagation behavior of a system area may dictate an engineer's CPI more than his organizational role or other factors.

To determine the effect of Component-CPIs on the Engineer-CPI, the engineers in this program were divided in two groups: those with absorber assignments and those with multiplier assignments. An engineer was said to have "absorber assignments" if the average Component-CPI of his assigned areas was negative (i.e., an absorber). Conversely, an engineer was said to have "multiplier assignments" if the average Component-CPI of his assigned areas was positive (i.e., a multiplier).

Figure 32 shows the distribution of Engineer-CPIs for each group of engineers. The distributions offer some evidence that the Engineer-CPI indeed depends on the Component-CPI of an engineer's assigned areas. In fact, 67% of engineers with absorber assignments had negative CPIs (i.e., were absorbers), and 75% of engineers with multiplier assignments had positive CPIs (i.e., were multipliers). The average CPI for each group was -0.12 (very weak absorber) and 0.44 (moderate multiplier), respectively.

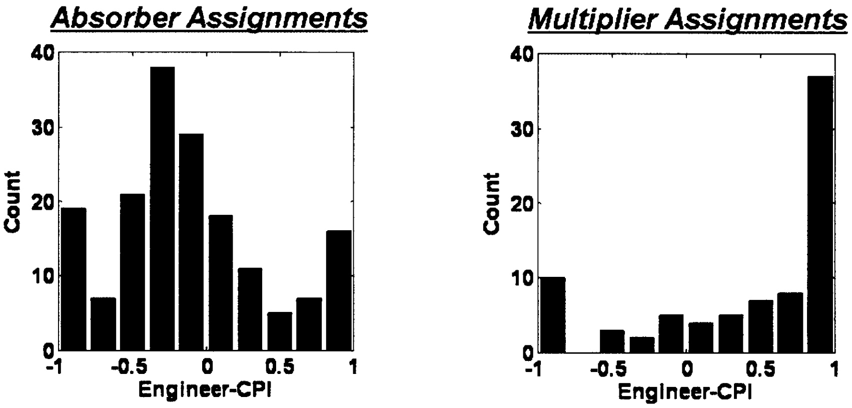


Figure 32. Assignment-based Distributions of Engineer-CPIs from Case Study

Thus, an engineer's CPI appears to be somewhat dictated by the propagation behavior, or Component-CPIs, of his assigned areas. That is, those engineers who work on multipliers and absorbers tend to be multipliers and absorbers themselves, respectively.

Nevertheless, there were some discrepancies. In fact, 25% of engineers with multiplier assignments were still absorbers, and 33% of engineers with absorber assignments were still multipliers. Delving a little deeper, Table 24 shows the Engineer-CPI of two representative engineers, along with the average Component-CPIs of their assigned areas.

Table 24. Propagation Behavior of Representative Engineers from Case Study

Engineer #	Average Component-CPI of Assigned Areas	E_{out} (#changes out)	E_{in} (#changes in)	Engineer-CPI
304	-0.23	8	1	0.78
63	0.31	0	2	-1.0

Each engineer in Table 24 exhibits a different propagation behavior relative to the propagation behavior of his assigned areas. For example, engineer #304 behaved like a strong multiplier (CPI = 0.78), yet his average assigned area was a weak absorber (CPI = -0.23). Meanwhile, engineer #63 behaved like a perfect absorber (CPI = -1), but his average assigned area was a weak multiplier (CPI = 0.31). As such, some engineers performed independently of the context of their work, while the majority of others, according to the distributions in Figure 32, performed as their average assigned areas dictate.

It should be noted that in the case of a perfect one-to-one mapping between engineers and areas (i.e., engineer *n* only implements changes in area *n*), an engineer's CPI will automatically equal the Component-CPI of his assigned area (i.e., Engineer-CPI_n = Component-CPI_n). Therefore, any comparison between the two would be redundant. In that case, it may be more appropriate to compare the Engineer-CPI with some other expectation of performance. Nevertheless, in this case study's program, most engineers worked on various areas throughout the system. Consequently, many different engineers contributed to each area's Component-CPI, and the Engineer-CPI of an individual engineer can differ from the behavior of his assigned areas.

Thus, neither the organization role nor the context of an engineer's work tell the entire story of his ultimate Engineer-CPI. In kind, this case study postulates that an engineer's technical competence may also be an influential factor.

5.4.1.1.4. Effects of Technical Competence

When the Engineer-CPI does not match an engineer's organization role or the context of his assignments, the engineer's technical competence may be making the difference. Qualitative observations throughout the literature have emphasized the impact of technical competence in the engineering change management process. Huang and Mak [18] observed that engineering firms consider individual skills to be extremely important factors in the processing of change requests. Other literature suggests that organizations who design complex products often suffer from poor system awareness among their engineers. Clarkson et al. [6] found that designers at Westland Helicopters often did not understand how their decisions affected the rest of the helicopter under development. They further discovered that even the chief engineers interviewed did not have a detailed comprehension of the entire helicopter design. Moreover, Jarratt et al. [19] concludes that mistakes due to inexperience, a lack of systems knowledge, communication breakdown, and even forgetfulness, are primary causes of change propagation. In fact, managers at a UK engine manufacturer [19] maintained that an understanding of possible change propagation "comes down to the experience of individuals." Therefore, it is reasonable to suppose that an engineer's technical competence will impact the propagation effects of his work.

An engineer's technical competence may explain some of the discrepancies discovered above with respect to organizational role and Component-CPIs. Unfavorable discrepancies (e.g., specialists who behave like multipliers) may be evidence of an engineer's incompetence or inexperience in his current position. These engineers are propagating more changes than are warranted by their job descriptions. By contrast, favorable discrepancies (e.g., systems engineers who behave like absorbers) may signify an employee's extraordinary skill as a change absorber. These highly competent engineers are able to find design solutions that cause less propagation than normally expected. Overall, the Engineer-CPI might help distinguish which employees need more training (e.g., an overview of the entire product) to guide their future design and redesign decisions. Likewise, the Engineer-CPI might help identify the best performers who should be assigned the most important tasks in the future. Qualitatively, the lead systems engineer interviewed in this case study agreed with the premise that technical competence might affect the probability of propagation. It was also explained that certain engineers in the program were considered go-to people, by virtue of their assessed skill and expertise, when a significant change request needed to be evaluated or implemented. Verification of the effects of technical competence on propagation behavior would require detailed data on the justification for individual changes, and whether viable alternative solutions existed that would have resulted in more or less propagation.

5.4.1.1.5. Effects of Human Resource Management

In the same way, the Engineer-CPI might also reflect the effectiveness of an organization's human resource management. For instance, if an employee is performing worse than anticipated, perhaps the organization has sloppily or unknowingly assigned him to tasks outside of his organizational role or expertise. Moreover, perhaps the employee is being overworked; Terwiesch and Loch [32] observed that oftentimes an overstretched engineer could have a backlog of one month's worth of work. Conversely, if an employee is performing better than expected, perhaps the organization exercised great managerial judgment; the organization has found the perfect niche for that worker. As such, the Engineer-CPI may translate to a management imperative. Implications for management policy will be discussed further in Chapter 6.

In summary, the Engineer-CPI is a complicated measure of an engineer's performance as an implementer of changes. The data for this case study indicates that the Engineer-CPI is partially dependent on an engineer's organizational role and the context of his assignments. Coders and engineers who worked on absorbers in the system tended to behave like absorbers themselves. Meanwhile, tester, integrators, and engineers who worked on multipliers in the system tended to behave like multipliers themselves. Nevertheless, these two factors did not completely determine the Engineer-CPI. The literature suggests that an engineer's technical competence, in addition to the organization's human resource management, also affects the propagation effects of an engineer's work. More rigorous statistical analysis and additional data would be required to tease out the contributions of all these factors in more detail.

5.4.1.2. Engineers as Proposers of Change

The other element of an engineer's work is the proposal of change requests. To measure an engineer's performance in this regard, this investigation employs the Proposal Acceptance Rate (PAR). As demonstrated by the data in this case study, the PAR can be combined with an engineer's proposal count to assess his performance on a two-dimensional scale. For more clarity, an organization can also compare the PAR with the CAIs of the areas targeted by the proposed changes.

5.4.1.2.1. Proposal Acceptance Rate

When an engineer proposes a change request, the request will ultimately be accepted or rejected. That is, an evaluation process determines if the change's benefits outweigh its costs from a systems perspective. The PAR measures an engineer's rate of acceptance as a proposer of change. Applying the metric to this program, Figure 33 shows a histogram of all the engineers' PARs. The average PAR is 71%.

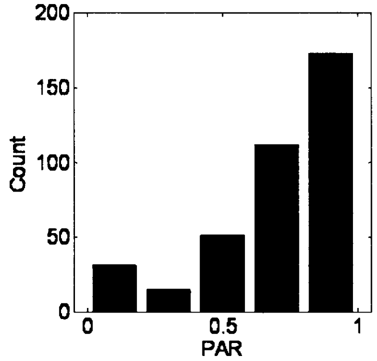


Figure 33. Distribution of PARs from Case Study

The PAR alone does not tell the entire story. An organization cannot necessarily distinguish its performers based on an engineer's PAR alone. Another factor to consider is the number of change requests proposed by each engineer. The evaluation of a change request requires time and effort. Consequently, even rejected change requests take their toll on an organization. In other words, an organization does not want to waste time evaluating a needlessly large amount of change requests. As such, this thesis proposes a two-dimensional scale for judging the performance of engineers as proposers of change. The scale's two dimensions are an engineer's PAR and the number of change requests he proposed. Figure 34 plots the position of the 382 engineers who proposed any changes on this two-dimensional scale.

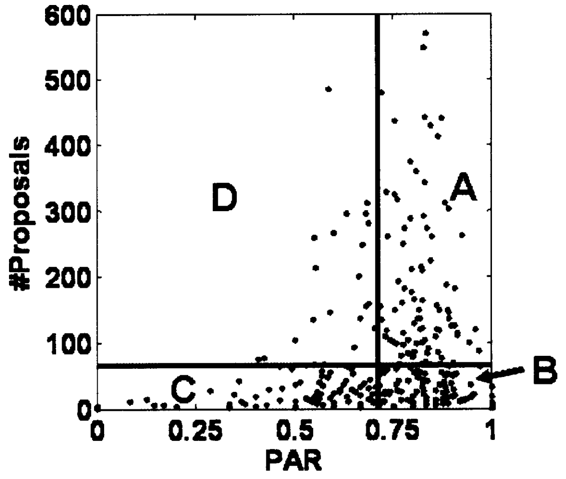


Figure 34. #Proposals vs. PAR from Case Study

Following the advice of the lead systems engineer interviewed for this case study, Figure 34 is additionally broken into four quadrants, A, B, C, and D. The quadrant boundaries are located at the average PAR and average proposal count of all 382 data points. Table 25 shows the number of engineers in each quadrant of the two-dimensional scale in Figure 34.

Table 25. PAR/#Proposals Quadrant Distributions from Case Study

Quadrant	Count
A	85 (22%)
B	151 (40%)
C	123 (32%)
D	23 (6%)
All	382 (100%)

Each quadrant has different implications for an engineer's performance, depending on his PAR and proposal count relative to the average engineer:

- *Quadrant A* contains engineers with high PARs and high numbers of proposals. These engineers might be termed "high performers."

- *Quadrant B* contains engineers with high PARs but low numbers of proposals. These engineers likely have great ideas and good systems awareness, since their change requests are usually accepted. However, for some reason, they propose a relatively low number of change requests. The reason for the low proposal count may lie in the engineer's organizational role, personality, or some other factor.

- *Quadrant C* contains engineers with low PARs and low numbers of proposals. These engineers are relatively passive with little success as proposers of change.

- *Quadrant D* contains engineers with low PARs but high numbers of proposals. There are two possible explanations for this troubling behavior. One is that the engineer tends to have lots of bad ideas. The alternative explanation is that the engineer is actually quite innovative, but the organization or product is stubborn or sluggish to change.

5.4.1.2.2. Effects of CAI

Other factors may also contribute to an engineer's performance as a proposer of change. For instance, an organization could compare an engineer's PAR with the Change Acceptance Rate (CAI) of the areas targeted by his proposals. The rational here is that some change proposals are inherently more or less likely to be accepted by virtue of the targeted area's CAI. As such, this thesis proposes an additional metric for assessing an employee's performance. The metric, R_{PAR}, is the ratio of an engineer's PAR to the average CAI of the areas targeted by his change proposals. The ratio is calculated as follows, where N is the number of proposed change requests, and CAI_n is the CAI of the area targeted by the *nth* proposal:

Equation 6

$$R_{PAR} = \frac{PAR}{\frac{1}{N}\sum_{1}^{N}\mathrm{CAI}_n}$$

Using Equation 6, Figure 35 displays a histogram of R_{PAR} values for all the engineers in the program. The majority (78%) of engineers have an $R_{PAR} \approx 1$, which would indicate that most engineers' PARs match closely with the CAIs of their targeted areas. A closer look at the data reveals that this result is an artifact of most engineers always proposing change requests in the same area. Consequently, the PARs and associated CAIs are essentially equal ($R_{PAR} = 1$). Still, 15% of engineers had $R_{PAR} > 1$. These engineers were able to achieve PARs higher than the average CAI of their targeted areas. These engineers may be particularly innovative since their ideas were accepted by relatively reflective areas in the system. By contrast, the 10% of engineers with $R_{PAR} < 1$ struggled to get changes accepted by relatively receptive areas. These engineers may not be quite as innovative or systems savvy.

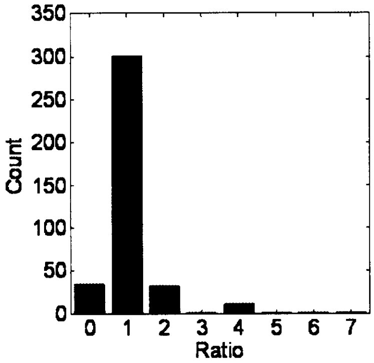


Figure 35. Distribution of R_{PAR} from Case Study

In summary, this investigation considered engineers as proposers of change. The data in the case suggests that a two-dimensional scale, based on an engineer's PAR and proposal count, can be used to evaluate an engineer's performance. Furthermore, the PAR can be normalized by the CAIs of the engineer's targeted areas, via the R_{PAR} metric, to better understand the significance of his performance relative to his targeted components.

5.4.2. Propagation Characteristics

The second topic for this case study is the general characterization of change propagation. The focus now shifts from the social layer to the product layer and change layer. The following investigation progresses through a line of inquiry regarding the nature and cause of change propagation. Table 27 (page 126) summarizes the results.

This investigation employs multilayer network tools and metrics to characterize change propagation in a holistic fashion. The first issue considered is the phenomenon of indirect propagation, whereby parent-child propagation occurs between nonadjacent system areas. Another issue explored is propagation extent, in terms of the number of generations flowing from initiating changes. Finally, propagation behavior is considered with respect to component centrality (a graph property). Overall, multilayer network tools and metrics are used to quantitatively confirm and offer counterexamples to many qualitative conclusions about change propagation in previous literature.

5.4.2.1. Indirect Propagation

Conventional wisdom about change propagation assumes that only direct propagation is possible; that is, a parent change in one component can only yield child changes in itself or adjacent components. This assumption forms the basis of Clarkson et al.'s CPM [6]. However, the program here experienced considerable indirect propagation, whereby child changes occurred in nonadjacent areas. This investigation uses the Product/Propagation DSM overlay and the Propagation Directness metric, in addition to some interview input, to gain insight into this non-intuitive phenomenon.

Figure 36 shows the program's Product DSM and Propagation DSM. Given the large range of values in the traditional Propagation DSM (bottom left), a binary version (bottom right) is included to give a better visual sense of where propagation actually occurred. An instance of parent-child propagation only appears in the Propagation DSM if the child change was ultimately accepted, regardless of the approval status of the parent change. Meanwhile, Figure 37 overlays the Product DSM with the Propagation DSM. An equivalent overlay was performed for this program by Giffin et al. in [16, 17], but without deeper investigation.

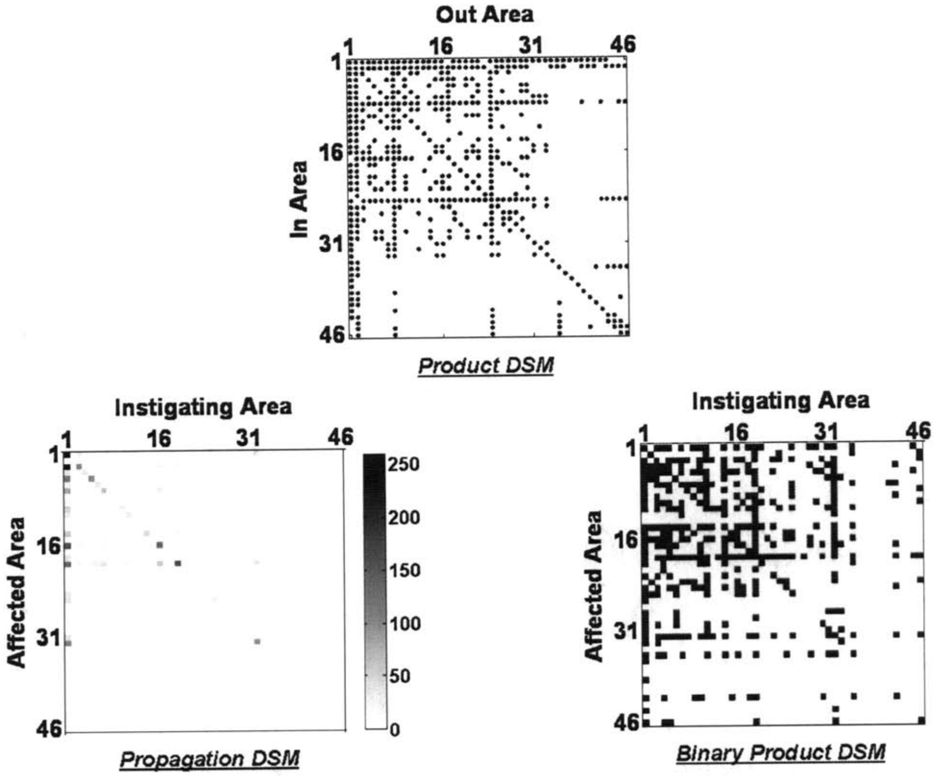


Figure 36. Product DSM and Propagation DSM from Case Study

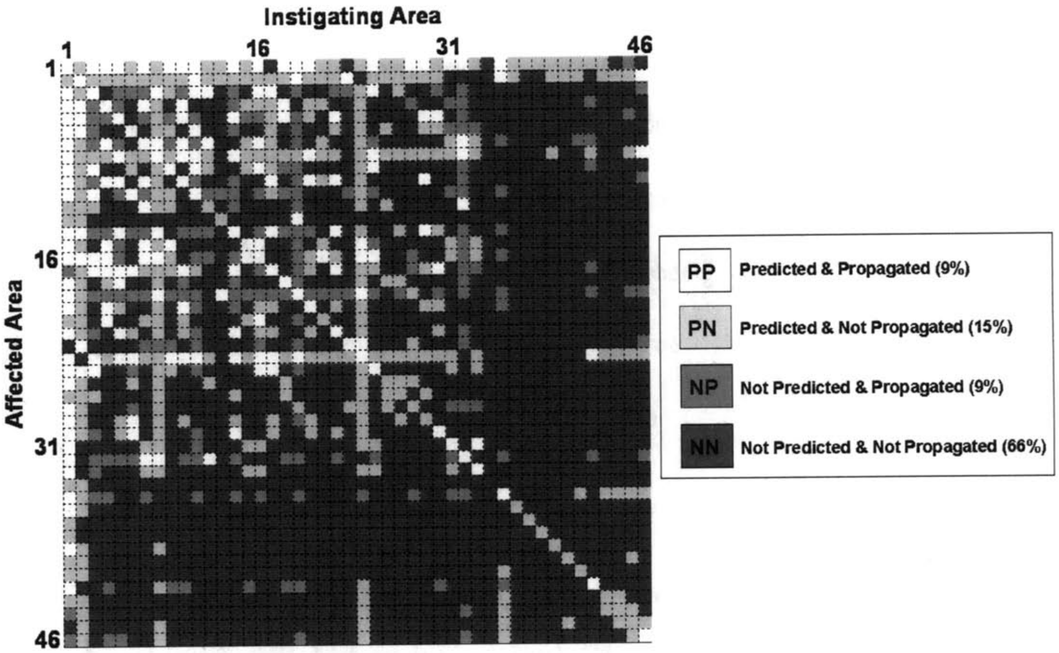


Figure 37. Product/Propagation DSM Overlay from Case Study

The overlay in Figure 37 exposes all four types of parent-child propagation behavior. Overall, 15%, 9%, 9%, and 16% of all pairs of components exhibited PP, PN, NP, and NN behavior, respectively. These behavior types were discussed theoretically in Chapter 4 (page 69), and now this program has demonstrated them in practice.

Where propagation did occur (PP and NP), it is meaningful to calculate the effective Propagation Directness (page 75). Propagation Directness (PD), a double-layer metric, refers to the number of product interfaces spanned by an instance of parent-child propagation. Figure 38 displays the distribution of Propagation Directness values, considering every instance of parent-child propagation in the program in which the child change was accepted (regardless of the parent change's approval status). The distribution reveals that 78% of all parent-child propagation in the program was direct (PD $\leq$ 1), while a surprising 22% was indirect (PD > 1). The vast majority of indirect propagation occurred across two interfaces (PD = 2) and a handful (3) occurred across three interfaces (PD = 3). It should be noted that the maximum possible Propagation Directness for this system was three because the product layer's network has a diameter of three; that is, the maximum geodesic path between any two areas is three.

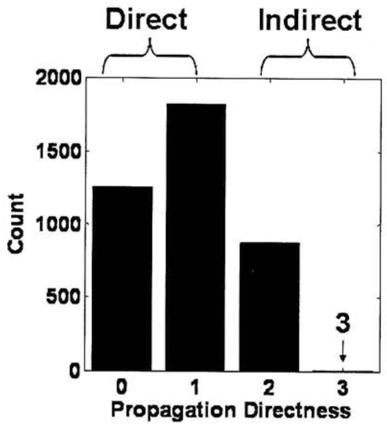


Figure 38. Distribution of Propagation Directness from Case Study

To further demonstrate the range of Propagation Directness experienced by the program, Figure 39 illustrates a few examples of parent-child propagation from the dataset. In each illustration, the change layer contains the parent change and child change connected by a directed intra-layer edge. Meanwhile, inter-layer edges connect these changes to the affected areas in the product layer. For PD > 1, the product layer also contains any unaffected areas on the shortest path between the two affected areas. All nodes are labeled as they appear in the raw data. For simplicity, the social layer is not included in these drawings, although the social layer's impact on Propagation Directness is something to consider in the future.

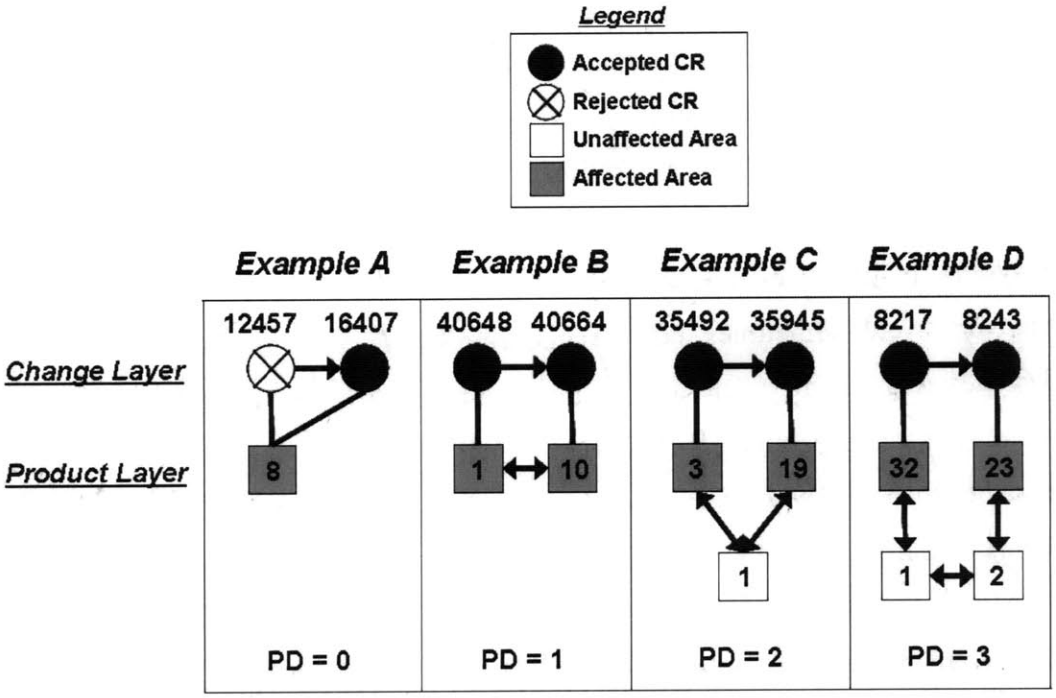


Figure 39. Examples of Direct and Indirect Propagation from Case Study

Each example in Figure 39 has a different Propagation Directness value, which should be clear from the number of product interfaces spanned by the propagation. In Example A, self-propagation (PD = 0) occurred in Area #8; interestingly, the parent change in this example was ultimately rejected. Next, Example B shows direct propagation between adjacent areas (PD = 1); a change to Area #1, which contains requirements documentation, caused a change in Area #10, a core technology area. Example C exhibits indirect propagation; Areas #3 and Areas #19 are separated by two interfaces (PD = 2) with Area #1 in between them. It should be noted that several geodesic (length-2) paths exist between Areas #3 and #19, besides the one through Area #1. Finally, Example D shows one of only three scenarios in the entire dataset with PD = 3. It's important to remember that in Examples C and D, the intermediate areas (connecting the two affected areas) were unaffected by any change, which is the non-intuitive feature of indirect propagation.

The phenomenon of indirect propagation contradicts conventional wisdom on change propagation. As such, one might conclude that if indirect propagation appears to have occurred, then the Product DSM must be missing some interfaces that actually exist; in other words, any observed indirect propagation is really direct propagation in disguise. If this explanation is true, then the Product DSM in this case study would shockingly be missing 192 interfaces. However, a lead systems engineer from the program explained that indirect propagation is a legitimate artifact of software system development. Apparently, engineers in this program would frequently violate the intended structure of the system in order to achieve a quick solution for a redesign. These ill-advised maneuvers were sometimes necessary during time crunches to meet development milestones (e.g., PDR, CDR, etc.). For example, one area of the system contained System Adjustable Parameters (SAPs). A SAP is a system variable kept in a loadable file, rather than in the software code itself. Many areas of the system were nominally disconnected from the SAP file. Still, on occasion, a hasty redesign effort would change the SAP file (e.g., adding an SAP), despite the lack of an interface between the SAP file and the parent area. Thus, indirect propagation, though sloppy, can and does occur during product development. Additional case studies are necessary to determine if indirect propagation is a common artifact among software systems only, or hardware systems as well.

5.4.2.2. Propagation Extent

Propagation extent is another interesting issue in the literature on change propagation. Eckert et al.'s [14] study of Westland Helicopters found that a change rarely occurs by itself, but usually propagates no more than four generations. These rules of thumb were based on interviews of several chief engineers and designers at the company. Fortunately, the case study in this thesis has the data available to quantitatively corroborate or challenge these rules of thumb about the extent of change propagation.

Analyzing propagation extent requires analysis of the change layer. The change layer for this program contains over 41,000 changes, too many to reasonably display the Change DSM (i.e., the DSM of the change layer) here. Still, the Change DSM is useful for tracing the descendants of each change, because each parent-child relationship is represented by an edge. Figure 40 shows the distribution of the number of generations flowing from each *un-parented change*. An un-parented change is an individual change that is not the child of another change, and may or may not have any child changes of its own. In other words, each count in Figure 40 corresponds with a distinct propagation chain, whether it contains one isolated change or a line of descendants. In all, the program generated 36,184 un-parented changes. The reader should note that Figure 40 uses a log scale on the vertical axis.

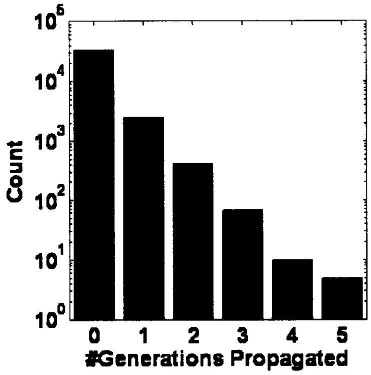


Figure 40. #Generations Histogram from Case Study

The histogram only partially conforms to Eckert et al.'s rules of thumb. On the one hand, change propagation in the system almost always (99.99%) halted after four generations, just as Eckert et al. found with Westland Helicopters. There was only a handful (5) of changes that yielded five generations of changes, which was the maximum number of generations experienced; in other words, change propagation always vanished after five generations. Examples of propagation chains from the dataset with four and five generations of descendants are illustrated in Figure 41. All the node labels correspond exactly with those in the raw dataset.

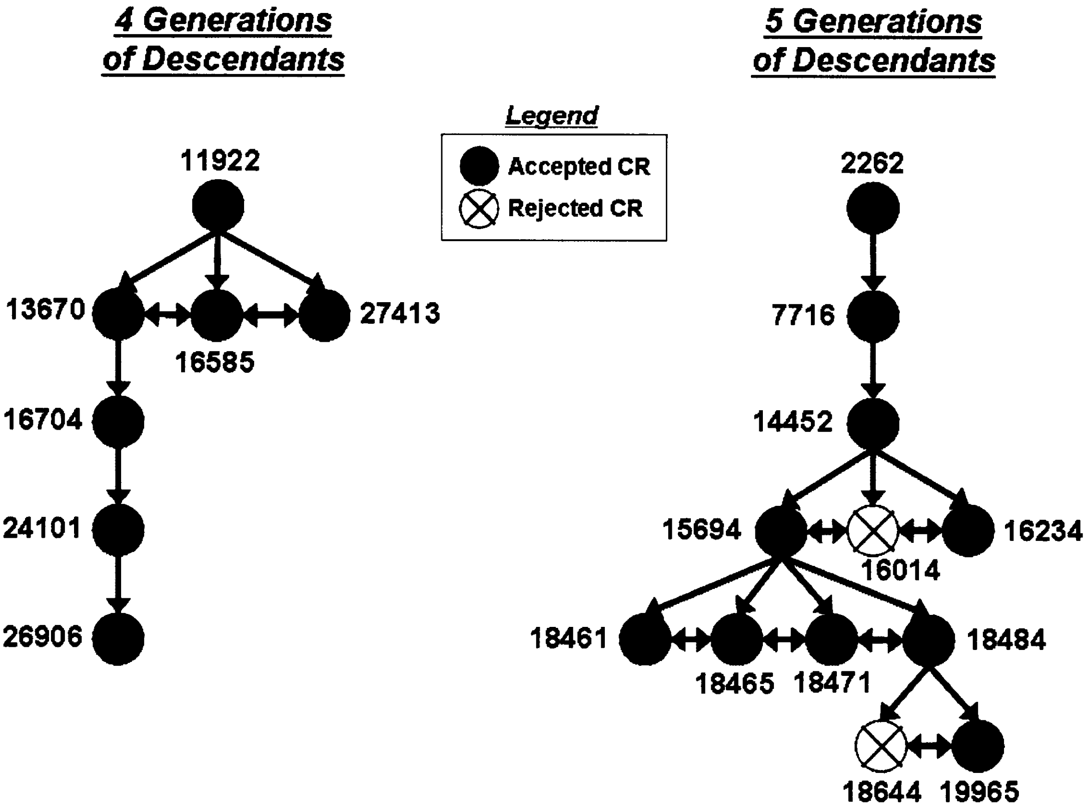


Figure 41. Examples of 4- and 5-Generation Propagation Chains from Case Study

On the other hand, Figure 40's results differ from Eckert's finding that a change rarely occurs alone. In fact, isolated changes were actually the norm for this system; 91% of un-parented changes (33,152 out of 36,184) did not have any children (i.e., zero generations propagated). A deeper look into the context of each change would elucidate these statistics more. For instance, the large majority (80%) of changes in this program were low magnitude (0 or 1 on a scale of 0 to 5), which may explain the generally low probability of propagation.

Overall, propagation extent likely stands as an extremely context-dependent feature of change propagation. This case study, at least, confirms that propagation vanishes after five generations of descendants, and rarely exceeds four generations.

5.4.2.3. Effect of Area Centrality

The consensus among researchers is that change propagation occurs primarily by virtue of the intricate dependencies among the components of modern products and systems [6, 11, 13, 31, 32]. One way to reinforce this intuition is by testing the effects of a component's centrality on propagation behavior.

Centrality is a graph property introduced earlier as a single-layer metric. A node's centrality is a gauge of its importance in a network, and can be measured by a node's degree or betweenness (among other metrics). A natural question to ask is whether a component's centrality in the product layer affects its propagation behavior. Intuition says that it likely does, since connectivity is the root cause of propagation.

To check this intuition in the case study, the degree (*C*) of each system area in the product layer was compared to its total propagation activity (*TPA*). An area's *TPA* is defined here as the number of times the area was affected by a parent change or child change. Figure 42 plots all 46 systems areas as (*C*, *TPA*) pairs.

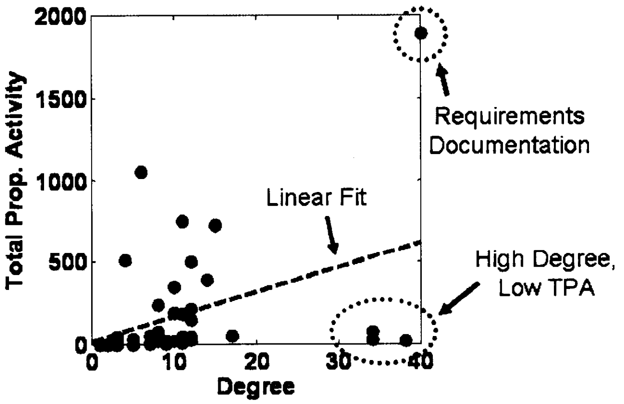


Figure 42. Total Propagation Activity vs. Degree for Case Study

In Figure 42, the data points appear to trend upward and to the right. A line has been fitted to the data (in a least-squares sense) that indicates a direct relationship between the two variables. In fact, the Pearson correlation coefficient between *C* and *TPA* is $\sigma_{C,TCA}$ = 0.4. Such a coefficient suggests a small to medium positive correlation. However, the data also suggest that system areas with high degrees do not always trigger a large amount of total propagation activity; a few outliers (as circled in Figure 42) have high degrees but relatively low TPA. There was also one significant outlier above the fitted line (also circled); this data point corresponds with Area #1 (requirements documentation) which levied constraints on most of the other systems and inherently participated in a relatively large amount of change propagation. In general, the single-layer metric of centrality has been used to quantitatively reinforce a conventional theory in the literature; namely, component connectivity can lead to change propagation.

5.5. Summary of Case Study

In summary, this chapter performed a case study of a real-world engineering project. The case under investigation was that of a technical program that developed a large scale sensor system dominated by software segments. A rich dataset extracted from the program's configuration management records facilitated extensive quantitative analysis using Chapter 3's multilayer network model and Chapter 4's baseline repository of tools and metrics.

The case study elucidated two major topics of interest to the research community and industry. One topic dealt with the socials layer's effects on propagation behavior and implications for engineering change management. The quantitative analysis here was a pioneering effort and illustrated the potential of the Engineer-CPI and Proposal Acceptance Rate as metrics for employee performance in the implementation and proposal of changes, respectively.

The second topic revolved around the general characterization of change propagation. Issues here included indirect propagation, propagation extent, and the effect of component centrality on propagation behavior. The system in this study experienced a significant amount of indirect propagation, but this behavior is potentially linked to complex software development. Propagation in general was infrequent and yielded a maximum of five generations of descendants (but almost always four or less). Finally, the centrality of a system area had a positive correlation with its total propagation activity. However, high centrality does not guarantee the occurrence of change propagation. The major findings of all these analyses are summarized in Table 26 and Table 27.

This concludes the case study for this thesis. Overall, the multilayer network model was demonstrated to be a useful paradigm for investigating change propagation and engineering change management. The next chapter broadens this discussion and considers the implications of the multilayer network model and general change propagation research for management policy.

Table 26. Summary of Investigation of Employee Performance

Topic	Multilayer Tools and Metrics Utilized	Major Finding
Engineers as Implementers of Change	• Engineer Propagation DSM • Engineer-CPI • Component-CPI	The Engineer-CPI appears to correspond with an engineer's organizational role and the Component-CPI of his (or her) assigned areas. Discrepancies between the Engineer-CPI and these factors may be due to an engineer's technical competence or an organization's human resource management.
Engineers as Proposers of Change	• Proposal Acceptance Rate • CAI	An engineer's performance can be effectively classified on a two-dimensional, according to his PAR and number of change he or she proposes. Normalizing the PAR by the CAIs of his targeted areas can help put an engineer's performance in context.

Table 27. Summary of Investigation of Propagation Characteristics

Topic	Multilayer Tools and Metrics Utilized	Major Finding
Indirect Propagation	• Product/Propagation DSM Overlay • Propagation Directness	Indirect propagation is possible, at least in software-dominated systems. Of all the parent-child propagation that occurred in the system, 78% was direct and 22% was indirect.
Propagation Extent	• Change DSM • Paths (graph property)	The vast majority of changes did not lead to any propagation in the system. All propagation vanished after five generations, and almost always after four.
Effects of Node Centrality	• Propagation-DSM • Centrality (graph property)	The degree (or centrality) of system area is positively correlated with its total propagation activity (as a parent or child), but a high degree alone is not sufficient to trigger significant propagation activity.

6. Management Policy Implications

This chapter discusses the policy implications of the multilayer network model and general change propagation research for engineering change management (ECM). Change propagation has been identified as a pervasive source of cost and difficulty in the ECM process. As such, the ideal ECM policy would have all design and redesign decisions carefully take into account potential propagation effects. However, meeting this policy objective is extremely challenging. According to several studies in the literature, engineering firms appear to be generally inconsistent at anticipating change propagation. These studies further suggest that organizations struggle to understand change propagation because no hard and fast methods exist for assessing the full impact of every individual change. As discussed throughout this thesis, the research community is responding to this shortcoming with the development of much needed theories, tools, and metrics. Overall, the literature suggests a three-pronged policy for dealing with change propagation: prevention, prediction, and control. The multilayer network model further emphasizes that the execution of this policy should consider the effects of all three layers (product, change, and social) on change propagation behavior. As such, the multilayer network model provides a holistic framework for practical use in the engineering industry and academia.

6.1. Industry's Struggle with Change Propagation

Change propagation is undoubtedly a real-world problem that warrants the careful attention of an engineering firm. In fact, the best ECM policies in the engineering industry recognize the need to account for change propagation in all significant design and redesign decisions. The Aberdeen Group, a company that conducts data-driven business research, found that engineering firms who exercise formal change impact analysis as part of their ECM process tend to perform better (in terms of achieving schedule, budget, and product quality goals) than companies who fail to do so [3]. Relative to lower performers, these "Best-in-Class" firms were more likely to consider the impact of individual changes on items such as other product components, manufacturing tooling, related documentation, and product requirements. Formal change impact analysis allows an engineering firm to keep tabs on their products' satisfaction of schedule, budget, and quality constraints, in light of propagation effects. As such, an appreciation of change propagation can make a significant difference in the success of an engineering firm.

However, the engineering industry as a whole still finds the threat of change propagation to be intractable. A 2006 study by the Aberdeen Group [2] found that less than two-thirds of surveyed engineering firms could "identify which items [in a product] are actually affected by changing another item." The same study reported that only 11% of engineering firms could generate a "precise list of impacted items...taking into account item interfaces." Clarkson et al. [6] highlighted similar weaknesses in their investigation of Westland Helicopters' ECM performance. The company's chief engineers estimated that as much as 50% of all changes were unexpected by the people involved in helicopter development. In other words, despite the company's efforts to anticipate change propagation, a lot of propagation still came as a surprise. In kind, Jarratt et al. [19] explored why some propagation is foreseeable, while other propagation is not. Interviewees at a UK engine manufacturer explained that unexpected propagation is usually the result of "stupid mistakes." That is, when small details are foolishly overlooked, the potential for propagation goes unnoticed. Interestingly, by contrast, high magnitude changes yield fewer surprises because these changes naturally receive more thorough attention. Overall, it seems that the engineering industry does not have a good handle on change propagation. In the end, unanticipated change propagation may lead to schedule slips, budget overruns, and decreased product quality. It should come as no surprise that change activities tend to drive the schedule, budget, and quality of product development [29].

The literature suggests that the engineering industry struggles with change propagation because no hard and fast methods exist for assessing the full impact of every individual change. Indeed, the extent of change propagation may fluctuate significantly with the context of the product or system being developed. As a result, companies cannot apply the same experienced-based rules of thumb to all types of products. For example, Chapter 5's case study demonstrated that a software-dominated system can exhibit much less propagation than the electro-mechanical systems studied in the literature (e.g., helicopters in [6]). Furthermore, an organization cannot always use the same rules of thumb for different areas within the same product. Jarratt et al. [19] found that engineers were able to develop standard (and successful) procedures for handling changes when the affected components were simple and well understood. However, no analogous standards existed for other aspects of the product. For instance, interviewees admitted they could not always predict the propagation effects associated with a combustion engine's vibration behavior. Indeed, the emergent properties of a complex system, like vibration dynamics, are likely the hardest propagation effects to predict in advance. In all, the engineering industry has yet to master rigid and infallible methods for predicting the full impact of any given change.

In the Information Age, it may seem surprising that industry cannot quickly and proficiently conduct change impact analysis. Apparently, however, even the powerful information technology (IT) of today cannot overcome the complexities of change propagation. IT that is tailored to ECM is widely available in the industry. For instance, Product Data Management (PDM) systems contain software for tracking and managing data on a particular product, including design details in the form of Computer-aided Design (CAD) drawings, cost data, and other specifications important for sourcing or manufacturing parts. PDM software enables engineering throughout an organization, no matter how fragmented, to share information with one another during product development. Theoretically, this type of communication infrastructure should assist an engineering firm in performing change impact analysis.

Nevertheless, in practice, IT does not solve everything. Change propagation remains a significant problem in industry, as indicated by the Aberdeen Group's 2006 study mentioned earlier [2]. The reason why IT cannot completely solve the challenges of change propagation may lie in the complexity of modern products and systems. In many cases, the dependencies among product components and subsystems are still too intricate to understand by engineers and teams working on different parts of a product. For example, Eckert et al. [14] explains that mere access to another team's documentation (e.g., via a PDM system) is not a perfect ECM solution; different teams use different design representations and conventions which can be too esoteric for outsiders to interpret correctly. As a result, one team cannot easily determine the consequences of its actions for another part of the product without significant expertise, experience, and verbal communication. Furthermore, IT cannot compensate for all human error or the mysteries of emergent system properties, both of which Jarratt et al. [19] have highlighted as major causes of change propagation. Ultimately, the engineering industry is in need of better ECM solutions for handling change propagation, technological or otherwise [26].

6.2. Policy for Handling Change Propagation

This thesis is part of an ongoing effort in the research community to address the engineering industry's struggle with change propagation. Previous research suggests a three-pronged ECM policy, or strategy, for dealing with change propagation: namely, prevention, prediction, and control. The multilayer network model further emphasizes that the execution of this policy should consider the influence of all three layers (product, change, and social) on change propagation behavior. Specifically, the *prevention* of change propagation can be accomplished through flexible design in the product layer and communication in the social layer. The *prediction* of change propagation requires analysis of the product layer for tactical purposes and a fundamental understanding of the change layer for more strategic considerations. Finally, the *control* of change propagation especially hinges on the management of the social layer. Overall, the multilayer network model offers a holistic framework for informing engineering change management policy.

6.2.1. Prevention

The first way an organization might reduce the negative effects of change propagation is to avoid it from occurring altogether. Change propagation can be prevented through effective management of both the product layer and social layer. Within the product layer, an organization can embed flexibility in a design to avoid change propagation wherever it appears economically viable. Within the social layer, an organization can foster necessary communication between teams and engineers who are designing interdependent parts of the product.

The product layer is an obvious setting for preventing change propagation. Suh and de Weck [31] provide guidelines for reducing propagation by embedding flexibility into a design. They draw on the Component-CPI for quantitative insight into where flexibility is needed in the product. One approach is to eliminate propagation at the source by converting multipliers (CPI > 0) into absorbers (CPI < 0). Another approach is to address the carriers (CPI = 0) that lie in the middle of dangerous propagation paths. Flexibility can be embedded into these critical areas of the product in several ways. As suggested by Eckert et al. [14], a "buffer" component could be inserted along a propagation path to absorb propagated changes rather than carrying them forward. For example, in the program from Chapter 5's case study, a buffer component shielded hardware segments from almost any propagation. A designer might also split a monolithic component into smaller components to increase its flexibility and decrease it potential to propagate [16]. All prevention strategies must strike a balance between the upfront investment required for embedding flexibility and the expected cost of propagation effects in the future [31]. As such, it is not necessarily economically viable to eliminate change propagation completely. Sometimes, the cost of flexibility might outweigh the expected cost of propagation.

Preventing propagation is also possible through management of the social layer. Communication among teams and engineers during product development is essential, especially among those who are designing interdependent parts of the product. Sosa et al. [28] reiterate that technical interfaces in the product layer (or domain) should be matched by communication interactions in the social layer. Consistency between the product and social layer is a critical means of preventing change propagation. As mention earlier, Pratt & Whitney suffered the consequences of an unattended interface when they had to disassemble, redesign, and rebuild several test engines. In light of the number of parts affected (i.e., change propagation), the redesign was estimated to add 1% to 2% total cost to the entire development program. Jarratt et al. [19] report a similar experience with a UK engine manufacturer, in which engineers had to redesign a gear-train twice because the first attempt triggered too much change propagation. A communication breakdown apparently caused this incident. The lesson here is that communication in the social layer is critical in the prevention of design mistakes that may ultimately lead to otherwise unnecessary change propagation.

6.2.2. Prediction

The next element of the three-pronged ECM policy is the prediction of change propagation. A prediction capability is at the heart of change impact analysis. The ability to predict change propagation has both tactical and strategic implications. *Tactical prediction* is useful in the short term, such as when an organization assesses the impact of individual change requests. Tactical prediction draws on analysis of the product layer through tools like CPM. Meanwhile, *strategic prediction* has long-term utility, such as life-cycle cost estimation during the earliest stages of product development. Strategic prediction requires a fundamental characterization of propagation behavior in the change layer.

The *tactical prediction* of change propagation is crucial for day-to-day ECM. Whenever an organization receives or generates a change request, its total impact should be evaluated in light of all potential propagation effects. According to the Aberdeen Group [2], a tactical prediction capability is exactly what the industry is lacking; the vast majority (89%) of engineering firms cannot produce a precise list of parts at risk of propagation from a given change request. Keller et al. [21] and Clarkson et al.'s [6] CPM is the most remarkable tool created for the tactical prediction of change propagation. Of course, other tactical prediction tools have been developed in the literature as well, including Cohen et al.'s Change Favorable Representation (C-FAR) [7] and Rutka et al.'s Change Propagation Analysis (CPA) [26]. Like CPM, both C-FAR and CPA hinge on a model of the dependencies among components to predict where changes may propagate. Still, CPM is unique among these and other prediction tools in that it is time efficient, allows for multiple propagation steps, and has been applied to a real-world complex product (i.e., a helicopter) with promising results [6]. Nevertheless, CPM does not allow for indirect propagation. As Chapter 5's case study demonstrated, indirect propagation may be an important aspect of software development. CPM's shortcoming here is reminiscent of Jarratt et al.'s [19] observation that no hard and fast rules exist for predicting propagation for all type of changes, products, or systems. Robust tools for tactical prediction likely must be tailored to the context of the product development underway.

Meanwhile, the *strategic prediction* capability can serve longer-term needs, especially in the estimation of life-cycle costs. NASA [23] defines life-cycle cost as "the total cost of the direct, indirect, recurring, nonrecurring, and other related expenses incurred...in the design, development, verification, production, operation, maintenance, support, and disposal of a project." As such, part of a project's life-cycle cost is the cost of change activity, including change propagation. A strategic prediction capability would enable an organization to estimate the total expected cost of change activity before a project begins in earnest. One general way to estimate costs is by analogy, as prescribed by NASA during the early stages of product development [23]. An *analogy approach* starts with the cost of analogous projects and makes adjustments depending on differences between the current project and the analogous one. In kind, a strategic prediction capability might look at the change behavior of analogous projects to estimate the total amount of change activity expected for an upcoming project. For example, Chapter 5's case study produced a statistical distribution of the number of generations propagated by all initiating changes (Figure 40, page 121). In preparation for developing an analogous software system, an organization could use these statistics to help estimate the expected cost of change activity. Accurate assessment of total change activity and life-cycle cost has significant implications for general project management and the acquisition of technology and systems by the government and military. Strategic prediction tools for cost estimation constitute an excellent subject for future research.

6.2.3. Control

The final element of the three-pronged ECM policy is the control of change propagation. Once a change request has been accepted, its successful implementation must account for all the anticipated and unanticipated propagation effects. As such, an organization might still be able to control the amount of change propagation that occurs. The control of change propagation has received some qualitative attention in the literature, most notably by Eckert et al. [14]. Meanwhile, this thesis has uniquely highlighted the importance of the social layer on propagation phenomena. An organization can employ quantitative analysis of the social layer to inform human resource management for controlling change propagation.

Eckert et al. [14] describes two distinct strategies for controlling change propagation (or "handling change" in [14]). One strategy is *forwards redesign*, by which an organization begins with a problem, considers alternative solutions in terms of propagation effects, and executes the planned solution with refinements if necessary. The other strategy is *backwards redesign*, by which an organization starts with a problem but "jumps" quickly to a solution and makes modifications as necessary with the same solution-first philosophy. The forwards redesign strategy is more careful and deliberate about its implementation plan. By contrast, the backwards redesign strategy seeks a quick solution; this strategy, though greedy on the surface, is not necessarily ill-advised. In fact, sometimes a backwards redesign is the best strategy, especially for familiar changes (e.g., software patches) or when an organization is up against a deadline. Of course, the risk here is that a quick solution may ultimately trigger more propagation than would have been necessary with a more careful process. Eckert et al. explain that an organization will likely practice a combination of forwards and backwards redesign strategies, depending on the familiarity and urgency of the change request.

Chapter 5's case study suggests that human resource management is another critical element of controlling change propagation. The program in the case study had a minority of engineers whose propagation behavior did not correspond with their organizational roles (roughly defined) or the propagation behavior of their assigned system areas. These results suggest that the engineers' technical competences or the organization's human resource management was making a difference. This thesis proposed the Engineer-CPI as a quantitative metric for evaluating where managerial attention may be warranted. Where unfavorable performance is found, the solution may be additional training, new communication protocols, or better workload distribution. Where favorable performance is found, an organization should promote any identified good practices in the future. Either way, quantitative consideration of the social layer may translate to important management imperatives to help control change propagation.

In summary, the research community is responding to the engineering industry's need for better ways to deal with change propagation. The literature suggests that industry adopt a three-pronged ECM policy that simultaneously fosters the prevention, prediction, and control of change propagation. Within this three-pronged strategy, the multilayer network model emphasizes a holistic, multilayer perspective.

This page has been intentionally left blank.

7. Conclusion

This chapter concludes the thesis with a summary of the research findings and recommendations for future work. This thesis proposed a novel multilayer network model of change propagation and established a baseline repository of single-layer, double-layer, and triple-layer tools and metrics. A case study of a real-world engineering project demonstrated the practical utility of the model. However, further research is necessary to prove the model's general applicability. A few notable avenues for future work include the development of better visualization techniques, additional probing of the social layer, and the development of prescriptive strategic prediction tools.

7.1. Summary of Research Findings

This thesis sought to answer the following primary research question:

- *What insights can be gained from a multilayer network model of change propagation?*

Two secondary research questions were also posed:

- *What are potential tools and metrics for analyzing a multilayer network model of change propagation?*
- *How can these tools and metrics inform future engineering change management policy in terms of design strategy, change request evaluation, and human resource management?*

In response to the first two questions, this thesis argued that a multilayer network model of change propagation facilitates extensive quantitative analysis of change propagation using a repository of tools and metrics. The multilayer network model proposed in Chapter 3 is composed of three layers, or domains, that contribute to the phenomenon of change propagation: namely, the product layer, change layer, and social layer. *Engineers* in the social layer work on *changes* in the change layer that affect *components* in the product layer. The baseline repository of tools and metrics developed in Chapter 4 enables single-layer, double-layer, and triple-layer analyses of the model. By incorporating a number of methods from the literature, the multilayer network model unites previous research on change propagation in a comprehensive paradigm. Finally, the model uniquely allows quantitative analysis of the previously overlooked, but significant, social layer of change propagation.

To demonstrate the hypothesized practical utility of the model, Chapter 5 conducted a case study of a real-world engineering project. The case study elucidated two major topics of significant interest to the research community and industry. One topic dealt with the socials layer's effects on propagation behavior and implications for engineering change management. The quantitative analysis here was a pioneering effort and illustrated the potential of the Engineer-CPI and Proposal Acceptance Rate as metrics for the employee performance in the implementation and proposal of changes, respectively. The Engineer-CPI was shown to correspond roughly with an engineer's organizational role. However, deviations from this correspondence suggested that other factors were also at play. For instance, the Engineer-CPI also appeared to depend on the Component-CPIs of his assigned areas. Furthermore, based on previous literature and interview input, it was also suggested that the Engineer-CPI might be a reflection of an engineer's technical competence and an organization's human resource management.

The case study's second topic revolved around the general characterization of change propagation. Issues here included indirect propagation, propagation extent, and the effect of component centrality on propagation behavior. The system in this study experienced a significant amount of indirect propagation, but this non-intuitive behavior is potentially linked to complex software development. Almost all indirect propagation spanned two product interfaces, but never more than three. Propagation in general was infrequent, which differs from the literature's experience with electro-mechanical systems. Initiating changes yielded a maximum of five generations of descendants (but almost always four or less), which is more consistent with previous research. Finally, the centrality of a system area had a positive correlation with its total propagation activity. However, high centrality did not guarantee the occurrence of change propagation.

To address the last research question, Chapter 6 considered the policy implications of the multilayer network model and general change propagation research. A comprehensive study by the Aberdeen Group [2] and several other case studies indicated that the engineering industry continues to struggle with change propagation during product development. The literature points to the industry's lack of hard and fast methods for effectively handling change propagation. The research community, including this thesis, has responded with an array of much needed tools and metrics. Although these methods have been minimally tested, they suggest a three-pronged policy, or strategy, for dealing with change propagation: namely, prevention, prediction, and control. The multilayer network model emphasizes a holistic perspective for ECM policy. Indeed, the prevention, prediction, and control of change propagation can draw on all three layers of the multilayer network model.

7.2. Future Work

This thesis creates several avenues for future work, including the following:

- Better *visualization techniques* for the multilayer network model are needed. Computers are obviously still able to process the underlying data without any visualization. However, a clearer drawing may reveal patterns and other insights more readily recognized by the human brain.

  A few options exist for better visualization techniques. One is to treat the multilayer network model as a single grand network using a spring embedding algorithm (as in Figure 24, page 84). This option will likely yield a less jumbled drawing, but the viewer will lose sight of the intended layered structure of the model. Another option is to keep the layered structure, but reposition the nodes in each layer using an augmented spring embedding algorithm. This option may confront a trade between the clarity of intra-layer edges and the clarity of inter-layer edges. The dataset from this case study provides an array of small and large change networks to test various multilayer network visualization techniques in the future.

- This thesis has only scratched the surface of the *social layer.* Many questions remain about the social layer's contribution to propagation phenomena. For instance, it may be insightful to consider an engineer's CPI over time; perhaps his (or her) propagation behavior is correlated with his workload (i.e., the number of change requests assigned to him) or milestones during product development.

  It would also be worthwhile to consider the communication patterns among engineers in the social layer. The major question here is whether communication between engineers who are in charge of interdependent components is a viable way to prevent or reduce change or change propagation.

- One of the chief questions underlying all change propagation research regards the predictability of change and change propagation. As discussed in Chapter 6, prediction capabilities can be both tactical and strategic. Tactical prediction has received the most attention in the literature. However, strategic prediction, by which an organization uses heuristic relationships to help predict the expected change activity for a new project, may also be a valuable endeavor.

References

1. Bartolomei, J. (2007). Qualitative Knowledge Construction for Engineering Systems: Extending the Design Structure Matrix Methodology in Scope and Procedure. PhD Thesis, Engineering Systems Division.

2. Brown, J. (2006). Managing Product Relationships: Enabling Iteration and Innovation in Design. Aberdeen Group, Boston, USA.

3. Brown, J., and Boucher, M. (2007). Engineering Change Management 2.0: Better Business Decisions from Intelligent Change Management. Aberdeen Group, Boston, USA.

4. Browning, T. R. (2001). "Applying the Design Structure Matrix to System Decomposition and Integration Problems: a Review and New Directions." *IEEE Transactions on Engineering Management* 48(3): 292-306.

5. Clark, J. and Holton, D.A. (2005). *A First Look at Graph Theory.* World Scientific.

6. Clarkson P.J., Simons, C., and Eckert, C. (2004). "Predicting Change Propagation in Complex Design." *Transactions of ASME* 126: 788-797.

7. Cohen, T., Navathe, S.B., and Fulton, R.E. (2000). "C-FAR, Change Favorable Representation." Computer Aided Design 32: 321-338.

8. Danilovic, M. and Browning, T.R. (2007). "Managing Complex Product Development Projects with Design Structure Matrices and Domain Mapping Matrices." *International Journal of Management* 25: 300-314.

9. de Neufville, R. (1998). "Thesis Definition and Preparation: Some General Guideline." Internal Publication. Engineering Systems Division, MIT.

10. Diestel, Reinhard. (2006). *Graph Theory.* 3rd Edition. Springer.

11. Eppinger, S. Whitney, D., Smith, R., and Gebala, D. (1994). "A Model-based Method for Organizing Tasks in Product Development." *Research in Engineering Design* 6(1): 1-21.

12. Eppinger, S. D. (2001). "Patterns of Product Development Interactions." ESD Internal Symposium, Cambridge MA, MIT Engineering Systems Division.

13. Earl, C., Eckert, C., and Clarkson, J. (2005). "Design Change and Complexity." 2nd Workshop on Complexity in Design and Engineering.

14. Eckert C., Clarkson P. and Zanker W. (2003). "Change and Customization in Complex Engineering Domains", *Research in Engineering Design* 15: 1-21.

15. Freeman, L.C. (2000). "Visualizing Social Networks." *Journal of Social Structure* 1.

16. Giffin, M. (2007). Change Propagation in Large Technical Systems. S.M. Thesis. System Design and Management Program, MIT.

17. Giffin, M., de Weck, O., Bounova, G., Keller, R., Eckert, C., and Clarkson, J. (2009). "Change Propagation Analysis in Complex Technical Systems." *Journal of Mechanical Design:* 131 (8), 081010.

18. Huang, G.Q. (1999). "Current Practices of Engineering Change Management in UK Manufacturing Industries." *International Journal of Operations and Production Management* 19(1): 21.

19. Jarratt T., Eckert C. and Clarkson J. (2005). "Pitfalls of Engineering Change: Change Practice during Complex Product Design." *Advances in Design*: 413-424.

20. Kirchain, R. (2010). "Course Introduction." ESD.80: Seminary in Technology and Policy Research. Spring 2010. MIT.

21. Keller, R., Eger, T., Eckert, C.M., and Clarkson, P.J. (2005). "Visualizing Change Propagation." *15th International Conference on Engineering Design*: 62-63.

22. Morelli, M.D., Eppinger, S.D., and Gulati, R.K. (1995). "Predicting Technical Communication in Product Development Organizations." *IEEE Transactions on Engineering Management* 42(2): 215-222.

23. NASA/SP-2007-6105 R1. (2007). *NASA Systems Engineering Handbook.*

24. Newman, J. (2003). "The Structure and Function of Complex Networks." *Society of Industrial and Applied Mathematics* 42(2): 167-256.

25. Pikosz P., Malmqvist J. (1998). "A Comparative Study of Engineering Change Management in Three Swedish Companies." *Proceedings of the DETC98 ASME Design Engineering Technical Conference*: 78-85.

26. Rutka, A., Guenov, M., Lemmens, Y., Schmidt-Schaffer, T., Coleman, P., and Riviere, A. (2006). "Methods for Engineering Change Propagation Analysis." *25th International Congress of the Aeronautical Sciences. 9.3.3.*

27. Sosa, M.E., Eppinger, S.D., and Rowles C.M. (2000). "Understanding the Effects of Product Architecture on Technical Communication in Product Development Organizations." MIT Sloan School of Management Working Paper. No 4130.

28. Sosa, M.E., Eppinger, S.D., and Rowles, C.M. (2007). "Are Your Engineers Talking to One Another When They Should?" Harvard Business Review: 133-142.

29. Sterman, J.D. (2000). *Business Dynamics: Systems Thinking and Modeling for a Complex World.* McGraw-Hill/Irwin.

30. Steward, D.V. (1981). "The Design Structure System: a Method for Managing the Design of Complex Systems." *IEEE Transaction on Engineering Management* 28(3): 71-74.

31. Suh, E.S., and de Weck, O.L. (2007). "Flexible Product Platforms: Framework and Case Study." *Research in Engineering Design* 18 (2): 67-89.

32. Terwiesch, C and Loch, C. (1999). "Managing the Process of Engineering Change Orders: the Case of the Climate Control System in Automobile Development." *Journal of Production Innovation and Management* 16: 160-172.

33. Wertz, J.R., and Larson, W.J. (1999). *Space Mission Analysis and Design* (3rd Edition). El Segundo, CA: Microcosm Press.

34. Wright, I.C. (1997). "A Review of Research into Engineering Change Management: implications for product design." *Design Studies* 18: 33-42.

35. Yin, R. (2009). *Case Study Research: Design and Methods.* 4th Edition. SAGE Publications, Inc.

This page has been intentionally left blank.

Appendix: Raw Data

The following pages contain the raw data for the change requests comprising 11-CR and 87-CR, as drawn in Figure 28 (page 101) and Figure 29 (page 102), respectively. The "Assignees" column for each change request (CR) may have the same engineer (e.g., eng001) listed multiple times, which indicates that the engineer was assigned to that CR multiple times over the course of the CR's evaluation and implementation or rejection. Empty cells indicate that no data were available.

11-CR

ID	Month Created	Month Last Updated	Area	Magnitude	Parent	Children	Siblings	Submitter	Assignees	Complete?
32496	91	92	10	2		36814	32821	eng302	eng234	-1
32497	91	98	10	1	32496		32573, 32852, 36814	eng302	eng302	1
32573	91	98	17	3	32496		32497, 32852, 35724	eng302	eng302	1
32821	91	92	14	0			32496	eng231	eng122, eng183	-1
32822	91	92	10	1	32821	36055		eng231	eng183	1
32852	91	105	17	3	32496		32497, 32573, 35724, 37008, 37635	eng302	eng302, eng310	1
35390	94	95	17	2			37635	eng301	eng275	1
35724	95	97	3	0			32852, 32573	eng172	eng388	-1
36814	96	105	10	1	32852		32497	eng302	eng122	1
37008			3	-1			32852		eng15 eng296	0
37635	97	99	17	3			35390, 32852	eng301	eng310	1

87-CR

ID	Month Created	Month Last Updated	Area	Magnitude	Parent	Children	Siblings	Submitter	Assignees	Complete?
8000	61	75	3	0			12156	eng015	engqual eng015 eng015 eng015 eng015 eng015 engqual engqual engqual	1
12156	68	114	3	0			8000, 13320, 22946	eng178	eng015 eng015	-1
13320			23	-1			12156		eng178	
22850	81	86	1	0		22946		eng013	eng013 eng013 eng013 eng013 eng013 eng013	1
22946	81	99	3	0	22850		31235, 12156, 26117, 30548	eng015		-1
23024	81	87	1	0		23942, 23945, 23992, 27169	23922	eng204	eng022 eng022 eng022 eng041 eng041 eng022 eng022 eng022 eng022	1
23729	82	88	3	1	23922		23821, 24980, 29731	eng302	eng276 eng260 eng276 eng276 eng276 eng276 eng276 eng276 eng276 eng276	1

23821	82	95	1	0		23922	23729, 24781	eng013	eng013 eng013 eng013 eng013	-1
23831	82	86	1	0		23925	24781, 25476, 25515	eng008	eng008 eng008 eng041 eng008 eng008 eng008 eng008 eng008	1
23922	82	95	3	0	23821	23729	23024, 27169	eng041	engqual eng276 eng276 engqual	-1
23925	82	88	14	3	23831	23831, 24781, 25476, 25515		eng041	eng183 eng183 eng183 eng183 eng183 eng183 eng183 eng183 eng183	1
23942	82	113	19	0	23024		23945, 23992, 27169	eng106	eng121 eng121 eng121 eng121 eng121 eng121	-1
23945	82	110	5	1	23024		23942, 23992, 27169	eng106	eng177 eng079 eng177 eng177 eng177 eng177 eng177 eng177	-1
23992	82	113	5	2	23024		23942, 23945, 27169	eng079	eng177 eng177 eng177 eng177 eng177 eng177 eng177 eng177 eng177	-1
24659	83	88	10	2		25053	24926	eng183	eng234 engqual eng222 eng222 eng234 eng234 eng234 eng234 eng234 eng234 eng234	1

24781	83	88	4	2	23925		23831, 25476, 25515, 23821, 25481, 29826	eng183	eng021 eng021 eng021 eng021 eng021 eng021 eng021 eng021 eng021	1
24926	83	88	10	1		24927	25463, 24659, 25053, 25481	eng183	eng234 eng222 eng234 eng234 eng234 eng234 eng234 eng234 eng234 eng234 eng234	1
24927	83	88	14	1	24926	24926, 25463		eng183	eng183 eng183 eng183 eng183 eng183 eng183 eng183	1
24980	83	88	3	2			23729	eng178	eng302 eng310 eng310 eng310 eng310 eng310 eng310 eng310 eng275 eng302 eng302 eng302 eng302 eng302 eng302 eng302	1
25053	83	95	10	0	24659		24926	eng073	eng234 eng234 eng234 eng234	-1
25463	84	88	10	2	24927		24926	eng122	eng234 eng234 eng234 eng234 eng234 eng234	1

25476	84	88	19	2	23925		23831, 24781, 25515	eng183	eng183 eng183 eng183 eng183 eng183 eng183 eng183 eng183 eng183	1
25481	84	88	10	1			24926, 24781	eng183	eng183 eng183 eng183 eng183 eng183 eng183 eng183 eng183	1
25515	84	88	20	2	23925		23831, 24781, 25476	eng183	eng183 eng183 eng183 eng183 eng183 eng183 eng183 eng183	1
26117	85	85	3	0			22946	eng318	eng275 eng302 eng275 eng275	-1
26331			3	-1		26333			eng168 eng168	
26333	85	90	3	2	26331		29711, 29226, 27023	eng244	eng244 eng244 eng244 eng244 eng244 eng244	-1
27023	86	95	3	0			26333	eng301		-1
27027	86	91	19	3		28695	28213, 28846	eng248	eng248 eng248 eng248 eng248 eng248 eng248 eng248 eng248 eng248 eng248 eng248	1
27169	86	113	19	1	23024		23942, 23945, 23992, 23922	eng177	eng121 eng121 eng121 eng121	1

27585	86	91	19	4			28007, 30143	eng087	eng176 eng176 eng176 eng176 eng176 eng176 eng176	1
27592	86	98	1	0			27952, 31966	eng343	company2 eng005 eng005 company2	0
27627	86	91	19	3		28878		eng030	eng030 eng030 eng030 eng030 eng030 eng030 eng030 eng030	1
27656	86	87	19	0		28528	28428	eng087	eng176 eng176 eng176 eng176	-1
27952	87	98	1	0		31966, 31967	28601, 30501, 27592	eng087	company2 eng005 eng005 company2	0
28007	87	91	3	2			27585, 28213	eng301	eng301 eng168 eng301 eng301 eng301 eng301 eng301 eng301 eng301	1
28009	87	90	1	0		30148	28067, 28428, 28531, 28821, 30465, 30548	eng013	eng013 eng013 eng013 eng013 eng013 eng013 eng013	1
28067			3	-1			28009, 28186		eng176 eng176 eng176	
28122	87	91	19	2		28153	28788, 28790	eng176	eng087 eng087 eng087 eng087 eng087 eng087 eng087 eng087	1

28153	87	89	3	0	28122		29711, 28213, 28878	eng244	eng244 eng168 eng244 eng244 eng244 eng244 eng244 eng244 eng244	-1
28162			35	-1			28601		eng087 eng087	
28166			3	-1			28567		eng251	
28186	87	91	19	1		28529	28067	eng176	eng176 eng176 eng176 eng176 eng176 eng176 eng176 eng176	1
28187	87	91	19	3		28213		eng176	eng176 eng176 eng176 eng176 eng176 eng176 eng176 eng176	1
28213	87	91	3	2	28187		28007, 28153, 27027	eng301	eng301 eng301 eng301 eng301 eng301 eng301 eng301 eng301 eng301 eng301	1
28428	87	91	19	2		28531, 28821	27656, 28009	eng299	eng176 eng176 eng176 eng176 eng176 eng176 eng176 eng176	1
28528	87	91	3	1	27656			eng168	eng294 eng294 eng294 eng294 eng294 eng294 eng294	1
28529	87	89	3	-1	28186		29711	eng168	eng244 eng244 eng244 eng244 eng244 eng244	-1

28531	87	91	3	2	28428		28821, 28009, 28878	eng168	eng294 eng294 eng294 eng294 eng294 eng294 eng294 eng294 eng294 eng294 eng294	1
28567	87	90	1	0		29538, 29547	28166	eng087	eng013 eng013 eng013 eng013 eng013 eng013 eng013	1
28601	87	98	1	0		31972	27952, 28162, 31973	eng087	company2 eng005 eng005 company2	0
28695	87	92	19	0	27027			eng309	eng309 eng309 eng309 eng309 eng309 eng309	1
28696	87	92	1	0		29226, 29227, 29744		eng013	eng013 eng013 eng013 eng013 eng013 eng013	1
28788			3	-1			28122		eng294 eng294	
28790			3	-1			28122		eng294 eng294	
28821	87	95	3	0	28428		28531, 28009, 29711	eng168	eng244 eng244 eng244 eng244 eng244 eng244 eng244 eng244	-1

28846	87	92	19	2		29399	27027	eng321	eng321 eng321 eng321 eng321 eng321 eng321 eng321 eng321 eng321 eng321 eng321 eng321	1
28878	87	92	3	2	27627		28531, 28153	eng244	eng244 eng244 eng244 eng244 eng244 eng244 eng244 eng244	1
29226	88	92	3	3	28696	29731	29227, 29744, 26333, 30126, 32289	eng106	eng301 eng301 eng301 eng301 eng301 eng301 eng301 eng301 eng301 eng301 eng301	1
29227	88	92	6	2	28696		29226, 29744	eng106	eng299 eng311 eng251 eng299 eng299 eng299 eng299 eng299 eng299 eng299	1
29353	88	98	19	2			29826	eng264	eng195 eng121 eng195 eng195 eng195 eng195 eng195 eng195 eng195 eng195 eng195	1
29399	88	95	19	0	28846			eng140	eng176 eng176 eng176	-1

29538	88	92	6	2	28567	30344, 30614	29547	eng106	eng176 eng087 eng176 eng176 eng176 eng176 eng176 eng176 eng176	1
29547	88	92	3	1	28567		29538, 29711	eng106	eng043 eng275 eng275 eng244 eng244 eng244 eng043 eng043 eng043 eng043 eng043 eng043 eng043 eng043 eng043	1
29711	88	92	3	0		30548	26333, 28153, 28529, 28821, 29547, 30148	eng244	eng244 eng244 eng244 eng244 eng244 eng244 eng244 eng244 eng244 eng244	1
29731	88	92	3	2	29226		23729	eng376	eng301 eng275 eng301 eng301 eng301 eng301 eng301 eng301 eng301 eng301 eng301 eng301	1
29744	88	91	19	2	28696		29226, 29227	eng299	eng299 eng299 eng299 eng299 eng299 eng299 eng299 eng299	1

29826	88	92	3	2			24781, 29353	eng275	eng264 eng264 eng264 eng264 eng264 eng264 eng264 eng264 eng264 eng264	1
30126	88	92	3	2			29226	eng301	eng301 eng301 eng301 eng301 eng301 eng301 eng301 eng301 eng301 eng301	1
30143	88	92	19	3			27585	eng030	eng030 eng030 eng030 eng030 eng030 eng030 eng030 eng030	1
30148	88	95	3	0	28009		29711	eng106	eng244 eng244 eng244	-1
30344	89	92	19	1	29538		30614	eng176	eng176 eng176 eng176 eng176 eng176 eng176 eng176 eng176 eng176	1
30465	61		11	-1			28009		eng421 eng421 eng421	
30466	68		11	-1			30503		eng013 eng013	
30501		98	1	0		30503	27952	eng013	company2 eng005 eng005 company2	0
30503	81	92	1	0	30501		30466, 30548	eng013	eng013 eng013 eng013 eng013 eng013	1

30548	81	92	3	3	29711		28009, 30503, 30771, 22946, 31471	eng296	eng296 eng296 eng296 eng296 eng296 eng296 eng296 eng296 eng296 eng296 eng296 eng296 eng296 eng296 eng296	1
30614	81	92	14	1	29538		30344	eng017	eng017 eng017 eng017 eng017 eng017 eng017	1
30771	82	92	1	0			30548	eng013	eng013 eng013 eng013 eng013 eng013	1
31235	82	92	3	0			22946	eng019	eng043 eng043 eng043 eng043 eng043	-1
31471	82	92	3]	0			30548	eng140	eng244 eng244 eng244 eng244 eng244 eng244	-1
31966	82	96	19	3	27952	32645	31967, 27592, 31972	eng087	eng251 eng087 eng176 eng176 eng251 eng251 eng251 eng251 eng251 eng251 eng251	1
31967	82	96	19	3	27952		31966, 31973	eng087	eng251 eng087 eng251 eng251 eng251 eng251 eng251 eng251 eng251 eng251 eng251	1

31972	82	95	19	0	28601		31966	eng087	eng251 eng251 eng251	-1
31973	82	95	19	0			28601, 31967	eng087	eng251 eng251 eng251	-1
32289	82	92	3	1			29226	eng296	eng296 eng296 eng296 eng296 eng296 eng296 eng296 eng296 eng296 eng296	1
32645	83	96	14	1	31966			eng261	eng261 eng261 eng261 eng261 eng261 eng261 eng261 eng261 eng261 eng261	1