COMPUTATIONAL REPRESENTATION AND REASONING SUPPORT FOR REQUIREMENTS CHANGE MANAGEMENT IN COMPLEX SYSTEM DESIGN

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Mechanical Engineering

---

by
Beshoy Wahib Morkos
May 2012

ABSTRACT

Requirements play a critical role within any design process and the activity of identifying and maintaining a system's requirements is essential. However, design is a complex and iterative process, where requirements are continuously evolving and are volatile. This change, if not managed, may result in undesired uncertainty within the design process leading to monetary losses and time delays, as the changing of requirements has been recognized as a major cause of project failure. In order to mitigate issues that arise due to requirement change propagation, this research presents a computational reasoning tool to help designers and engineers predict change propagation in the requirements domain. The developed tool makes use of requirements syntactical elements to build relationships between requirements. Two heterogeneous industry case studies, spanning four engineering change propagations, are used to both explore the use of requirements in predicting change propagation and generalize an automated prediction tool. Using design structure matrices and graph theoretic based metrics a predictive model is generalized from 491,520 relationship and metric permutation combinations. The developed tool makes use of an RMS scoring algorithm to rank requirements in order of most likely to change due to previous requirement changes. The developed tool is tested against a third industry case study where five engineering changes are predicted. Results indicate the tool can predict sixty percent of change propagation within the top four percent requirements scoring and predict all change propagation within the top thirteen percent scoring.

DEDICATION

Fulfilling a promise I made a long time ago. Dedicated to my father, Wahib Morkos.

ACKNOWLEDGEMENTS

I want to first and foremost thank my Lord, God and Savior Jesus Christ "for with God nothing will be impossible" (Luke 1:37)

I would like to wholeheartedly thank my advisor, Dr. Joshua Summers, who was has assisted me in growing as a researcher, a professional, and a human being. Without his dedication I would not be where I am today, and I hope that one day I will become as influential to my students as he has been to me. Echoing the statements of past students, I sincerely thank him for his steadfast support. With greatest of respect, I want to thank Dr. Gregory Mocko for his continuous involvement in my development. He has contributed to my research in many ways and I am fortunate to have someone as perceptive as he on my committee. I would also like to extend thanks to Drs. Georges Fadel and Joel Greenstein for their assistance in my Ph.D. completion. A deep gratitude to Dr. Laine Mears for dedicating so much of his time to helping me reach my academic and developing my career goals. I want to thank him for all the mentoring he provided me when he didn't need to. He is a model for what I wish to become in the future.

CEDAR lab has been instrumental to my success the past five years, providing many opportunities for stimulating dialog, deep friendships, and the chance to learn from my lab mates. With profound respect, I want to specifically thank: David Stowe, Sudhakar Teegavarapu, Shraddha Joshi, Prabhu Shankar, Chiradeep Sen, Ben Caldwell, James Mathieson, and Essam Namouz.

I want to, with love, thank my mother Sanaa Morkos for being my motivation and support mechanism throughout my life. Thanks are also due to my sister, Ereni Morkos, for being a source of inspiration and enthusiasm during this process. Without them and their sacrifices for me, I could not have accomplished this goal. I apologize for seemingly abandoning you as I entrenched myself in research, and promise to never do that again. I thank you for keeping me focused on the task at hand, even when I was led astray. I also want to thank the love of my life, Maryam Mishrigi, for her constant support and compassion. This is for you…for us.

I want to thank my priest Father Shenuda Morkos for his reminder of faith, focus, and fortitude. You have kept me humble during my highs, and perseverant during my lows. For my professional development, I want to specially thank George Ishak, Dr. Yousry Azmy, Nagi Seedhom, and Dr. Adly Girgis whom I wish were here to see this.

Finally, I acknowledge my close friends in life. Though they may not realize, I needed their friendship to keep me afloat during my hardships, academically and personally. Without them, my stress and circumstances would have surely overpowered me. When I was on the verge of failing, you offered support and you reminded me of whom I am. These are my best of friends: Michael Haje, Ryan Gaston, Brandon Sevener, Andy Mauk, Chris Terry, Mark Mikhail, Michael Bassily, Dr. Michael Beshay, Peter and Christine Riyad, Rani Seedhom, Michael and Kendall Wahba, Veronica Jacob, Maryan Mikhael, Mina and Bishoy Tadros, Mina and Marco Sidhom, Michael and Joe Messieha, Cindy and Melissa Youssef, Michael and Peter Beshay, Tony Abdelmalek, Kerollos and Kristy Malak, Abanob Khiralla, Mina Gad, Mike and Moura Messiha, and Mirna Rezkalla. I am forever grateful for their support and hope I can one day return the favor.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1.
OPEN ISSUES IN REQUIREMENTS CHANGE PROPAGATION

Identifying, managing, and satisfying requirements play a critical role within any design process [1,2,3,4,5]. The activity of identifying a user's or system's need is essential to the success of a project as one of the initial steps in the design process, the correct identification and specification of requirements is critical [6]. Requirements define what the stakeholders, users, customers, suppliers, developers, and businesses need and how each need is to be satisfied [5]. As one of the initial documents of any design project, requirements are captured, in many instances, in the form of a client contract and are maintained throughout the process to ensure project completeness. However, requirements are often neglected, and not properly maintained throughout the design process. This negligence includes, but is not limited to, not maintaining, ignoring changes, and not validating requirement satisfaction. Consequently, monetary and time losses are involved with such negligence. The focal of this research addresses requirement change propagation through the development of a tool to predict requirement change propagation.

Design is a complex and dynamic process [7,8,9,10,11,12]. A requirement document is not fixed in the data it contains and is used by multiple individuals over the span of a design project. For example, during a sponsored project by BMW to develop an LED headlight, great concern was given to the environment in which the LEDs were placed. Specifically, the temperature of the environment had significance on the ability of the LED to operate efficiently. Initially, it was assumed the maximum temperature the

LED substrate may reach is 120°C. This requirement was later changed to 90°C. In making this change, greater constraints were placed on the performance of the LED. This resulted in changing the design to incorporate more LEDs which cumulatively operate at a decreased power to avoid temperature build up. As a result, the concept grew in dimensional size and many more LEDs were needed. This requirement change resulted in the change of other requirements pertaining to geometry, spatial constraints, power requirement and number of LEDs. The basic concept of this project is detailed in [13,14] but the details of the requirements evolution is not yet reported in the literature.

Requirements change over time as a design project progresses from its conceptual to detailed stages, changing in structure, abstraction, and quantity. Most design projects tend to evolve as they are an iterative process, and as a result, requirements will do the same [15,16,17]. The change within requirements may cause undesired uncertainty and complexity within the design process. Further, in many instances, different project stakeholders use different subsets of a requirements document and make changes without recognition of how this change may cross boundaries. For instance, during the design and build of a BMW X5, there are engineers who pertain to braking while others who are concerned with the interior. Though they may seem mutually exclusive, there are requirements (and requirement change propagation) which may cross boundaries between the two subsystems of the vehicle that an engineer is not aware of.

Change is expected and can occur within any stage of the product life cycle [18], and, as a result, is recognized as a major cause of project failure [15,19]. Change occurs when companies request, implement, and affect changes to products, documents, components, manufactured or purchased parts, processes, or even supplies [20]. Requirements change, and this change continuously occurs and roots from different sources involved within the design project. This results in a need for requirements management, specifically to address the change within requirements to mitigate the negative consequences that may occur due to change propagation.

These negative consequences include, but are not limited to, the increase complexity [21,22], opportunity for data loss [23], and cost and time wasted [24]. The result of requirement changes introduces many challenges that become barriers for successfully completing a design project that accurately meets the needs of the client. A designer could save time and money if it were possible to make a quick, yet accurate, assessment about the overall effects of a design or requirement change before committing to implementing a change [25]. As a result, the need for a systematic and methodical manner of managing requirements change [15,26,27] and mitigating the negative consequences of requirement change propagation is needed [28,29].

In order to mitigate or eliminate issues that arise due to requirement change, this research has developed a tool capable of predicting requirement change propagation. While the occurrence of requirement change has been recognized, the understanding and modeling of this change has not been thoroughly researched and has been an active research topic as it aims to enhance the quality of the designed product and to improve the efficiency of the design process [30]. However, great difficulty is involved with managing requirements change, specifically with how this change can be represented and reasoned. This is due to their evolving and dynamic nature and the many characteristics that could be modeled, including change type, requirement relationships, and impact of change. Specifically, example characteristics may include the element of change, time of change, source of change, propagation of change, or employees associated with the change; all factors which contribute to change propagation. As this research demonstrates, the use of requirements as a change propagation predictor is possible and can provide many benefits to a designer or engineer.

The use of a traditional requirements list does not accommodate the need for the ever evolving requirement lists of design problems. Traditional requirement lists are of static nature, simply revealing the requirements of that specific time stamp. This lacks in the ability to view change occurrence, type of change, or change propagation. A requirements document, in its current state, is the most ubiquitous way to model requirements. This is evident by the “contracts” used to document and manage requirements in most industry design projects.

This research resolves issues identified with requirements change propagation by developing a means to systematically relate requirements and predict their propagation based on this relationship. Requirement change propagation is predicted by making use of a syntactical tool, analyzing how the change in one requirement may affect other requirements in a requirements document. The capacity to predict requirement change propagation lies within the requirement relationships developed between requirements. A syntactical approach is used to develop relationships between requirements based on their part of speech (POS) tagging. This automated tool is benchmarked against the initially developed manual tool of developing requirement relationships based on their subject and keyword.

1.1. Research Objectives and Deliverables

Motivating this research, the primary objectives are listed in Table 1.1. Additionally, the corresponding outcomes for each objective are generally described. Each objective is associated with a completed outcome, in the form of a conference or journal paper, so the satisfaction of this dissertation can be measured through the deliverables detailed. Each objective is addressed in at least one section of the dissertation body. The associates sections are noted in the table.

Table 1.1: Objectives, Outcome and Completed Deliverables

I.D.	Objective	Outcome	Deliverable	Dissertation
A	Survey change propagation and design impact	List of methods and techniques used in managing change propagation	Conference Paper [23]	Section 2.3 and 2.4
B	Survey of requirements modeling tools	Review change modeling tool to find adequate tool to incorporate within syntax change representation developed.	Conference Paper [31]	Section 4.2
C	Explore the use of requirement syntactical elements for their capacity to predict change propagation	Studies which accurately show requirements syntactical elements may be used to predict change propagation.	Conference Paper [24]	Chapter 5
D	Investigate requirement change propagation with respect to requirement relationship	Review and sampling of different types of propagation characteristics based on the type of requirement relationship.	Journal Paper [32]	Chapter 7 Chapter 8 Chapter 9
E	Develop requirement change prediction tool making use of a POS tagging.	Robust tool accepting raw requirements document and making use of POS tags to develop requirement relationships most conducive to requirement change propagation prediction	Journal Paper [33]	Section 5.2

1.2. Benefits of Modeling Requirements Change Propagation

The benefits of a tool to model and predict requirements change propagation lies within three specific areas: (1) predicting change, (2) analyzing requirement sensitivity, and (3) evaluating requirement change impact. Each area introduces a potential use for the developed requirement change propagation prediction tool. The specific areas of gain are discussed and an example for each is provided.

1.2.1. Predicting Change

The chief, transformative contribution of this research is the ability to predict requirement change propagation. If designers and engineers are able to identify requirement relationships and view the propagation characteristics of such relationships, change propagation may be predicted. This propagation prediction tool will allow a designer to analyze the affects a requirement may have on the remainder of the design and other subsystems, irrespective of the designer's familiarity with the system. For example, a designer may wish to make a requirement change to improve the manufacturability of a product housing. However, this change could affect many related requirements to the product housing. Propagation analysis allows the designer to analyze the risk involved with making the change and to evaluate effected requirements before implementation. By foreshadowing requirement change propagation, the designer is better suited to address the change propagated in a timely and efficient manner. Further, the capacity of this tool is not limited to a requirement change as a designer may make a design change in the physical domain. If the designer is able to relate the physical change to the requirements domain, requirements may be propagated to view the potentially effected requirements. For example, the case studies presented in Section s Chapter 7, Chapter 8, and Chapter 9 detail the ability to predict potential requirement changes through an initial change in the physical domain. This initial change is tied to a requirement change which is propagated. This propagation is then analyzed to view if subsequent changes could have been predicted through the propagation prediction tool. The ability to predict requirement changes will assist in assessing the consequences and benefits of changing requirements.

1.2.2. Requirement Sensitivity

In an ideal condition, a designer would benefit from a single set of requirements that does not change. However, this is not likely in most instances as requirements will change to adapt to the many external and internal factors associated with any design process. Based on the network of relationships between requirements, there exists graph theoretic metrics to evaluate connectedness of requirements, which effectively identifies how related a requirement is to other requirements. Those requirements which are heavily connected, if changed, may more than likely cause change propagation to occur. Whereas other requirements may not be heavily connected, their change will have a minimal impact on the requirements document. Analyzing how sensitive a requirement is to change assists in determining how those requirements should be addressed. It could be that those requirements should be satisfied near the end of the design project as they have a high likelihood of changing due to their numerous connections. If satisfied early, there is a chance they will need to be resatisfied at a later time when they change due to propagation. A designer is able to view critical requirements which have the greatest effect on the design process and how their change must be controlled. While change may not be completely controlled, the ability to evaluate the value improvement of the change is of benefit to the designer.

1.2.3. Evaluating Requirements Change Impact

A change does not need to take place for the requirements change propagation prediction tool to be effective. The evaluation of requirement change allows a designer to view how specific changes within requirements will affect the remaining requirements. For example, a requirement change to assist in reducing product cost may necessitate re-satisfying multiple requirements that may have been affected by the changed requirement. In such a case, more cost may be associated with re-satisfying the propagated requirements, effectively negating the cost saving or increasing the time needed to implement the change.

For example a NASA sponsored project entailed the development and build of a cryogenic testing apparatus simulating the environment of the moon. Many of the requirements were fulfilled at an earlier time. In an attempt to develop a more robust design, changes were made to adjust the design concept. This resulted in evaluating the requirements satisfied and identifying if changing a requirement was worth reevaluating all the requirements previously satisfied [34].

The ability to predict requirements change is of particular important for large design projects in which the designer may not be directly associated with other designers or engineers. The larger and more complex the system is and the more people working on the project, the more likely it is that a problem will occur [35]. Requirement change must also be represented so an understanding of why a requirement was changed, how it was changed, the type of changed encountered, its relationship with other requirements in the system, and its affect on the total system may be analyzed. Such a tool facilitates the viewing of change impact and considers the value addedness of requirement changes.

This research will allow for those individuals involved within a design project to view change propagation and the possible change that may occur as a result. Managing requirement changes ensures the latest version of modifications, product and process data items and specifications are correct, a critical task in engineering management [20]. The ability to manage requirement changes will assist in preparing for requirement changes that may have at one time not been foreseen. An example of such a case in presented in the case studies where a requirement was changed and a propagated requirement change was not identified until over a month later. This research is not aimed to reduce the time in which propagation is identified; rather it is to provide the designer a tool for predicting requirement propagation before a change may occur. This allows a designer to understand what other requirements may be affected so they are addressed immediately, not at a later, unanticipated time.

1.3. Developed Requirement Change Propagation Prediction Tool

In order to mitigate or eliminate issues that arise due to requirement change, this research computationally predicts requirement change propagation using the implemented method shown in Figure 1.1. The developed tool operates by: (1) preprocessing requirements documents to prepare them for analysis, (2) identifying all permutations for requirement relationships, (3) selecting high performing POS and keyword relationship types, and (4) using the relationships to identify potential requirements changed due to propagation.

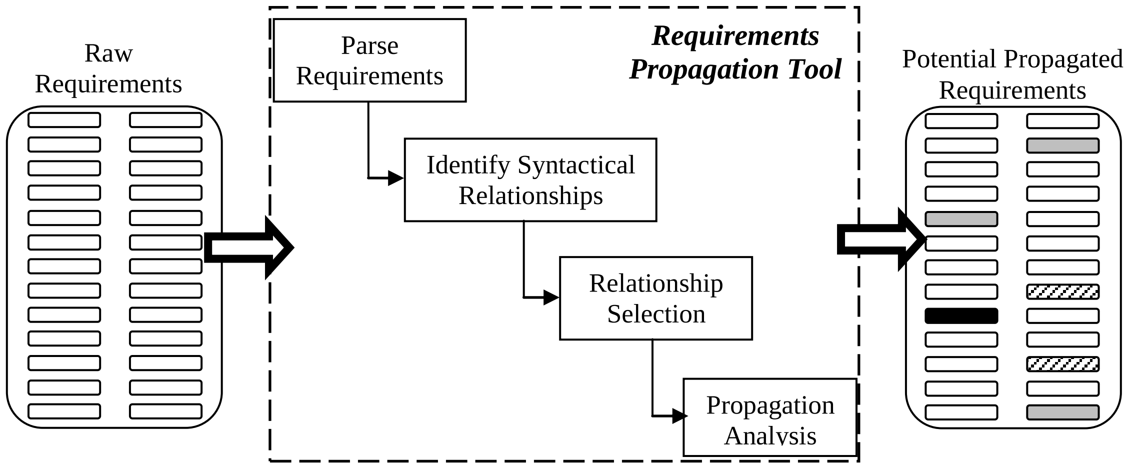

Raw Requirements
Requirements Propagation Tool
Parse Requirements
Identify Syntactical Relationships
Relationship Selection
Propagation Analysis
Potential Propagated Requirements

Figure 1.1: Requirement Change Propagation Method

In some instances, requirement change propagation is apparent while in many instances it is not as it has a multiple order effect on the requirements. Propagation occurs because requirements may be related to one other and this relationship may be based on subject, function, component, energy, or other relaters. Each relationship type will yield a different type of propagation model, which will in turn cause different propagation results. The correct and accurate selection of requirement relationships is critical to accurately predicting change propagation. Details of the tool and the method for selecting the relationships used will be provided in Section 5.3.2 of this dissertation.

CHAPTER 2.
RELEVANT RESEARCH ON REQUIREMENTS

It is widely accepted that requirements development is an integral part of the design process as they play a critical role in design decisions during the remainder of the project. Requirements may be developed in multiple fashions, however all include decomposing the system into subsystems and defining how these subsystems should operate and interact [36,37].

Before requirements change can be thoroughly researched and presented in this dissertation, an understanding of requirements and their role within the design process must be understood. With this understanding, research can be directed toward accurately predicting requirement changes and viewing its effect on the design process. The use of requirements in the design process can be broken down into four sections: elicitation, specification, analysis, and verification [38]. The elicitation of requirements is important as it is when requirements are first formally introduced to the design process. After eliciting requirements, they are recorded for later use and maintained as needed. Requirements may need to be translated into specifications at a point throughout the design process. Specifications are requirements written in specific engineering context, in a manner that verifiable and its satisfaction can be measured. Requirements analysis is the process of identifying a user's needs and determining what to build in a system, which includes a host of design activities which incorporate requirements, such as QFD. Their use in subsequent stages of the design process may vary greatly and is highly dependent on the type of project and design environment. As the requirements evolve, their changes are recorded and a need for requirement change management is introduced. This section details how requirements affect the design process, specifically in the following areas of interest: elicitation and recording, change, change propagation and modeling.

2.1. Requirements Role Within Design

Requirements are one of the initial documents generated during the design process and are of significance as they present the first set of information that represents the communication between the designer and stakeholder. The Rational Unified Process defines a requirement as “a condition or capability to which a system must conform; either derived directly from user needs, or stated in a contract, standard, specification, or other formally imposed document” [37]. The International Council on Systems Engineering (INCOSE) defines requirements as statements that identify system or product constraints deemed necessary for stakeholder acceptance [39]. As seen from Figure 2.1, the Pahl and Beitz process introduces requirement specifications early within the design process. They are introduced immediately after the task is clarified, and problem is introduced. This is similar with all other design processes as requirements must be introduced early in the design process before proceeding with other design activities, as they have great influence on several downstream design activities including idea generation, verification, and testing [31].

Because of their constant evolution, a requirement list not only reflects the initial position but serves as an up to date working document of the design process [1]. This requirements document could serve as a tool depicting the state of the design process, a transient document from abstract to detailed. As the requirements become less abstract and more measurable, the design process and problem is, in turn, more understood. Further, as more requirements are satisfied, the design is closer to completion.

Requirement statements identify critical attributes, characteristics, capabilities, or functions of the design in order to improve the understanding and focus of the designer [40]. This research will make use of attributes, specifically the syntactical POS, to develop requirement relationships. Serving many roles, requirements may be used as a benchmarking tool that assists in evaluating existing solutions for their ability to meet the customer needs. They may also serve as concept selection tools in their ability to use criteria in measuring different concept. By defining requirements, an expectation of the design solution is developed which constraints the solution space [41].

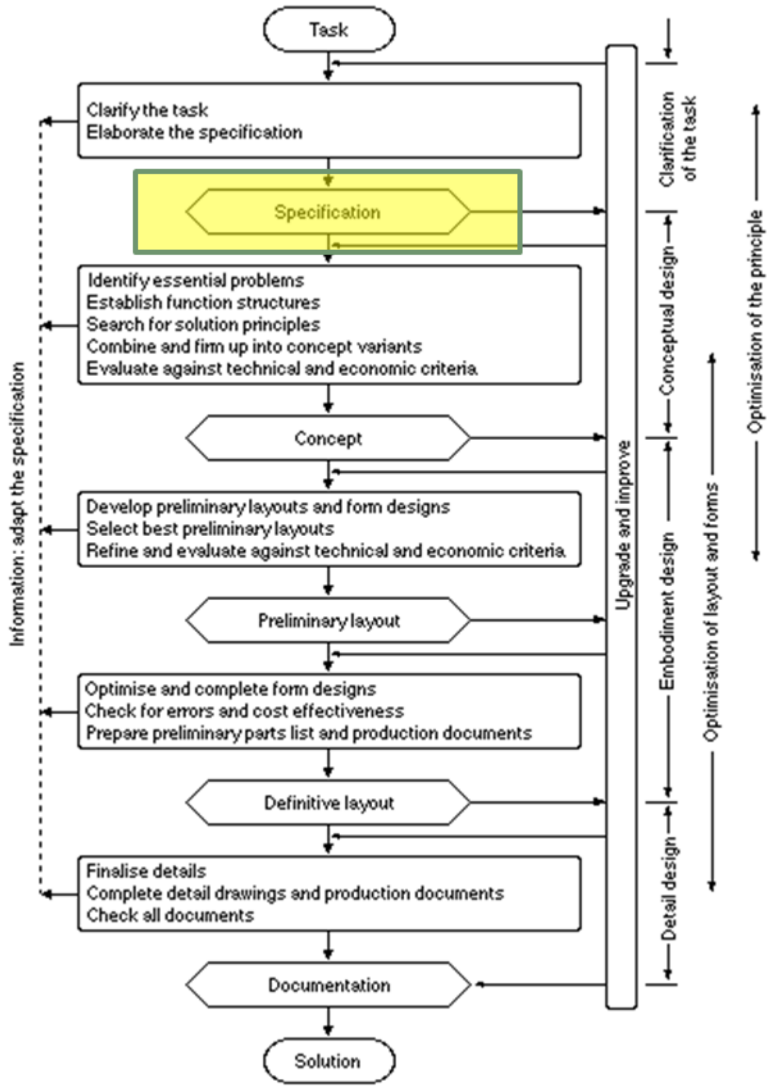

Task
Clarify the task
Elaborate the specification
Specification
Identify essential problems
Establish function structures
Search for solution principles
Combine and firm up into concept variants
Evaluate against technical and economic criteria
Concept
Develop preliminary layouts and form designs
Select best preliminary layouts
Refine and evaluate against technical and economic criteria
Preliminary layout
Optimise and complete form designs
Check for errors and cost effectiveness
Prepare preliminary parts list and production documents
Definitive layout
Finalise details
Complete detail drawings and production documents
Check all documents
Documentation
Solution
Information: adapt the specification
Upgrade and improve
Clarification of the task
Conceptual design
Embodiment design
Detail design
Optimisation of the principle
Optimisation of layout and forms

Figure 2.1: Pahl and Beitz Design Process Outline [1].

Ulrich and Eppinger provide a step by step procedure for requirements development [2]. This procedure allows the designer to define the influential members, stakeholders, of the project so that they may be interviewed. It provides information as to how raw data may be collected from the stakeholders so that requirements are elicited. The raw data is then interpreted and written in the form of engineering specifications that may be understood by those designing the project. Such specifications allow for writing the requirements in measurable form. This is important as the customer raw data will not always provide the designer with sufficient, accurate, and technical information as to what each component of the system must satisfy [42]. The requirement elicitation processes within in the Ulrich and Eppinger methodology is as follows [2]:

- Defining the scope of efforts
- Gathering raw data from users
- Interpreting raw data in terms of needs
- Organize the needs into a hierarchy of primary, secondary and tertiary needs
- Establish relative importance of the needs

Table 2.1 documents the takeaway of this section, which includes the significance of requirements in the design process. This is of importance here as the importance of requirements within design must be realized before developing a requirement based reasoning tool. Managing requirements is important because they affect many other areas of design where they are used to support design tasks and activities. Alongside the takeaways shown in Table 2.1, the recognition of requirement pertinence to the design process is important. As noted, requirements may be used for a host of design activities and their potential role in other activities and tools has yet to be realized. It is through change propagation that requirements are introduced as a potential domain to manage change propagation. This introduces multiple requirement domains that must be researched such as requirement change, change propagation, and how requirements are currently modeled.

Table 2.1: Takeaways of Requirements and Their Role within Design Research

Take Away	Ref
Requirements are an integral part of the design process	[2,42],
Requirements support many of the design tasks/activities within the design process	[6,37],
Serves as an up to date working document of the design process	[1]

2.2. Elicitation and Recording

The design process begins with the identification of a societal need [43] which may come in the form of a market gap, an opportunity for product redesign, or the introduction of a novel product. In the initial stages of the design process, based on a request made by customers, designers begin to identify the customer's true need from the product. This may prove to be of great difficulty as this need is not static and there exist a varying host of different types of customers, even within the same market segment. Nonetheless, requirement data from customers must be collected in order to gather all explicit and implicit requirements that the product has to satisfy [6]. Designers understand requirements to be statements about situations of use, which implicitly describe the customer's needs, as well as envisioned situations [44]. The manners in which requirements are elicited are highly dependent on the nature of the project and those stakeholders involved with the project. Many design approaches define their own means of capturing client raw data, which in turn is transformed into a requirement document.

Requirements elicitation is a process in which tacit information about what must be designed and developed is obtained from the user and their environment [45]. In some design instances, a design statement or problem statement will be provided to the design team, where the requirements will have to be elicited by the team. In other instances, a list of requirements will be provided to the design team for solution design and development. The design team, in this scenario, is to design and develop an artifact that satisfies the requirements provided. The primary goal of requirements elicitation is to objectify the nature as well as the boundaries of the problem domain that would be encapsulated within the proposed information system [46]. The study of requirements elicitation serves a means for achieving high quality, successful products through accurate development of stakeholder needs. It is the designer's responsibility to identify their stakeholders so that they may retrieve the appropriate requirements [11]. Though there are multitudes of design processes, from varying disciplines, two will be presented in this dissertation in the following sections.

2.2.1. Ullrich and Eppinger

To ensure that many of the development issues are addressed, the stakeholders, those who are affected by the product's specifications and characteristics, are identified as one of the initial steps of requirements elicitation. The stakeholders are end users of the product or individuals whom may place a constraint on the product such as those who sell or sponsor the product [2]. Further, it is not possible to write formal requirements until the stakeholders and their goals are identified [47]. The customer selection matrix serves as a tool that considers the requirements of every individual, end user or not, who will be influenced by the product so they may be interviewed. Interview question and setup is different for each stakeholder, based on the expected needs and use of the designed system [42]. Stakeholders are identified through each market, an individual or groups of individuals who will seek benefits from using the product [2]. This includes members such as the project clients, users and investors [42]. The stakeholders and market are identified and recorded through a customer selection matrix, tabulating the individuals that the design team must acquire raw data from. An example of a customer selection matrix from a health project is seen in Table 2.2.

Table 2.2: Example Customer Selection Matrix

	Nurses:	**Network Manager**	**Lab Manager**	**Director**	**Quality Manager**	**Customer Rep.**
Health Center	4	1	1	1	1	1
Insurance Corporation		1		1	1	1

The stakeholders populated in the customer selection matrix serve as individuals from whom which raw data can be collected. Design teams may use interviews as an efficient data collection technique. Interviews require less man hours and generate the same amount of information as focus groups [42,48]. The raw data collected is interpreted into statements that are pertinent to the design project, effectively documenting them as serviceable requirements.

2.2.2. Pahl and Beitz

A different systematic approach, Pahl and Beitz, recommends a guideline for eliciting requirements [1]:

1. Identify the requirements

2. Arrange the requirements in clear order
3. Enter the requirements list on standard forms and circulate among interested departments, licensees, directors, etc.
4. Examine objections and amendments and, if necessary, incorporate them in the requirement list.

In identifying the requirements, the designer must check all available information regarding the retrieval of customer data, including customer contracts or sales documents. Within those, technical documentation may be found which relates to the system requirements. Further, the designer may have to determine their own set of requirements, in which they seek stakeholders to collect this data from. In other instances, the designer may develop scenarios that consider the different uses of the product, under different environments by varying users within the market segment. In developing the requirements, the designer must specify if a requirement is a demand (constraint) or wish (criterion).

The second step requires arranging the requirements in clear order. This order assists the designer in identifying the major objective of each set of requirements and may be clustered by subsystem or assembly. The requirements are then inserted into a standard document which can be viewed by all stakeholders involved. Alongside the guidelines, Pahl and Beitz provide an example set of requirement classifications to assist the designer when eliciting requirements. This classification, seen in Figure 2.2, helps designers seek out specific information regarding the system to assist them in identifying requirements.

Main Headings	Examples
Geometry	Size, height, breadth, length, diameter, space requirement, number arrangment connection, extension
Kinematic	Type of motion, direction of motion, velocity, acceleration
Force	Direction of force, magnitude of force, frequency, weightk, load, deformation, stiffness, elasticity, stability, resonance
Energy	Output, efficiency, loss, friction, ventilation, state, pressure, temperature, heating, cooling, supply, storage, capacity, conversion.
Material	Physical and chemical properties of the initial and final product, auxiliary materials, prescribed materials (food regulations etc.)
Signals	Direct safety principles, protective systems, operational, operator and environmental safety
Safety	Direct safety principles, protective systems, operational, operator and environmental safety
Ergonomics	Man-machine relationship, type of operation, clearness of layout, lighting aesthetics.
Production	Facory limitations, maximum possible dimensions, preferred production methods, means of production, achievable quality and tolerances
Quality Control	Possibilities fo testing and measuring, application of special regulations and standards
Assembly	Special regulations, installation, siting, foundations.
Transport	Limitations due to lifting gear, clearance, means of transport (height and weight), nature and conditions of despatch.
Operation	Quietness, wear, special uses, marketing area, destination (for example, sulphurous atmosphere, tropical conditions).
Maintenance	Servicing intervals (if any), inspectin, exchange and repair painting, cleaning
Recycling	Reue, reporocessing, waste disposal, stoarge.
Costs	Maximum permissible manufacturing costs, cost of tooling, investment and depreciation.
Schedules	End date of development, project planning and control, delivery date.

Figure 2.2: Pahl and Beitz Requirement Types [1]

Pahl and Beitz and Ulrich and Eppinger are not the only design processes which prescribe a means of eliciting customer requirements. There are multiple other techniques from varying disciplines used to obtain requirements from customers; and selecting the suitable techniques according to the characteristics of the project is important [49]. In many instances, an integrated approach for eliciting requirements may be needed based on those working on the design team [42,50]. The development of requirements specifications is conceived as an incremental process, in which the stakeholders successively add requirements [51].

2.2.3. Elicitation Process

Designers are not expected to be familiar with the pool of users and products they design for, nor are they expected to have experience using every product they design [52]. It is well known researchers have linked the problem with insufficient requirements to poor communication among designers and users [53]. In cases where the designer lacks familiarity with the end user, a persona may be used. A persona is a user whose goals and needs are representative of a particular group of people [54]. A persona does not provide requirements for a system; rather a persona is a source for which designers can extract useful raw data [55]. A persona description is not a list of tasks that a particular person completes. Personas are a narrative that describes the flow of a potential end user's day.

The takeaways of this section are shown in Table 2.3, where the importance of the requirements in ensuring a satisfactory project is noted. The requirements document will describe the needs of the stakeholder, and if this document is not managed correctly, accurate maintenance and satisfaction of their needs may be jeopardized. A major challenge of system requirements is identifying and clarifying them in a volatile and uncertain environment [56].

Table 2.3: Takeaways of Requirement Elicitation and Recording Research

Take Away	Ref
Gathers requirements the product must satisfy	[2,6,42,45,11,49]
Describe stakeholder needs	[2,42,44,45,11,49]
Define boundaries of the problem	[2,46]

2.3. Change

Research within the field of requirement change assists in understanding how requirements and their change shape the design process [57]. It has been shown that more than half of a system's requirements will change before completion [58,59], and at such high frequency, this change must be properly managed. Depending on the type of project, requirements may change internal or external to a project [18,60,61]. For example, an internal requirement change may occur when a subsystem requires greater packaging space and results in a design change, which subsequently leads to a change in requirements. An external change may occur when government regulations change on a vehicle safety standard. Changes may be initiated by an engineering redesign, the customer's ever changing needs, competition, or the need for internal improvement [62,63]. Additionally, changes in the understanding of the problem or internal effects such as budget considerations can also cause requirement changes [46]. In many instances, decisions are made due to requirement changes rooting from, but not limited to, technology, trends, perceptions, and regulations [64,65].

Requirement change has not gone unnoticed as product development paradigms exist and operate on the concept of expecting requirements to change at a rapid rate [64,12,66] as they explicitly address and recognize the volatile dynamic evolution of requirements. While the occurrence of requirement change has been recognized, the managing and modeling of this change has not been thoroughly researched [15,27,26]. Great difficulty is involved with managing requirements change, specifically with how this change can be modeled due to their evolving and dynamic nature and the many elements of a requirement that could be modeled.

Table 2.4 notes the takeaway of this section. Requirements change can have detrimental monetary affects on the design process if not managed correctly. This is due to the occurrence of change that may come both internally and externally. Though it has been recognized as an area of need, minimal research has been performed in the field of requirement change. However, it is not the change that can be detrimental to design; it is the unmanaged and unanticipated change propagation that may occur.

Table 2.4: Takeaways of Requirement Change Research

Take Away	Ref
Requirement change does not always affect elements of a design	[7]
Consequence of unmanaged change can be costly	[7,67]
Requirement change is needed to adapt to external and internal changes	[18]
Minimal research has been performed in the realm of change through requirement propagation	[68,60]

2.4. Change Propagation

In the design of complex systems, typically incorporating multiple diverse individuals, a change may result in propagating uncontrolled dynamic changes to occur [69]. Change propagation is a process in which a change to one element of a design results in additional changes either within or different parts of the design when otherwise this change would not have been required [60,68] . In many instances, the change initiator is not aware of the propagation consequence of the change [7]. Change propagation research stems from studies performed in change management, engineering design, product development, complexity, graph theory, and design for flexibility [70,71]; however, few make to use requirements as a means for managing change. Of those which do make use of requirements for managing change, it is primarily for software systems.

Relevant methods for predicting change propagation in design have appeared, primarily in the field of software engineering [72,73]. These methods decompose a program into pieces that are then linked in a propagation graph. The research of this dissertation uses requirement relationships as a means of linking requirements to one another to predict propagation occurrence. Within mechanical design, these pieces might be subsystems or specific components. However, this method of breaking down a system is not detailed as one component or "piece" may consist of dozens or hundreds of requirements. Such requirements may relate to a component's features or appearance, hence providing a finer resolution for the specific change propagation that may occur. Nonetheless, the technique of decomposing a system highlights where subsequent, immediate changes might be necessary, presupposing that the subsystems are generally independent. In software redesign programming variables are relied upon to indicate changes and is not appropriate for mechanical design where the parametric links between parts may be less explicit or the prediction of change involves more than one step [18]. Furthermore, predicting change in complex systems such as automobiles is difficult as the consequences of change are often hard to predict, especially when device subsystems cross boundaries and are highly interrelated [25].

Requirement changes are one of the reasons for engineering changes (ECs) where ECs are defined in multiple ways by different researchers. However, in this research the following definition is used [74]:

> *"An engineering change is an alteration made to parts, from embodiment design stage to production stage of the product life cycle, in its form or fit or function, drawing or software that has already been released. The change can be of any size or type, can involve any number of people, and can take any length of time."*

These changes occur when companies request changes to products, documents, components, manufactured or purchased parts, processes, or even supplies [20]. Hence, research related to engineering change has been reviewed for different change propagation models and tools making use of the models. The majority of the work performed in engineering change has been to define and characterize engineering change propagation [68], but none in the form of requirements. Alongside the greater resolution of requirements, predicting change propagation in the requirements domain allows the use of the tool at virtually any stage of the design process; whereas component based models require a physical architecture to be defined.

Shown in Table 2.5 are the takeaways of this section. It is recognized that change propagation occurs because of a change to one element resulting in subsequent changes which otherwise would not have occurred. This becomes harmful when this propagating change is not anticipated. As a result, a means for managing change propagation is needed; however most of the work done is in the field of software systems. In electromechanical systems, research has been performed in managing change propagation in the component domain; however none have used to the requirements domain. The requirements domain allows for the use of the tool at early conceptual phases when system architecture has yet to be developed. The advantage of the component domain is the existing models which can be used, whereas the modeling of requirements is limited.

Table 2.5: Takeaways of Requirement Change Propagation Research

Take Away	Ref
Change Propagation is a process in which a change to one element results in subsequent changes	[68,60,71]
Change initiator is not always aware of consequences	[7]
Change propagation is prominent in the field of software engineering, not mechanical design.	[18,72,73]
Change propagation research stems from change management, engineering design, product development, complexity and graph theory.	[75,70,76]
Predicting change in complex mechanical systems is difficult	[25]

2.5. Modeling

The ability to model information, such as requirements, allows for analyzing data in a manner not possible in unmodeled form. Requirement's modeling is a means for making it easier to capture, communicate, track, analyze, verify, validate, view, and manage the hundreds of hierarchical and interrelated engineering requirements necessary for large and/or complex systems [37]. Modeling requirements can significantly reduce the cost of systems development and the probability and severity of cost and schedule growth. Note that the term "model", with respect to requirements modeling, does not imply a simulation model, it implies to a means to manage requirements through a computational, graphical representation outside of the traditional textual form.

Examples of "partial" requirements models are behavioral hardware, algorithm simulations such as Mathworks and MATLAB, and block diagrams [35]. Others models have targeted requirements management such as IBM Doors and MagicDraw SysML [31]. Both can model requirements, however these models are not developed for change propagation reasoning. Though elements of a requirement may be linked using requirements management programs, the user is not able to define relaters nor is it automated. For instance, requirements relating to a mechanical pump could be modeled to capture many elements of the requirements: functional, architectural, energy, geometry, and interface. All these elements may serve as relaters between other requirements. These elements may be important and relevant to the designer at various points during the design process. Because a requirement is meant to be a textual description of what the system must do, a requirements model is the only place where all the models necessary for a system's development can be represented together [35].

Although existing approaches propose different representations in modeling customer requirements, constraints, design tasks, design intent, design goals and objectives to better understand a design problem, few can provide it through use of a systematic and integrated framework [6]. Nonetheless, model based reasoning has been used to generate and evaluate design changes and their accompanying side effects [25]. In engineering design, requirements may be modeled using a Quality Function Deployment (QFD) management approach [77,78]. The HoQ is a conceptual map that relates the requirements, design, and manufacturing information through the following sequence [77,78,63]:

1. customer requirements → engineering requirements,
2. engineering requirements → part characteristics,
3. part characteristics → key process operations
4. key process operations → manufacturing requirements.

Though the HOQ may serves as a customer requirement elicitation tool, it primarily serves as a managerial tool, mapping the strength of the relationship between customer requirements and engineering requirements. The sequence of domains used in the HoQ is shown in Figure 2.3.

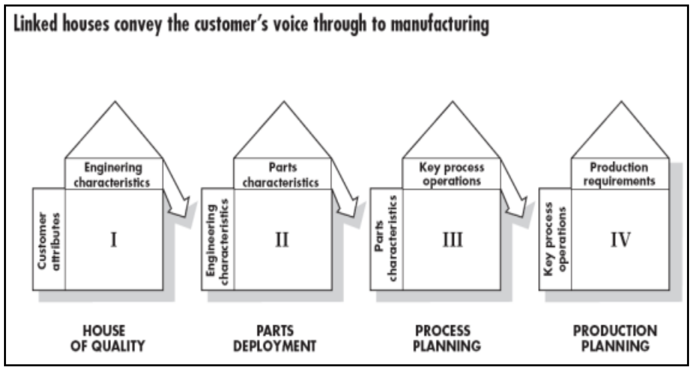


Figure 2.3: House of Quality Modeling Scheme[1]

The takeaways of requirements modeling is shown in Table 2.6. Modeling allows for the development of representations beyond that of the basic textual format. Further, with models, analysis can be performed using different foundations such as graph theoretic. Modeling is already performed in many design activities and including it to requirements could open an avenue for requirements analysis. Requirements have been partially modeled through the use of blocks and diagrams, based on modeling languages, but none of the models are used to explicitly address change propagation.

Table 2.6: Takeaways of Requirements Modeling Research

Take Away	Ref
Modeling is a necessary part of any design process or activity	[37],
Requirements may be partially, in blocks, modeling through use of language models and algorithms.	[6,35]
Requirements, as a whole, may be modeled through requirements management tools	[31]

CHAPTER 3.
RESEARCH EXECUTION

This dissertation focuses on understanding requirement change within a requirements document to support the development of a requirement change propagation prediction tool. In developing this tool, different types of requirement relationships are be analyzed to determine the relationships most conducive to predicting requirement change propagation. All relationship permutations will be assessed to determine which relationships, or combination of, could be used to most accurately and consistently form predictable propagation. The relationships are filtered to identify the relationships which most accurately predict the change propagation. To add differentiation to the results, a scoring system is developed to identify, of the selected requirements, which have the highest potential for change.

This tool is developed through two industry studies, Toho and Pierburg, and validated against a third, EVRAZ. Multiple case studies were performed on the initial studies which contributed to the automated syntactical tool.

A requirements list is developed detailing what the system must do to operate as a sufficient tool for designers and engineers. Further, the research questions and hypothesis of the research are presented and the tasks used to address the questions are tabulated. The research approach is presented and all the completed studies are graphically represented.

3.1. Objective

The objective of this research is to develop a computational reasoning tool able to built relationships between requirements that could be used to predict change propagation. This research investigates and develops a requirement change propagation prediction tool to investigate how a change to one requirement may affect related requirements. A requirement change propagation tool is developed to support designers and engineers throughout the dynamic design process. This is instrumental to any design due to the change requirements experience, independent of the type of change, through the life span of the project. This allows designers to model requirement changes over time that, traditionally, could not be viewed through a static requirements list and goes beyond any requirements management software available.

In completing this research, the objectives satisfied are shown in Table 1.1. The objectives are to: (1) identify types of requirement change, (2) survey change propagation techniques, (3) survey requirements modeling tool, (4) explore the use of requirements for change propagation prediction, and (5) develop an automated requirement change propagation prediction tool making use of the syntactical structure of the requirement. Each research question presented addresses at least one of the objectives. This research will start with the appropriate research on design change, specifically within the domain of requirements and their change. Nine studies are completed to evaluate and support using requirements to predict change propagation. This research will investigate the phenomenon of change throughout requirements evolution and develop a computational representation and reasoning tool for meaningfully modeling such change so it may be used to assist designers predict change propagation through the design process.

The most critical factor in this research is how requirement relationships are developed. Such relationships will control how requirements are related to one another, which in turn affects its propagation characteristics. In studying requirement change, different requirement relationship permutations will be analyzed and assessed for their predictive ability.

The tool developed must satisfy the requirements elicited in Table 3.1 The tool must be capable of supporting multiple types of requirement format. This includes requirements written in simple single statement form or requirement specifications, detailed in paragraph form. This is of importance as the tool must be robust to function in multiple design environments and uses. The parsing of each requirement's POS is not in the scope of this tool. The parser used here, as explained in Section 5.2.1 is the Stanford natural language parser. This tool must be capable of using the POS tagging output of the parser to develop relationships between the requirements. The requirement relationships developed through the POS tags will be used to develop a DSM of requirement relationships that can subsequently be used to identify change propagation. This propagation will be identified through multiple change orders, as described in Section 5.4. To granulate the results of the potential requirement propagated, a weighting or ranking system is used. The purpose of this is two folds: (1) to filter the potentially propagated requirements into a narrow, more evaluable list and (2) develop separation between requirements so designers and engineers know which requirements to review first, in ranking order.

Table 3.1: Change Propagation Prediction Tool Requirements

Tool Requirements	Description	Justification
Must be capable of supporting all requirements documents formats.	Tool must be capable in inputting, storing and updating all requirements within	To support varying types of projects: industry, academic, and industry sponsored.
Must develop relationships between requirements	Make use of syntactical POS (parsed externally) of each requirement to develop the relationships needed to predict change propagation	This is needed so requirements which are related may serve as propagation sources or intermediate requirements for propagation
Must be able to propagate change into multiple orders.	Tool must use relationships to develop propagation on multiple orders.	Using the relationships developed, the tools will highlight potential critical requirements that my change. This will be performed at multiple orders. User will select the numbers of orders for propagation.
Must incorporate weightings within potential requirements changed.	After developing model of propagation, tool must incorporate post propagation weightings to identify critical potential requirements change.	As propagation is analyzed, weightings are needed to narrow selection of potentially changed requirements down to a set of key requirements. This will improve the tool's ability to accurately highlight high potential requirements.
Rank critical requirements based on scoring.	Using the requirement relationships and weightings, high potential requirements will be selected that have the highest potential of subsequent change, in ranking order.	High potential requirements are needed to assist the designer in managing the requirement change. Potentially propagated requirements will be scored and ranked.

3.2. Research Scope

In relating the research questions, this research will present requirement change at the local and global level. Localized requirement changes are defined as those changes that occur to the specific syntactical structure of a requirement. Globalized requirement changes are defined as those changes that occur to a requirement document, propagating from a localized change. While both are important in their own respect for their ability to identify propagation at respective scale, this research finds greater contribution in global requirement change propagation. As a result, this dissertation will focus on requirement change propagation while including a survey of localized requirement change, shown in APPENDIX A and reported in [79]. As a result, the term requirements change will refer to global requirement changes in this dissertation.

The core contribution of this dissertation is the development of an automated requirement change propagation prediction tool, which makes use of a single or set of requirement change(s) to propagate forward to other requirements which may change. This research is carried out by performing multiple cast studies which eventually lead to the development of the automated syntactical tool.

3.3. Research Questions & Hypothesis

Each research question will be associated with a hypothesis based on preliminary research that was performed and tested against subsequent studies. A validation approach is listed with each research question to address how each question will be answered in this dissertation. Further, an objective ID from Table 1.1 will be provided to relate each question to the proposed research objective and deliverables.

Table 3.2: Research Questions

I	Question	Can requirements be used to predict change propagation?
	Hypothesis	Syntactical elements within a requirement may be used to relate requirements in a manner that can predict change propagation.
	Validation Approach	Through industry case studies of and case study analysis viewing industry change notifications, change prediction through use of requirements will be evaluated.
	Objective ID	A, B, C (see Table 1.1)
II	Question	What types of relationships between requirements predict change propagation?
	Hypothesis	Relationships between requirements exist within the subject or function of the requirement. Additionally, keywords within the individual requirement may be used to relationships. Subjects and keywords may be extracted through POS tagging. These relationships can be used to predict change propagation.
	Validation Approach	Develop several propagation models for the same requirement change through different relationships to analyze results.
	Objective ID	D
III	Question	Can the propagation tool be tuned to narrow the selection of potential requirement for propagation?
	Hypothesis	Weightings, rankings, or factors can be used both before and after propagation to target critical potential requirement change propagated
	Validation Approach	Using similar change scenarios with different weighting, ranking or factors applied both before and after propagation. Results will be analyzed to view if tool yielded propagation at a greater accuracy.
	Objective ID	E
IV	Question	Can a scoring system be used so potentially propagated requirements are ranked in order of review?
	Hypothesis (null)	A scoring system can be used to sort requirements in order of greatest likelihood of change.
	Validation Approach	Perform scoring algorithm to each requirement change study to identify if scoring system can highly rank requirements which change.
	Objective ID	E

3.4. Tasks

A list of tasks is developed to ensure each research question is addressed through specific activities detailed in this dissertation. The task list, shown in Table 3.3, details specific tasks needed to address each research question.

Table 3.3: Research Questions and Tasks

Task		Task Description
I	Task I.a	Develop a taxonomy through Survey and Review of requirement changes
	Task I.b	Industry and academic case study evaluating if all changes can be identified through developed taxonomy
	Task I.c	Develop syntactical representation of all requirement change covering taxonomy of changes
II	Task II.a (*iterative*)	Explore permutations of types of requirement relationships
	Task II.b	Compare relationships and combinations to determine most accurate predictor of propagation (retrospective case studies)
	Task II.c	Validate prediction tool through a historical virgin case study
III	Task III.a (*iterative*)	Explore permutations of propagation weightings, factors or rankings
	Task III.b	Intersect different relationship from task II.b with weightings, ranking, or factors identified in task III.a
	Task III.c	Cross case analysis to applicability scope of using relationship types and weightings, factors or rankings
IV	Task IV.a	Develop scoring system to incorporate important propagation characteristics
	Task IV.b	Analyzing scoring system ranking to determine depth of requirement review needed to find requirements propagated

3.5. Research Approach

To answer the research questions and test the associated hypotheses, three basic research phases will be executed: (1) initial evaluation of using requirements to predict change propagation through manually selected requirement relationships, (2) development of an syntactical tool for capturing, reasoning and predicting requirement change; and (3) development of a scoring algorithm to granulate results of propagation and provide ranking of most requirements with greatest likelihood of propagated change. The phases will be performed through the use of three heterogeneous industrial case studies with Automation Engineering Corporation (AEC).

This dissertation will make use of three varying industry projects with requirements elicited by different corporations and given to AEC, as seen in Table 3.4. The three projects totaled 560 requirements and seventeen engineering changes where nine are predicted (ten requirement change propagations). Two of the projects, Toho and Pierburg, make use of two approaches in predicting the requirements propagation, a manual and automated syntactical approach. As discussed in Section 6.1, a case study research approach is sought out to perform these studies.

Table 3.4: Industry Projects Summary

Project	**Approach**	**Requirements**	**Requirements Predicted**
Toho	Manual – Subject	160	2
Pierburg	Manual – Keyword	214	3
EVRAZ	Automated Syntactical	186	5
3 Projects	**Three Approaches**	**560**	**10**

3.6. Studies Completed

In completing this research, multiple studies were performed. Some studies, such as the statistical analysis, were used to validate and support other studies. The studies are shown in Figure 3.1. As proposed during the research proposal of this research, a localized requirement change survey is complete and provided in the appendix. Studies on localized requirement change is not in the body of the paper as it is not the core contribution of this research, rather it supports future work. A statistical analysis is performed on the capacity to use requirement syntax words to propagate change propagation. Two relationship creation approaches are used: (1) manual and (2) syntactical. The manual approach has two relationship types: (1) subject and (2) keywords. Both relationship types are studies against an industry project and the keyword relationship type possess two additional studies to evaluate keyword selection and consistency. The syntactical approach will be studies against three industry projects and a statistical analysis will be performed on the results as well. The section each of the studies is completed in is noted within the table. In total, nine studies are performed and presented in this research dissertation to address and support the research questions.

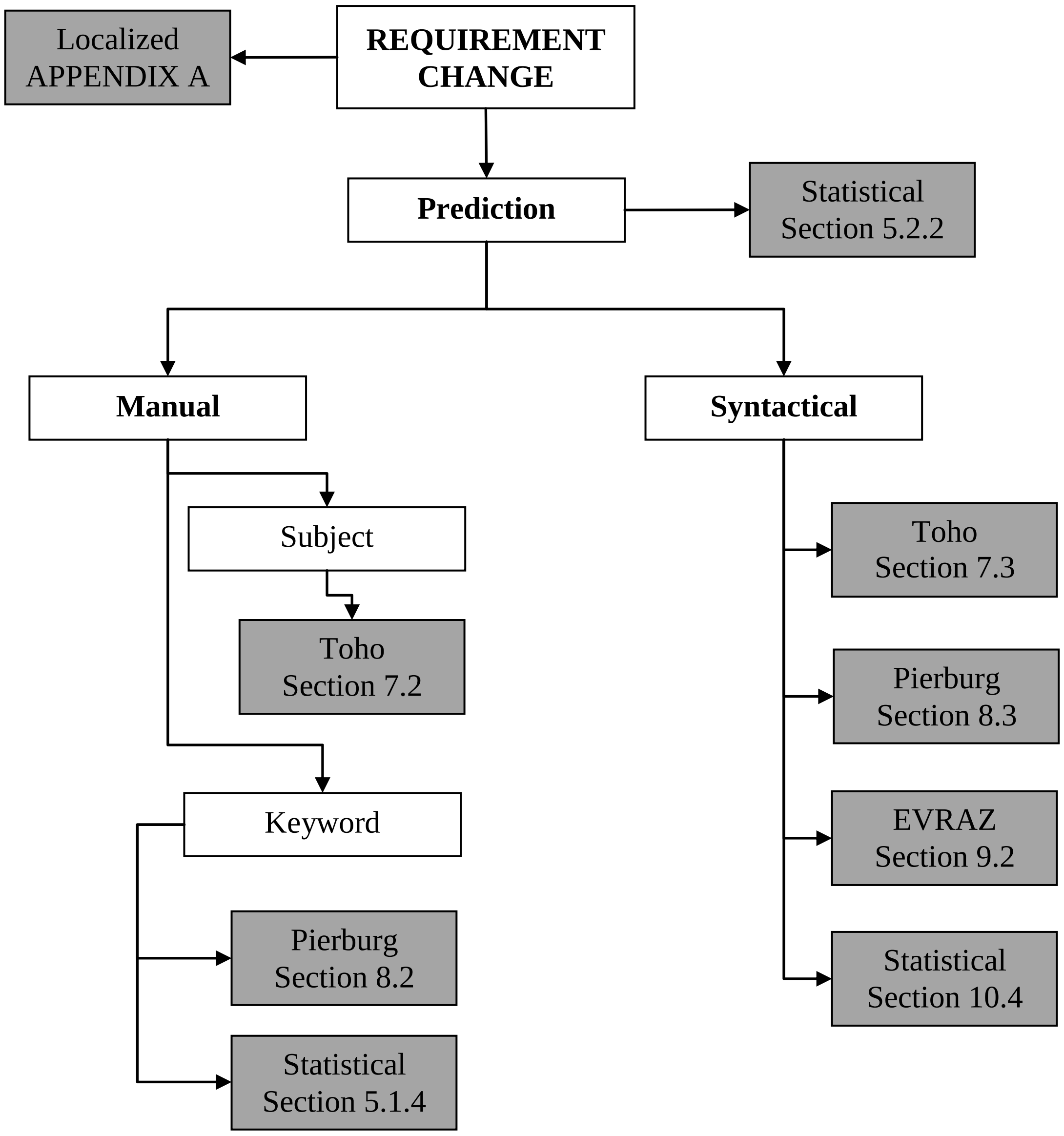


Figure 3.1: Studies Completed in Research

CHAPTER 4.
REQUIREMENT CHANGES

Requirement changes are those changes which occur because of an initial requirement change that propagates to other requirements. This change may be defined as subsequent changes which otherwise would not have changed. It is important to note that not all change is due to a previous change. Some change may be external in nature and independent of all previous changes. The change investigated in this study is that which occurs, unexpectedly, because of previous changes. This type of change propagation may occur through two mechanisms: (1) single requirement propagation or (2) cumulative propagation. Requirement propagation occurs when a change in a single requirement, due to its relation with other requirements, causes unforeseen propagation. This research has identified this to be the more popular of the change propagations. An example of this is when a change to the braking system spatial requirements of a vehicle causes a change in the geometry of the suspension to prevent physical interference. A cumulative change occurs when cumulative changes results in the need to make a change to another requirement. For example, a requirement relating to the maximum occupancy of a vehicle may change from four to five passengers. At a later time, a requirement change may occur to the size of the fuel tank, increasing it from thirteen to sixteen gallon capacity. Cumulatively, both changes result in a need to change the maximum supported weight by the suspension, whereas one of those changes individually would not have warranted a change in suspension weight rating.

4.1. Requirements Management and Modeling Software

Requirements engineering is an important activity in the design process and involves eliciting, modeling, analyzing, and communicating engineering requirements. First, requirements may be generated from multiple perspectives and standpoints which could overlap, complement, or contradict requirements [80,81]. Secondly, requirements may be generated by decentralized and distributed design teams, affecting the ability to seamlessly communicate and share the requirements. Requirements may change throughout the design process and cause other requirements to change as a result. Outside of requirement change, requirements are suitable for modeling because of their management needs. Specifically, there exist three key challenges in requirements engineering: (1) eliciting requirements from stakeholders, (2) modeling requirements into engineering specifications, and (3) analyzing requirements [82]. These issues drive the need for formal software and tools to support requirements management.

The management tools examined here are the Unified Modeling Language (UML), Systems Modeling Language (SysML), and IBM Rational DOORS. The three models were selected due to their different uses in different design environments.

UML is presented here for its ability to model requirements. UML is a modeling language that provides a host of different models which designers can use to represent a system. Primarily used in the software industry, it models aspects of a requirement that traditional requirement sentences and paragraphs cannot. This includes how the system interacts with its environment and the possible scenarios a system may experience. Such models, used alongside requirements can be helpful to designers. It is important for this research to present this modeling language because it formalizes a means for modeling requirements. Though this modeling is not meant for the purpose of change propagation, it is a representation of requirements nonetheless.

SysML is presented to introduce a systems engineering approach to modeling, which builds on the success of UML. A systems engineering modeling language, this tool affords the use of requirements in a host of different graphics, including diagrams and tables. A key feature of SysML is its ability to model requirements in a manner relatable to one another. This is unique in modeling languages as requirements where not modeled to one another before this. This introduces both a unique modeling feature and adds reasoning to the modeling of requirements. Further the modeling language allows for the modeling of requirements with different elements such as components, product specific problems, and validation tests. Using this, designers and engineers can use such a model to illustrate how specific requirements relate to other aspects of the design process in a graphical manner.

The IBM Rational DOORS requirement management tool uses a centralized database for managing engineering requirements. DOORS largely serve as requirements publishing tool, documenting all system requirements in one location. DOORS is widely used in industry, though there exists many other management tools used to model requirements. Many of these tools can be found in the INCOSE survey [39] where they are evaluated based on different metrics. In many cases, their specific purpose is not to model requirements change, they are used to manage a requirements document. As a preliminary study, this research includes the study of IBM Rational DOORS [31]. This section will investigate the modeling tools available for requirements management.

As noted previously, a future work of this research may be incorporating the developed tool as an add-on feature to some of the existing modeling tools which do not have a change propagation prediction mechanism.

4.1.1. Unified Modeling Languages (UML)

Software quality is defined as its ability to meet its system requirements [83]. UML is a graphical language used for visualizing, specifying, constructing, and documenting the artifacts of a software system [84,85,86,87]. UML principals are implemented because it has become a standard language for modeling software requirements and design [88]. UML is widely used throughout the design process of software systems [89]. Within UML, use case, statechart, and sequence diagrams are found to be beneficial to the programmers of software [90]. UML was specifically design for visualizing, constructing and documenting several aspects of a software system [85].

Programmers needed a visual tool alongside the textual requirements document to develop the software. A tool that could reveal scenario descriptions of requirements in terms of interactions between the system and its environment is of great use to programmers [91]. Diagrams such as use case, which model the interaction of the system and its environment, are useful because it provides a representation of the multiple scenarios the system could experience throughout its operational environment. Use case refers to a complete sequence of scenarios in the system, as understood by the user of the system [92]. These diagrams, derived from UML principals, are part of a relatively expressive language that addresses all the views needed to develop a system [84]. This is important to requirements because a successful set of requirements are ones that enable the user to develop an application and interface that meets the needs of its users [48].

UML tools such as use case, statecharts, and sequence diagrams are needed in software development as a modeled representation of requirements [90]. UML encompass three kinds of building blocks with the development of their diagrams: entities, relationships, and diagrams [84]. Entities may include users or adjacent computer systems, while relationships are the connection that ties different entities together. Diagrams are the different forms of representation entities and relationships.

Use case diagrams are typically used to simulate interaction between the user(s) and system during different scenarios [93,94] and are a popular modeling technique within UML [92,95]. Use cases are most useful during the analysis phase of a project to identify system functionality through interaction between the user and system [96]. However, literature stresses one of the early steps in object modeling is to build a use case model [97]. Users here may be any human being such as a customer, associate, or developer. Other users may be devices such as computers, pieces of hardware or secondary software. The user supplies stimuli to that part of the system, and receives outputs from it [96]. The key advantage to use case diagrams and their use with requirements is there are some aspects about a software system that a developer cannot understand unless they build models to supplement the textual programming language

[84]. Using UML tools, designers are able to develop software that satisfies its intended purpose, by meeting and engaging users, to expose the real requirements of the system [84]. Further, use case diagrams identify activities and interactions the system experiences so they may be accommodated [84]. An example of a use case diagram of interactive vehicle sales program is shown in Figure 4.1. As seen in the figure, there is a customer, sales associate, and financial associate. The lines in the figure indicate the relationship between the entities. Differing line types (dashed, thickness, color) may indicate a different type of relationship.

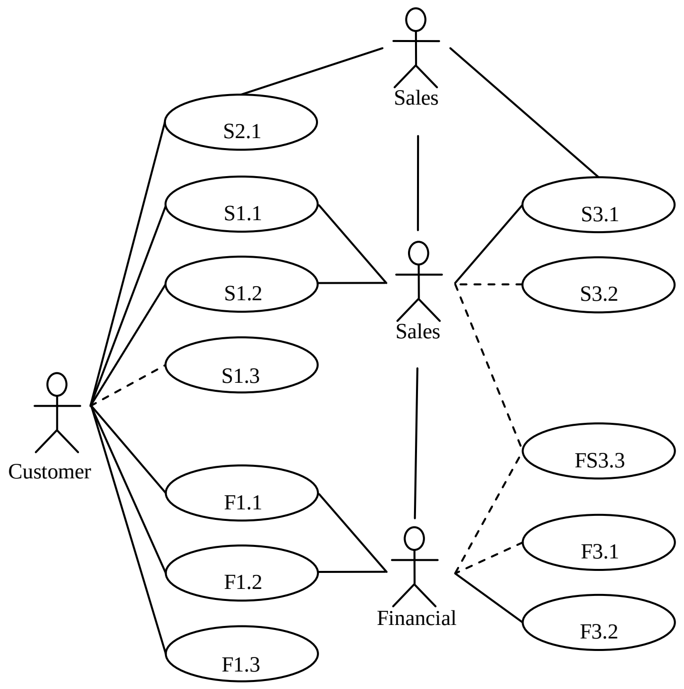


Figure 4.1: Example Use Case Diagram [50]

Using use case diagrams, coupled with the use of a text based requirements specification document, requirements may be developed that are utilized by all constituents within a project development team [42]. Use case diagrams are used to document requirements through the use of visual representation as opposed to a collection of sentences and paragraphs [98]. Alongside use case diagrams, statecharts may be developed to model the behavior of the system. However, use case diagrams are first needed as they are the driver for the rest of the UML diagrams [99].

Statechart diagrams are central to modeling different elements of a system [100,85,101]. Statecharts are a popular visual technique for modeling the behavior of reactive systems [102,103,104]. Statecharts were introduced over twenty years ago as a visualization tool [101,102,105]. A statechart diagram models the event flow of control from one state of a scenario to another as it describes the sequence of events [89]. An example statechart from a design project is shown in Figure 4.2. Statecharts come in many forms, ranging from simple to complex states where each state may have several parts to it and transitions may have multiple affects. Advantages to statecharts are that they are both graphical and textual. The graphical part of the statechart describes the sequence of events while the textual part describes the events and conditions that cause that state change [106].

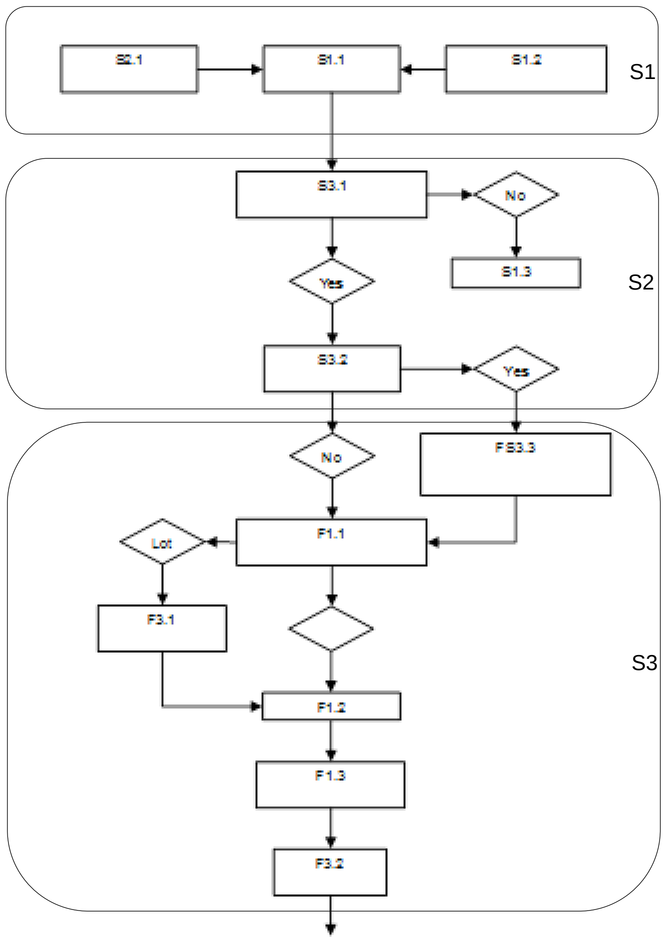


Figure 4.2: Example Statechart [50]

4.1.2. System Modeling Language (SysML)

A limitation of UML is that it is aimed to assist software engineers and is not entirely relevant to systems engineers who work with products outside the software domain [107]. A version of UML suitable for modeling a broad range of systems, including hardware, software, data, personnel, procedures, and facilities was needed [108]. SysML was developed UML in cooperation with the Object Management Group (OMG) and the International Council of Systems Engineering (INCOSE) [109]. SysML was developed to address such limitations and to provide a “standard modeling language for systems engineering to analyze requirements, design, and verify complex systems [108]. In doing so, it was able to bridge much of the gap between software systems and systems engineering. Specifically, the requirements diagrams introduced in SysML can be used as a bridge between requirements engineering and UML use cases [110]. Additionally, its models include parametric models, requirements relationships, causal analysis, verification models, and decision trees. [108]. Some of the more widely used areas in SysML pertain to requirements modeling. SysML provides a means to model and represent textual requirements and relate them to other elements [109]. Within requirements modeling, SysML maintains some of the existing UML diagrams, such as use case and block diagrams and uses them together. [107]. UML is presented first in this background study because those who do not have UML background may find difficulty in learning SysML diagrams, due to their heavily detailed models [111].

Outside of requirements modeling, SysML has also been used as an elicitation tool and requirement template [112]. Requirements modeling in SysML may come in many forms: graphically, tabular, or as a tree structure. Shown in Figure 4.3 is an example requirements diagram for a Hybrid Sports Utility Vehicle (HSUV). The diagram shows the breakdown of the requirements specification document into pertinent requirement groupings, illustrated in a hierarchical manner.

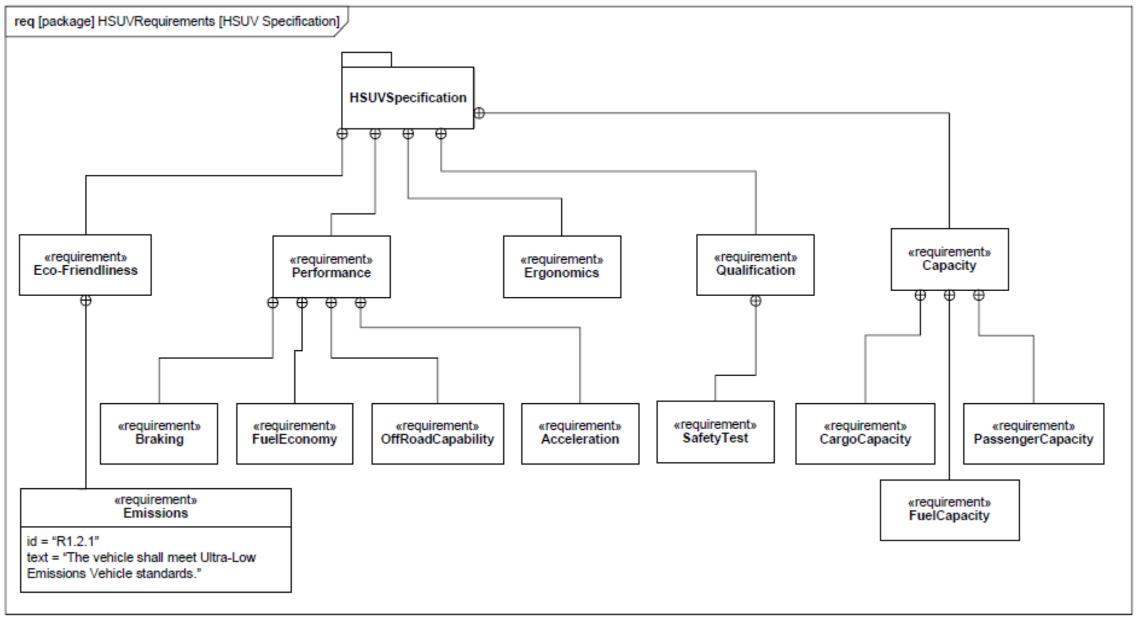

req [package] HSUVRequirements [HSUV Specification]
HSUVSpecification
«requirement» Eco-Friendliness
«requirement» Performance
«requirement» Ergonomics
«requirement» Qualification
«requirement» Capacity
«requirement» Braking
«requirement» FuelEconomy
«requirement» OffRoadCapability
«requirement» Acceleration
«requirement» SafetyTest
«requirement» CargoCapacity
«requirement» PassengerCapacity
«requirement» FuelCapacity
«requirement» Emissions
id = "R1.2.1"
text = "The vehicle shall meet Ultra-Low Emissions Vehicle standards."

Figure 4.3: Example Requirement Diagram [109]

Another type of requirements model presented in SysML is a derived requirements model. Such models illustrated the requirements in a manner specifically related to the product. An example of this is shown in Figure 4.4 where requirements such as fuel economy are related back to product specific problems such as the vehicle power needed for acceleration. Further, this model also allows for relationship between requirements such as the requirement for range and fuel capacity.

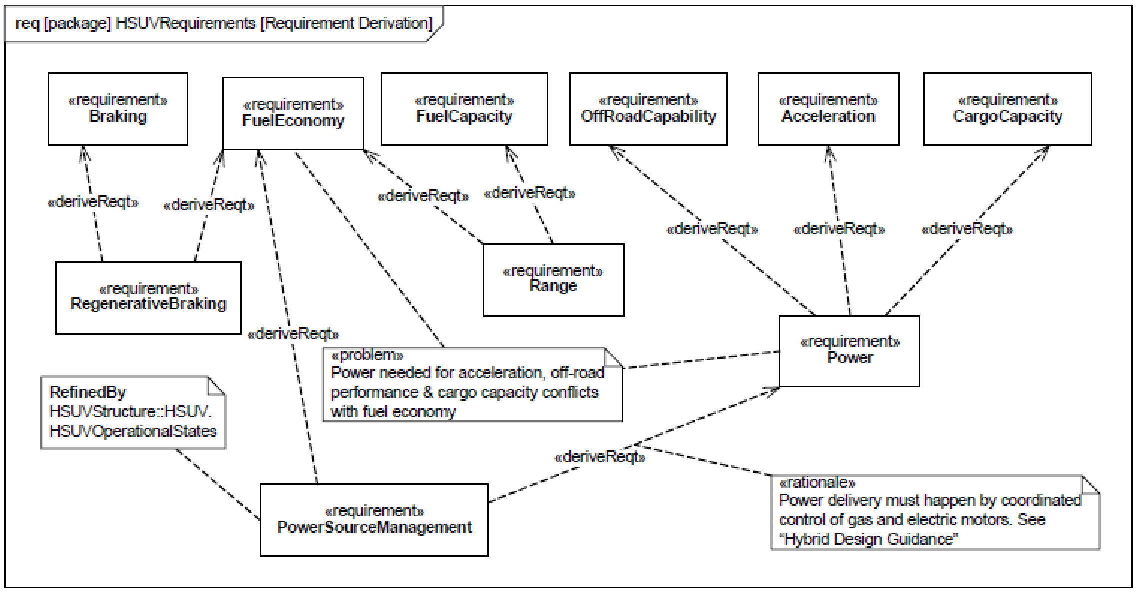

req [package] HSUVRequirements [Requirement Derivation]
«requirement» Braking
«requirement» FuelEconomy
«requirement» FuelCapacity
«requirement» OffRoadCapability
«requirement» Acceleration
«requirement» CargoCapacity
«deriveReqt»
«deriveReqt»
«deriveReqt»
«deriveReqt»
«deriveReqt»
«deriveReqt»
«deriveReqt»
«requirement» RegenerativeBraking
«requirement» Range
«deriveReqt»
«problem» Power needed for acceleration, off-road performance & cargo capacity conflicts with fuel economy
«requirement» Power
RefinedBy HSUVStructure::HSUV. HSUVOperationalStates
«deriveReqt»
«requirement» PowerSourceManagement
«rationale» Power delivery must happen by coordinated control of gas and electric motors. See "Hybrid Design Guidance"

Figure 4.4: Example Derived Requirements Diagram [109]

If a designer is only concerned with, for instance, the acceleration requirement of a system, a diagram can be extracted which illustrates only those elements relating to the acceleration requirement. An example of this is shown in Figure 4.5 where all elements relating to acceleration are captures. This includes other requirements such as those pertaining to power, validation tests such as max acceleration, use cases, and physical subsystems. As seen, such modeling capabilities allows for relating varying elements to one another, a feature not possible before SysML.

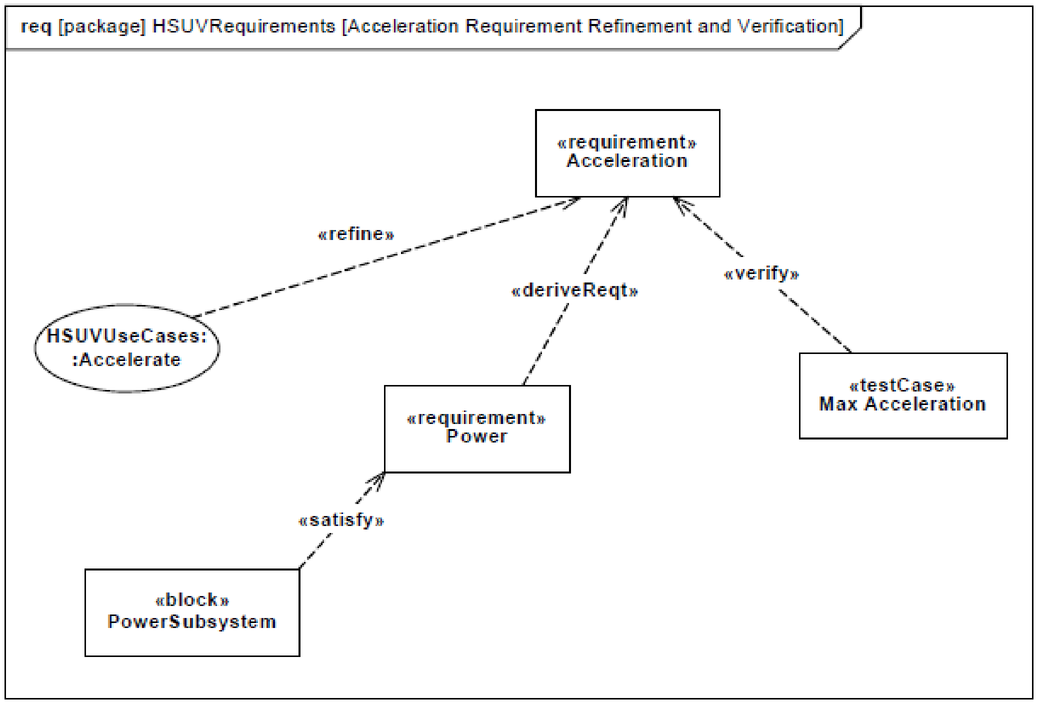

req [package] HSUVRequirements [Acceleration Requirement Refinement and Verification]
«requirement» Acceleration
«refine»
«deriveReqt»
«verify»
HSUVUseCases: :Accelerate
«requirement» Power
«testCase» Max Acceleration
«satisfy»
«block» PowerSubsystem

Figure 4.5: Example of Acceleration Requirement Diagram Extract [109]

4.1.3. <u>IBM Doors Capabilities</u>

IBM Rational DOORS is a Requirements Management tool that allows users to input different sets of requirements within its database and provides users the ability to view specific information traditionally not viewed in requirements specifications documents. Available for multiple users, DOORS allows users to access the database and view specific requirement documents.

Information is placed in DOORS through modules. A module, as seen in Figure 4.6, will list a specific requirement list. In the case below, this module is used to represent the “BMW LED Design” Requirements list. For larger projects, it is beneficial to have multiple modules to accommodate all the requirements needed such as architectural, system, and user requirements, so that they may be classified in their appropriate grouping. Located to the left of the window in Figure 4.6 is the module directory. This is a folder structure that saves different projects under the DOORS Database.

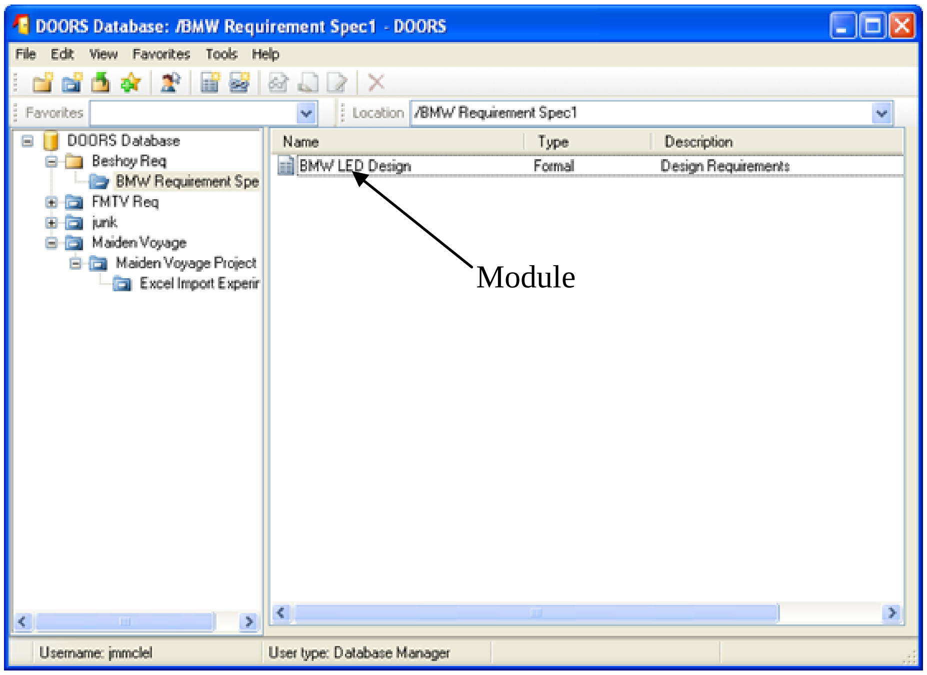
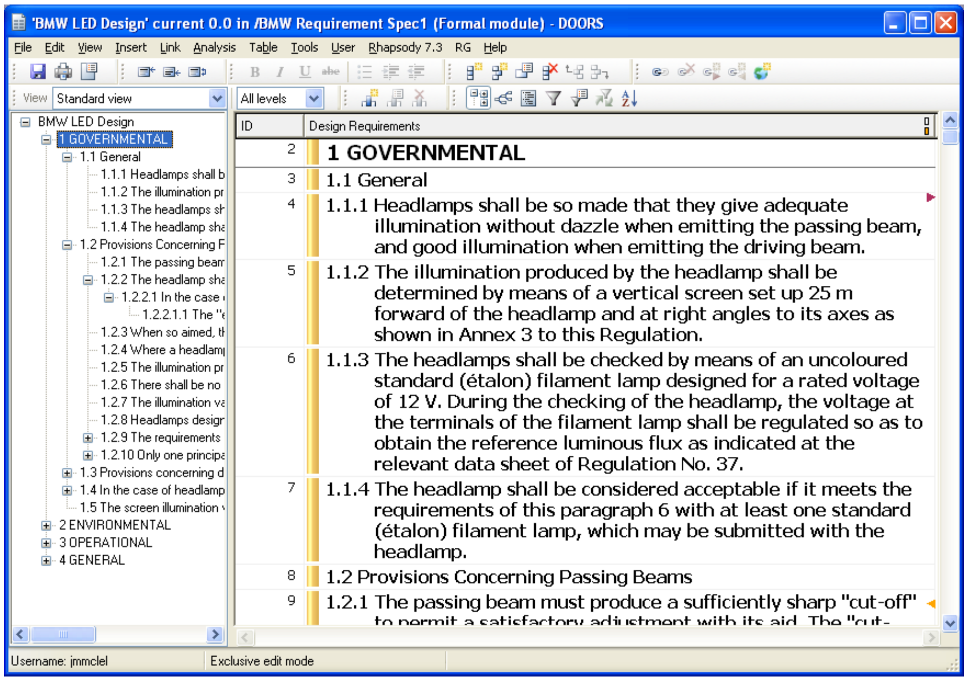


Figure 4.6: Initial Display windowing DOORS Database Directory [113]

Double clicking the module name will open the module. The module, seen in Figure 4.7, shows a list of requirements in a hierarchical manner. This hierarchy is dictated by the user. In this case, the first level or requirements is Governmental. Underneath this are requirements that are "General" and those "Provisions concerning passing beams." The hierarchy may include several other sets of requirements placed above or below other requirements. Also seen in Figure 4.7 is the document map of requirements. This map provides easy navigation to different requirements within the module.


'BMW LED Design' current 0.0 in /BMW Requirement Spec1 (Formal module) - DOORS
File Edit View Insert Link Analysis Table Tools User Rhapsody 7.3 RG Help
View Standard view
All levels
Username: jmmclel
Exclusive edit mode

BMW LED Design
1 GOVERNMENTAL
1.1 General
1.1.1 Headlamps shall b
1.1.2 The illumination pr
1.1.3 The headlamps sh
1.1.4 The headlamp sha
1.2 Provisions Concerning F
1.2.1 The passing bean
1.2.2 The headlamp sha
1.2.2.1 In the case
1.2.2.1.1 The "
1.2.3 When so aimed, tl
1.2.4 Where a headlam
1.2.5 The illumination pr
1.2.6 There shall be no
1.2.7 The illumination va
1.2.8 Headlamps design
1.2.9 The requirements
1.2.10 Only one principa
1.3 Provisions concerning d
1.4 In the case of headlamp
1.5 The screen illumination
2 ENVIRONMENTAL
3 OPERATIONAL
4 GENERAL

ID	Design Requirements
2	1 GOVERNMENTAL
3	1.1 General
4	1.1.1 Headlamps shall be so made that they give adequate illumination without dazzle when emitting the passing beam, and good illumination when emitting the driving beam.
5	1.1.2 The illumination produced by the headlamp shall be determined by means of a vertical screen set up 25 m forward of the headlamp and at right angles to its axes as shown in Annex 3 to this Regulation.
6	1.1.3 The headlamps shall be checked by means of an uncoloured standard (étalon) filament lamp designed for a rated voltage of 12 V. During the checking of the headlamp, the voltage at the terminals of the filament lamp shall be regulated so as to obtain the reference luminous flux as indicated at the relevant data sheet of Regulation No. 37.
7	1.1.4 The headlamp shall be considered acceptable if it meets the requirements of this paragraph 6 with at least one standard (étalon) filament lamp, which may be submitted with the headlamp.
8	1.2 Provisions Concerning Passing Beams
9	1.2.1 The passing beam must produce a sufficiently sharp "cut-off" to permit a satisfactory adjustment with its aid. The "cut-

Figure 4.7: Opened Module [113]

Requirements are entered into the system by inserting an object, where requirements are placed. After inserting an object, double clicking the space provided allows the user to insert a Requirement Heading and Requirement Text. This can also be performed by right clicking each object space and entering the data through the properties command. An example of this is shown in Figure 4.8. Within this property, the user may insert a Requirement: Heading, Short Text, and Object Text. The heading is the main title of the requirement. This is important for hierarchy based requirements documentation as this could serve as a title; the object is simply used as a heading. This is seen in object 1, 1.1 or 1.2 in Figure 4.7.

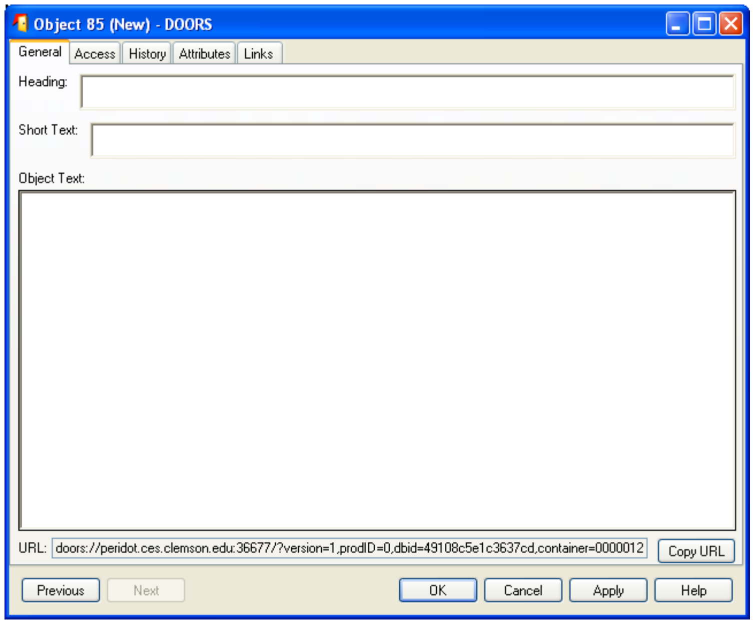
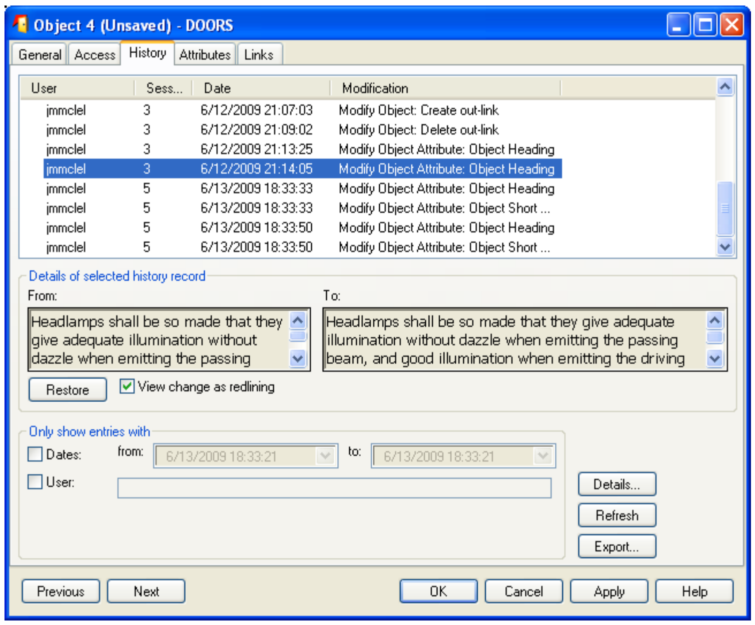


Figure 4.8: Object Properties [113]

The Access tab provides the user with information as to who is allowed to view the object. The history tab provides information as to the history of the object. An example of the information contained in the history tab is shown in Figure 4.9. This shows all the changes that have occurred with the object. This information details the user of the change, the date of occurrence, the type of modification and the details of the modification. The details of the modification reveal the requirement before and after the change. The attributes tab reveals information regarding the object's level of hierarchy, date it was created, who created it, last date it was modified, who was last to modify it, the text within the object and the object number. The final tab lists any linking the requirement may have. Linking will be described in a later section.



User	Sess...	Date	Modification
jmmclel	3	6/12/2009 21:07:03	Modify Object: Create out-link
jmmclel	3	6/12/2009 21:09:02	Modify Object: Delete out-link
jmmclel	3	6/12/2009 21:13:25	Modify Object Attribute: Object Heading
jmmclel	3	6/12/2009 21:14:05	Modify Object Attribute: Object Heading
jmmclel	5	6/13/2009 18:33:33	Modify Object Attribute: Object Heading
jmmclel	5	6/13/2009 18:33:33	Modify Object Attribute: Object Short ...
jmmclel	5	6/13/2009 18:33:50	Modify Object Attribute: Object Heading
jmmclel	5	6/13/2009 18:33:50	Modify Object Attribute: Object Short ...

Details of selected history record

From: Headlamps shall be so made that they give adequate illumination without dazzle when emitting the passing

To: Headlamps shall be so made that they give adequate illumination without dazzle when emitting the passing beam, and good illumination when emitting the driving

[✓] View change as redlining

Only show entries with

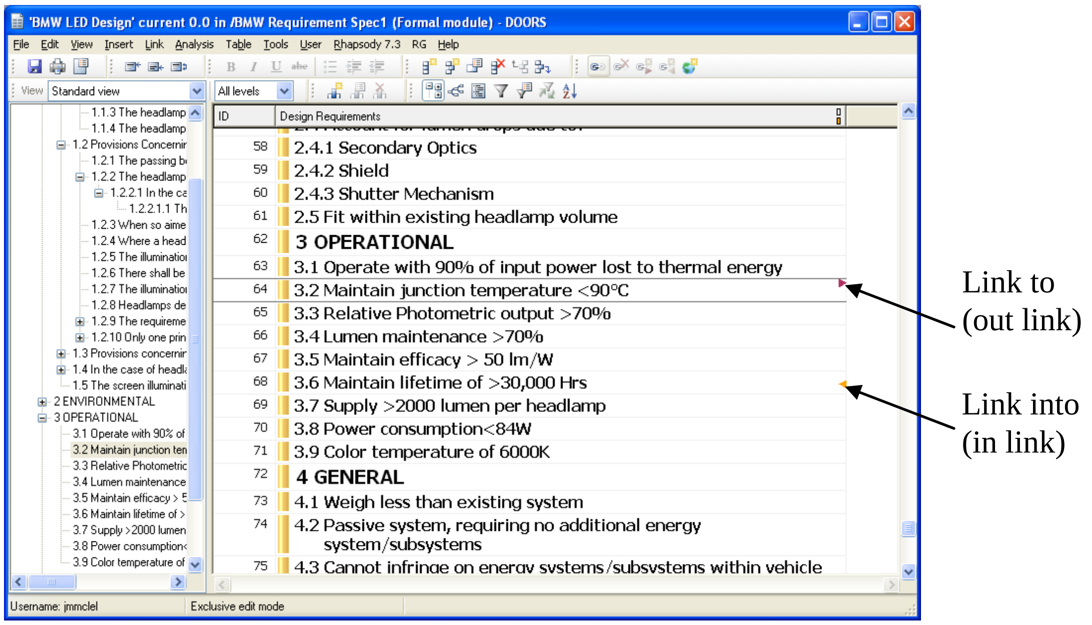
[ ] Dates: from: 6/13/2009 18:33:21 to: 6/13/2009 18:33:21

[ ] User:

Figure 4.9: History of an Object [113]

As discussed in the section above, there may be instances when different requirement lists are developed and inserted within different modules. While distinct, many requirements within may possess a relationship with a requirement in a different module. In order to indicate this relationship, a link is used to designate a push or pull relationship from one requirement to the other. For instance, as seen in Figure 4.10, the arrowheads located to the right of each object indicate a linking. This linking is denoted as linking from a requirement, arrowhead pointing to the right, and linking to a requirement, arrowhead to the left. In the example in Figure 4.10, maintaining a lifetime over 30,000 hours will depend on if the requirement for maintaining a junction temperature of less than 90°C can be met.

In the linking example shown, the links are within the same module. Linking can occur between or within modules. To view the linking of a requirement, the user may view the properties windows and view the links tab, as described in the section above. Right clicking on the arrow will also provide the user with information to all the links associated with a file. Linking provides the user many benefits. For example, if a change occurs to a linked requirement, the associated linked requirements are flagged. In the example shown in Figure 4.10, if we change object 3.2 from "Maintain junction temperature <90C" to "Maintain junction temperature <75 C" a warning for the change in the linked requirement should occur. This is in fact the case as seen in Figure 4.11. A change indicator appears flagging a change in a linked requirement.

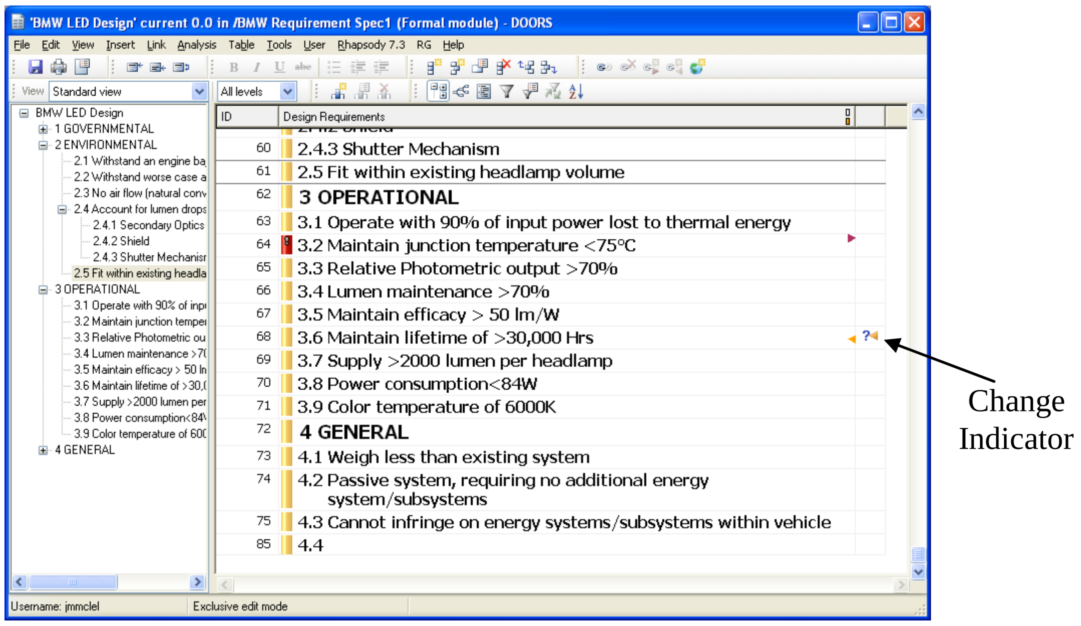

'BMW LED Design' current 0.0 in /BMW Requirement Spec1 (Formal module) - DOORS
File Edit View Insert Link Analysis Table Tools User Rhapsody 7.3 RG Help
View Standard view
All levels
Link to (out link)
Link into (in link)

```
1.1.3 The headlamp
1.1.4 The headlamp
1.2 Provisions Concernir
  1.2.1 The passing b
  1.2.2 The headlamp
    1.2.2.1 In the ca
      1.2.2.1.1 Th
  1.2.3 When so aime
  1.2.4 Where a head
  1.2.5 The illuminatio
  1.2.6 There shall be
  1.2.7 The illuminatio
  1.2.8 Headlamps de
  1.2.9 The requireme
  1.2.10 Only one prin
1.3 Provisions concernir
1.4 In the case of headl
1.5 The screen illuminati
2 ENVIRONMENTAL
3 OPERATIONAL
  3.1 Operate with 90% of
  3.2 Maintain junction ten
  3.3 Relative Photometric
  3.4 Lumen maintenance
  3.5 Maintain efficacy > 5
  3.6 Maintain lifetime of >
  3.7 Supply >2000 lumen
  3.8 Power consumption<
  3.9 Color temperature of
```

ID	Design Requirements
58	2.4.1 Secondary Optics
59	2.4.2 Shield
60	2.4.3 Shutter Mechanism
61	2.5 Fit within existing headlamp volume
62	**3 OPERATIONAL**
63	3.1 Operate with 90% of input power lost to thermal energy
64	3.2 Maintain junction temperature <90°C
65	3.3 Relative Photometric output >70%
66	3.4 Lumen maintenance >70%
67	3.5 Maintain efficacy > 50 lm/W
68	3.6 Maintain lifetime of >30,000 Hrs
69	3.7 Supply >2000 lumen per headlamp
70	3.8 Power consumption<84W
71	3.9 Color temperature of 6000K
72	**4 GENERAL**
73	4.1 Weigh less than existing system
74	4.2 Passive system, requiring no additional energy system/subsystems
75	4.3 Cannot infringe on energy systems/subsystems within vehicle

Username: jmmclel   Exclusive edit mode

Figure 4.10: Linking [113]


'BMW LED Design' current 0.0 in /BMW Requirement Spec1 (Formal module) - DOORS
File Edit View Insert Link Analysis Table Tools User Rhapsody 7.3 RG Help
View Standard view
All levels
Change Indicator

BMW LED Design
- 1 GOVERNMENTAL
- 2 ENVIRONMENTAL
  - 2.1 Withstand an engine ba
  - 2.2 Withstand worse case a
  - 2.3 No air flow (natural conv
  - 2.4 Account for lumen drops
    - 2.4.1 Secondary Optics
    - 2.4.2 Shield
    - 2.4.3 Shutter Mechanisr
  - 2.5 Fit within existing headla
- 3 OPERATIONAL
  - 3.1 Operate with 90% of inpu
  - 3.2 Maintain junction temper
  - 3.3 Relative Photometric ou
  - 3.4 Lumen maintenance >7(
  - 3.5 Maintain efficacy > 50 ln
  - 3.6 Maintain lifetime of >30,(
  - 3.7 Supply >2000 lumen per
  - 3.8 Power consumption<84\
  - 3.9 Color temperature of 60(
- 4 GENERAL

ID	Design Requirements
60	2.4.3 Shutter Mechanism
61	2.5 Fit within existing headlamp volume
62	3 OPERATIONAL
63	3.1 Operate with 90% of input power lost to thermal energy
64	3.2 Maintain junction temperature <75°C
65	3.3 Relative Photometric output >70%
66	3.4 Lumen maintenance >70%
67	3.5 Maintain efficacy > 50 lm/W
68	3.6 Maintain lifetime of >30,000 Hrs
69	3.7 Supply >2000 lumen per headlamp
70	3.8 Power consumption<84W
71	3.9 Color temperature of 6000K
72	4 GENERAL
73	4.1 Weigh less than existing system
74	4.2 Passive system, requiring no additional energy system/subsystems
75	4.3 Cannot infringe on energy systems/subsystems within vehicle
85	4.4

Username: jmmclel   Exclusive edit mode

Figure 4.11: Change Indicators [113]

IBM Rational DOORS has shortcomings in its inability to differentiate requirement relationship types. For instance, two requirements may possess a function to function relationship, both serving the same function such as converting energy, while another two requirements may have a component to component relationship, both relating to a specific component, such as a drive shaft. Within DOORS this is noted as a requirement relationship. No type or weighting or relationship exists, an aspect this research will address. This is of great importance to this research as the identification of requirement relationships directly correlates to Research Question II. Requirement relationships will be used as the foundation for predicting requirement change propagation. DOORS is currently incapable of providing specific relationship types. All requirement relationships within the tool are noted as binary, existing or not. DOORS, however, is able to use this relationship to highlight when a change occurs, as seen in Figure 4.11.

While outside the scope of this research, it is important to note the problems that exist with input validation of DOORS. DOORS uses a text-based requirements representation. Thus, the ability to assess the quality, identify errors, and support advanced analysis is limited [31]. This is a major shortcoming of current requirements management tools that can lead to poorly communicated requirements. Further, this inability to validate requirements input may become an issue when attempting to predict requirement changes.

4.2. Change Representation and Modeling Tools

There exists change representation tools realized through research. Some are augments of existing tools, such as the Delta DSM while others are innovative means for managing design change, such as C-FAR. No modeling tools exist for specifically modeling requirement changes and their propagation, nonetheless many of the existing design change modeling tools can compared to the requirements model developed through this research. This section of the research dissertation will investigate existing modeling tools for their use as a change modeling, managing, or predictive tools.

Table 4.1 lists of the modeling tools which were investigated in this research. Each of the tools makes use of a different element change, for instance the Change Propagation Method (CPM) uses system components to model and predict change. The methods listed in the table were those which most closely aligned with the method used in this research and could be used for comparative reason.

Table 4.1: Explored Change Propagation Techniques

Tool	Ref	Author(s)	Subject of Change
Change Propagation Method	[18]	Clarkson, et.al.	Components
Change Design Structure Matrix	[60]	Giffen, et.al.	Subsystem areas
Change Favorable Representations	[7]	Cohen and Fulton	System/Subsystem attributes
Product Metamodel and ReChaP	[114,115,116,117]	Ibrahim, et.al.	Requirements
Requirements Relation Model	[118]	Chen, et.al.	Requirement Attributes
Impact propagation structure	[119,120,121]	Lock and Kotonya	Component

The change propagation and change design structure matrix by Clarkson and Giffen are discussed in Section 4.2.1 of this dissertation. The Change Favorable Representation model by Cohen and Fulton is presented in the following section, Section 4.2.2. Ibrahim's product metamodel and requirement change propagation model are presented in Section 4.2.3. The requirements relation model by Chen is described in Section 4.2.4. The final propagation model investigated is the impact propagation structure by Lock and Kotonya, presented in Section 4.2.5. Section 4.3 details a description of the model used in this research, Higher Order DSMs. An example is provided to demonstrate its use.

4.2.1. Requirement Change - Delta DSM

There exist few techniques used to model requirement change. A popular method used is use of a Design Structure Matrices (DSM) to develop relationships between subsystems, components, or requirements [122,123]. DSM can assist the designer by providing them with a means for modeling relationships between requirements. DSMs provide a technique for identifying the parts of a product or design and the parametric relationships between them [124]. An example of a DSM is shown in Figure 4.12. Each cell in the matrix may contain a numerical or binary representation of the link between one in a column to another row heading [18]. A DSM may not necessarily be symmetric based on the directionality of the relationships. For instance, Requirement A may relate to Requirement B, but not vice versa, causing a lack of symmetry.

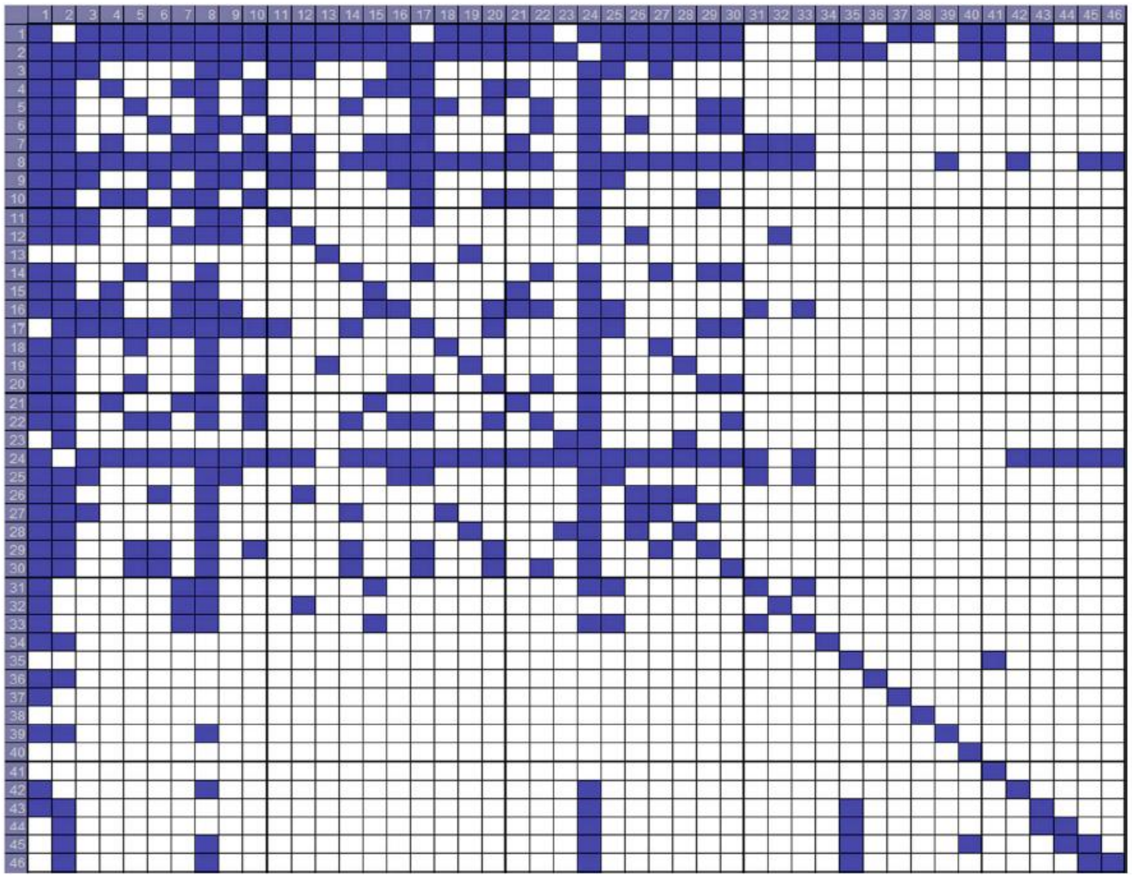


Figure 4.12: System Structural DSM [60]

A significant study performed by Giffen et.al evaluating the ability to propagate engineering change in complex systems through use of a Delta DSM [60]. While this study is not to the resolution of engineering requirements, it investigates specific "subsystem areas." The first step in the process requires a system block diagram, similar to that shown in Figure 4.13. The block diagram shows all relationships between different subsystems. The relationships are placed in a DSM to identify the relationship between the subsystems.

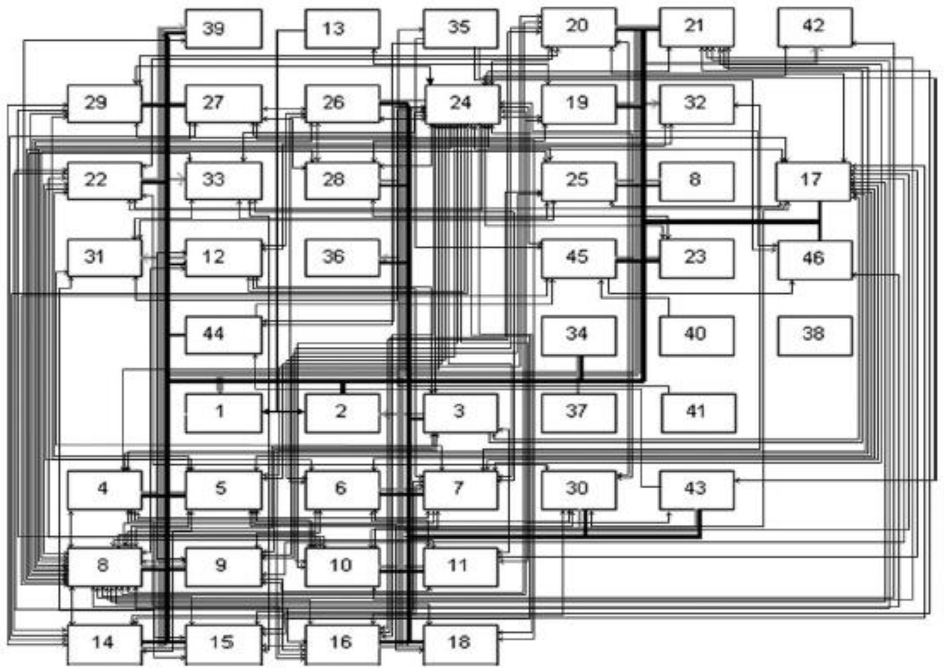


Figure 4.13: System Block Diagram [60]

The change requests are reviewed to identify if the relationships could have been used to predict the change propagation. Different types of change path are identified between each subsystem. In order to create a map of how changes affected the system, a Delta DSM is created [60,125]. An Example of a Delta DSM is shown in Figure 4.14. The Delta DSM identifies where change should propagate from one subsystem to another. The purpose of the Delta DSM is to identify what occurred to the subsystems after the change. There were three possible situations described within the study:

- *There were cases when the propagation was predicted and propagation did in fact occur.*
- *There was an identified relationship, change occurred yet no propagation occurred.*

- *Where a connection was not present, yet propagation was found*

Fundamentally, all scenarios could only fall under those three descriptions identified above, except for scenarios where propagation results in the addition of a subsystem. Currently, all propagation techniques cannot complete account for the addition of a requirement. This is primarily due to the inability to predict requirement addition due to the lack of relationship with a requirement that has yet to exist.

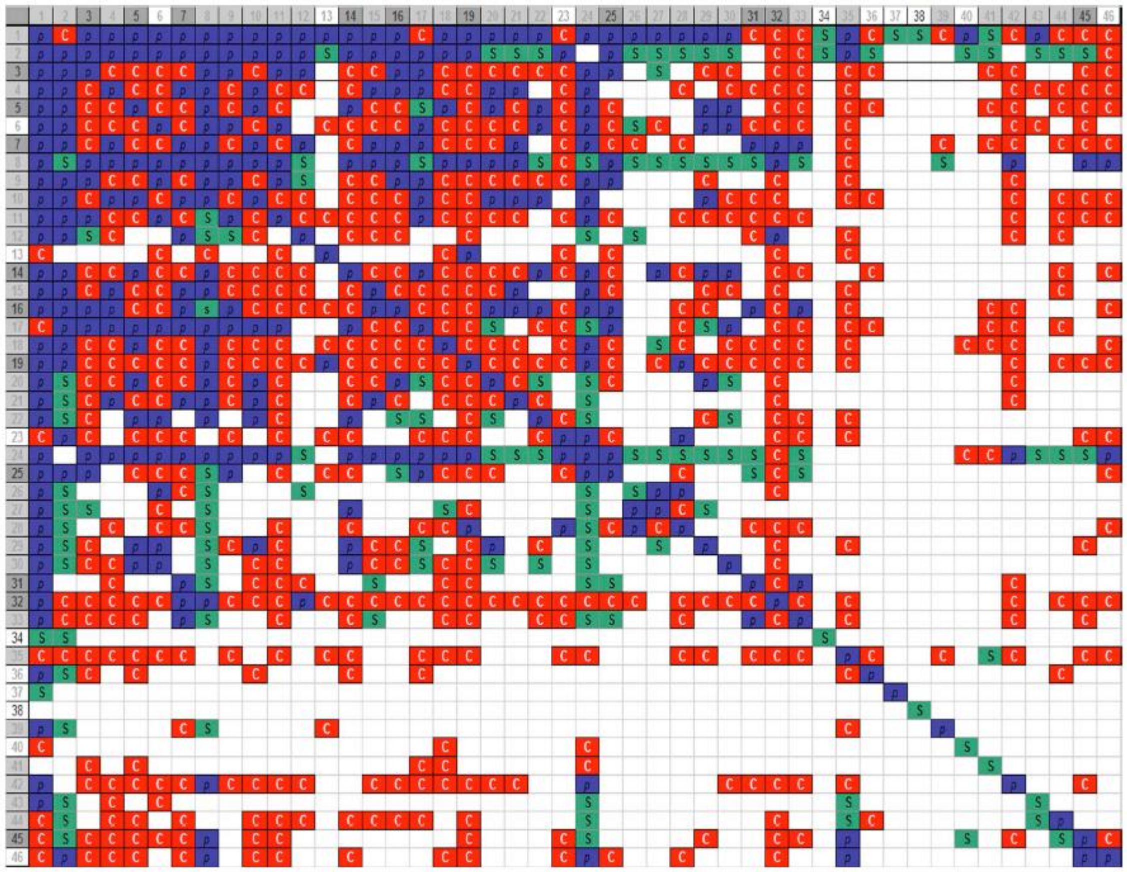


Figure 4.14: Delta DSM [60]

In addition to the Delta DSM populated, each one of the changes came with a Change Propagation Index (CPI). A subsystem area CPI controls whether or not propagation is carried through the relationship:

- *If you are a constant or carrier, you are not increasing or decreasing propagation, and as a result, your index is 0.*
- *If you are a propagation absorber, CPI <1.*
- *If you are a multiplier, CPI >1.*

The limitation of this approach is the use of the subsystems, instead of working on change propagation in the requirements domain. This is limited because it requires system architecture to be developed and this architecture is not of the resolution of requirements. Where as a single component may have tens or hundreds of requirements, a requirement will emphasize on component behavior, feature, or function. ***The Requirement Change Propagation Prediction Tool presented in this dissertation will allow for prediction of change within the requirements domain, allow for greater resolution of change propagation, and be suitable for early design use (before a system architecture has been defined).***

4.2.2. <u>Change Favorable Representation</u>

Another change propagation tool is the Change FAvorable Representation (C-FAR), a new and different methodology of representing design information so that changes can be dynamically anticipated and evaluated [7]. Fundamentally, C-FAR uses attributes, elements and relationships to develop its prediction. It uses a schema that defines the entities, their relations between entities; and their attributes. Figure 4.15 shows an example C-FAR model of a bottle. The bottle is the entity of interest here, where its attributes are size and material.

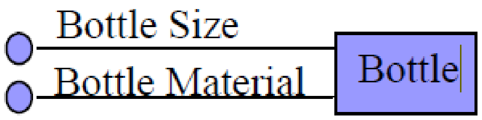


Figure 4.15: C-FAR Using EXPRESS Framework [7]

If this example is expanded to multiple entities, such as the liquid contained within the bottle, the C-FAR shown in Figure 4.16 is developed. This C-FAR illustrated the bottle, liquid, their relationship, and their respective relationship with their attributes. A C-FAR Matrix is developed to provide links between the attributes of one entity and the attributes of another entity. The components used to construct the C-FAR matrix are called linkage value [7].

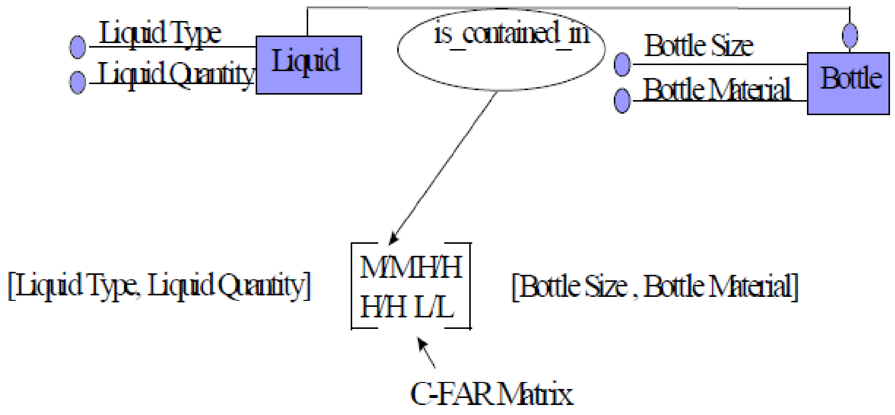
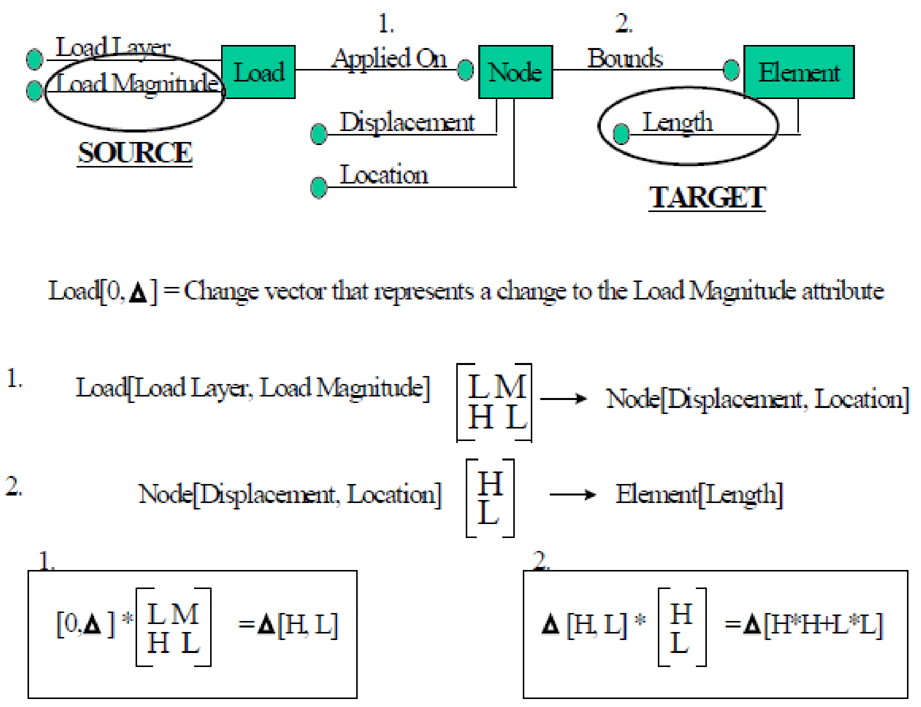


Figure 4.16: Expanded C-FAR [7]

A linkage value is used to represent the relationship between two attributes of different entities. The purpose of this is to identify how a change in one entity attribute affects the attribute of another entity. An example of this is shown in Figure 4.17 where the Load Magnitude changes and the affect this has on the length of the element is measured.



Load Layer
Load Magnitude
Load
SOURCE
1.
Applied On
Node
Displacement
Location
2.
Bounds
Element
Length
TARGET
Load[0, Δ] = Change vector that represents a change to the Load Magnitude attribute
1. Load[Load Layer, Load Magnitude] [L M / H L] → Node[Displacement, Location]
2. Node[Displacement, Location] [H / L] → Element[Length]
1. [0,Δ] * [L M / H L] = Δ[H, L]
2. Δ [H, L] * [H / L] = Δ[H*H+L*L]

Figure 4.17: Example Change with C-FAR Schema and Matrix [7]

C-FAR is effective in computationally measuring the effect of one attribute to another using its matrix relationship. However, C-FAR uses an existing product information model to facilitate change representation, propagation, and qualitative evaluation. ***Unfortunately, product information may not always be available during the design process. Requirements, however, are an initial document generated and will***

always be available. While it may constantly change, its existence is nonetheless evident.

4.2.3. Requirement Change Propagation (ReChaP)

The Requirement Change Propagation (ReChaP) method was developed by Ibrahim and colleagues as a means to manage requirement change in software systems. It is important to note that this is not a requirement change propagation *prediction* method, rather a means for managing inevitable requirement change. Though this type of change propagation approach is intended for software systems, it is unique in its use of requirements. This approach was developed to enhance requirement change management throughout the volatile software development process [126]. Requirements change within the software domain may occur through the following mechanisms: new requirements are introduced into an existing system, modifications are performed for requirements, and a change in the operating environment of the system [127]. Using this, a change propagation model is developed by Ibrahim. The motivation for this model was to better understanding of the realistic process during requirement change propagation, resulting from theory to real practices [128]. Requirements are used here because they are a more efficient mechanism to maintain the relations and consistencies between varying artifact granularity [129]. This is demonstrated through the use of requirements relating to class and state diagrams. Before defining the approach, a brief introduction to requirement change propagation in software design is discussed.

In software systems, change propagation is defined as “changes required to other entities of the software system to ensure the consistency of assumptions in a software system after a particular entity is changed” [130,131]. Effectively, it is a process in which modifications are implemented to reestablish system consistency by completing a set of estimated consequent changes [132,133]. As a result, when a part of the software is changed and modified, other parts of the software need to be identified and changed as well as they may have been impacted by the original change [130]. Fundamentally, the model used here is similar to the proposed model of this research, an initial set of requirements affected is analyzed, and subsequent requirements (or artifacts) that need to be changed are identified. Such artifacts include class and state diagrams, as shown in Figure 4.18. As seen from the figure, when a requirement changes, this may propagate to other artifacts of the design which can be modeled through class and state diagrams [134]. In object oriented systems, it is not uncommon to refer to relationships between software artifacts and requirements during different phases of the software development [135].

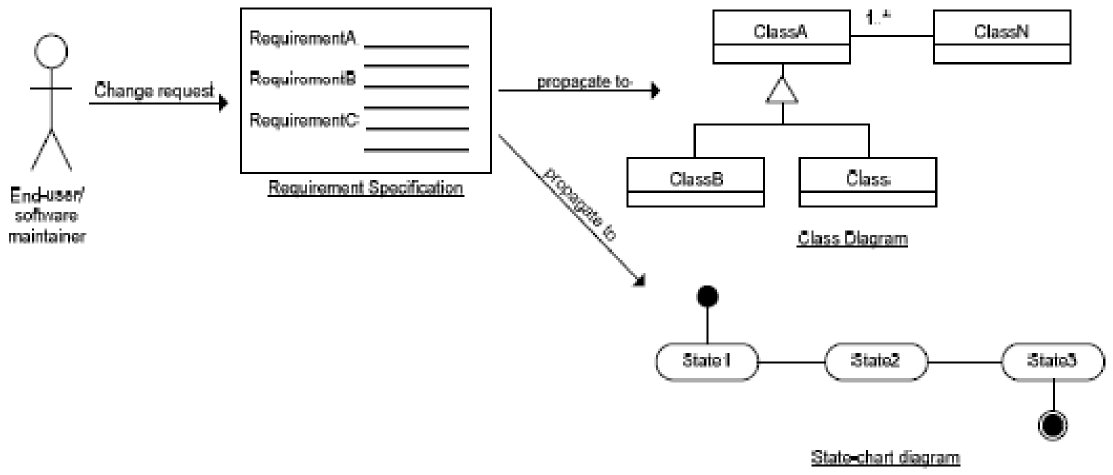


Figure 4.18: ReChaP Process [134]

ReChaP was developed to simplify the error prone process of change propagation due to requirement change and to provide a means for supporting change throughout the software development lifecycle. ReChaP is developed using two approaches, a product metamodel consisting of theoretical specifications and process model consisting of implementation specifications.

In particular, the foundations of ReChaP approach are available in twofold: product metamodel (theoretical specification) and process models (implementation specification). Figure 4.19 illustrates the ReChaP approach making use of the two models. The Theoretical specifications are used to better understand the solutions for software developers by developing a product metamodel. The process model is used to develop an implementation specification, defining the roles of the designers and activities which must be completed within the design process. Requirement change propagation occurs in the product metamodel when changes occur to specific requirements.

Subsequently, the process model is when change propagation occurs, as requirements will experience a process flow of propagation [114].

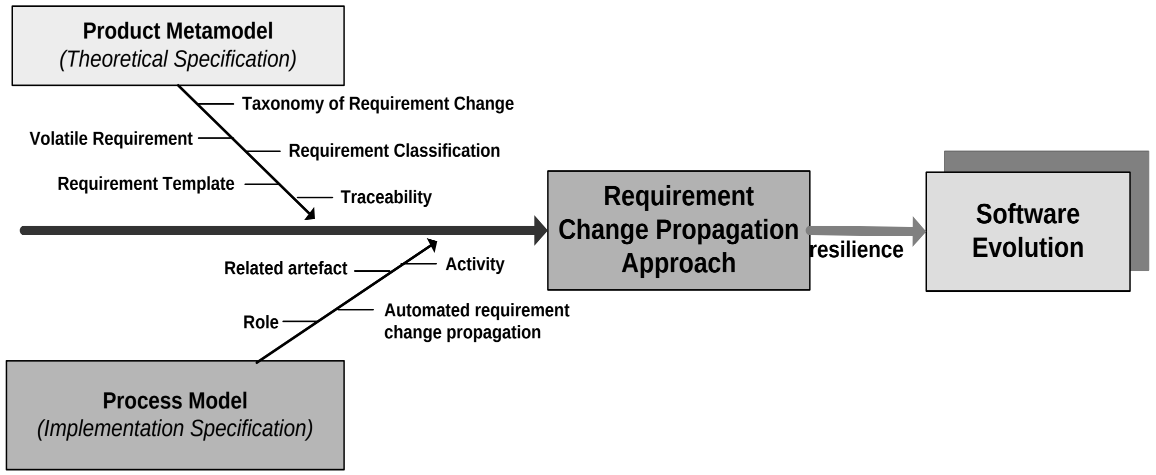

Product Metamodel
(Theoretical Specification)
Taxonomy of Requirement Change
Volatile Requirement
Requirement Classification
Requirement Template
Traceability
Requirement Change Propagation Approach
resilience
Software Evolution
Related artefact
Activity
Role
Automated requirement change propagation
Process Model
(Implementation Specification)

Figure 4.19: ReChaP Approach Framework [114]

The ReChaP approach is not a predictive approach, simply a requirement change propagation method. This method attempts to alleviate much of the risks involved with mismanaged change propagation by providing the designer or engineer with a model that assists in determining areas where propagation may occur. Figure 4.20 shows the ReChaP approach used which incorporates a class and state diagram in conjunction with the metamodel. This approach functions by relating the requirements to other parts of the product through relations developed by the diagrams used.

The designer or engineer is to subjectively develop this model through the knowledge they have about the system and its design. The class diagram illustrates a static part while the state diagram illustrates the dynamic process of system operation. The metamodel identifies the requirement classification which categorizes the requirements based on their type: functional or nonfunctional requirements [136]. Also in the metamodel is the change request written and their source, reason, and type of change. The information in the metamodel is used with the diagrams to develop connections between the requirements, their change, and the associated class and state diagrams.

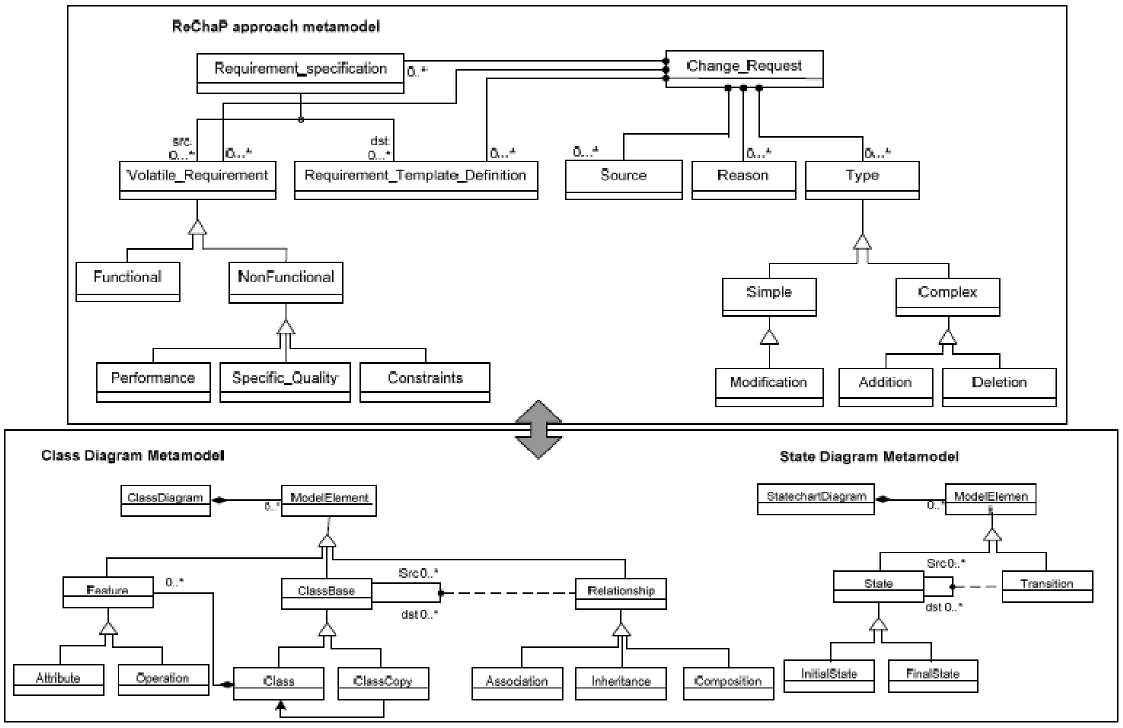
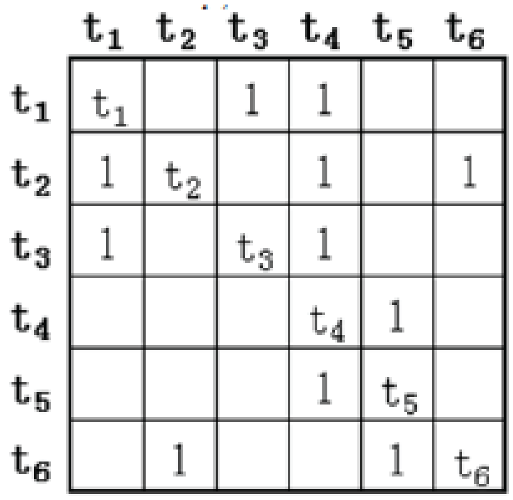


Figure 4.20: Model for ReChaP Approach [134]

While this approach does make use of requirements to manage change propagation, there are two major limitations. This approach was developed for software systems alone and this approach does not predict requirement change. ***The developed tool in this research is suitable for other types of products outside of software systems and is able to predict change propagation.***

4.2.4. <u>Requirement Relation Model</u>

The requirements relationship model developed by Chen is of unique interest to this research because it introduces the use of requirement dependencies. The success of the research proposed in this dissertation relies heavily on the dependencies, or relationships, developed. The limitation with the Requirement Relation Model is, as with many requirements change tools, intended for use in a software development environment. Similar to other DSM approaches, requirements are placed on a DSM and related against one another. An example of this is shown in Figure 4.21.



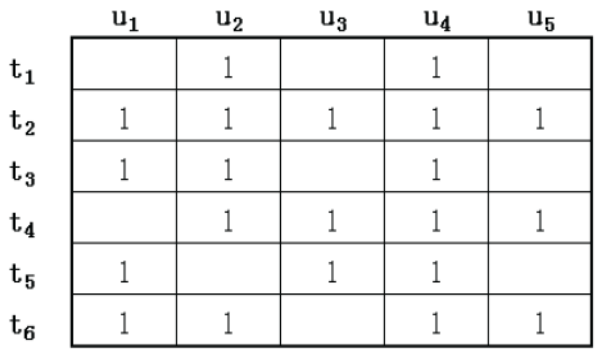
	t_1	t_2	t_3	t_4	t_5	t_6
t_1	t_1		1	1		
t_2	1	t_2		1		1
t_3	1		t_3	1		
t_4				t_4	1	
t_5				1	t_5	
t_6		1			1	t_6

Figure 4.21: Requirement Dependency DSM [118].

In this model, the author makes use of "trustworthy" requirements which are requirements composed by various trustworthy attributes [137]. Trustworthy requirements are defined as "quality requirements of trustworthy software and overall quality checks which have back act to functional requirements, and their change can produce big impaction to the quality of trustworthy software product" [118].

Trustworthy requirements have two kinds of influence relations, inherent and non-inherent [137]. Trustworthy requirements are used here because most requirements based methods use only functional requirements, whereas trustworthy requirements have a great influence on one or more functional requirements.

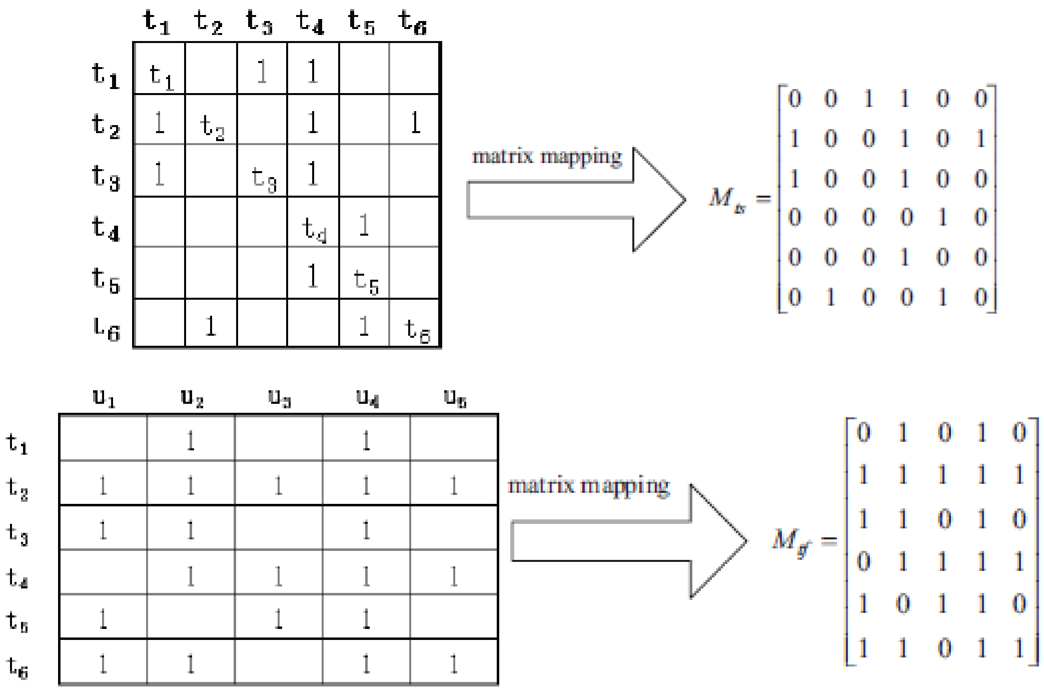
A matrix is developed to correlate the trustworthy requirements to use cases, as seen in Figure 4.22. The use cases are used to describe function requirement, considered to be core concerns of the system. This models finds that there is high correlation between trustworthy and function requirements. While the first DSM, Figure 4.21, shows how a change to one trust worthy requirement may affect another, the incident matrix, Figure 4.22, illustrates how a change in a trustworthy requirement may cross boundaries and affect function requirements.



	u_1	u_2	u_3	u_4	u_5
t_1		1		1	
t_2	1	1	1	1	1
t_3	1	1		1	
t_4		1	1	1	1
t_5	1		1	1	
t_6	1	1		1	1

Figure 4.22: Incident matrix of requirements vs. use cases [118].

Relationship matrices, as seen in Figure 4.23 are developed so the results of the relationships can be analyzed. Both matrices are matrix multiplied to develop a reachability matrix. The resultant reachability matrix, seen in Figure 4.24 indicates how each trustworthy requirement may reach (or propagate) to another trustworthy requirement. An example interpretation of the reachability matrix, the first column, for trustworthy attribute t_1 is related to all relations on that column. In this example t_1 is reachable to t_3, t_4, and t_5.



	t_1	t_2	t_3	t_4	t_5	t_6
t_1	t_1		1	1		
t_2	1	t_2		1		1
t_3	1		t_3	1		
t_4				t_4	1	
t_5				1	t_5	
t_6		1			1	t_6

$$M_{tt} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

	u_1	u_2	u_3	u_4	u_5
t_1		1		1	
t_2	1	1	1	1	1
t_3	1	1		1	
t_4		1	1	1	1
t_5	1		1	1	
t_6	1	1		1	1

$$M_{tu} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Figure 4.23: Relation Matrices [118]

$$M_{R^+} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 4.24: Reachability matrix [118]

Additional matrix manipulation may be performed on this approach to determine other propagation metrics outside of element reachability. A summation may also be taken of each trustworthy attribute to determine which requirement has the most significance on other requirements. Continuing the example above, trustworthy attribute t_2 in this case would have the greatest influence as it correlates to every other trustworthy attribute.

The limitation to this model is that only select types of requirements can be used in this model. Further, this model is developed for software systems and may not be applicable to electromechanical systems. ***In the tool developed in this research, all requirements may be used and it is suitable for product development outside of software systems.***

4.2.5. Impact Propagation Structure

The impact propagation structure developed by Lock and colleagues investigates change impact. This model makes use of different relationships between design entities to develop impact propagation. The impact of propagation in complex system relationships makes the process of assessing the effect of change tedious and expensive [119]. Impact analysis is used in this method because of its previous research use in determining the extent and complexity of changes during various stages of a product cycle [138,139,140,141,142,143]. However, impact analysis is predominantly used in software systems as the relationships and their propagation impact can be determined.

Impact analysis is a process used to determine how the change in one element may affect other, related, elements. There are many approaches to this, however the approach used in the impact propagation structure is that of Moreton's who makes use of a generalized model relevant for most software systems [144,145]. The stages of Moreton's impact analysis process are illustrated in Figure 4.25.

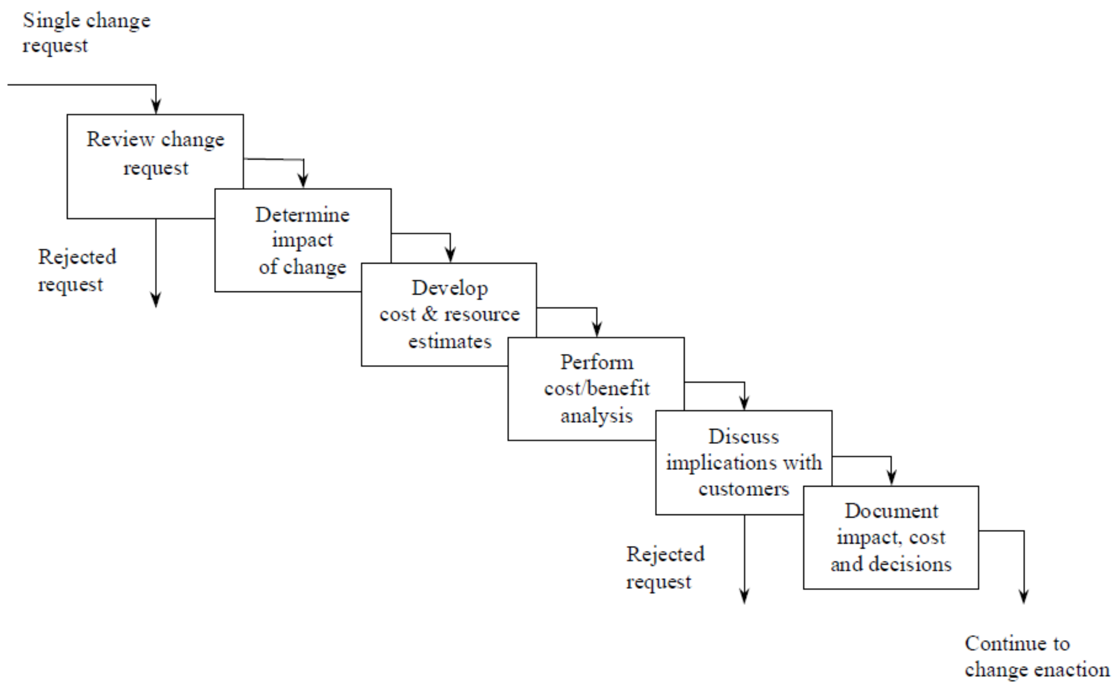

Single change request
Review change request
Rejected request
Determine impact of change
Develop cost & resource estimates
Perform cost/benefit analysis
Discuss implications with customers
Document impact, cost and decisions
Rejected request
Continue to change enaction

Figure 4.25: Impact Analysis Process [119]

The process follows a series of steps, with two chances of rejecting the change. The process is as follows [119,121]. The process starts with a review in the change request. This is an initial evaluation of the change request to determine what must be performed and implemented. This is followed by determining the impact of change by analyzing the target system and viewing the related constituents which may be affected. With an initial change and the affected areas identified, a cost and estimation of the total change is calculated. This includes the potential change propagation from the initial change. This only takes into account the change the designer is certain will propagate. Because the total cost of change may be significant, it is important to compare this cost against the benefit of the initial change (if the change was initiated for cost reduction). In some instances, this step is removed as the change must be completed, for instance a change in federal regulation. Once the change and the associated costs are identified, this is presented to the customer for negotiations (if needed) and approval. The customer here may reject the change if it is not mandatory. If approved, the change is documented in the appropriate forms, and the changes are implemented. The stage most critical in this process is the accurate determination of the impact. This is performed subjectively, based on the designer's knowledge of the system. If a designer rejects the change in the early stages, the cost of the change must still be recorded as this information is pertinent to subsequent change processes [146].

As a result, determining impact is crucial to the effectiveness of this model. Because many systems are complex in nature, depending on a designer for impact information is not sufficient. Further, a balance must be maintained between producing analysis results with too much impact and not enough impact for proper appreciation of propagation impact [119,121]. Traceability from various sources is used to identify the impact of change propagation between entities. Traceability is effectively a relationship between two artifacts (or elements) within the system that may be vulnerable to propagation impact. Any information which may relate to a specific artifact may be a relating entity to that artifact. The traceability links are used to develop an impact determination process by which the traces between artifacts are analyzed for their impact propagation. An example of this is shown in Figure 4.26.

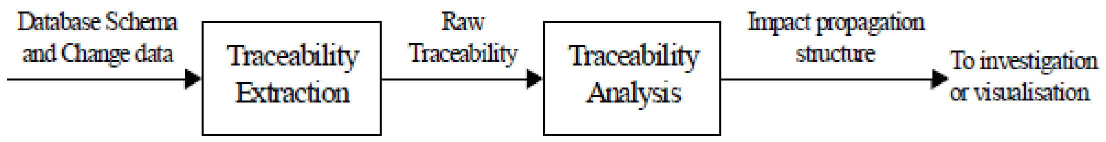

Database Schema and Change data
Traceability Extraction
Raw Traceability
Traceability Analysis
Impact propagation structure
To investigation or visualisation

Figure 4.26: Impact Determination Process [119]

In using the impact determination process, an understanding of the system, is needed. This is maintained through a database schema of all artifacts (elements) and relationships. The first step in this process is a traceability extraction. The term traceability extraction is defines as an “extraction of individual traceability relationships from the model” [119,121]. This is followed by a traceability analysis which makes use of the raw traces extracted to produce a representation model of the possible change impact.

The impact analysis and determination process are integrated to develop a model. By integrating both processes, their strength are used and limitations are mitigated. Because of the subjectivity of traceability extractions, relationships may not be realized. To ensure missing relations are not excluded, a multiple layer approach where traceability is extracted from multiple models is used. The models includes: Schema, Interaction, and Knowledge base models.

The first step in the integrated propagation structure, shown in Figure 4.27, is to extract traceability information from multiple models. The Schema model is used for performing prerecorded analysis. The interaction model is used for performing dependency analysis. The knowledge base model makes use of past experience to develop traces. The traces are extracted from each of the models and processed to produce an impact propagation structure. The structure involves three stages, lateral composition, vertical composition, and duplication resolution.

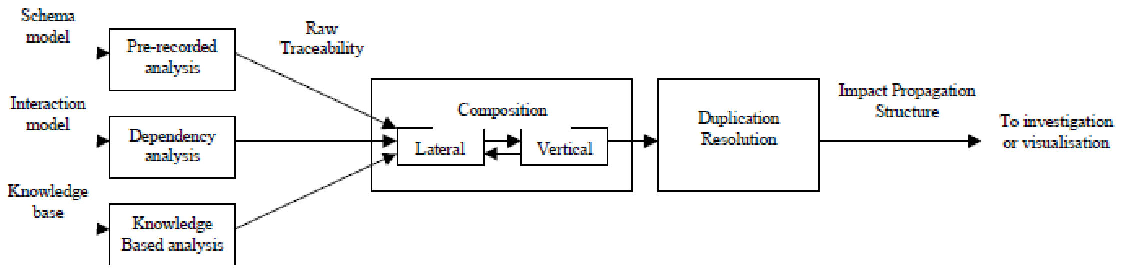


Figure 4.27: Propagation Structure [119]

The vertical composition of the used to develop the propagation structure is a model which considers the cycle of each impact to determine the propagation from one entity to another in adjacent cycles [147]. A maximum depth of three cycles is used here. An example of the vertical composition is shown in Figure 4.28.

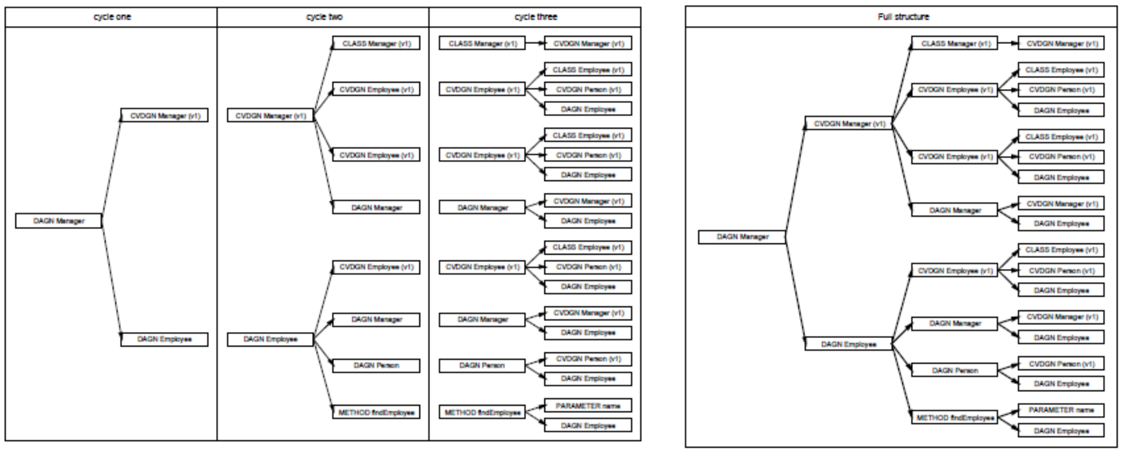


Figure 4.28: Example Vertical Composition [119]

A lateral composition is used in impact determination to combine the three forms of traceability extractions. Since this model makes use of multiple traceability extractions, a means for combining them is needed. Lateral composition sums the impacts detected by the schema model, interaction model, and knowledge base. An example of this is shown in Figure 4.29.

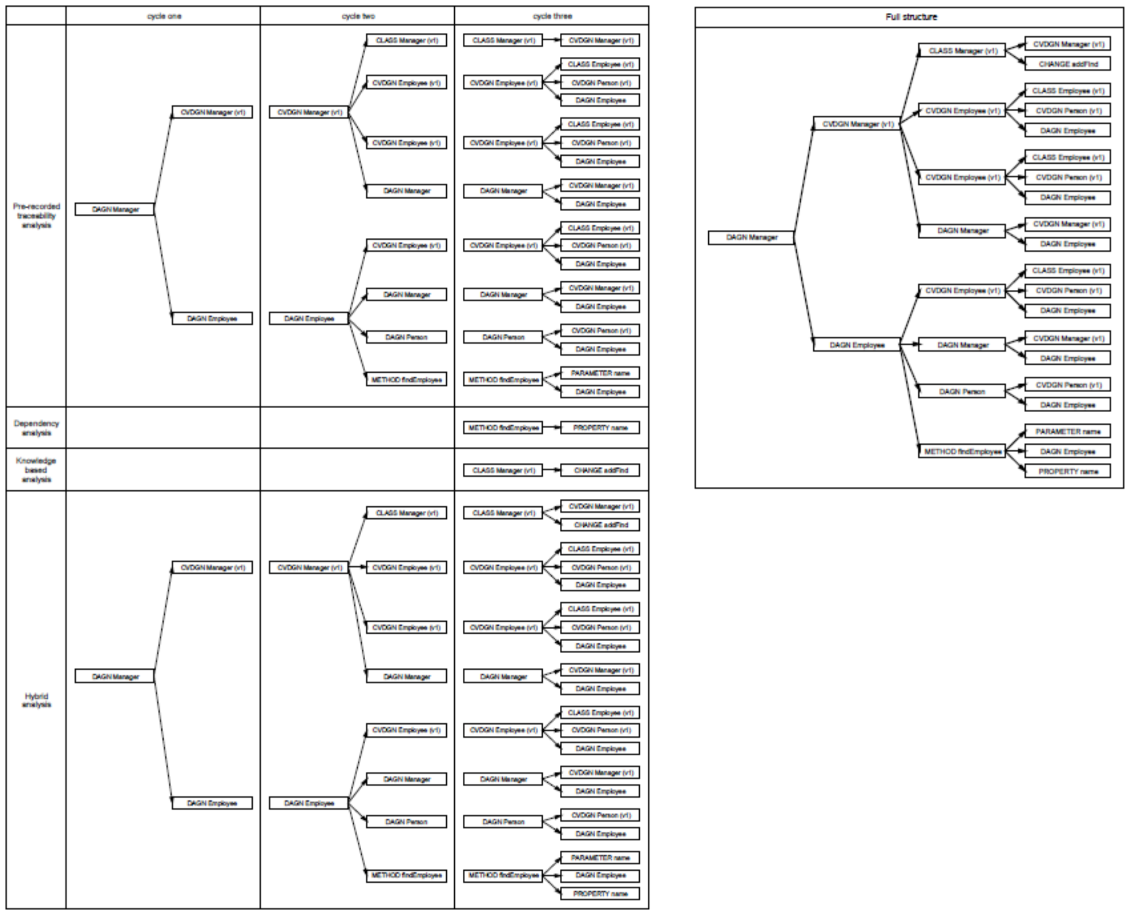


Figure 4.29: Example Lateral Composition [119]

The two composition models may result in duplicate propagations path to be identified. The duplicate resolution stage removes all duplicates to ensure this does not distort the summation of path developed in the lateral composition. A final impact propagation structure is developed after completing the three stages. As seen in Figure 4.30, the structure identifies the total possible consequences of implementing a change on the database.

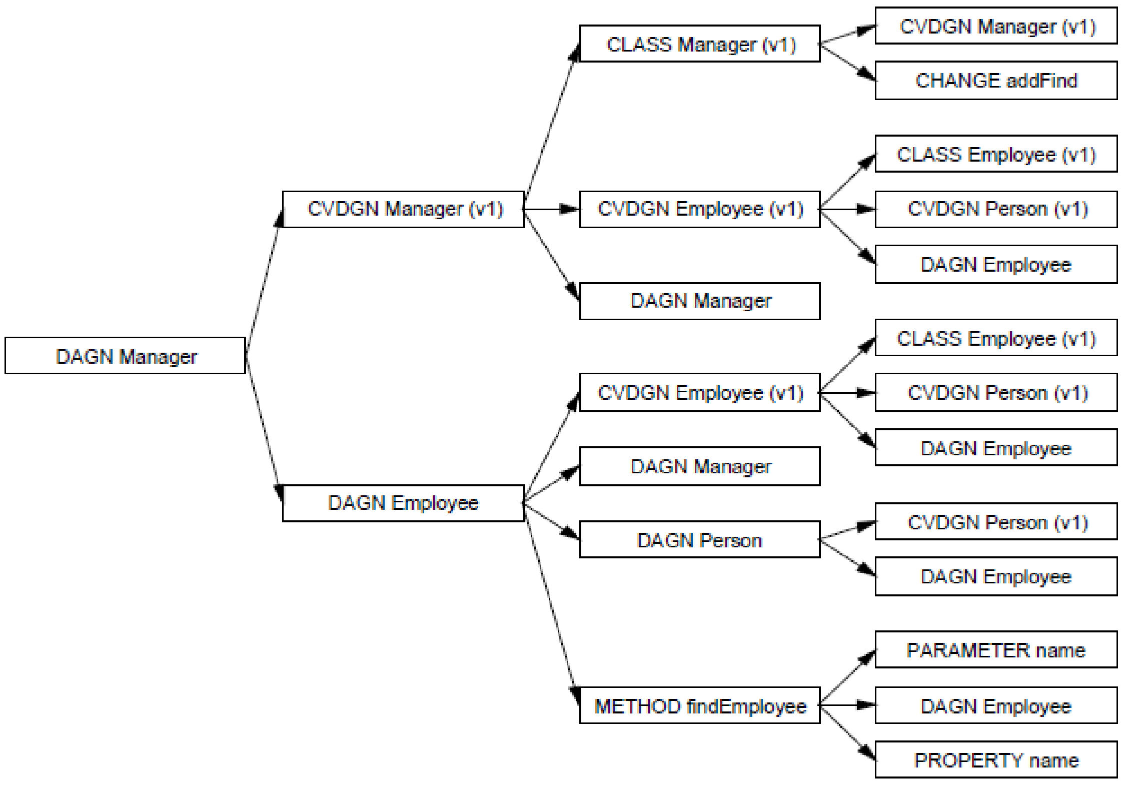


Figure 4.30: Example Impact Propagation Structure [119]

This structure will assist designers in [119]:

- Determining the accurate cost of change
- Producing a representation of change, change propagation, and change impact
- Guide the implementation of change across artifact cycles

However, this integrated approach maintains limitations similar to its peers. It is again suitable for software systems where relationships are between entities are easily realized. As noted, the relationships here are subjectively developed. ***The tool developed in this research automatically generates relationships for complex systems where relationships may not be apparent.***

4.3. Higher Order DSMs – Foundation for Proposed Tool

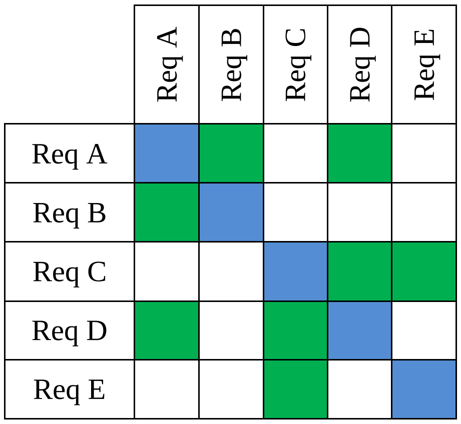
Change propagation has been predicted using a Design Structure Matrix (DSM) in complex systems termed change prediction model (CPM) [60] which makes use of the CPI as discussed in Section 4.2.1. CPM uses the probability of change in a subsystem area on others as elicited by experienced engineers. Although this model is capable of predicting the likelihood of changes in other subsystem areas, it is not to the resolution of engineering requirements, as it investigates specific subsystem areas, whereas requirements may be able to identify the specific features of the component. Therefore, a more systematic approach from the requirements paradigm is sought for which higher order DSMs are explored for its ability to predict change propagation through requirements.

DSMs have been used to model change propagation before [148,149,150,74,151], however it has yet to be performed formally through the use of requirements. This research makes use of DSMs differently as a DSM is used here by analyzing how change in one element of a DSM can propagate throughout a system. This propagation is illustrated through a higher order DSM. The advantage of using requirements is it does not depend on the system architecture, allowing designers to use it early in the conceptual design phase.

A higher order DSM is used to develop relationships between subsystems, components, or requirements [122,123,152]. In this research, where requirements are related, the DSM functions by listing the requirements on both axis and highlighting all cells where the requirement on a row is related to a requirement on a column. DSMs can assist the designer by providing them with a means for modeling, visualizing, and clustering relationships between design elements. Further, DSMs provide a tool for identifying the parts of a product or design and the parametric relationships between them [124,153]. Each cell in the matrix may contain a numerical or binary representation of the link between one in a row to another column heading [18]. A DSM may not necessarily be symmetric based on the directionality of the relationships. In the DSMs presented in this section, a cell highlighted in green indicates a relationship exists between requirements.

A DSM here can be considered as a zeroth order DSM as it serves as a baseline matrix. In order to create a map of how changes affected the system, a higher order is created. A higher order DSM may be that of a first, second, or third order depending on the complexity, population, and coupling of the requirements. While a higher order DSM is capable of propagating requirement changes, its current limitation is it cannot predict the addition of a requirement. The inability to predict requirement additions is due to the lack of relationship with a requirement that has yet to exist. For example, a requirement regulating the use of an electric motor may change to adapt an internal combustion engine instead. This propagation may lead to subsequent changes; however, the addition of a requirement to sanction exhaust emissions cannot be predicted.

First order relationships are those which a requirement is directly related to another requirement. These are highly dependent on how relationships are formed. For example, a requirement may be related to another requirement because they pertain to the same component, or they share a similar function. As seen in Figure 4.31, a DSM is used to represent the relations between five requirements, A through E. All cells highlight in green indicate a relationship between requirements, such as that between C and E. The original DSM may be considered a zeroth order relationship matrix. If a change is made to requirement E, all immediate relations are highlighted in red, as seen in Figure 4.32. These relationships are termed first order relationships because of their direct relation with the requirement changed. As seen, requirement E has one first order relation, requirement C. Second order relations are those illustrated in Figure 4.33 where due to the potential propagation from requirement C, all requirements related to requirement C are highlighted. This indicates that requirement D is second order related to the requirement E, where requirement C acts as the mediator.



	Req A	Req B	Req C	Req D	Req E
Req A	🟦	🟩		🟩	
Req B	🟩	🟦			
Req C			🟦	🟩	🟩
Req D	🟩		🟩	🟦	
Req E			🟩		🟦

Figure 4.31: Example Baseline DSM (zeroth order)

	Req A	Req B	Req C	Req D	Req E
Req A	🟦	🟩		🟩	
Req B	🟩	🟦			
Req C			🟦	🟩	🟥
Req D	🟩		🟩	🟦	
Req E			🟥		🟥

Figure 4.32: Example First Order DSM

	Req A	Req B	Req C	Req D	Req E
Req A	🟦	🟩		🟩	
Req B	🟩	🟦			
Req C			🟦	🟨	🟥
Req D	🟩		🟨	🟦	
Req E			🟥		🟥

Figure 4.33: Example Second Order DSM

Higher order DSMs will be implemented in the propagation prediction approach developed in this research. Further, many of the limitations discussed in the existing change representations and models will be addressed in the developed tool. The takeaway is to view existing requirement change management solutions and to develop a tool which is capable of enhancing change management through predictive capabilities. The developed tool is also automated, so designer subjectivity is not involved. However, a manual aspect is available for entering specific keyword based relaters. The developed tool manages requirement change, predicts change propagation, and addresses the limitations of the presented available tools.

4.4. Complexity Within Requirement Change

It is important to present a brief discussion on complexity as it has a significant effect on the propagation characteristic of the higher order DSM. This includes, but is not limited to, the orders in which relationships should be measured and the types of relationships that should be developed. In practice, the degree to which change propagates through a product depends on the complexity of the product itself. A complex system will contain a greater number of requirements with a multiple possible relationships avenues (referred to as relater in this research). The complexity of a product can be measured in terms of the relationships between components [154]. This is pertinent here because requirements will be used to relate the components when predicting engineering changes (the physical domain) using the requirements domain. Further, requirements for complex systems are themselves complex information structures [155], making the processing of requirements of complex systems a sensitive process.

While complexity is not a focal of this research, it is important to recognize its significance during the development of a propagation technique. Complexity may affect this research in two forms: prior to propagation and after propagation. Complexity may make it difficult to develop relationships between requirements and forming weightings to requirements. As a product increases in complexity, there is greater difficulty in identifying the types of relationships that may be important in the analysis of propagation. Further, in completing the propagation analysis, complexity may result in much noise in the computation of potential requirements change. This could come in the form of superfluous potential requirements or inaccurate selection of potential requirements change (false positives or false negatives).

CHAPTER 5. CREATING RELATIONSHIPS BETWEEN REQUIREMENTS

Presented in this chapter are two means for populating the requirement DSMs and how each resulting DSM is measured in terms of connectedness. A scoring algorithm for requirement relationships is presented to weigh and rank requirements to narrow the highlighted requirements for propagation consideration. The two methods for DSM creation are manual and syntactical. Manual creation of DSMs represents a traditional method of populating DSMs through human decision making and provides a baseline by which to compare new methods. This method is initially used in predicting change propagation on two industry case study [24,32] . Syntactically, using an automated part-of-speech tagger to identify nouns and verbs augmented with manually selected keywords for each requirement, is used as the second method to generate requirements relationship DSMs.

The manual method includes a subject based approach where requirements are related by the subject and a keyword based approach where five keywords are used. The manual methods were used to determine if propagation prediction is possible [24,32]. Ultimately, both manual methods are used to develop the foundation for the automated syntactical approach. The manual approaches demonstrated that relationships could be developed between requirements to predict requirement change. Alongside the nouns and manually selected keywords, the syntactical approach evaluates the use of requirement functions, the verbs, for their ability to predict requirement change propagation.

In the syntactical approach, five nouns and five verbs are automatically extracted from each requirement. Additionally, five keywords are manually selected for each requirement. The fifteen total nouns, verbs, and keywords and their combination sets are evaluated for their ability to predict accurate requirement change. This required the analysis of 32,768 possible relater combination sets for the requirement change propagations that occurred in the industry studies. The three best combinations in all studies were then combined to develop a single requirement to requirement DSM. The protocol for identifying the best combinations and their overlaying will be discussed in section 5.2.2.

5.1. Manual Creation Approach

In the manual method, requirements are related to one another through manually identified requirement subjects and keywords, both used in two separate case studies. The Toho industry project (see Chapter 7) made use of a subject based relationship, relating all requirements which shared a subject. In most instances, requirements pertaining to a specific subsystem or component were identified as relating, as both shared a common subject. The selection of the subject within each requirement is manual and may be subjective based on those tagging the subject in the requirement.

The second manual creation method was developed based on requirement keywords, words the designer or engineer selected as pertinent and important to a requirement. This project dealt with the Pierburg industry project (see Chapter 8). While the former manual creation approach demonstrated that a subject relationship between requirements could predict requirement change propagation, there were deficiencies that needed to be addressed. For example, an important requirement needing constant consideration may be a federal regulation. However, such a requirement may not have a specific subject; rather it possesses regulation or standard numbers. As a result, a means for relating requirements based on relaters outside of their subject alone was needed. Further, in the manual approach subject base study, only one subject could be selected. In many requirements, there were several subjects that were important to the requirement. Thus, a refinement of the relationship types between requirements is explored where keywords are used.

Keywords were selected by reviewing and interpreting the semantics of the requirements rather than their syntactical subject. This was performed by studying the requirements document and understanding how each requirement specifically affected the system design or which keywords may be pertinent to the requirement. Upon studying the requirements document, each requirement was tagged with five keywords which the requirements document reviewer felt were relevant to the requirement and the overall system. While the selection of keywords is subjective, a set of common words were identified after investigating the requirements that might be used as "seed keywords" in future projects to reduce subjectivity.

5.1.1. Assumptions of Manual Creation

Inherent within this method is the subjectivity of the relations created. In the subject based relationship, it is assumed all requirements are related through a single subject. Though this is found to be a limitation as requirements may pertain to multiple subjects, it is addressed in the syntactical creation approach. It is assumed the keywords selected are sufficient in representing the important artifacts of the requirement. Further, it is also assumed that each requirement can support the needed number of keywords to build relationships. The variation of keyword selection due to the subjectivity of their selection is tested and detailed in Section 5.1.4. This is performed to ensure keyword selection is not discounted due to the variation that may be presented between engineers.

5.1.2. Subject Based Manual Creation Protocol

The subject based approach required the designer or engineer to review the requirements document and manually select the first subject, be it a subsystem or component pertaining to the specific requirement. An example of this can be extracted from one of the engineering change notices used in this study. Requirements 9.3.9 and 9.3.10 of the Toho project state:

> *9.3.9: A **Yarn Comb** for (22) ends shall be provided for each layer of bobbins.*
>
> *9.3.10: A **Yarn Comb** for (220) ends shall be provided for each of the two (2) PAN sheets.*

Highlighted in bold are the subjects selected for each requirements. Due to the shared subject, yarn combs, both requirements are related. Once this is identified for each requirement, a DSM is created for shared subjects relationships. The DSM is binary (identifying whether the subjects match or not) and symmetric (the relationships cannot be unidirectional as both subjects have to match). An extract of the subject based study

DSM is shown in Figure 5.1, where relating requirements are highlighted in green and blue indicates the DSM diagonal. The final DSM generated for the Toho project's subject based DSM is shown in Figure 5.2. The clustering seen in the figure is as a result of the manner in which the requirements were documented. The requirements here are listed in a hierarchical manner where requirements with similar subjects are written adjacently.

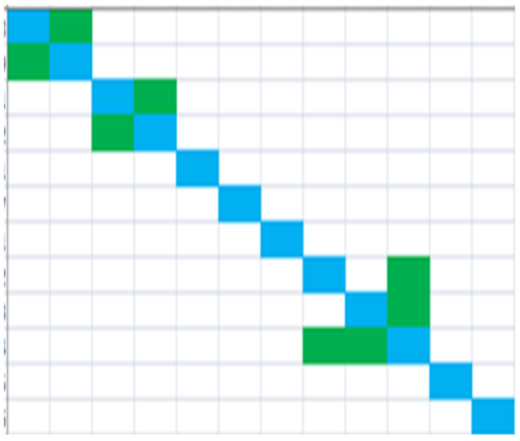


Figure 5.1: Small Segment of DSM for Subject Based Study

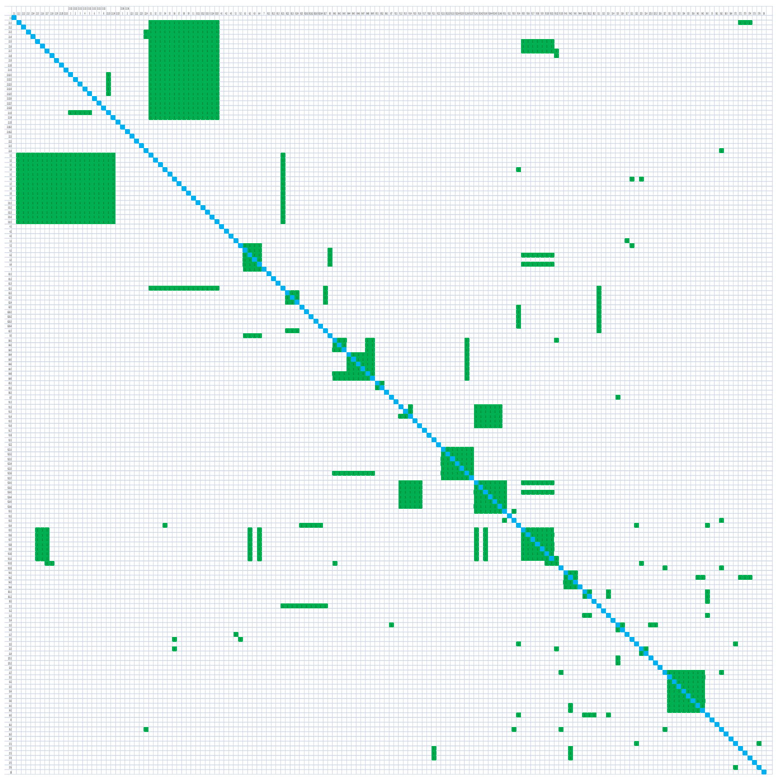


Figure 5.2: Toho Subject Based Manual Creation Study DSM

5.1.3. <u>Keyword Based Manual Creation Protocol</u>

The purpose of keywords is to allow designers or engineers to manually select words they feel are pertinent to a design. A keyword is a single word (sometimes defined as a unigram), or a grouping words [156]. Keywords are very common in most text and their purpose is to give a brief description of the text [157]. Within this approach, the designer must tag each requirement with a set of keywords. Keyword tagging is defined as the act of associating a term with an information object so that the term is a description of the information object [158].

Keywords serve to summarize the text into a predefined set of words [159,160], where requirements are the text here. The keyword selection protocol is subjective, requiring the engineer or designer to be familiar with the requirements document. Every requirement is reviewed at least once to understand the overall goal of the system and the major subsystem and component involved. After the initial review, the requirements are analyzed for potential keywords. To demonstrate the keyword selection process, consider requirement 2.1.12 from the Pierburg project:

> *2.1.12: Vibration dampening level pads will be provided with a +/- 2-inch height adjustment capability.*

When reviewing this requirement for keywords, it is important to identify what terms may cause this requirement to be related to another requirement. Further, it is important to recognize which parts of the requirement may change in the occurrence of a requirement change. The keywords selected for this requirement are: *vibration*, *level pads*, *dampening*, *height*, and *adjustment*. *Vibration* is selected as any system, subsystem, or component which experiences vibration could require dampening pads. Further, *level pads* were selected as this requirement affects this specific component and other components which may also use level pads. *Dampening* was selected as a keyword addressing the working principle of the level pads as there may be other dampening mechanism which relate to this due to their shared objectives. *Height* is selected as a keyword because of its overall dimensional effects on the system because of other spatial constraints that may be placed within other requirements. *Adjustment* is selected as a keyword because it was important this system afford adjustability to satisfy the requirement.

Initially, each requirement is tagged with five keywords as it was difficult to select an additional keyword when examining the requirements document as a whole. A total 1070 keywords (407 unique) were selected from the requirements of the Pierburg project, which entailed 214 requirements.

The goal is to reduce the number of requirements the designer or engineer has to review after propagation analysis. The greater the number of relationships, the more reviewing the designer or engineer has to perform before implementing the change. Though in this manual study, three keywords are used, the syntactical approach uses five keywords in evaluating the highest performing noun, verb, and keyword combinations.

Initially, five keywords were selected, though the appropriate number was not yet determined. These five keywords were sorted from highest to lease frequency keywords. For example a requirement may state "All Mechanical drive components shall have timing marks to verify alignment." The keywords within this requirement are: mechanical, drive, component, timing, alignment. The keywords are listed in their sequence of syntactical position. However, these keywords are then rearranged based on the frequency of keyword appearance of each word. In this scenario, the keyword list would be as follows: component, drive, alignment, mechanical, and timing. To identify the correct number of keywords to use, the number of relationships generated per keyword is analyzed.

The propagation analysis was performed with each possible number of keywords to identify how many keywords were needed to develop the relationships needed to predict change propagation. It was identified that using 3, 4, or 5 keywords would yield a propagation analysis which required the designer and engineer to review the requirements which changed. As a result, investigation of the keywords and their propagation sensitivity revealed that the minimum number of keywords needed to propagate to the appropriate requirements was three. Figure 5.3 illustrates the number of relationships vs. the number of keywords used. As the number of keywords increased beyond three, a fourth keyword would increase the number of relationships by approximately 4% while a fifth keyword would increase 6% over that of three keywords. Over populating the DSM would require the designer or engineer to evaluate multiple false positive propagation results.

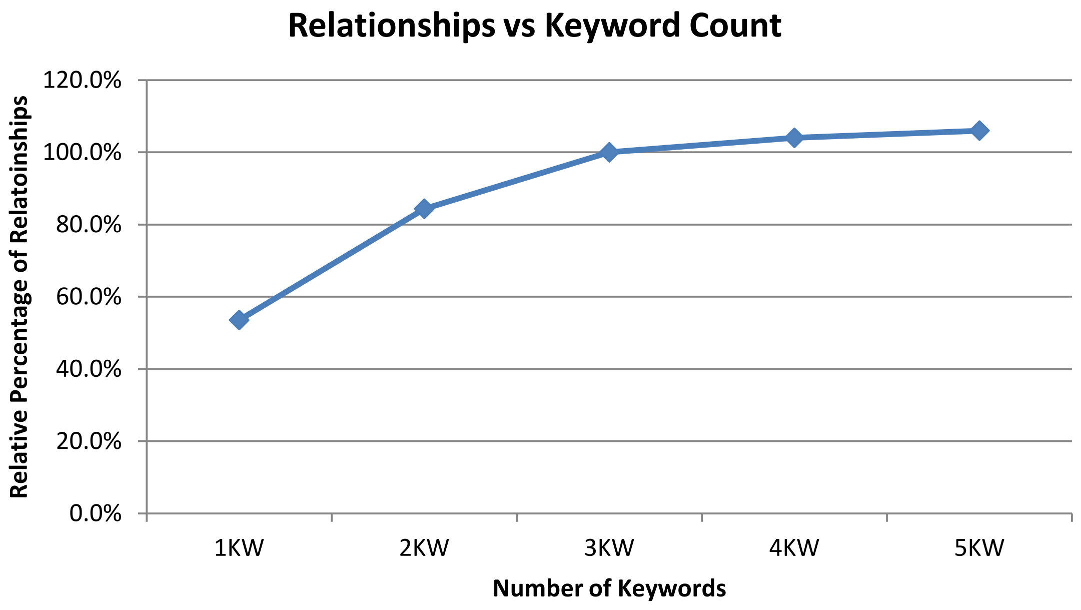


Figure 5.3: Percentage Number of Relations per Keyword w.r.t. Three Keywords

The Pierburg made use of three keywords during the analysis of the change propagation. This was performed as too many keywords will excessive relationships while too few may result in missed relationships. To systematically reduce the five identified keywords to three keywords for relationship generation, a frequency approach is used. The three keywords for each requirement that occur most frequently in the collection of keywords (1070) are used. This is to ensure the keywords most likely to possess relationships due to their high frequency, and therefore potentially more meaningful, are selected.

For each requirement, the identified keywords are compared against the complete text of all other requirements. If a match is found, then the two requirements are related. In contrast to the subject based relationship, the relationships are not all bidirectional and the developed DSM is asymmetric. For instance, the requirements 2.2.1 and 2.9.14 from the Toho project may have the keywords:

2.2.1: Tooling or fixtures switched during changeover shall attach to a sub-plate in accordance with "single minute exchange die" (SMED) design philosophies.
Keywords: Tooling, Fixtures, Changeover
2.9.14: Fragile Parts (Sensors, plastic parts, plastic gears etc..) or parts touching fragile parts (e.g. gear to gear assembly) must be assembled with tooling incorporating force control (and/or spring loaded mechanisms) to prevent part damage during the assembly.
Keywords: Fragile, Touching, Force Control

These keywords cause a relationship from requirement 2.2.1 to 2.9.14 because requirement 2.9.14 has the word "tooling" within its requirement text. However, this relationship is not bidirectional as none of the keywords belonging to requirement 2.9.14 are located within the text of requirement 2.2.1. Figure 5.4 shows an extract of the keyword based manual creation approach while Figure 5.5 shows the full DSM.

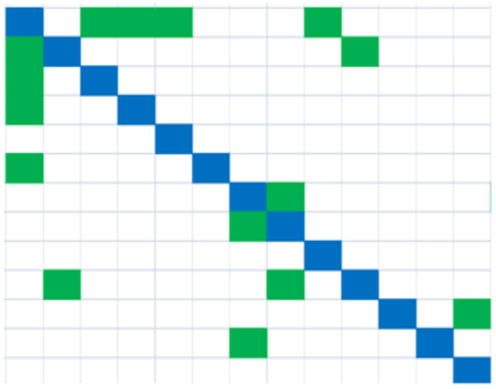


Figure 5.4: Extract of DSM for Keyword Based Study

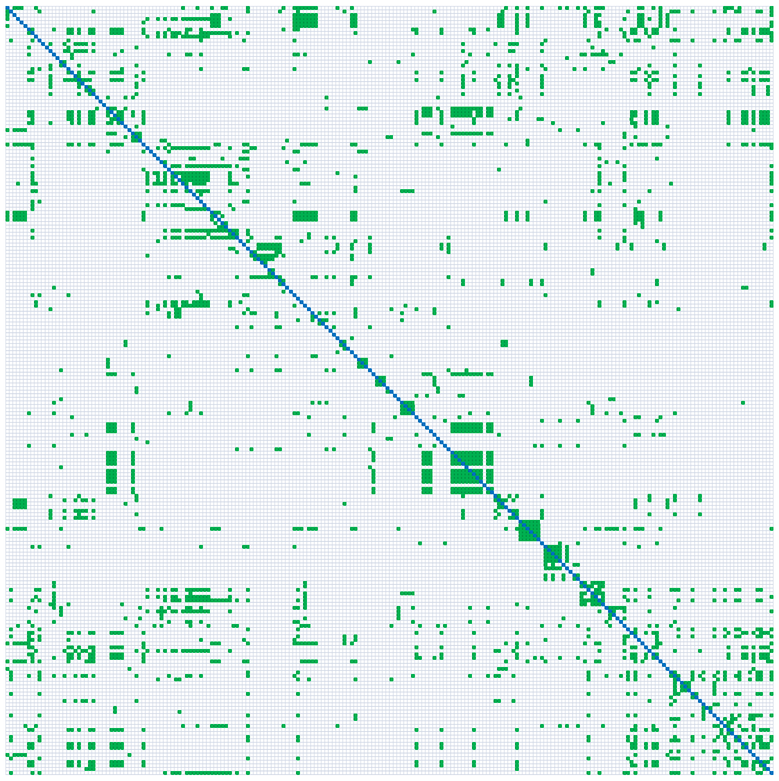


Figure 5.5: Pierburg Keyword Based Manual Creation Study DSM

Both subject and keyword DSM creation methods result in a binary DSM which identifies if a match exist between requirements. An additional DSM that can be generated with the keyword creation is the degree of freedom (DoF) DSM which represents the density of relationships. With the keyword selection, a requirement may possess anything from zero to up to three keyword relaters. To differentiate between the relationship strength, a DoF DSM is developed. The DoF DSM generated for the keyword based manual creation is shown in Figure 5.6. In the figure, the color blue indicates a single keyword match, yellow indicates two keywords and red indicates all three keywords match. As seen, there are very few requirements (with the exception of the diagonal) where all three keywords match.

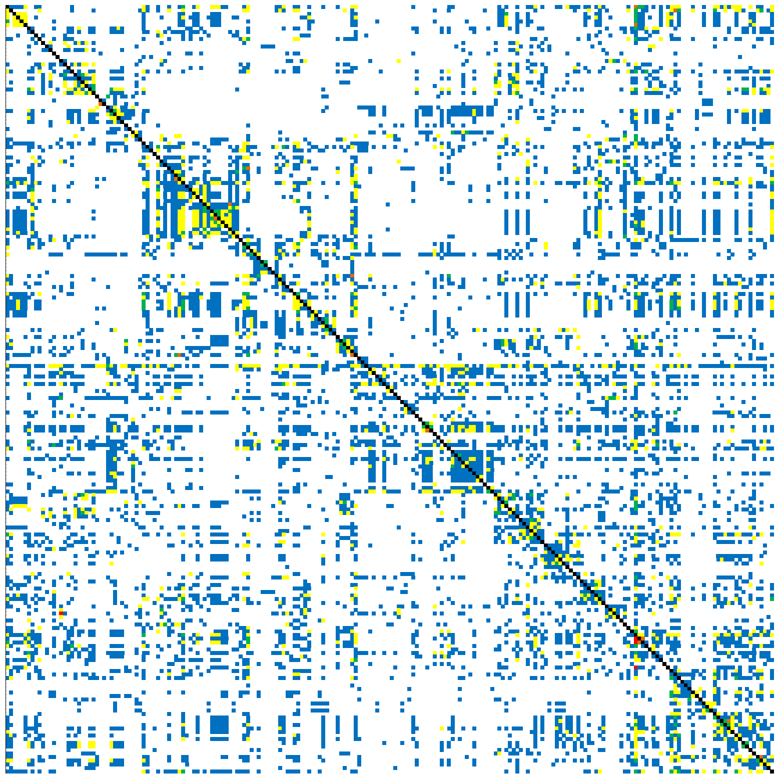


Figure 5.6: Excel Matrix Graphic

The number of keywords and the types of words selected may vary depending on the type of project and breadth of requirements. Further, the format in which requirements are written may also have an impact on this. For example, paragraph type requirements may possess a greater number of keywords than sentence form requirements. Keywords function by relating with any other requirement which possess that keyword in its entire requirement syntax. This results in an asymmetric DSM, in contrast to a subject based relationship DSM. The types of keywords selected will have an impact on the relationships developed and the accuracy of those relationships to predict subsequent change propagation.

Though keyword selection, in its current state is subjective, may be automatically selected as previous research has investigated the use of models to summarize text into specific words. Several keyword selection algorithms make use of word frequency. Terms are extract and if a term appears with a particular subset of frequent terms, the term is likely to have a meaning [75].

5.1.4. Statistical Validation of Keyword Selection

The selection of keywords is subjective, and to some degree may require a level of expertise with the system or familiarity with the requirements document. The goal of the developed syntactical tool is to be capable of functioning with a robust set of requirements documents and be independent of the user expertise. As a result, it is important to measure how consistently keywords are selected from a requirements document. To test this, segments of each of the three requirements document was given to graduate students in mechanical engineering at Clemson University. The students were asked to read through each requirement and select five words that they felt were significant or pertinent to the requirement. In total, there were 37 requirements used from the Toho, Pierburg, and EVRAZ project. Thirteen students completed the study by selecting five keywords for each requirement. The test subjects were not given specific instructions on the selection of the keywords. They were allowed to read the entire document before selecting keywords while others selected the keywords as each requirement was read.

It was important separate projects be used here as each requirement was written by a different customer, ensuring a variety of requirements document. Further, the individuals completing the study were not familiar with the system. If consistency could be shown between requirement selection in such a conservative case, confidence could be placed on keyword selection when used in industrial practice.

The purpose of the test was to identify if there was consistency between how keywords were selected. This indicated that for a specific group of people, the same set of keywords should be selected if keyword selection is consistent, meaning that there was association between how individuals selected keywords. Association here indicates that the selection of keywords is (1) not random and (2) similar to that of others.

To test the selection of keywords, a chi square test is used. The test is used because if the selection of keywords was completely random, it is known which words are selected and how many times their selection would occur throughout the data points (individuals who completed the study). For example, reviewing requirement 2.2.1 in the Pierburg projects states:

> *Tooling or fixtures switched during changeover shall attach to a sub-plate in accordance with single minute exchange die (SMED) design philosophies.*

The possible words that could be selected as keywords here are: tooling, fixture, switch, changeover, attach, sub-plate, SMED, design, and philosophies. Since thirteen data points are observed, each data point will select five keywords. The results of the keyword selection for this requirement are shown in Figure 5.7. As seen in the figure, of the thirteen individuals, all selected the work tooling, while 92% (12 of the 13) selected fixture and SMED. Ten of the thirteen selected the word subplate while about half selected the word attach. The remaining selections were distributed amongst switch and changeover.

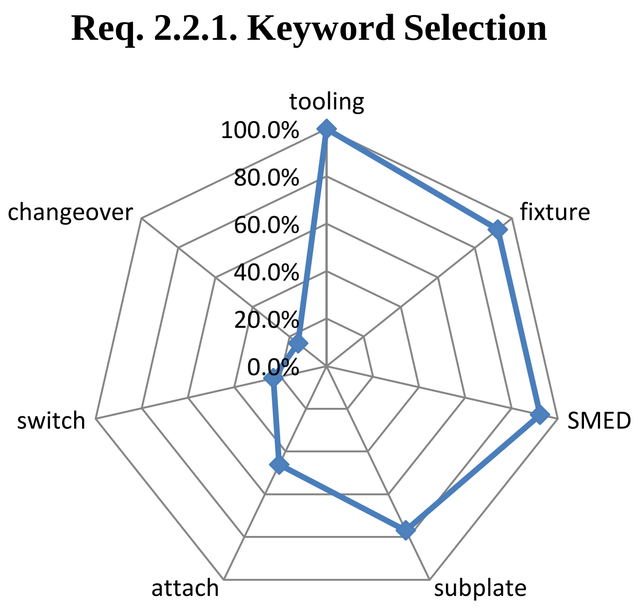


Figure 5.7: Example Keyword Selection Result

If the selection of keywords was completely random, each word would have an equal opportunity to be selected. Since there are a total of nine possible keywords and each of the thirteen data points will select five, then a normally distributed random selection of keywords would result in each word to be selected approximately seven times. This there is an observed amount of selections for each word which can be tested against the theoretical amount. The hypothesis tested here are:

H_0 = Selection of keywords is random and lacks any association between data points
H_A = The selection of keywords is not random and there is an association between how keywords are selected

Since an observed and theoretical selection of words is identified, a chi square test is performed to identify if it is random and lacking association. An $\alpha = 0.001$ is used here as it is an aggressive statistical significance but is needed to ensure quality in the research. The results of the chi square test are shown in Table 5.1. As seen, a **p-value of 0.00138** is observed which concludes there is sufficient evidence that the selection of keywords is not random and there is association involved in how keywords are selected.

Table 5.1: Chi Square Test Results

Chi Square Test Statistic	**P-value**	**Conclusion**
31.05	0.000138	Reject H_0

Though the chi square test identified that the selection of keywords is not random and there is association involved, this does not necessarily indicate that the same keywords will be selected each time. A binomial test is performed to identify if in fact there is association involved and what this association means. For instance, the association may be that users are consistent in not selecting a specific set of words for keywords, which would not contribute to analyzing the consistency of the selected words. As a result, another statistical test is needed to identify if this association is with respect to the selected keywords. To address this, a binomial test is performed to investigate the selection of the keywords between data points.

The purpose of the binomial test is to identify if this association is the selection of keywords between each user. Since only five keywords are used in the syntactical analysis, the test is used to determine the consistency of selection amongst data points. The binomial test results indicate that users have a greater probability of selecting the same keywords to a significance of **p-value of 1.467e-05**.

The results of the statistical analysis reveal that the selection of keywords is not random and there is association between data points. Further, there is statistical evidence to suggest that data points selected the same keywords consistently.

5.2. Development of Syntactical Approach

The syntactical creation approach builds on the success of the previous approach by automating the process, seeking to increase objectivity. The manual creation method and previous study recognized the potential for subjects and keywords to relate requirements in a manner conducive to predicting change propagation. However, the selection was manual, tedious, and was prone to subjectivity related discrepancies. An automated manner for selecting words such as nouns and verbs, while offering the selection of keywords was required. Further, a major limitation of the manual creation process was the lack of granularity between requirements highlighted as potentially propagating and the number of requirements which needed review. A weighting and ranking system is employed in this approach to incorporate strength of relationship between requirements and is used to rank the requirement relationships in order of most likely to change as a result of propagation.

The syntactical protocol, which is detailed in Section 5.3.2, follows a four step process, as seen in Figure 1.1. This process includes (1) parsing the requirements to extract pertinent POS tagging and selection of keywords, (2) identify requirement relationships based on the relaters selected, (3) highlighting the potentially propagated requirements, and (4) perform post propagation analysis to determine ranking of requirements which may propagate. This section will detail the research and analysis performed to ultimately develop this method. Alongside the detailed description of the analysis performed, the program code run on MATLAB will be referred to, located in APPENDIX B, APPENDIX C, and APPENDIX D.

5.2.1. Development of POS and Keyword Relater Combinations

Each requirement is tagged with a POS as illustrated in Figure 5.8, and keywords are manually selected within each requirement that a designer or engineer may wish to incorporate in addition to the POS. Parsing is performed through the Stanford POS tagger and passed into a MATLAB code to use the nouns and verbs for analysis. Upon parsing, each requirement will have five nouns, five verbs, and five keywords. Using the example shown in the previous approach, the following requirement and its POS output are examined:

> *2.2.1: Tooling or fixtures switched during changeover shall attach to a sub-plate in accordance with "single minute exchange die" (SMED) design philosophies.*
> ***Keywords: Tooling, Fixtures, Change over***

*POS Output***: Tooling/NN** *or/CC* ***fixtures/NNS*** *switched/VBN during/IN* ***changeover/NN*** *shall/MD attach/VB to/TO a/DT sub-plate/JJ in/IN* ***accordance/NN*** *with/IN ``/`` single/JJ* ***minute/NN*** *exchange/NN die/VB ''/'' -LRB-/-LRB- SMED/NNP -RRB-/-RRB- design/NN philosophies/NNS*

As seen from the example, the first five nouns selected by the POS tagger are tooling, fixtures, changeover, accordance, and minute. Tooling, changeover, accordance, and minute were denoted as NN (singular noun) while fixtures was denoted as NNS (plural noun).

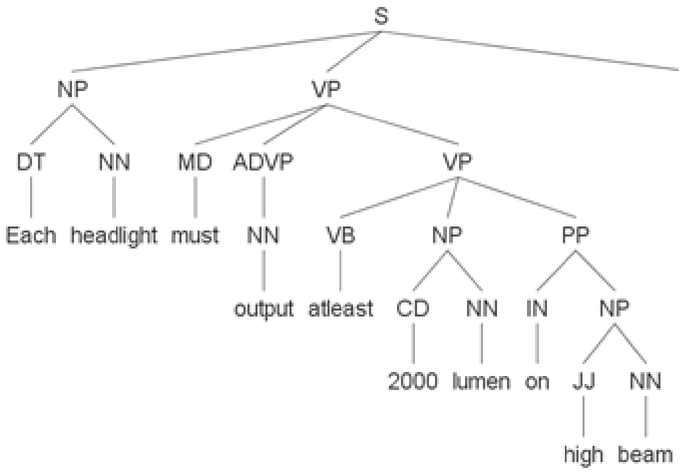


Figure 5.8: Example POS Requirement Tagging

Shown in APPENDIX B is the MATLAB code used to extract noun, verbs and keywords (manually) from each requirement. Since the code was process intensive, a multicore system was used and it is recommended a parallel configuration be used. A progress bar is inserted into the program as well as to monitor the progress and approximate time remaining for the program. These two steps are of significance here as running the program takes approximately two to five days on requirements documents varying from 150 to 214 requirements.

The first step was to load the requirements, keywords, and engineering change item numbers. The requirements document is first loaded into MATLAB. The requirements, which are in excel form, are placed in a matrix so they may be called at a later time. It is important to note that the manner in which requirements are written will have no effect on the performance of the program. MATLAB here must input the requirements through an excel document. Each row on the excel document represents a specific requirement. In this section, requirements may be referred to as “line items” as each requirement will be a line item on a row. The manually elicited keywords are also placed into a matrix for later use. An engineering change (EC) matrix file is created at an earlier point and opened in the program for later use in evaluating the performance of each combination. The EC Matrix file contacts an n *x* n matrix, where n is the number of propagations that we wish to predict. Each cell in the matrix contacts an array of line items used to evaluate the combinations performance at predicting requirement change propagation. The EC Matrix file for the Toho and Pierburg projects are tabulated in Table 5.2 and Table 5.3 respectively. As an example, consider the EC Matrix file for the Pierburg project shown in Figure 5.1. The first column indicates the specific line items which changed and the second column views the propagation which may have occurred due to the changes in the first column of that row. In this case, the first engineering change made changes to requirement line items [2,17,138] which could have caused a change to the second engineering change (requirement line item [17,25]). The second row represents all previous change, the first and second engineering change, and its propagation to the third engineering change, requirement line item 85.

Table 5.2: Toho EC Matrix

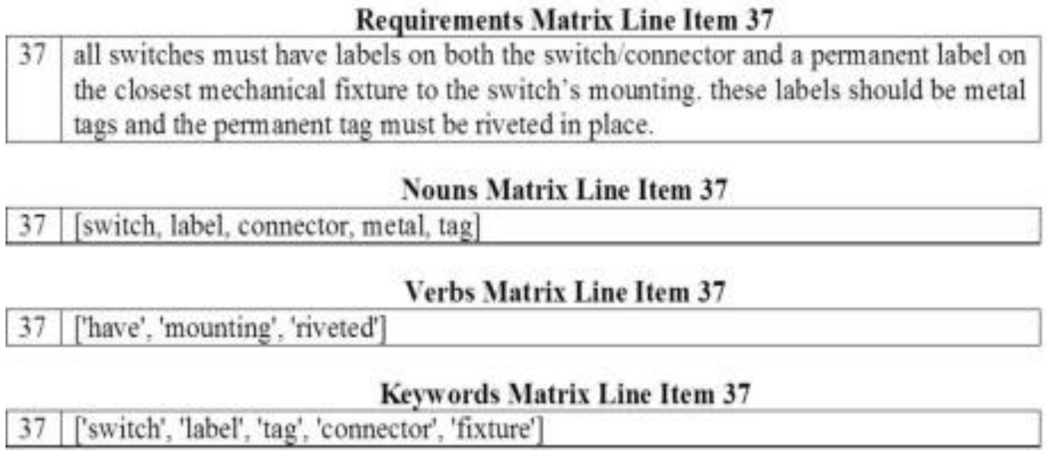
	Prior Changes	Propagated Changes
Predicting Second Engineering Change	{91}	110,111
Predicting third Engineering Change	{91}, {110, 111}	113
Total Requirement Changes Predicted		***3***

Table 5.3: Pierburg EC Matrix

	Prior Changes	Propagated Changes
Predicting Second Engineering Change	{2, 17, 138}	17,25
Predicting Third Engineering Change	{2, 17, 138}, {5}	97
Total Requirement Changes Predicted		***3***

An empty DSM is created with an empty matrix of r *x* r where r is the number of requirements. Further, the number of keyword is defined by determining the number of keywords written for each line item in the keyword excel document. The POS function is run, which is listed in Appendix B.2. Within this function, the nouns, verbs, and keywords for each requirement are extracted. Beforehand, each requirement is manually run through the Stanford parser and the results are listed as line items in an excel sheet. The POS.m function removes any duplicate nouns and verb and develops a matrix for each POS.

The requirements document is placed in a matrix and each of the relater (nouns, verbs, and keywords) have a matrix with five POS for each requirement. An example of this is shown in Figure 5.9 for line item 37. As seen in the example, not every requirement will always have five nouns and verbs. This will depend on the syntactical structure of the requirement. This tool was developed to accommodate this by skipping blanks in relaters. Every requirement had up to fifteen possible relaters (five nouns, verbs, and keywords).



Requirements Matrix Line Item 37

37	all switches must have labels on both the switch/connector and a permanent label on the closest mechanical fixture to the switch's mounting. these labels should be metal tags and the permanent tag must be riveted in place.

Nouns Matrix Line Item 37

37	[switch, label, connector, metal, tag]

Verbs Matrix Line Item 37

37	['have', 'mounting', 'riveted']

Keywords Matrix Line Item 37

37	['switch', 'label', 'tag', 'connector', 'fixture']

Figure 5.9: Example Line Item

There are multiple possible relationships which can be made from the relaters. To identify all possible permutations of relationships, a combination matrix is developed. This combination matrix identifies all possible combinations of the fifteen possible relaters. The total number of relater combinations possible is a simple $2^n = 2^{15} = 32,768$. A combination matrix is developed (extracts shown in Table 5.4). As seen in the extracts, combination 39 makes use of verbs 5, keyword 3, and keyword 4 as relaters. A DSM for each relater combination is developed. The DSMs are later evaluated for their effectiveness in predicting change propagation.

Table 5.4: Extracts of Combination Matrix

Combination number	Noun 1	Noun 2	Noun 3	Noun 4	Noun 5	Verb 1	Verb 2	Verb 3	Verb 4	Verb 5	KW 1	KW 2	KW 3	KW 4	KW 5
39										●			●	●	
1952					●	●	●	●			●	●	●	●	
6142			●		●	●	●	●	●	●	●	●	●		
17995	●				●	●			●			●			
27657	●	●		●	●								●	●	

Four DSMs are developed for each of the combinations. A binary DSM is developed to determine if a relationship between the two relationships exists. This can be measured at a specified depth, such as first or second order. A relationship order DSM is developed to measure the order at which two requirements are related, also termed shortest path. A path count DSM is generated to determine the number of possible relationships at a specific order between two requirements A maximum flow DSM is created to determine all the possible path of relationship between two requirements. Each DSM is evaluated in order to identify those combinations which are consistently performing well, while outputting minimal number of requirements for review (those requirements which a designer has to review for propagation). Each combination and its respective DSMs are filtered to determine the optimal combinations based on their prediction performance. This performance analyzed how both Pierburg and both Toho changes, a total of four changes, spanning the propagation of five requirement changes.

In total, three DSMs (binary, relationship order, path count, and maximum flow) are developed for each of the 32,768 combinations. Since this tool was tested against two heterogeneous projects, different DSMs had to be created for each requirements document. Six total requirement changes are predicted using this tool. In total **655,360 DSMs** are analyzed for their ability to predict the 5 requirement changes (32,768 combinations x 4 DSMs x 5 changes).

5.2.2. <u>Identifying Optimal Combinations</u>

Each combination was evaluated based on its ability to accurately predict the requirement changes and to provide the least noise or superfluous relationships. The program code used to identify the optimal combinations is shown in APPENDIX C. This code used the results from the code discussed in section 5.2.1 and filtered the combinations based on their performance to propagate the five requirement changes from the four engineering changes of both industry studies. Three filters are used in identifying the optimal combinations. The filter MATLAB codes are presented in Appendix C.2, C.3, and C.4. It is important to note that each project had its own set of 32.768 combinations which both, in parallel, ran through these filters. After running through the final filter, the combinations for both projects are intersected to identify which combinations performed effectively on both projects and their respective requirement changes.

The first filter (change identification and depth: filter1.m) is used to determine, first, if the known subsequent changing requirement is related to the initiating requirements and, second, if the changing requirement is found within a path length of four steps. This target of four steps was not initially fixed in the study as different cutoffs were created to analyze the sensitivity in comparison with other filters. The path length of four was the longest path length that still yielded suitable results. Less than 1% of all combinations were eliminated due to this filter. This would increase if the relationship order threshold were made shorter. However, for research purposes, one wants to consider as many possible options in order to ensure that the best performing combinations are captured. Therefore, a conservative filter of path length of four is maintained. A second filter is used to evaluate the false and true positives of the requirement change propagation prediction.

The second filter (true positive minus false positive = filter2.m) evaluates the number of requirements the designer or engineer has review for propagation based on the results of the combination. Since only five requirement change propagations occurred, ideally a tool which offered those five and none other would be preferred. The metric used in the second filter compares the true positive to false positive returned requirements as shown in Equation 1.

$$\frac{1}{TP} - \frac{1}{FP} = F_2$$ **Equation 1**

Where: TP = Average True Positive Relationship Order or Path length

FP = Average False Positive Relationship Order of Path length

F_2 = Filter 2 Scoring

The multiplicative inverse of the relationship orders are used to keep the numbers between 0 and 1. As an example, the first change propagation for the Pierburg project was requirements 17 and 25. If, for a specific combination, the average relationship order between requirements 17 and 25 and the previous requirements changed (2, 17, 138) is 1.67, the inverse of this is 0.599. The false positives that the previous requirements changed (2, 17, 138) identified for possible propagation are similarly analyzed to determine their scoring. Continuing the example, if the average relationship of the false positives is 2.2, the inverse of this is 0.455. As a result, the difference between the two values is 0.144. A high F_2 score indicates a difference between the true positive and false positive relationship depth, which is a more effective combination at determining the true positives at a shorter depth than false positives.

A histogram of the F_2 scores distribution for the Toho and Pierburg studies are shown in Figure 5.10 and Figure 5.11 respectively. As seen from the figures, most of the F_2 scores have a central tendency near the 0.03-0.05 range. The initial filtration was those values less than zero, which indicated the combination, was identifying false positives at a shortest path length than the true positive relationships. Since higher scores were an indication of a more effective combination at predicting requirement change propagation, while minimizing the occurrence of false positives (or increasing relationship order), the top 10% of scores were selected while the remaining were cut in this filter. While this number is arbitrary selected, multiple thresholds were evaluated during the testing.

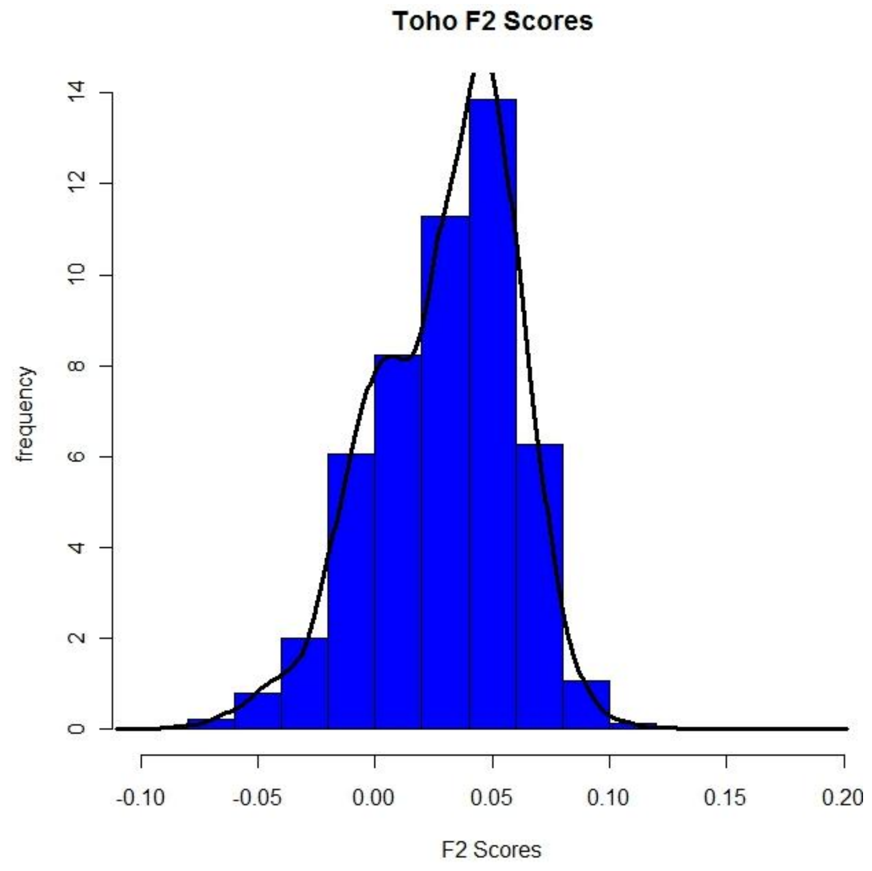


Figure 5.10: Histogram of Toho F_2 Scores

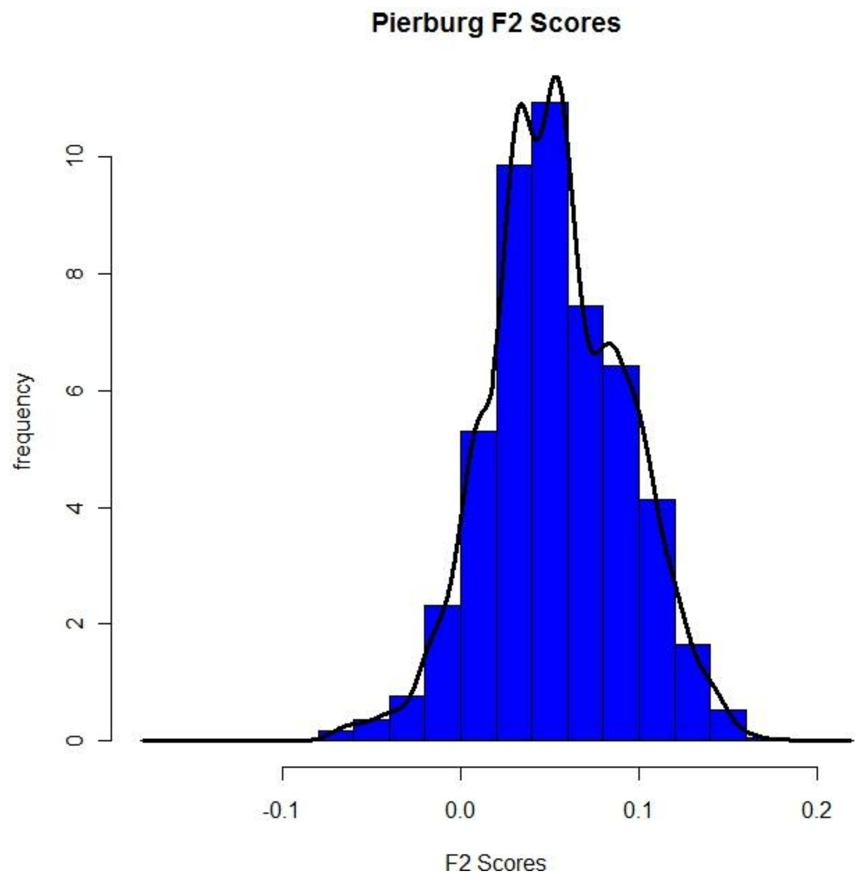


Figure 5.11: Histogram of Pierburg F_2 Scores

The histograms developed for the Toho and Pierburg study revealed an interesting observation. Both histograms which are separate distributions from two different projects indicate words have a higher tendency to relate requirements in a manner conducive to requirement change because their central tendency sits greater than one. Before analyzing both populations, it is important to recognize that both are separate distributions, an indication of the heterogeneity of the projects. Though they may appear as normal distributions, this is due to the large number of data points presented [161]. As discovered by the central limit theorem, any population with a large number of data points will have an approximate normal distribution [162,163]. As a result, testing was needed to view how the distributions for the Toho and Pierburg varied. Further, proving the studies are heterogeneous will reveal the system is capable of functioning on requirements documents which may vary in how nouns, verbs, and keywords are used in a sentence.

The first step is to perform a Wilcoxon test used to analyze data from studies with repeated matches' subjects [164,165]. The goal of this is to determine whether the scoring of F_2 changed significantly between requirements documents. The results of the Wilcoxon resulted with a test statistic value of 380,922,959 and a p-value of 2.2e-16. The p-value of 2.2e-16 is not exact but is used because the lowest tabulated p-value available for this test is 2.2e-16.

The test identifies that both histograms are statistically different from one another. The advantage to having different distributions is to identify that the tool was (1) not generalized through two similar projects and (2) the developed tool is capable of supporting varying requirements documents. The only difference between the Toho and Pierburg from a protocol standpoint is the types of nouns, verbs, and keywords selected for both projects, which is a reflection of the requirements document.

Further, another interesting observation is their similar appearance and tendency. Another statistical test was performed to identify if using POS were relaters conducive to change propagation. As noted, the F_2 is a score of the combinations to yield a shorter path length with the true positive propagation relationships minus the path length with the false positive propagation relationships. A positive score indicates the words can be used to identify change propagation at a shorter path length while a negative score indicates the opposite. If POS are not effective relaters between requirements where change propagation occurred, there would be a central tendency around a score of 0. Effectively, this would be a random distribution, centralized around the score of zero. However, it is hypothesized that the POS can in fact, for two statistically heterogeneous requirements document, be used as relaters to predict change propagation.

To reveal this, a statistical hypothesis test is performed to test if the F_2 scores of both studies equal to zero. The results of the test, shown in Table 5.5, indicate that there is high statistical evidence the F_2 score for both the Toho and Pierburg are not equal to zero, and since they both sit above zero, the mean for both studies is greater than zero.

Table 5.5: T-Test (H_o: $\mu = 0$)

	N	**Mean**	**Std. Dev.**	**Min**	**Max**	**P – value**
Pierburg F_2 Scores	32768	0.0536	0.0388	-0.1653	0.2058	2.2 e-16
Toho F_2 Scores	32768	0.0297	0.0306	-0.0998	0.1920	2.2 e-16

The statistical tests performed here reveal that the F_2 score distribution for both studies are different, which is a residual of the type of requirements document. Further, another statistical test identified that the use of a requirements noun and verb POS and the manually selected keywords are an effective relater to predict change propagation.

The third filter (true positive over false positive = filter3.m) makes use of a similar metric using the true and false positives values calculated for each combination. This filter removes yet another set of combinations based on the true positive over false positive values, as shown in Equation 3.

$$\frac{\left(\frac{1}{TP}\right)}{\left(\frac{1}{FP}\right)} = F_3 \qquad \textbf{Equation 2}$$

Where: F_3 = Filter 3 Scoring

Equation 2 used a subtraction approach between the true and false positive to determine which combinations yielded a great difference between the true positive and false positive relationship order. Another filter was needed because a subtraction lacks enough information regarding the combination score. For instance, combinations 100 may have the following scores: TP=0.5 and FP = 0.4. Combination 200 may have the following scores: TP = 0.8 and FP=0.7. While both combinations yield the same difference in true positive to false positives, they vary in the order of the relationships. As a result, a ratio is taken to evaluate the separation in relationship order between the two values.

Filter 3 also used a threshold that was iteratively developed for each of the distributions. Shown in Figure 5.12 and Figure 5.13 are the Toho and Pierburg F_3 scorings respectively. The central tendencies for both graphs are above a score of one. Likewise to the F_2 scoring, this indicates that the true positive relationship path length was shorter than that of the false positive.

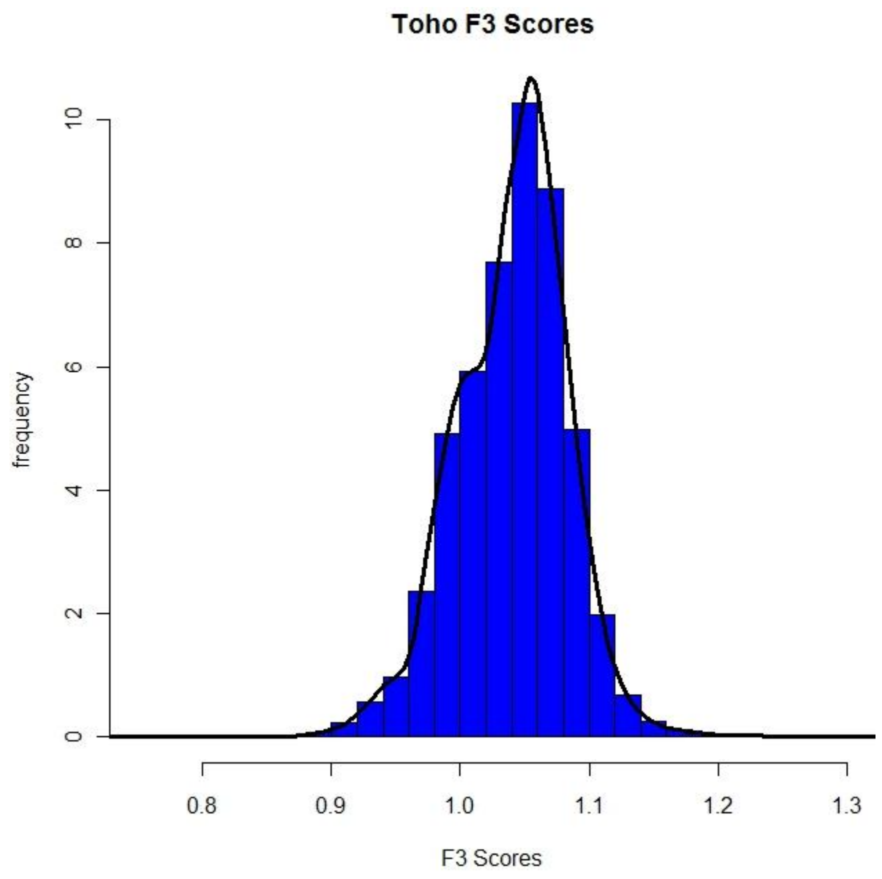


Figure 5.12: Histogram of Toho F_3 Scores

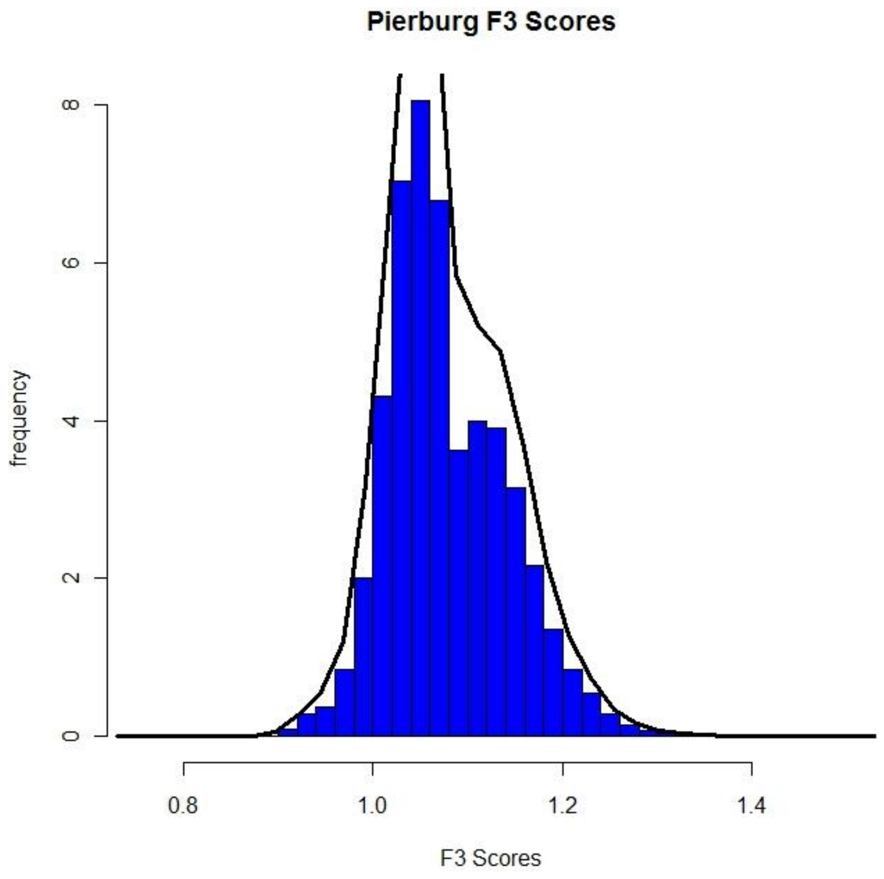


Figure 5.13: Histogram of Pierburg F_3 Scores

The second and third filter use thresholds to cut combinations from the combination set. Thresholds are developed based on a statistical method using graphical models [166] of the sorted scores of the second and third filter to determine a cut off for both. Through iterations, it is observed the appropriate cutoff value to be approximately the top 5% of scores for each filter. Since this approach will vary between the Toho and Pierburg combinations as each one yielded different filter scores, the thresholds varied slightly.

Using the thresholds developed and after filtration, Pierburg maintained 134 combinations (0.4%) and Toho maintained 96 combinations (0.3%). The results yielded revealed three combinations which overlapped between both Pierburg's and Toho's filtered combinations. The combinations are tabulated in Table 5.6.

Table 5.6: Matching Top Results for Pierburg and Toho

Combination number	Noun 1	Noun 2	Noun 3	Noun 4	Noun 5	Verb 1	Verb 2	Verb 3	Verb 4	Verb 5	KW 1	KW 2	KW 3	KW 4	KW 5
5													•		
8197		•											•		
24578	•	•													•

The three combinations were overlaid to develop a single DSM. A weighting was incorporated within the combination to account for relationship order. The overlaying process combines the three DSMs formed by each combination and combines them into a single, summed DSM. The order of each relationship is weighted based and the scores are summed to develop one overlaid DSM.

5.2.3. Post Propagation Weighting and Scoring

Recognizing the importance of requirement relationship order from the manual creation study, the syntactical approach extended requirement order analysis by incorporating weightings. A weighting is developed based on the results of the Pierburg and Toho studies. A 9-3-0 scoring system was found to yield the most accurate results in both studies, where a first order relationship had a score of 9 and a second order a score of 3. Third order and beyond were not score because of their inability to assist in predicting change propagation in the study. Further, at an order of 3, the DSM was highly saturated with superfluous relationships which distorted the propagation data.

It is important to note that this scaling may be altered for systems of varying complexity; however this was not investigated in this research. It is hypothesized system of high complexity will require greater relationship orders. An overlay DSM is created by summing the relationship order weighting for each requirement relationship in the DSMs generated by the three combinations. The MATLAB code used to perform the overlaying is detailed in D.2. An example extract of the three combinations combined to form an overlay DSM is shown in Figure 5.14, where 9-3-0 are used to indicate a first, second and third order relationship respectively.

$$\overset{\textbf{Combination 5 DSM}}{\begin{array}{|c|c|c|}\hline 9 & 3 & 3 \\ \hline 0 & 9 & 3 \\ \hline 9 & 3 & 9 \\ \hline\end{array}} + \overset{\textbf{Combination 8197 DSM}}{\begin{array}{|c|c|c|}\hline 9 & 3 & 9 \\ \hline 3 & 9 & 9 \\ \hline 0 & 0 & 9 \\ \hline\end{array}} + \overset{\textbf{Combination 24578 DSM}}{\begin{array}{|c|c|c|}\hline 9 & 3 & 0 \\ \hline 3 & 9 & 9 \\ \hline 0 & 9 & 9 \\ \hline\end{array}} =$$

$$\overset{\textbf{Overlay DSM}}{\begin{array}{|c|c|c|}\hline 27 & 9 & 12 \\ \hline 6 & 27 & 21 \\ \hline 9 & 12 & 27 \\ \hline\end{array}}$$

Figure 5.14: Example of Overlay DSM

As noted previously, there are multiple types of DSMs. The overlay DSM may be created for:

- Binary Overlay DSM: A binary summation of where relationships existed in each combination.
- DoF DSM: Sum all the possible relaters used in each combination
- Relationship order (shortest path) DSM: Sum of the shortest path between all combinations. Example shown in Figure 5.14
- Max flow DSM: Summing all the possible avenues for simultaneous data flow between requirements in all three combinations.

An example of all the DSMs discussed is shown in the EVRAZ study DSMs shown in APPENDIX H. This extensive way of building relationships through Overlay DSMs was explored for their utility in predicting requirement change propagation. It was identified that the Relationship Order Overlay DSM was most consistent and simplest to explain as it was clear as to how propagation occurred through using this type of DSM. An overlay DSM simple sums the cell values for each combination DSM into a single DSM, example shown in Figure 5.15.

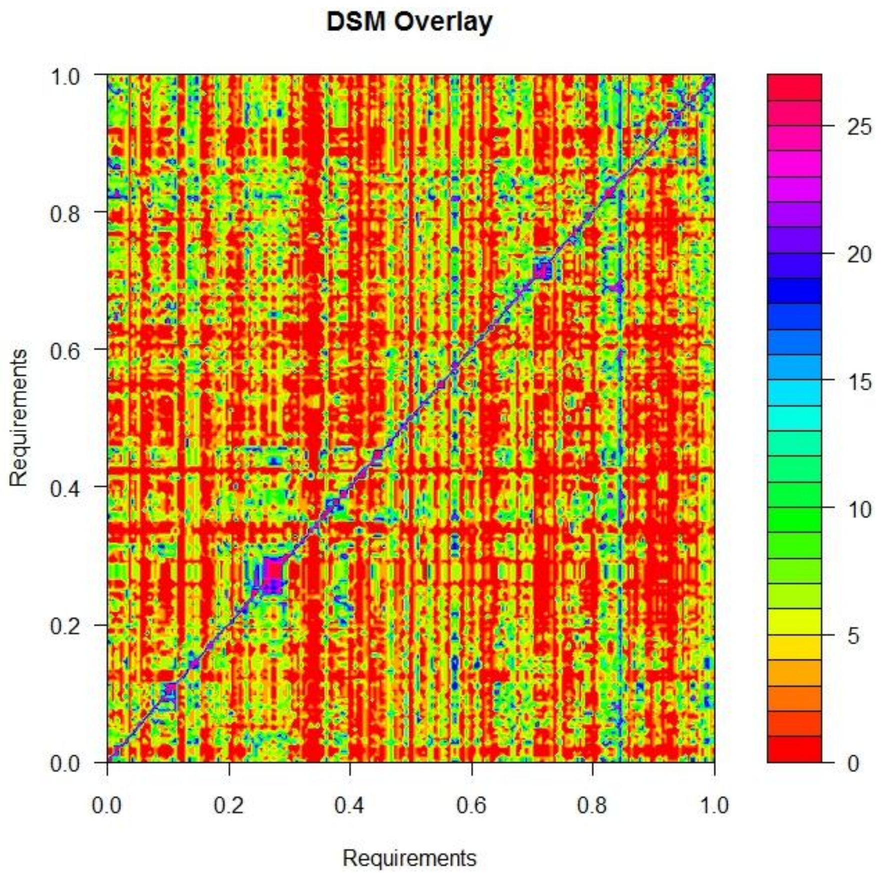


Figure 5.15: Pierburg Overlay DSM

The syntactical approach predicts requirement change propagation using all previous requirement changes, cumulatively. Whereas one change instance may cause another change instance to occur, it is important to predict change propagation that occurs from multiple previous changes. In doing this, propagated changes are analyzed by predicting change resulting from all previous changes up to that point. As a result, a change may be propagated from all previous engineering changes and their associated requirements. Consider the following example: The BMW X6 suspension system is designed to support a given weight. A change is made to the rear seat requirement, supporting three passengers in the rear seat, as oppose to the existing two rear passenger configuration. This change alone, with the addition of another passenger, does not merit a change in the suspension of the vehicle. Another change, changing the fuel tank capacity from 16.0 gallons to 22.5 gallons is made to support greater vehicle range. This change alone will not merit a change in the suspension capability of the vehicle. However, both changes combined, will require a change to the suspension because of the cumulative additional weight. This must be accounted for in the change propagation to ensure change stemming directly from one requirement, a set of requirements, or separate requirements at different times is accounted for.

To ensure the significance of the requirement which caused the change is amplified (as to not treat equally with other requirements) in the analysis, a root mean square (RMS) approach is taken. RMS is a fundamental measurement of magnitude in most fields such as statistics [167] and physics and is often termed the quadratic mean of data [168] The quadratic mean is the square root of the arithmetic mean of squared values [169,170]. It is primarily used as a variation from the arithmetic mean when weighting must be considered and are of great importance between values, such as in this study. RMS is used here to maximize the significance of first order relationships between previously changed requirements while incorporating the other requirements which may not be of great significance but must be maintained for cumulative propagation. The scoring for this propagation is calculated as followed in Equation 3. Each previous requirement changed is RMS summed to find a propagation score. As noted, the weighting is based on the relationship order with the previous requirement, which is based on a 9-3-0 scale. Effectively, scores which share any first order relationships with previous requirements changed will have a high propagation score without neglecting the effect of second order relationships as well.

$$S_{Propagation} = \sqrt{\frac{{W_{R1}}^2 + {W_{R2}}^2 + {W_{R3}}^2 + \ldots + {W_{Rn}}^2}{n}} \qquad \textbf{Equation 3}$$

Where: S = Propagation Score for potentially propagated requirement
W_{Ri} = Weighting of previous requirements i based on relationship order
n = number of previous requirements

The MATLAB code used to develop a score for each requirement relationship is detailed in APPENDIX D. During propagation analysis, every requirement is scored using Equation 3. This scoring is performed to each of the requirements to identify which requirements, based on the RMS score, has the highest chance of propagating forward because of their relationship with the requirements changed. This scoring will incorporate self-propagated requirements as well, meaning requirements which were changed, may change again at a later time. The result of the propagation analysis on each requirement is ranked amongst the remaining requirements. The result is a ranking of requirements, based on their score, which high the greatest likelihood to propagate due to their relationship with the changed requirements.

A depth of review term is introduced which determines how much of the requirements document must be reviewed before reaching the requirement which did in fact propagate. For instance, during a retrospective study, if a requirements document has 200 requirements and the propagated requirement scored as the 18 highest, this means the designer or engineer has to review a depth of 9% of the requirements before finding the requirement which did in fact change. The goal is to decrease the depth of search a designer or engineer has to perform before identifying the requirements which propagate forward.

5.3. Syntactical Creation Approach

The syntactical approach makes use of a POS tagger to identify specific nouns and verbs within the requirement. Since not all the nouns and verbs can be used as they would result in overpopulating of the relationships, only the first five nouns and verbs are selected for subsequent processing. Five nouns and verbs were initially selected, however it was identified through subsequent analysis that two would suffice and any additional POS would be overpopulating the DSM [76]. The manual selection of keywords is maintained in this study to include words that would not naturally be tagged as nouns and verbs. The syntactical approach makes use of the POS tagger developed by the Stanford Natural Language Processing Group [171,172].

The syntactical approach is taken here because requirement sentences are not structurally different than other sentences, following the same grammatical rules [173,174], allowing for a robust syntactical analysis. Though grammar has multiple parts of speech, such as nouns, verbs, adjectives, pronouns, and adverbs, this approach is only concerned with nouns and verbs. Nouns were selected to maintain the subject based approach from the previous study and enhance it by implementing it in an automated manner. Verbs were introduced to identify if the function of the requirement was of significance to change propagation.

5.3.1. Assumptions

Only five nouns and verbs are used from the POS tagger. The nouns and verbs selected are the first five to appear in a sentence. As a result, it is assumed the nouns pertinent to a requirement are those which are more likely to appear in the beginning of a sentence. Though the sequence of the set of nouns may not be important, the set of nouns used in the analysis are of significance. Though the use of manual keywords prevent the system from being completely automated, they assist in selecting those words, which may include additional nouns and verbs, that were not selected in the POS tagging. For instance, a pertinent noun may be of importance but resides as the eighth noun in the sentence. Since only five nouns are used, it is important this one is included through another avenue, such as keywords.

5.3.2. Syntactical Creation Protocol

The development of the syntactical tool allows for a systematic approach for inputting requirements and chance instances to output requirements which may propagate. The syntactical approach followed a systematic, partially automated method for developing relationships between requirements. This process is partially automated because the designer or engineer is required to manually develop keywords for each requirement. The implemented tool of the approach is shown in Figure 1.1, which operates by (1) parsing requirements to prepare them for analysis, (2) performs relationship matching based on POS tagging and keywords, (3) selects potentially propagating requirements, and (4) performs propagation analysis to identify potential requirements changed due to propagation in ranked order.

The input of the tool is a requirements document of any format, acceptable in any textual form. It is important the tool is robust enough to accept different types, both contextually and format, of requirements to ensure it is appropriate in all design applications. The requirements are parsed to extract all nouns and verbs and keywords are selected for each requirement. Using the optimal combinations found and detailed in Section 5.2.2, an Overlay DSM is generated. The scoring metric algorithm is used to determine which requirements are most likely to propagate due to their relationship order. The scores are ranked to identify to the engineer or designer which requirements to review.

5.4. Measures and Evaluation of Connectedness

When evaluating the approaches presented, graph theoretic can be used to understand how requirements relationships are evaluated. Figure 5.16 is used as a reference in this section. Each requirement is interpreted as a node within the graph defined by the DSM. The edges between nodes represent the relationship between requirements. Some nodes are uni-relationship, such as between nodes A and B, while others are bi-relational such as C and D.

Graph theory offers several avenues for exploring the relationship between two requirements and how they can be evaluated, however the term path is most popular. Path is defined as a sequence of connected nodes and edges between two points such that no node or edge is repeated [175]. Three measurements on the path properties between nodes are addressed in this research: shortest path length, path count, and maximum flow capacity. Since graph theoretic will be used to evaluate connectedness between requirements. Each node is related to other nodes through edges.

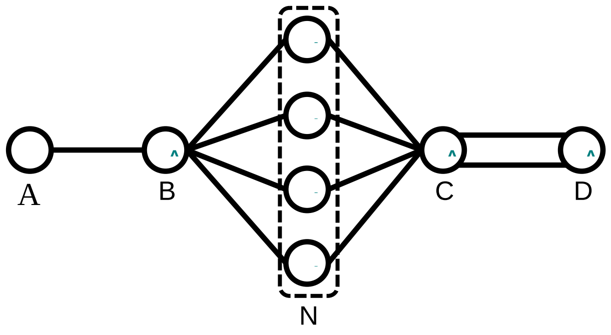


Figure 5.16: Example Graph [33]

The shortest path length between two nodes is a measure of the minimum number of edges that must be traversed to travel from one node to another [176,177]. For example, in Figure 5.16 nodes A and B have a shortest path length of one as they are directly connected, through a single edge. Examining another example, nodes B and D have a path length of three: B to N, N to C, and C to D must be traversed. Shortest path may also be termed relationship order in the context of this research. The relationship between A and B here is first order related, whereas the relationship between B and C is second order related.

The second measurement used is path count, which is the combinatorial evaluation of the number of possible paths between any two nodes [175]. Returning to the example in Figure 5.16, it can be seen that there is only one possible path between A and B, while between B and D there are as many possible paths as there are nodes in layer N that act as intermediaries.

The last measure, maximum flow capacity, is a network property on the amount of information that can flow between two nodes at any one time [178]. It is the measure of the maximum number of paths that may be simultaneously active between the two nodes. Returning to Figure 5.16, while there are N possible paths between A and C, only one can function at any given time because of the single edge between A and B. Whereas nodes B and C can have up to N number of flows at any given time.

Not all the metrics discussed in this section will be used in evaluating requirement change propagation. The manual approach is limited in using some of the graph theoretic approach while syntactical approach is able to use them.

5.4.1. Manual

The initial manual creation method used a simple evaluation of relationship order (path length) to identify if requirements related, regardless of relationship order. In this approach, the relationship order and path count were both considered. As a preliminary study, it was important to identify if change propagation could be predicted using a relationship between the initial requirement and the subsequently changed requirement. If a relationship did exist, the order of this relationship was noted. The manual creation approach output was a simple relationship order. Further, to differentiate between potential requirements for propagation, path count rankings are developed and such requirements are sorted. For added granularity, requirements with high path counts have a higher chance of propagation than those of low path counts.

5.4.2. Syntactical

A scoring system is used in the syntactical approach alongside weighted relationship orders to identify change propagation. This scoring system is developed through the relationship order DSM which identifies the order of relationship between requirements (shortest path) and weights each based on a 9-3-0 system. A RMS scoring algorithm is then used to differentiate between requirements and develop a scoring system which may then be ranked. These rankings are evaluated against observed instances of propagated change by the depth of search required to identify those requirements.

CHAPTER 6.
INDUSTRY CASE STUDY BACKGROUND

This need to manage requirement change is apparent in industry. Within design, it is known requirements change over the span of a project [18,60]. Most design processes will accommodate this through requirement iterations within the requirements elicitation and management. Sequent Computer Systems Company investigated five hundred managers and found 76% had experienced project failure as a result of requirement change [28,179]. Many of the costs involved with managing requirements are a result of change that occurs and the lack of preparing for such change earlier. The greatest proportion of affecting requirement costs can be traced to change management [180,28].

It is assumed, in this research, that all engineering change may be related back to a requirement or set of requirements that are affected by this change. For instance, a change in suspension travel may affect all requirements relating to the suspension and spatial boundaries. Further, because a requirement is affected does not mean a requirement must change, as a requirement may be able to absorb change. Referring to the example, a requirement relating to the mounting of the suspension is affected because of the change, but does not necessarily warrant a change as this requirement can still be satisfied.

This research will present three industry projects in which nine change instances causing ten requirement change propagations. The key takeaway is to view how this change could have been predicted, managed and handled with a tool such as that proposed by this research. The first two studies presented, Toho and Pierburg, analyzed using a manual based approach. The results of the approach motivated the need to develop an automated tool. A syntactical approach was developed through the Toho and Pierburg changes which is validated against a third industry project, EVRAZ. Further, Toho and Pierburg were later analyzed using the syntactical approach to compare against the manual approach.

An introduction to the case study approach and background on the corporation where the study was performed is presented. A method is presented for which the prediction approach is able to transition from the physical to requirements domain where the propagation and prediction is performed.

6.1. Case Study Research and Questions

The objectives of this study align closer with those of case study analysis than user studies in which patterns are sought that might be suggestive and foundations for subsequent experimental studies [181,182,183]. Case study research is an empirical inquiry that investigates a phenomenon within its practical context [184] Case studies also examine the behavior of designers and engineers with a system [185], an important component of this research. This research began as an exploratory study into the use of requirements as change propagation mediums and has led to the development of a tool used to predict change propagation. Further, an instrumental approach was taken as the research sought to understand beyond what was obvious in change propagation [186] and develop an advance understanding of the phenomenon [187]. As a result, case study research was ideal because it pertained to research which required a holistic in depth investigation [188].

The specific question of importance here was to explore if the requirements domain could be used to predict change propagation in both other requirements and the physical domain. If so, can engineering change propagation be predicted systematically through the use of a requirements change propagation prediction tool?

6.2. Corporation Background and Change Protocol

The study was performed on Automation Engineering Corporation’s (AEC) design process to identify the potential of requirement as change predictors. An initial study of their in house data management system was performed to view the operation of the system and identify if deficiencies existed. This system contains files stored for each project and any information that may be relevant to the project such as initiation documentation, requirement specifications, vendor information, task and activity allocations, and budget updates. AEC stores all of the project files within a designated file directory where multiple users may access the system to input, revise, and update information. The case study focused on the means in which project requirements are managed and analyzed the propagation of requirement change. A preliminary analysis of the system was reported in a conference proceeding [23]. The results were used on a subsequent study which analyzed the requirement change propagation of the project and is reported in a separate proceeding [24].

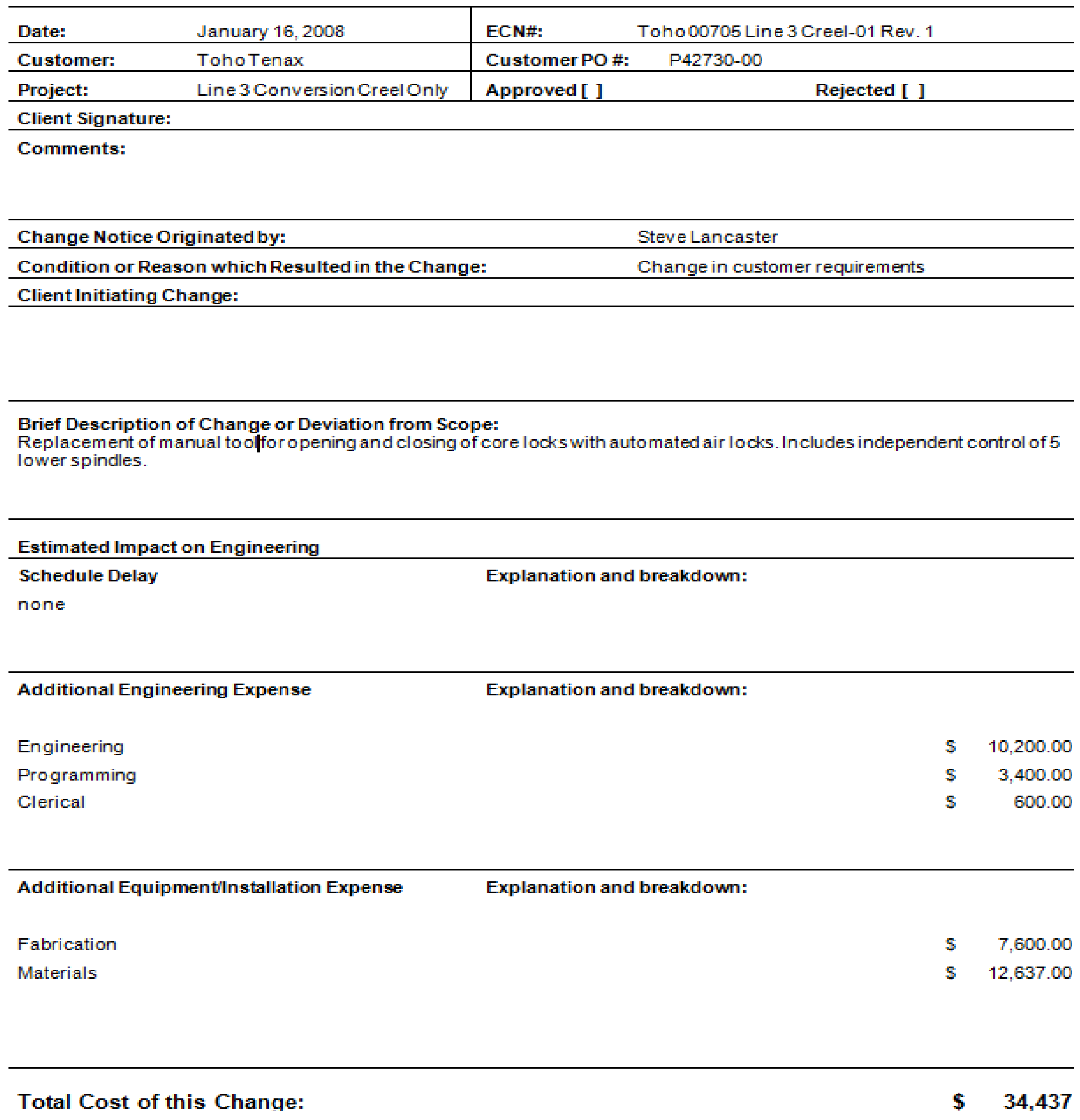
An important observation identified through the preliminary analysis of the system was the lack of requirements linking. Requirement linking here describes the link between requirements and information within the project files. This may be directly, linking immediately back to the requirements, or indirectly, linking back to the requirements through other data in between. The lack of requirements linking becomes problematic as there exists no means of measuring change propagation. Requirement linking is important for this study as all design activity and tasks are in place to address a specific requirement. Without identifying this specific relationship, propagation is difficult to analyze. All reengineering changes documented within AEC is found in a engineering change notification document, which was of significant interest to this research as it contained much of the information and data needed to perform the propagation analysis.

AEC is an appropriate target for this type of study because they offered heterogeneous projects which could be investigated and the projects were large, yet manageable. As this tool is intended to assist designers and engineers in industry, it was important to ensure it was developed through and validated against industrial applications. The potential benefit and readily available information at AEC made it an ideal industrial source for case studies. Multiple studies pertaining to this research and from AEC have been published in conference proceedings and journals.

The current AEC requirements change process entails completing an engineering change notification (ECN) form which contains metadata concerning the change and the associated requirement. The change notification form is seen in Figure 6.1. All data pertaining to engineering change was localized within ECN forms documented by the engineers. There is external information located in emails between the client and corporation where discussion of the change takes place. When a change is initialized, the corporation collects this information from the client and summarizes it in an ECN. This ECN form is exchanged and negotiated with the client until a final change is approved by both parties. The changes in this study were all initiated by the customer.

In order to study change propagation, the documents which pertain to engineering change are sought. The specific detail that is analyzed within each ECN is the cause, date, and requirements effected by each change. A change is initiated when a manager within the corporation or the client identifies a change required in the system. For example, the client may wish for their manufacturing equipment to carry more pallets on its line. The change starts through an exchange of conversation between the corporation and the client. At this time, an ECN form is documented which details the change information and all related monetary and time delays. If this is approved by the client, a permanent design is developed to address the change in the final design of the system. The ECNs must be documented by the project or operations manager of the project who is responsible for contacting the client to ensure ECN completeness and approval. Though the ECNs are accessible to all associates for viewing, the initiator of the ECN is the only associate allowed to make any changes to it.

This form is authored by the project manager, and approved by upper management within AEC and the client. As seen in Figure 6.1, within this form is pertinent information such as the date the ECN was initiated, the client or customer, and the specific project. Each ECN has an ECN# associated with it so it may be easily tracked. Further, an ECN does not require approval to be documented. Many ECNs are initiated but never implemented. Each form has a designation to indicate whether this ECN has been approved or rejected. This specifically documents the personnel involved when initiating this ECN. The type of change that took place is also documented; however in all instances it was found that a "change in customer requirements" was the condition for change. A description of the change is written in the form that explains the specifics of the change so it may be implemented by the engineers. This description is a brief of what must take place to satisfy this ECN. A section titled "Impact on Engineering" is located within the form that entails the delays and expenses that will result from this ECN. The delay may come in the form of time required to completing a specific sub system or developing a component. Another type of impact documented within the form is the engineering expenses incurred. This includes expenses such as additional manufacturing, assembling, clerical, or programming. A similar category is the equipment and installation expenses which details additional expenses such as material or fabrication. Each ECN indicates a total cost for the change. When AEC completes this form, it is sent to the client so it may be approved before any changes can be implemented.



Date:	January 16, 2008	**ECN#:**	Toho 00705 Line 3 Creel-01 Rev. 1
Customer:	Toho Tenax	**Customer PO #:**	P42730-00
Project:	Line 3 Conversion Creel Only	**Approved [ ]**	**Rejected [ ]**

Client Signature:

Comments:

Change Notice Originated by:	Steve Lancaster
Condition or Reason which Resulted in the Change:	Change in customer requirements
Client Initiating Change:	

Brief Description of Change or Deviation from Scope:
Replacement of manual tool for opening and closing of core locks with automated air locks. Includes independent control of 5 lower spindles.

Estimated Impact on Engineering

Schedule Delay	**Explanation and breakdown:**
none	

Additional Engineering Expense	**Explanation and breakdown:**	
Engineering		$ 10,200.00
Programming		$ 3,400.00
Clerical		$ 600.00

Additional Equipment/Installation Expense	**Explanation and breakdown:**	
Fabrication		$ 7,600.00
Materials		$ 12,637.00

Total Cost of this Change:	**$ 34,437**

Figure 6.1: AEC's Engineering Change Notification (ECN) Form

The ECN is analyzed to identify which requirements this change effects. This required viewing the client requirements and identifying where a requirement may be influenced by this type of engineering change. The author of the research familiarized himself with the requirements as to correlate each ECN to a requirement or set of requirements. The requirements are written in a hierarchical format making the identification of relevant requirements convenient. For example, a change relating to wiring would be located within the electrical controls grouping of the requirements document; however this was not the case for all ECNs.

A typical change propagation that occurred at the corporation is illustrated in Figure 6.2 along with the proposed method to predict change propagation. Currently there were instances where change propagation did occur unnoticed and an ECN is implemented at a later time. In the proposed method, the DSMs are used to predict the subsequent changes that may occur. It is important to note not all ECNs are due to propagation, some may come in the form of a change independent of any prior changes [74,189].

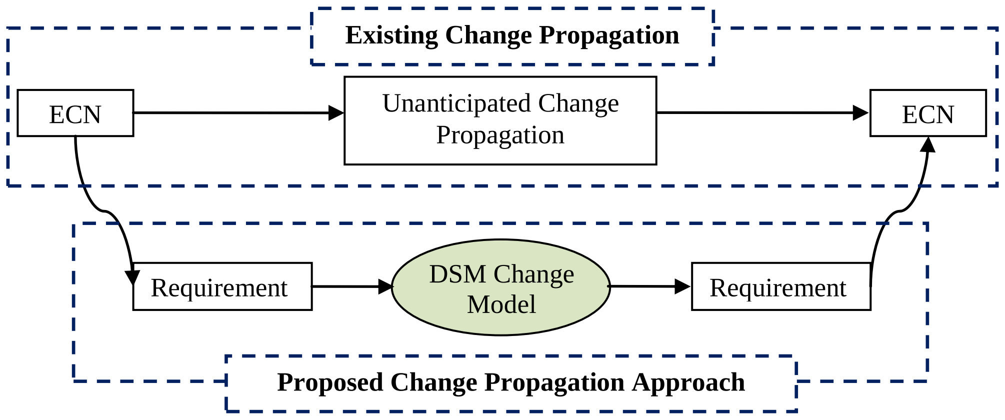

Existing Change Propagation
ECN
Unanticipated Change Propagation
ECN
Requirement
DSM Change Model
Requirement
Proposed Change Propagation Approach

Figure 6.2: Proposed ECN Prediction Method

If the ECN results in a change in requirements, these requirements are processed through the change propagation tool. This method includes propagating the requirements using a DSM for each affected requirement to predict subsequent changes. From this DSM, the potential requirement(s) which may change as a result of an initial requirement change is identified. As this study is of a historical nature, subsequent ECNs are analyzed to view if they have affected requirements which were predicted by the DSM.

The ECN forms are used to extract information from the Toho, Pierburg, and EVRAZ project. The first project investigated was the Toho, which made use of a subject based relationship. The Pierburg project advanced the ability to predict requirement change propagation through the use of manually selected keywords. Using both manual approaches, a syntactical approach was developed to which incorporated both the Toho and Pierburg project. The syntactical approach was validated against the EVRAZ project.

It is important to note that not every engineering change can be predicted as some changes are independent of all previous changes. Only the changes which were propagated will be analyzed. It is known which of the requirements are propagating from feedback from the client, AEC, and documentation in the project. This documentation is dependent on the project and the associates involved with the project. However, when an ECN is completed and is revisited at a later time, this is an indication of change propagation due to a change that was not recognized at the time the ECN was developed. Each project varies in documentation and will be described in more detail during each study.

It is equally important to note that some changes propagate without the corporation's recognition. Though this will not be specifically studies, it is apparent in some of the change prediction that there was a relationship between the initial and propagated requirement. An example of this is the prediction of ECN03 presented in Section 7.1.1.

6.3. Manual Approach Prediction

Two projects are studied in the manual approach, the Toho subject based and Pierburg keyword based study. Both studies make use of a manually developed set of words which are used as relaters between requirements. The results of both studies aid in the development of the syntactical tool. The goal of both studies is to determine if requirements can be used to predict change propagation. Both projects are heterogeneous as they pertain to the development of a different system for different customers. Further, the requirements documents are written by the customer and not changed by AEC.

6.4. Syntactical Approach Prediction

Developed by the Toho and Pierburg project, The EVRAZ project is used to validate the syntactical approach. Further, the syntactical approach is used to analyze the Toho and Pierburg projects. The protocol for analyzing change propagation using the syntactical analysis is described in Section 5.3.2. The results of the syntactical approach are presented as a requirements document search depth.

CHAPTER 7.
REQUIREMENT CHANGE PROPAGATION PREDICTION: TOHO

The Toho project dealt with the creation of yarns on a spool through an automated creel system. This spool is commonly referred to as combs in the textile industry. A full comb of yarn is shown in Figure 7.1. A creel is used to hold all of the combs as the textile yarn fill each one of them simultaneously. The yarn comb sits on a spindle as it collects yarn. An example of a full creel system is shown in Figure 7.2.

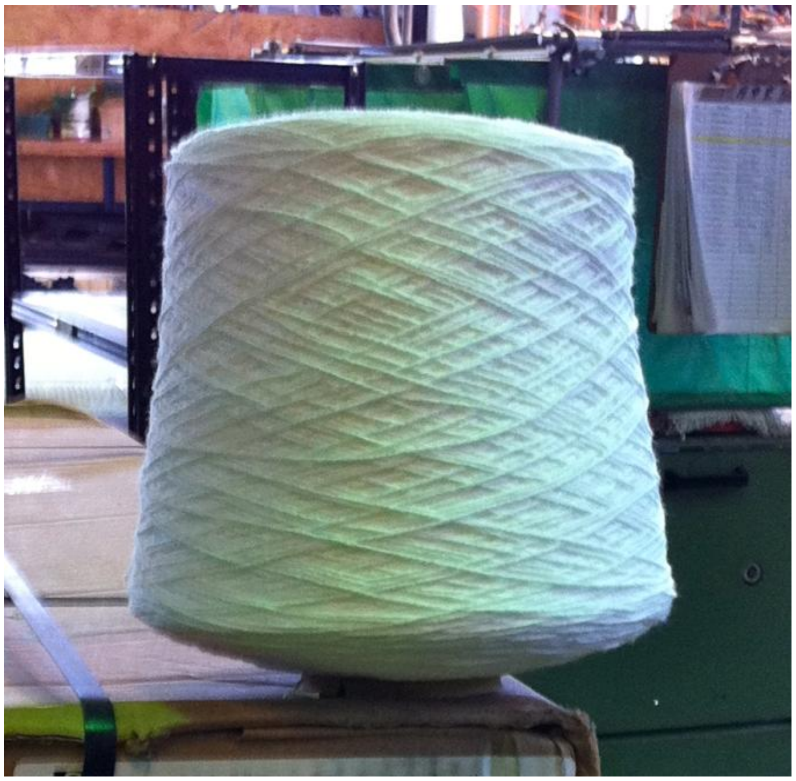


Figure 7.1: Example Comb of Yarn [190]

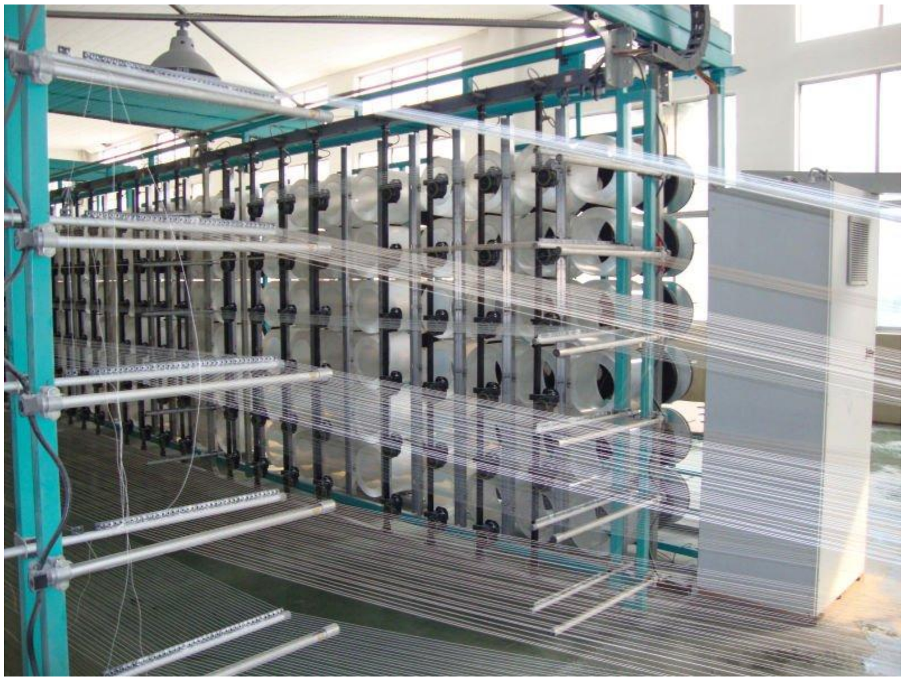


Figure 7.2: Example Creel System [191]

The project duration of the initial study spanned approximately fifteen month and included fifteen managers, engineers, and business associates. The total cost of the project was over two million dollars. The engineers were responsible for working on a specific subsystem of the product. The project client provided the corporation a contract detailing 160 requirements.

7.1. Toho Engineering Changes

Though multiple ECNs were proposed throughout the span of the project, only three were implemented. Table 7.1 shows each ECN, their description, the date they were implemented and the requirements affected by this change. Each ECN has a specific ECN# associated with it, for instance ECN01 relates to the replacement of the manual tool which occurred on January 16, 2008, affecting requirement 9.2.3.1. The requirement numbering scheme was taken from that of the requirements document.

Table 7.1: Toho ECNs

Approved ECNs	Description	Date	Requirements Affected
ECN01	Replacement of manual tool for opening and closing of core locks with automated air locks. Includes independent control of 5 lower spindles.	1/16/08	9.2.3.1
ECN03	Fabricate additional combs. Fabrication drawings will be provided for approval before fabrication.	10/2/08	9.3.7
ECN04	Design, fabricate and install mounting brackets for combs	11/7/08	9.3.10

7.1.1. ECN03 Background

ECN01 made changes to the core locks. The core locks were originally manual and this was changed to an automated set of locks. These locks are used to grab the creel, and hold them in the spindle. Further, a change was made to the five lower spindles to allow for independent control. It is unknown why this change was made, however the customer requested the change. The requirement affected by this change is requirement 9.2.3.1 which states:

> *Spindle design shall incorporate a means to lock bobbin spools in place during unwinding operation. Self-locking core chuck or similar design shall be used to allow freedom of movement when loading and unloading bobbins without requiring manual locking or releasing*

The feedback meeting with AEC did not indicate this change could have affected the subsequent requirement to fabricate additional yarn combs as the primarily concern was the propagation from ECN03 to ECN04. As a result, this was not investigated thoroughly during the manual approach analysis. However, this change was returned to during the syntactical analysis to identify if this could have been a change not recognized by the AEC associates as potentially propagating to ECN03.

7.1.2. ECN04 Background

The feedback from AEC and documentation from the data management system supported the change propagation from ECN03 to ECN04. ECN03 made a change to fabricate additional combs for the ends of the bobbin. This change occurred on October 2nd 2008, describing a need to fabricate additional combs for the machine. There are different bobbin sizes and each required the fabrication of a greater number of combs. Analyzing this, this change was related back to requirement 9.3.7. which relate to the number of combs in the system. The requirements stated:

A Yarn Comb for (22) ends shall be provided for each layer of bobbins.

Reviewing subsequent change notifications revealed that on November 7th, 2008, 33 days after original requirement change, ECN04 was implemented, stating “design, fabricate and install mounting brackets for combs.” The requirement this ECN affected was requirement 9.3.10 due to their relevance to the mounting of the combs. The requirements stated:

Yarn Rollers shall be mounted a) at the end of each layer of bobbins and b) at the production line infeed. The yarn rollers shall be positioned to set the yarn elevation before entering combs. Rollers shall be 80 mm diameter by 500 mm long. Rollers shall be 120 mm diameter by 2200 mm long.

Feedback from AEC indicated this was a change that could have propagated as the need to install mounting brackets was not recognized during the request to develop a greater number of combs. As a result, this was analyzed during the propagation analysis to determine if this change could have been predicted.

7.2. Manual Approach Prediction

The first step of the study was to complete a DSM of the requirements, based on a component to component relationship between requirements. Using this relationship, the DSM seen in Figure 7.3 was created. The DSM represents a requirements vs. requirements mapping. The DSM is color coded for interpretation where all cell shaded a color green indicates a relationship exist between the requirements. Since all relationships are based on their subject, this DSM is symmetric. A small segment of the DSM is shown in Figure 7.4 for visibility. The remaining DSMs generated for the Toho subject based study are shown in APPENDIX F.

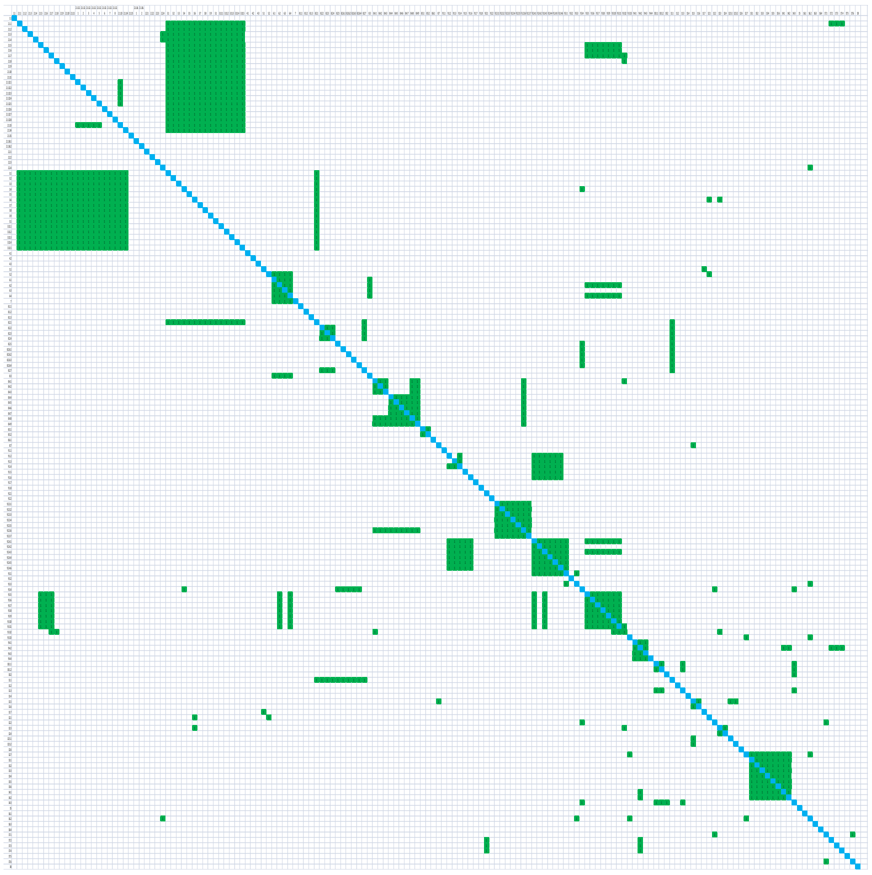
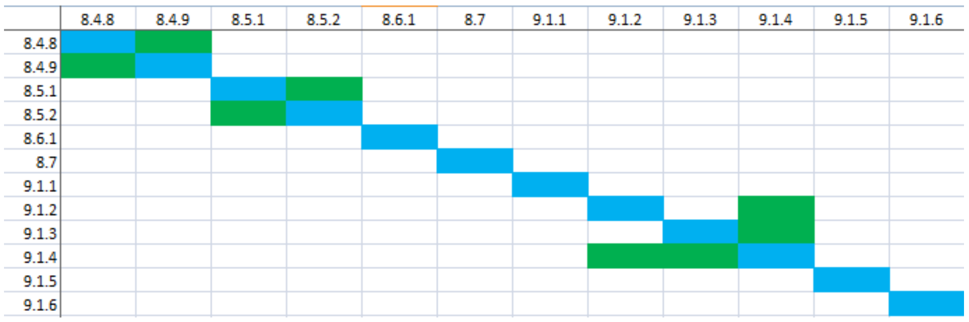


Figure 7.3: DSM for AEC Toho Study



	8.4.8	8.4.9	8.5.1	8.5.2	8.6.1	8.7	9.1.1	9.1.2	9.1.3	9.1.4	9.1.5	9.1.6
8.4.8	🟦	🟩										
8.4.9	🟩	🟦										
8.5.1			🟦	🟩								
8.5.2			🟩	🟦								
8.6.1					🟦							
8.7						🟦						
9.1.1							🟦					
9.1.2								🟦		🟩		
9.1.3									🟦	🟩		
9.1.4								🟩	🟩	🟦		
9.1.5											🟦	
9.1.6												🟦

Figure 7.4: Extract of DSM for AEC Toho Study

The DSM shows the relationship between requirements based on their subject, which in this case was usually the component or subsystem of the requirement. The requirements in the DSM are listed in the order they were written by the client. The client contract documented the requirements in a hierarchical manner. For example, all requirements under 9.0 are “creel design.” The creel design requirements are further separated into subsystems as all requirements under 9.1 refer the “bobbin and packaging”. This type of requirements documentation is apparent in the DSM as there are many large clusters because of the hierarchy.

The higher order DSM, shown in Figure 7.5, illustrated the propagating effect of the requirement change. Furthermore, Figure 7.5 only shows the first order propagation, indicated by red shading in the higher order DSM. This is referring to the propagation that occurs to requirements directly relating to the changed requirements. The second order DSM shows the changes that could occur due first order requirement change propagation. This second order propagation of the AEC study is shown in Figure 7.6, highlighted with yellow shading. Figure 7.6 illustrated both first order and second order propagation.

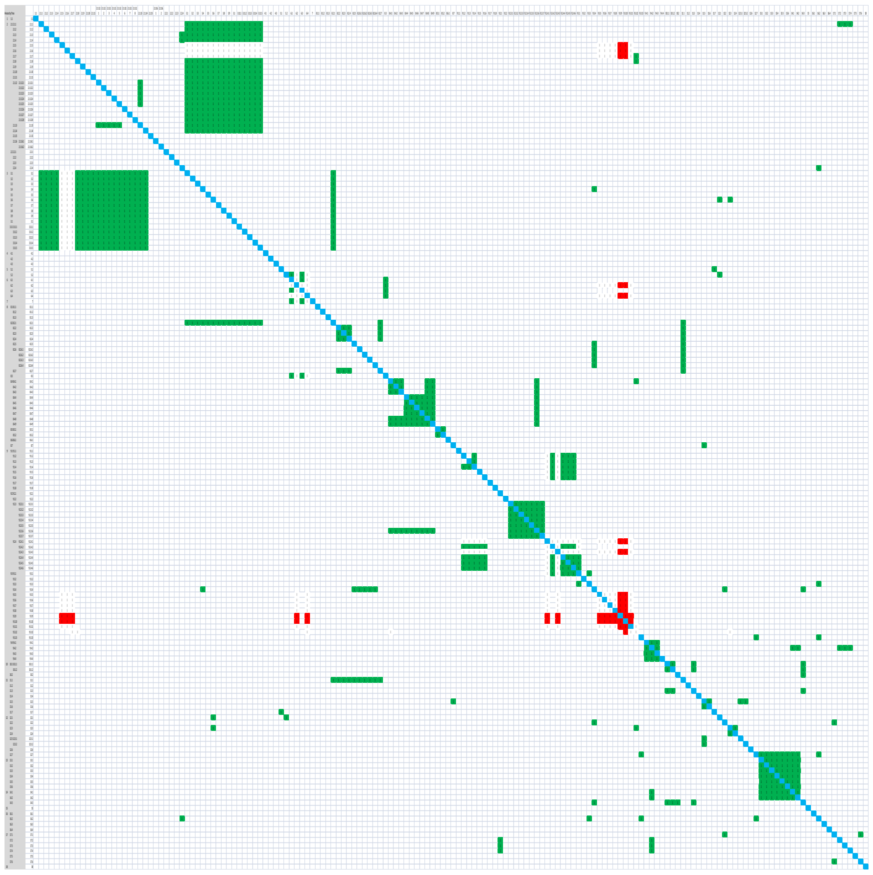


Figure 7.5: First Order Delta DSM for AEC Toho Study

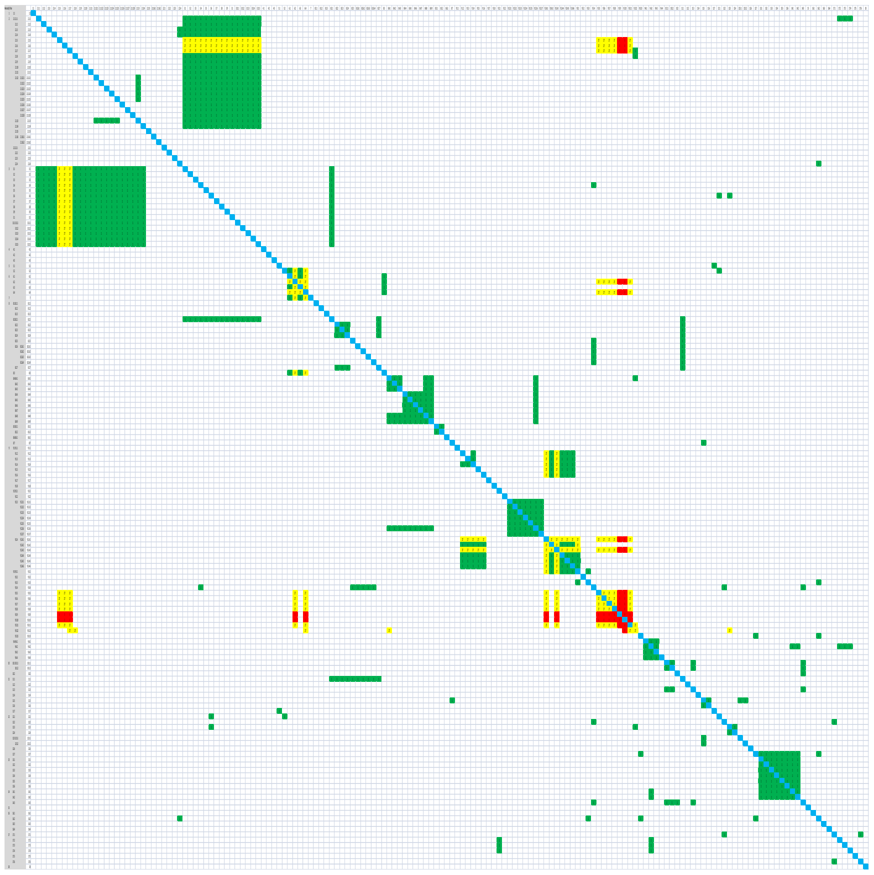


Figure 7.6: Second Order Delta DSM for AEC Toho Study

A third order DSM could be developed to further view possibilities of requirement propagation, however this seems to dilute the propagating impact of requirement change. Using the information from Figure 7.6, possibilities for further requirement change is seen.

The DSM revealed that this change could have been propagated as a second order relationship did exist between the changed requirement and those which propagated. This study did not provide further granularity to differentiate between those requirements which had second order relationships. Rather, it was used to identify if a relationship, regardless of order, did exist.

7.3. Syntactical Approach Prediction

An overlay DSM was developed for the Toho project, as shown in Figure 7.7. During change propagation analysis, all previous changes were inputted into the propagation scoring algorithm introduced in section 5.2.3. The results of the analysis are shown in Table 7.2. The first propagation analyzed was from ECN01 to ECN03, requirement 9.2.3.1 to requirement 9.3.7. After a change in requirement 9.2.3.1, the scoring system ranked requirement 9.3.7 as the 16th ranked requirement to change. It is important to note in the case of a scoring tie, the ranking system assumes it is the last requirement reviewed in that series of numbers. This yields a worst case, conservative scenario. This meant the designer or engineer would need to review a depth of ten percent of the requirements document to address this propagation.

The second change occurred to ECN04, affecting requirement 9.3.10. Using the information gathered from previous changes (ECN01 and ECN03) results in a propagation scoring ranked 5th. At such a ranking a designer or engineer would need to review a depth of only 3.1% of the requirements document.

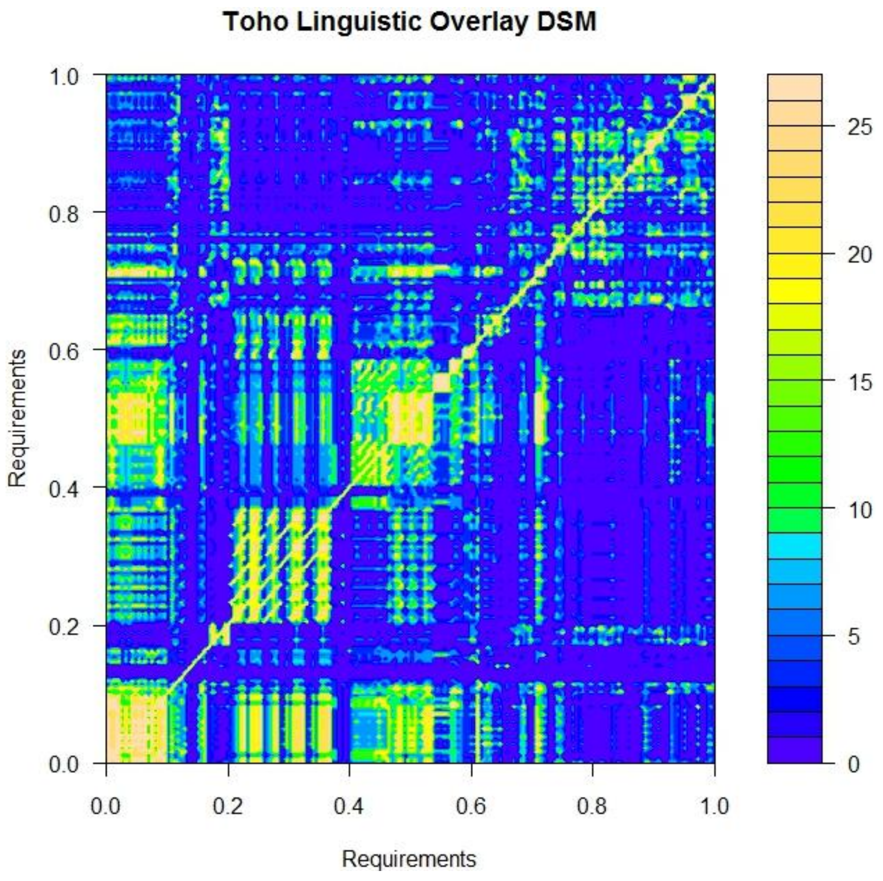


Figure 7.7: Toho Syntactical Creation DSM

Table 7.2: Results of Toho Syntactical analysis

Change	Requirement Changed	Score Propagation	Rank	Depth
ECN03	9.3.7	21	16th	10.0%
ECN04	9.3.10	27	5th	3.1%

The syntactical method and scoring system used for the Toho project would require an engineer or designer to review, at most, 10% of the requirements document to identify change propagation for such instances of change.

CHAPTER 8.
REQUIREMENT CHANGE PROPAGATION PREDICTION: PIERBURG

The Pierburg project dealt with an assembly line design for the development of an exhaust gas recirculation bypass flap, as shown in Figure 8.1. The requirements document was developed in a manner pertinent to each assembly station. Alongside the fabrication of each station, the transportation of the bypass flap from station to station was important. Further, change over for different bypass flap models, as shown in Figure 8.2 was important to the design of the assembly line. The Pierburg project included 214 total requirements, approximately a third more than the Toho project. The budget of this project was approximately one million and spanned nearly one year.

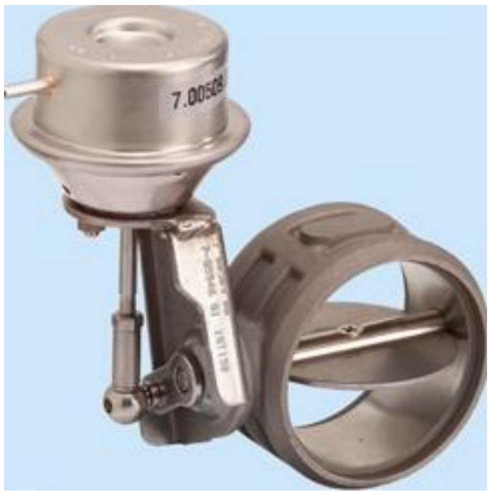


Figure 8.1: Pierburg Exhaust Gas Bypass Flap [192]

Figure 8.2: Pierburg Exhaust Gas Flap Models [193]

8.1. Pierburg Engineering Changes

A list of ECNs were collected from the project and summarized in Table 8.1. The ECNs were used in identifying which requirements they affected as this was not documented in the ECN documents. The ECNs simply stated “change in customer requirements” without specifically stating which requirement or set of requirements changed.

The ECNs were sorted based on their documentation date and their status. All the implemented ECNs, noted as “approved” were analyzed. A total of 16 ECNs are documented, of which six are approved. It is not documented why the remaining 10 were not approved and therefore they were not addressed in this study. However, it is important to note that the rejection was not due to change propagation. Changed requirements and date of change were extracted from each ECN. A description of the change is detailed so this may be understood by those involved with the project.

Of the six approved ECNs, only three ECNs were analyzed for the requirements they affected. This is due to the available information documented in the ECN to assist in identifying the changed requirement. While interviews may help expose the missing information, not all engineers associated with this project are still employed within the corporation. However, the president of the corporation did recognize the conflict between ECN01 and ECN07, which will be discussed. Additionally, the Pierburg project documentation is used to identify where propagation may have occurred. The three jettisoned changes are not used because the research must ensure that proper documentation is available to validate the results of the analysis.

Ultimately, ECN01, ECN07, and ECN11 are studied to determine the predictability of requirement change based on the keyword based DSM. Identifying the affected, or changed, requirements was performed by viewing the initial set of requirements to identify which requirement could have been effected or changed as a result of the change noted in the ECN. For example, ECN01 states “Remove additional PLCs”. Any requirement pertaining to the PLC could also be considered as affected requirements. A list of the affected requirements for the three ECNs investigated is seen in Table 8.1. As seen in the table, some ECNs affected multiple requirements.

Table 8.1: Pierburg ECNs

Approved ECNs	Description	Date	Requirements Affected
ECN01	1. Removed additional PLCs, Substituted C-More HMI for A/B Panel View 2. Design labor and material reductions due to line layout changes associated with increasing machine cycle time from 25s to 50s. The cycle time increase reduced the number of operators from ~10 to ~6, eliminating tooling nests 3. Gasket And Housing Assembly (screw driver stations) - Removed pallets and pallet return conveyor and replaced with fixed tooling nests for Scorpian and Griffin tolling plates. 4. Added storage shelves under stations to hold unused tooling nests	6/10/08	2.5.8 - 2.1.2 - 2.9.2 - 2.1.14
ECN07	Tool Changer at Station 3	08/15/08	2.1.14 - 2.2.6
ECN11	Stack Lights at Each Station	08/15/08	2.7

8.1.1. <u>ECN07 Background</u>

Examining ECN07, it could be inferred the change of requirement 2.9.2 during ECN01 influenced and propagated the change of requirement 2.1.14. Requirement 2.9.2 states:

Transport pallets shall be used to transport the product. Pallets will not be used as fixtures for critical operations. Client must approval deviations from this specification. If pallets are used as fixtures then each measurement must have a capability (Cpk) > 1.33. The measurement report has to be provided to client.

It is important that this requirement was highlighted through the propagation as an immediate conflict is recognized. ECN01 states "removed pallets and pallet return conveyor and replaced with fixed tooling nests." While the requirement states "pallets will not be used as fixtures," this change specifically states to place a fixed tooling nest on a pallet return conveyor while the requirement stated this should not occur. Nonetheless, reviewing the requirement affected in ECN07, requirement 2.1.14 states:

> *The entire base plate where tooling and fixtures are mounted must be completely removable for each process in such a manner that a new base plate with new tooling can be interchanged.*

Again, it is seen that this requirement states that tooling and fixtures which are mounted must be completely removable. This is in direct conflict with ECN01 as it called for the addition of a fixed tooling nest. This change was imminent as it directly conflicted with requirement 2.1.14. Consequently, requirement 2.1.14 was changed during both ECN01 and ECN07 to address this. Feedback from AEC during a review stated that this was in fact a conflict that was not recognized when the change was made. Both documentation and feedback from the client support the propagation of this change due to the conflict.

The second ECN07 affected requirement, requirement 2.2.6 states:

> *Supplier will design equipment for fast change over time (5 minutes or less total line change overtime) using quick change out tooling and fixtures.*

It could be inferred requirement 2.9.2 or 2.1.2 propagated to requirement 2.2.6 because those requirements again relate to the design of tooling and fixtures, a set of components which changed during ECN01. The second requirement from ECN01 which shared high relation with requirement 2.2.6 was requirement 2.1.2. Requirement 2.1.2 states:

Tooling or fixtures switched during change over shall attach to a sub-plate in accordance with "single minute exchange die" (SMED) design philosophies

Attaching a sub-plate to a fixture may have influenced the ability for the fast change over time stated in requirement 2.2.6 as it introduces a new process (changing of sub plate) to the changeover process.

8.1.2. <u>ECN11 Background</u>

Reviewing the final change, ECN11 affected requirement 2.7 which pertains to station lights, stating. This requirement states:

Status lighting at every station must be mounted for good visibility (Top down: Red-Yellow-Green). Module Status Indicator lighting will be as follows:

- ➢ *Green Light (Solid): No Faults present on Module*
- ➢ *Green (Flashing): Quality Check is switched off*
- ➢ *Yellow (Solid): Manual Mode – Not in automatic*
- ➢ *Yellow (Flashing): Parts Bin out of parts*
- ➢ *Red and Yellow (Flashing): Station out of parts*
- ➢ *Red (Solid): Station is faulted*
- ➢ *Red (Flashing): Station Stopped - Part failed caused by % counter of faults*
- ➢ *Green and Yellow and Red (Solid): Station is deactivated*
- ➢ *Green (solid) Yellow (flashing): no faults, cycle time above Limit*

In this ECN, requirement 2.7 was selected as the affected requirement because of its significance to the lighting over each station. This was important because ECN11 states to make a change in which lights will be stacked at each station. This is a potential indicator that the change to requirement 2.5.8 propagated to requirement 2.7. Requirements 2.5.8 states:

Individual Module and/or system operations can be PLC controlled as long as the data transfer, collection, and management is PC based and does not slow down the speed of the system.

Initially examining those requirements directly, one does not immediately identify a relationship with the status lighting stations. Unlike the previous examples where the relationship was apparent, the strength of using a change propagation prediction tool is realized in situations such as this.

8.2. Manual Approach Prediction

In developing relationships, requirement keywords were compared against the text of other requirements to identify if the text included those keywords. A requirement may only be related to another requirement if at least one its keywords exist within the text of the related requirement. In this manner, the relationships are not bidirectional and the resulting DSM is asymmetric.

The DSM shown in Figure 8.3 was developed based on keyword based requirement relationships. An extract from the complete DSM is illustrated in Figure 8.4. Since up to three keywords can be used to relate requirements here, this allows for the development of a DoF DSM which identifies how many of the keywords matched in the text of a requirement. This goes beyond the typical binary relationship DSM. A DoF DSM is shown in Figure 8.5 where blue indicates relationships where one keyword matches, yellow indicates the relationship of two keywords, and red indicates all three keywords. While it may be noted that multiple keyword matches would suggest a stronger relationship, this is reserved for investigation presented in later sections.

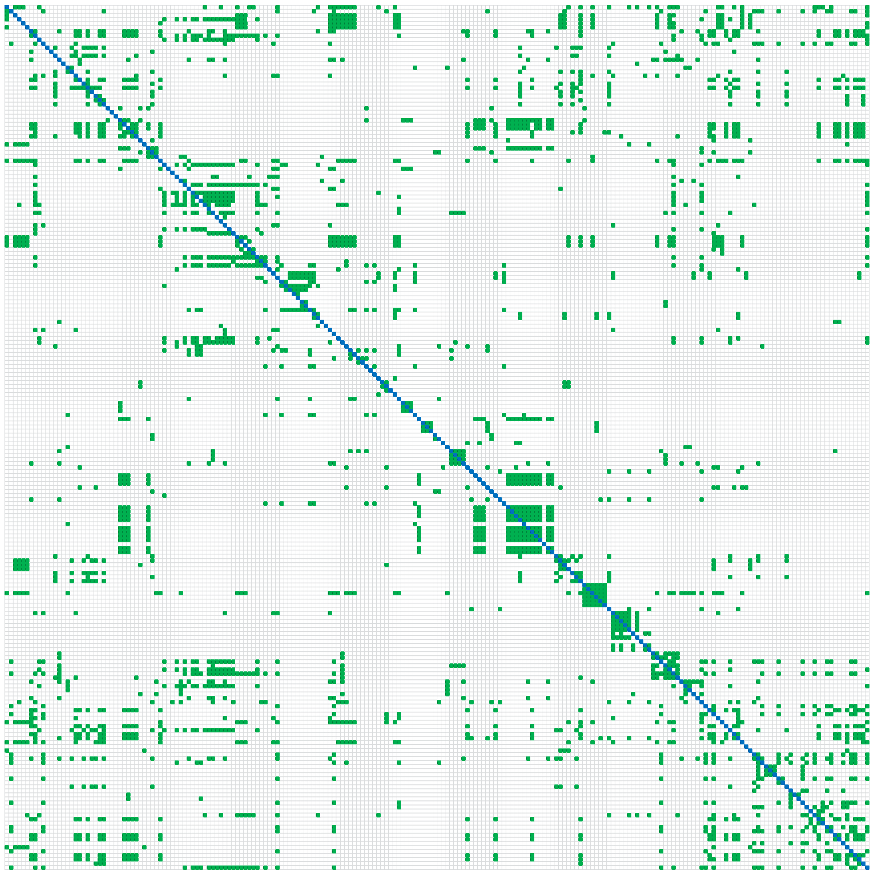
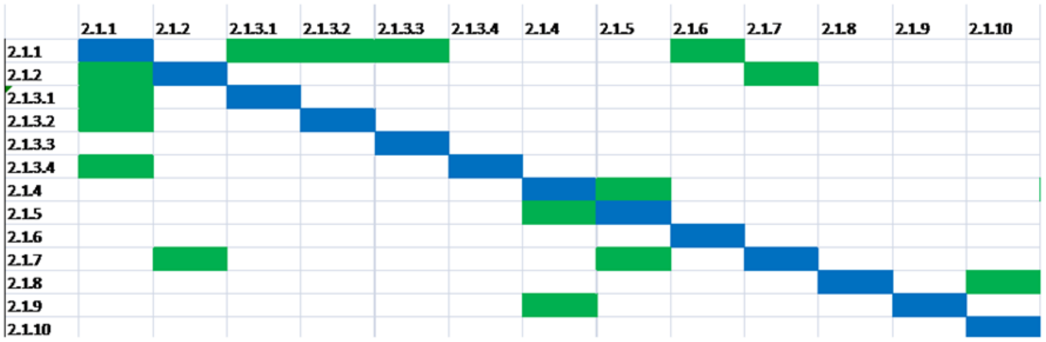


Figure 8.3: DSM for Keyword Based Study



	2.1.1	2.1.2	2.1.3.1	2.1.3.2	2.1.3.3	2.1.3.4	2.1.4	2.1.5	2.1.6	2.1.7	2.1.8	2.1.9	2.1.10
2.1.1	🟦		🟩	🟩	🟩				🟩				
2.1.2	🟩	🟦								🟩			
2.1.3.1	🟩		🟦										
2.1.3.2	🟩			🟦									
2.1.3.3					🟦								
2.1.3.4	🟩					🟦							
2.1.4							🟦	🟩					
2.1.5							🟩	🟦					
2.1.6									🟦				
2.1.7		🟩						🟩		🟦			
2.1.8											🟦		🟩
2.1.9							🟩					🟦	
2.1.10													🟦

Figure 8.4: Extract of DSM for Keyword Based Study

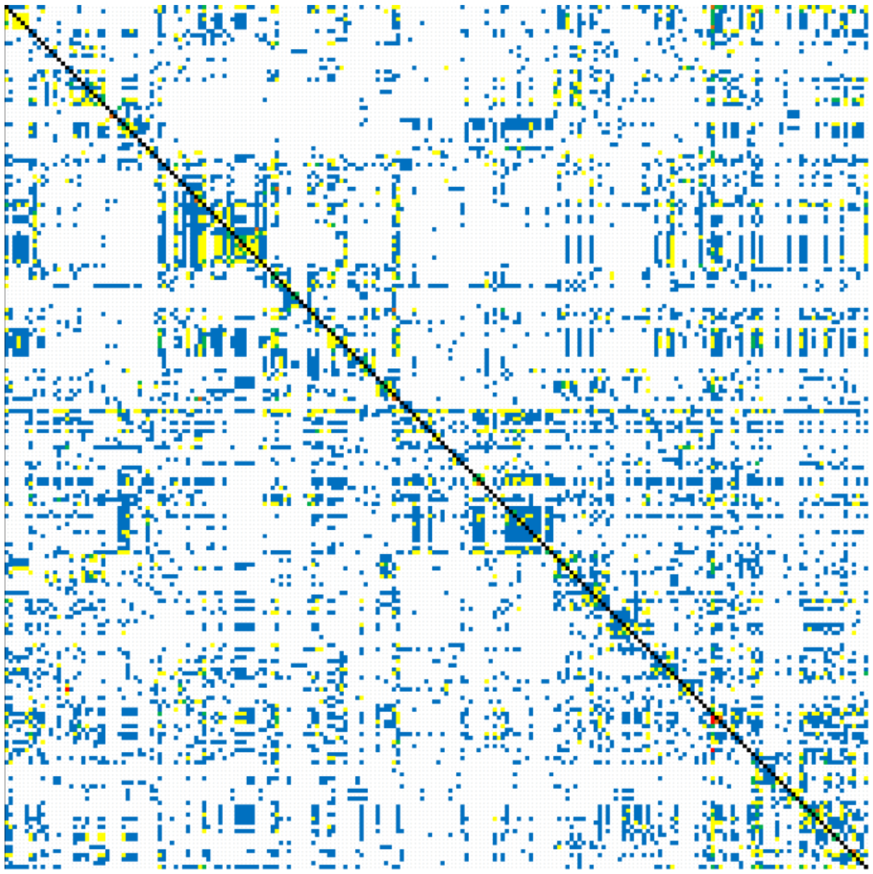


Figure 8.5: Pierburg Keyword DoF DSM

The DSM modeled 2,839 relationships between the 214 requirements. On average, each requirement had 13.3 relations with one requirement relating to 51 other requirements and other requirements being completely independent with zero relations.

A higher order DSM, allowing the user to view the changes that propagate through the requirements based on keyword relations, was created for each of the requirements affected by an ECN. A total of five higher order DSMs, for requirements 2.5.8, 2.1.2, 2.9.2, 2.1.14, and 2.2.6, are developed to address and model the requirements affected from ECN01 and ECN07. A higher order DSM is not developed for the requirements affected in ECN11 as it is the last ECN in the study and is not useful for predicting subsequent change. The developed higher order DSMs for the Pierburg manual keyword based study can be found in APPENDIX G. The higher order DSM uses the requirement relationship in the original DSM to propagate the requirement changes. Using this change propagation, the requirements changed in subsequent ECNs will be analyzed to identify if their change could have been predicted. All cells shaded in red indicate are first order propagation, while those shaded in yellow indicate a second order.

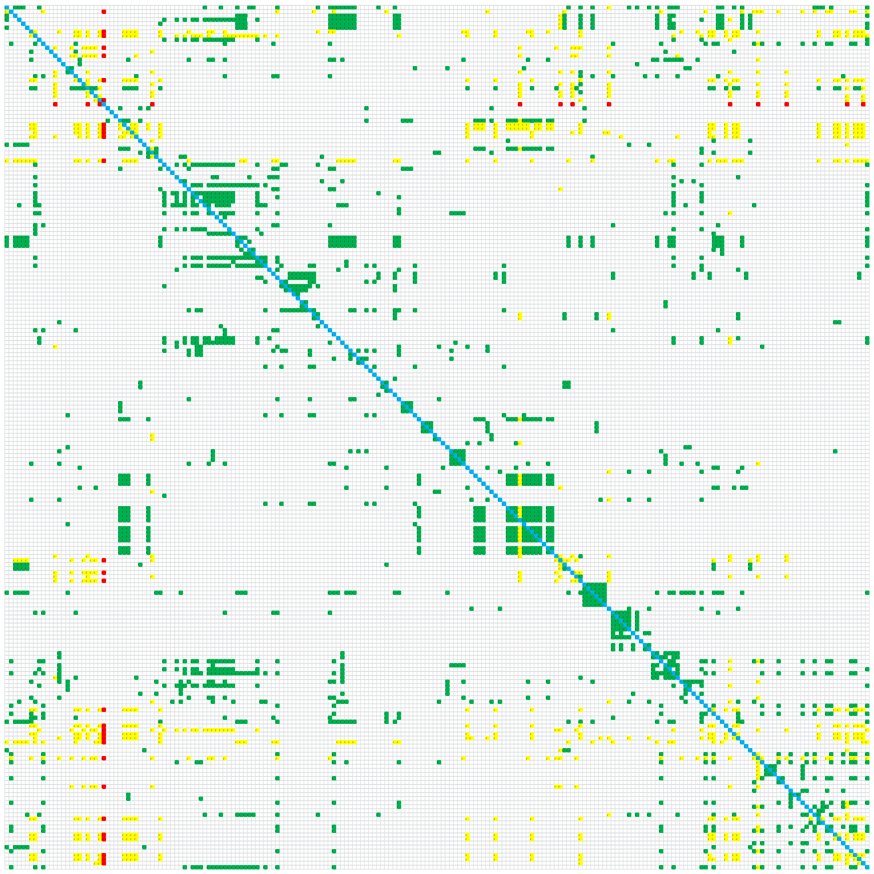
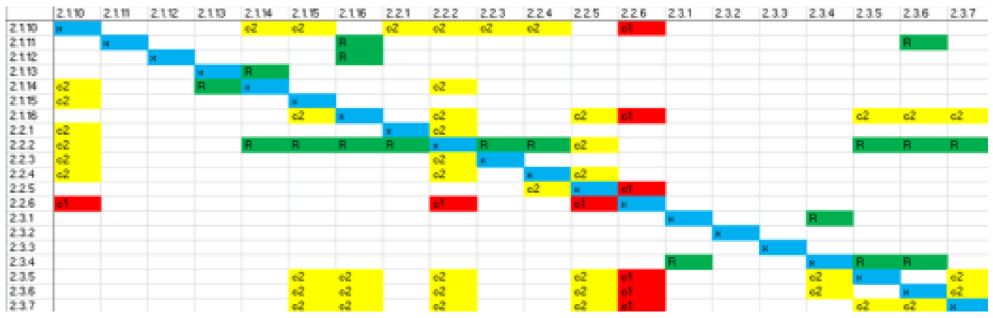


Figure 8.6: Example higher order DSM for Keyword Based Study



	2.1.10	2.1.11	2.1.12	2.1.13	2.1.14	2.1.15	2.1.16	2.2.1	2.2.2	2.2.3	2.2.4	2.2.5	2.2.6	2.3.1	2.3.2	2.3.3	2.3.4	2.3.5	2.3.6	2.3.7
2.1.10	x				c2	c2		c2	c2	c2	c2		c1							
2.1.11		x					R												R	
2.1.12			x				R													
2.1.13				x	R															
2.1.14	c2			R	x				c2											
2.1.15	c2					x														
2.1.16						c2	x		c2			c2	c1					c2	c2	c2
2.2.1	c2							x	c2											
2.2.2	c2				R	R	R	R	x	R	R	c2						R	R	R
2.2.3	c2								c2	x										
2.2.4	c2								c2		x	c2								
2.2.5											c2	x	c1							
2.2.6	c1								c1			c1	x							
2.3.1														x			R			
2.3.2															x					
2.3.3																x				
2.3.4														R			x	R	R	
2.3.5						c2	c2		c2			c2	c1				c2	x		c2
2.3.6						c2	c2		c2			c2	c1				c2		x	c2
2.3.7						c2	c2		c2			c2	c1					c2	c2	x

Figure 8.7: Extract of higher order DSM for Keyword Based Study

The three requirements, from ECN07 and ECN11, could be predicted in this study. The ECNs analyzed were ECN07 which comprised of changes to requirement 2.1.14 and 2.2.6 followed by ECN11 which comprised of changes to requirement 2.7.

To highlight high potential requirements, defined as those requirements possessing a great number of relationships, a relationship ranking is developed. This is used to sort the relationships in comparison to the remaining requirements which also share a relationship. The ranking gives insight as to the strength of relationship, based on the number of relationships, compared to other requirements.

As seen from the propagation results in Table 8.2, many requirements shared a great number of relationships with the requirements affected by ECN07 (requirement 2.1.14 and 2.2.6). The results indicated a relationship did exist as each of ECN07's affected requirements could have been predicted through a previous ECN's affected requirement. The total number of first and second order relationships for each of the previous ECNs is shown in the second and third primary column of Table 8.2 and Table 8.3 respectively.

The first order relationship pathways are all first order path that a requirement has with related requirements. Since first order relations are direct relationships, each path is to an individual requirement. For instance, requirement 2.5.8 had thirty one first order relations through thirty one separate pathways. Second order pathways are those possible pathways of connection to second order relations. A requirement may be related to another requirement in the second order through multiple intermediate requirements, increasing the number of second order relationship paths. For example, requirement 2.9.2 had 487 second order relationship pathways, of which twenty were to requirement 2.1.14, meaning there are twenty requirements which have a relation to both requirement 2.9.2 and 2.1.14. To further understand this, twenty of requirement 2.9.2's thirty eight first order relations are first order related to requirement 2.1.14.

ECN07's requirement 2.1.14 possessed a first order and twenty second order relationships with requirement 2.9.2. There exist no ranking for first order relations as it is binary and the second order relationship ranked 13th amongst all other requirements.

As ECN01 influenced requirement 2.9.2 and 2.1.2, both of which address the use of fixtures and their attachment, the higher order DSM could have assisted in predicting an influence such change would have had on requirement 2.2.6 which relates to the change out of tooling and fixtures. Neglecting requirement 2.1.2, the lowest ranking requirement relationship with that between 2.1.14 and 2.26, ranked as the 16th highest. At this ranking, a search depth of 7.5% of the requirements document is needed to predict the change propagation. This was recognized by the higher order DSM as a critical requirement. Further, this was recognized by AEC after their review.

Table 8.2: ECN07 Propagation Analysis

Total Relationship Pathways			**Relationships with ECN07 *Requirement 2.1.14***		
ECN01	1st Order	2nd Order	1st Order	2nd Order	2nd Order Ranking
2.5.8	31	249	-	-	-
2.1.2	37	539	-	2	56
2.9.2	38	487	1	20	13
2.1.14			N/A	N/A	N/A
Total Relationships Pathways			**Relationships with ECN07 *Requirement 2.2.6***		
ECN01	1st Order	2nd Order	1st Order	2nd Order	2nd Order Ranking
2.5.8	31	249	-	-	-
2.1.2	37	539	1	29	6
2.9.2	38	487	1	40	3
2.1.14	22	375	-	18	16

The last change occurred during ECN11. In analyzing this change, all previous engineering changes are considered. Finally, ECN11 is examined with the results illustrated in Table 8.3. Reviewing the final change, ECN11 affected requirement 2.7 which pertains to station lights, stating:

This change was incorporated into the higher order DSM for analysis and it was found that the subsequent requirement change with the greatest influence was requirement 2.5.8, as seen in Table 8.3. Requirement 2.5.8 had a first order relation with requirement 2.7 and also had twenty eight second order relationships, more than any other requirement.

Within the ranking, numbers in parenthesis represent the number of requirements tied for this rank. For instance, requirement 2.1.2 has 4 second order relationships, which ranks as a tie for thirty first with 11 other requirements.

Table 8.3: ECN11 Propagation Analysis

Total Relationships Pathways			**Relationships with ECN11 Requirement 2.7**		
ECN01	1st Order	2nd Order	1st Order	2nd Order	2nd Order Ranking
2.5.8	31	249	1	28	1
2.1.2	37	539	-	4	31(12)
2.9.2	38	487	-	4	31(12)
2.1.14	22	375	-	-	-
ECN07					
2.2.6	42	889	-	3	50(10)

In subsequent review with the automation corporation, they recognize that there is perhaps a relation between these requirements, but readily admit that this relation is not apparent. Nonetheless, with a functioning tool, it is difficult to consider this relationship as coincidental because of the high number of relationships and ranking of relationships between requirement 2.5.8 and their potential propagated requirement 2.7.

8.3. Syntactical Approach Prediction

An overlay DSM was generated for the Pierburg study based on the syntactical analysis performed. The DSM, shown in Figure 8.8, is used to score requirements based on the propagation scoring algorithm. The results of the scoring algorithm are shown in Table 8.4. The first Pierburg engineering change caused a change in four requirements. A subsequent change occurred two month later related to two requirements. The propagated requirements, 2.1.14 and 2.2.6, were scored based on the previous changes. Requirement 2.1.14 was found to score 7th highest amongst all requirements while requirement 2.2.6 was scored as the highest, indicating it was the most likely to change.

For this specific change, an engineer is required to review at least 7 requirements, 3.3% of the requirements document. The next engineering change affected a single requirement, requirement 2.7. Using the six previous requirements changed (from both ECNs), this requirement was found to score the 23rd highest score amongst other requirements. At this score, an engineer or designer would need to review at least 11% of the requirements document.

Table 8.4: Results of Pierburg Syntactical analysis

Change	Requirement Changed	Score Propagation	Rank	Depth
ECN07	2.1.14	17.7	7th	3.3%
	2.2.6	20.3	1st	0.5%
ECN11	2.7	14.9	23rd	10.7%

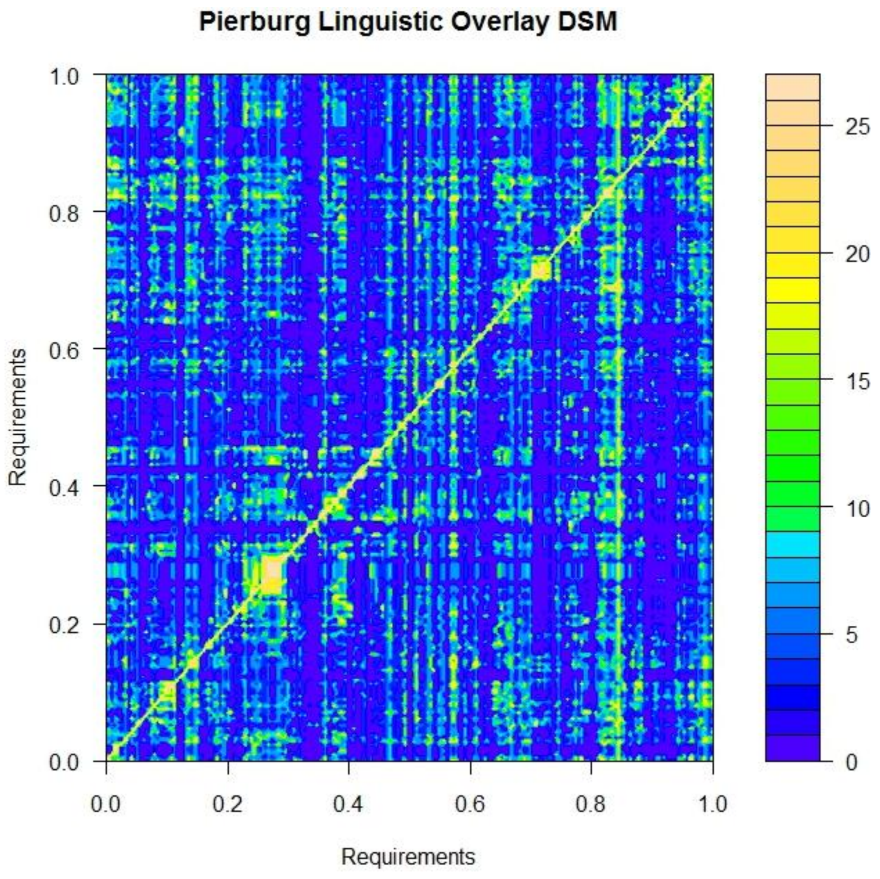


Figure 8.8: Pierburg Syntactical Creation DSM

Overall, for the entirety of a project, the syntactical approach and scoring method used were sufficient as long as the engineer or designer addressed the top 11% requirement scores.

CHAPTER 9.
REQUIREMENT CHANGE PROPAGATION PREDICTION: EVRAZ

The EVRAZ project was a project that lasted approximately one year and ended during the end of October 2011. This project pertained to the development of a threading station to be used for pipes. The project was for customer EVRAZ, in the Red Deer division who makes pipes. The EVRAZ Red Deer facility has an annual capacity of 150 ktons of pipe. Threading, as seen in Figure 9.1, is required on some of the pipes and AEC was tasked with developing the automation line to place such threads on the pipe. The pipes are laid on a “Skid” table where they are threaded and inspected. This process of threading is referred to as “Pin End” processing. The pipes ranged from 4.5” to 13.5” diameter and 25’ to 45’ in length made of varying material grades.

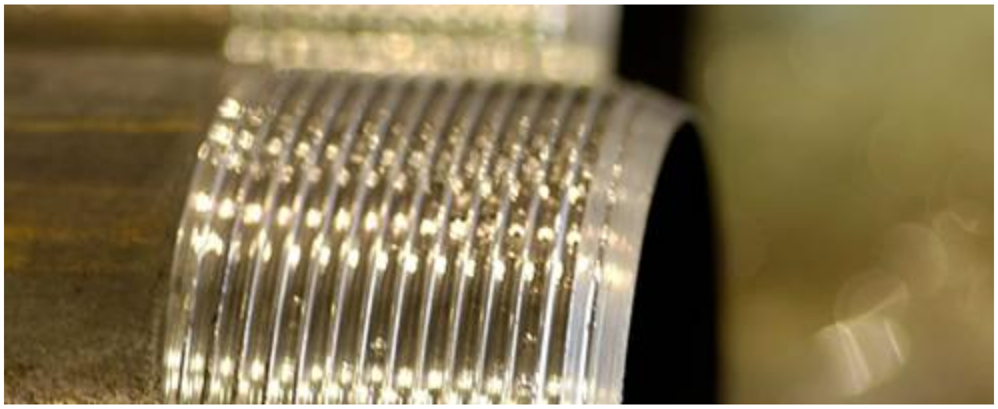


Figure 9.1: EVRAZ Pipe Threading [194]

The EVRAZ project entailed 186 requirements and multiple requirement changes. EVRAZ was a volatile project, encountering multiple changes throughout. However, this project was very well documented and document revisions were maintained. Many of the changes were not formally recorded as ECNs as they were performed without requiring authorization from the client. Documentation indicates many of such changes were done to correct or address previous changes, potential requirement change propagation.

9.1. EVRAZ Engineering Changes

Table 9.1 lists all of the changes that took place during the project span. The associated date with every change is listed and the changes are sorted in this order. It was noticed within EVRAZ that not all changes were documented as ECNs. As a result, some of the changes were found in other locations, such as emails and meeting minutes. To address this, they are noted as change instances. A change instance is when a set of changes occur on the same date. The ECNs which were documented are also recorded and noted as an ECN alongside its change instance. Some of the ECNs, such as ECN01, perform multiple changes on varying dates, causing different instances for the same ECN. The requirement number changed is also noted in the table.

Table 9.1: EVRAZ Changes and ECNs

Date	Change Instance	Description	ECN #	Requirement Changed
5/26/11	1	Proceed with Nord motors in place of Eurodrive.		70
5/26/11	1	Section 1 & 8 Table slope to be 5/16" per foot.		39
5/26/11	1	Use of urethane rolls in conveyors accepted.		75
6/9/11	2	Motors need to be CSA or ULC approved		70
6/13/11	3	Motors IP66 with synthetic oil	ECN01	70
6/20/11	4	Use 2HP gear motors on threading tables, 1.5 on remaining		88, 89
6/23/11	5	Place order for spare Nook electric cylinder	ECN01	34
6/23/11	5	Place order for 6 spare v-rollers	ECN01	34
6/23/11	5	Place order for 2 spare steady rest rolls	ECN01	34
7/7/11	6	Clearance requirement for electrical cubicle: 1m clearance North 4m clearance east of cubicle		22
7/7/11	6	Section 1 & 8 Load and Unload Rails to have 3 legs rather than 4		39
7/15/11	7	Positional feedback required to locate end of pipe in lathe		6,10,173
7/15/11	7	Index stops to select one pipe at a time into the threading table		119
7/15/11	7	Centering Roll Mechanism		133
7/21/11	8	Add encoders to Steady Rests	ECN02	89
7/21/11	8	Replace Static V nest with Static V rollers and motors to coupling stations	ECN03	98
7/21/11	8	Add 6th pivot V roller without motor at pin inspections	ECN04	89
8/8//11	9	Motors changed to higher voltage (600v)	ECN05	70
9/19/11	10	Fabricate and Supply Shipping Fixtures	ECN06	131
10/13/11	11	Provide additional field supervision for installation	ECN07	127

Though all of the changes are used in the tool, not all of the changes presented were documented well enough to determine if the change was due to propagation. As a result, only a select few of the changes are analyzed for propagation if the available documentation supports propagation may have occurred. For example, the change takes place during Instance01 replaces the Eurodrive motors with the Nord motors. After analyzing the documentation, it is identified that this change required a change in the type of motor oil used, affecting this requirement. This change was not recognized during the change of motors and, as a result, required another change instance. An example of a change which did not propagate is the purchase of spare v-rollers in Instance05. It is important to note that analysis was not performed before deciding which requirements to use for propagation, rather the requirements which the documents identified as potential propagators were used in the analysis to determine the tool's ability to predict changes.

The changes shown in Table 9.2 are those which are analyzed for change propagation. Most of the changes experienced and propagated are due to the motor changes that were encountered multiple times throughout the process. Motors where changed twice, once from Eurodrive to Nord motors and then changed again to upcharge them from 450 to 600 volts. This resulted in many parts associated with the motors to be affected. In total, there were eleven unique motors which changed at least once some point to accommodate these engineering changes.

Table 9.2: EVRAZ Change Instances

Date	Change Instance	Description	ECN #	Requirement Changed
6/13/11	3	Motors IP66 with synthetic oil	ECN01	70
7/15/11	7	Centering Roll Mechanism	--	133
7/21/11	8	Replace Static V nest with Static V rollers and motors to coupling stations	ECN03	98
7/21/11	8	Add 6th pivot V roller without motor at pin inspections	ECN04	89
8/8//11	9	Motors changed to higher voltage (600v)	ECN05	70

9.1.1. Instance03 and Instance09 Background

Instance03 and Instance09 are coupled here because both are a result of the initial change to the motors occurring in Instance01. Change Instance03 resulted in the documenting of an engineering change. This change dealt with the oil used in the motors used on the assembly line. The motors were changed during change Instance01, changing the motors from Eurodrive to Nord. This changing of the motors required a reconsideration of the oil used as the Nord motors needed synthetic oil. This change occurred in an email from the customer which stated that a change to the oil must be performed because of the previous changes.

During Instance09, the motors were once again changed to a motor of higher voltage. Eleven motors (two turning roll, four steady rest, two centering roll, and three kick out motors), placed throughout the assembly line were changed to upcharge the voltage from 450 to 600 volts. This change again was made due to the initial change to change the motors. In this scenario, multiple propagations occurred that were addressed at a later time. Both Instance03 and Instance09 stemmed from the initial change of the motors. However, it is important to note that other, previous changes in between the initial motor change at Instance01 and the subsequent instances could have contributed to the propagation.

9.1.2. Instance07 Background

Instance07 deals with the centering rolls of the threading station. Specifically, a centering roll mechanism was required for this part of the threading station. The requirement affected by this change is requirement 133 which states:

> *centering rolls for both pin end and box end threading stations. allows handling time to be optimized (faster cycle time) by guiding pipe into lathe at high speed with minimal risk of interference and chuck damage.*

The centering rolls of the threading station operate with a centering roll motor. This motor was changed during the motor change of Instance01. It is interesting to note here that the effects of second order propagation are apparent as the requirement does not list the word motor, yet it is the motor change that required a different centering roll mechanism for this part of the threading station. Further, because of the changes to the centering roll mechanism, an electric drive was eventually used here. A pneumatic cylinder was also considered for this change at one point.

9.1.3. Instance08 Background

Instance08 pertain to two separate ECNs. The two ECNs are marked as Instance08 because they occurred on the same date. Both ECN03 and ECN04 dealt with the V rollers and V nests of the assembly line. The two requirements affected by ECN03 and ECN04 are requirements 89 and 98 respectively, which state:

> *89: The box end and pin end threader require five v rollers w/ individual gear motors, height adjusting feature with a common electric drive, pneumatic shaft brake*
>
> *98: box end and pin end inspection four (4) turning roll units that rotate pipe, mechanically linked, gear motor*

This change occurred once again to the box and pin end threader which had previously experienced the change in Instance07 when the centering rolls and the box end and pin station motors were changed. Requirement 89 experienced a change in the increase of the number of v rollers, without increasing the number of motors on that roller. Requirement 98 specifically states to make a change to the V rollers and motors, which again was due to the propagation of the previous motor changes.

Requirements 89 and 98 both pertain to the gear motors used on the threading line. This was selected as a potential propagation because of the multiple changes which occurred to the motors on the line.

9.2. Syntactical Approach Prediction

Binary, relationship order, and maximum flow DSMs were generated for the EVRAZ study, as shown in APPENDIX H. The DSM shown in Figure 9.2 shows the Overlay DSM for the EVRAZ project with the Propagation score incorporated. Five requirements were propagated in the EVRAZ study. The results are shown in Table 9.3 which indicate the largest depth of review is 13% of the requirements document for requirement 70 in instance09. As noted, all previous changes are used to propagate forward to the next change.

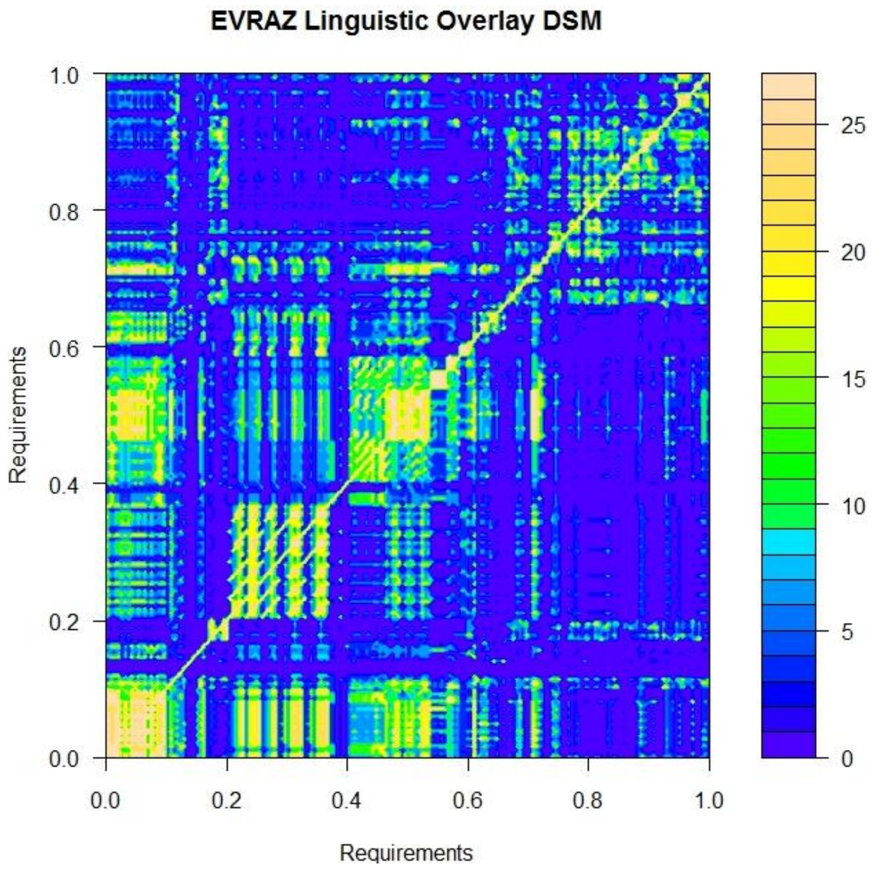


Figure 9.2: EVRAZ Syntactical Creation DSM

The first instance of change of requirement 70 was ranked as the seventh highest, with a score of 15.6. Requirement 133 during instance07 had a relatively high propagation score of 31.8, yet ranked as the sixteenth highest score. The high score is an indication that the previous changes were highly related, causing high scores amongst all the changes. Instance08 were related to the previous changes through a score of 16.1 and 16.8 for requirement 98 and 89 respectively. Those two requirements where ranked as the fourth and first highest, yielding a low requirement document search depth. Instance 09 had the highest requirement depth, with a score of 12.5, ranking twenty fourth.

In the EVRAZ project, the designers and engineers would need to review 13% of the requirements document during each change to ensure they address all propagation.

Table 9.3: Results of EVRAZ Syntactical Analysis

Change	Requirement Changed	Score Propagation	Rank	Depth
Instance03	70	15.6	7	3.8%
Instance07	133	31.8	16	8.6%
Instance08 – ECN03	98	16.1	4	2.2%
Instance08 – ECN04	89	16.8	1	0.5%
Instance09	70	12.5	24	13.0%

CHAPTER 10.
RESULTS

Two approaches, manual and syntactical are analyzed based on the protocol given for each approach in Sections 5.1 and 5.3 respectively. The result of each approach is detailed and a comparison for each approach is discussed to compare the different means for developing requirement relationships capable of predicting change propagation. Nine change instances causing ten requirement changes, from three separate industry projects are evaluated against a syntactical approach. Two of the projects, Toho and Pierburg, are also evaluated in a preliminary manual approach. The results are given in terms of search depth required to identify the subsequent changes. The search depth is determined from the propagation scoring algorithm developed. This scoring is ranked and the depth is given in the form of a percentage, measuring the percentage of the requirements document that must be searched to address the propagated requirement. As it is possible that a propagated requirement change may have the same propagation score as other requirements, the search depth reported is the depth of the last element of the same ranking.

Three industry case studies are presented in this dissertation, all presenting a requirement change propagation tool exploring different types of requirement relationships. The relationships are based on different word relaters between the requirements. The finest resolution of relaters is the syntactical words of requirements as individual letters cannot be used. As a result, different permutations for word selection are explored in an attempt to determine those which are most conducive to building requirement relationships accurate for predicting change propagation. The manual approach made use of two separate types of relaters, (1) subjects and (2) keywords while the syntactical approach made use of a POS tagging and manual keyword selection.

The results of the manual and syntactical approach are discussed in this section. A comparison is also provided to compare between the manual and syntactical approach. Included in this comparison are the limitations of both approaches. As this tool was motivated and developed for industry, its implication on design and requirements management practice is presented. Additional findings pertinent to requirement change propagation are presented.

Four research questions are addressed in this research. The research questions and their results are shown in Table 10.1. The questions are all addressed through the studies performed in this research. Requirements were found to be capable of predicting change propagation. This is with the assumption that components may be translated over to the requirements domain where propagation analysis may be performed. An analysis of requirement relationships is performed by first exploring subjects and keywords through a manual based approach. A syntactical approach is used where 32,768 permutations of nouns, verbs, and keywords are analyzed for their propagation capabilities. The results of the requirements analysis is narrowed through scoring metrics provided by graph theoretic and a scoring algorithm developed to use all previous changes to propagate to subsequent changes.

Table 10.1: Research Questions Results

Research Question	Question	Results	Section
I	Can requirements be used to predict change propagation?	The requirements domain is capable of predicting change propagation.	Chapter 7 Chapter 8 Chapter 9
II	What types of relationships between requirements predict change propagation?	Relationship permutations are explored to identify relationships that are most accurate predictors of change propagation.	Chapter 5
III	Can the propagation tool be tuned to narrow the selection of potential requirement for propagation?	Graph theoretic metrics are used to analyze the relationships as to identify which are most likely to propagate.	Section 5.4
IV	Can a scoring system be used so potentially propagated requirements are ranked in order of review?	A scoring algorithm is developed that is capable of ranking requirement change propagation.	Sections 5.2.3, 6.4, 7.3, and 8.3

10.1. Manual

The first relationship technique of the manual approach made use of the subject relationship of requirements. This resulted in a symmetric DSM and gave general insight as to the propagation of requirement and their use in predicting engineering change through requirement propagation. The first study indicated that change most often goes unnoticed during second order propagation. Designers and engineers are able to identify first order propagation with relative ease, however second order propagations may go unaddressed. This is specially the case there are many second order relationships to another requirement. There were limitations with the manual, subject based approach as it missed on some important words between requirements. As a result, an approach was needed that was less constrictive than the subject only approach, and allow for a greater number of relaters.

A subsequent study was performed to explore different types of requirement relationships, through keywords. The study was able to confirm that the propagation of requirements can be used to predict future engineering change requests. The keyword approach was found to be of great benefit as it allowed the user to identify the words pertinent to a requirement and use those as relaters. Through two change propagation scenarios, it was found that using a requirements change propagation prediction tool could have given the designers and engineers an indication of the potential propagation due to an initial requirement change. The use of a tool in this instance could have saved time lost between ECNs, prevented inaccurate estimation of change cost, and reduced the risk involved in approving changes.

The manual creation metrics measured the order at which requirements were related, such as first and second order relationships, and the number of such relationships (path count). A discussion on the Toho and Pierburg study results are presented here to identify findings and performance. Overall, following these manual studies, the tool was found to allow designers and engineers to taking into considerations requirements otherwise neglected. While the higher order DSM does provide the designer insight as to the change which may occur due to propagation, all affected requirements may not necessarily change. A requirement may be affected by a change to a related requirement; however this does not always merit a subsequent requirement change.

10.1.1. <u>Toho</u>

The Toho industry study revealed, through use higher order DSMs, that requirements change propagation may be predicted. The use of the DSM in the Toho project (ECN04) change instance could have saved over a month of time and the cost associated with the subsequent ECN needed.

The results of the Toho manual creation study indicated through the 160 requirements, all engineering changes could be predicted through a maximum relationship order of two (second order). At a second order relation, 43 requirements must be evaluated for possible change propagation. Though it was found that second order was the greatest relationship order needed to predict propagation, this indicated a designer or engineer would have to review 27% of the requirements document. There was no further granularity evaluated for this approach on the Toho project. Though it was accurate in predicting change propagation, the return of 27% was too high to make the approach useful. Further, there was no difference between requirements that could propagate. The Pierburg project added this granularity by incorporating relationship order path count ranking.

As the first study taken in using requirements to predict change propagation, it was successful as it answered the first research question. It was also realized that words could be used to relate requirements so they may be used for change propagation. Though only subjects were used here, other syntactical elements are explored in the Pierburg industry study.

10.1.2. Pierburg

A DSM and subsequent set of higher order DSMs were developed for the Pierburg project. To propagate the requirement change, a different type of relationship was developed than that of the Toho. The Toho project used subject to subject relationships in relating requirements together, effectively relating all requirements relating to the same subsystem or component. Realizing the influence of requirement relaters on requirements propagation prediction, the Pierburg project used a multiple keyword based relationship. The use of keywords allows a less restrictive and multiple relaters.

The DSM was developed based on keyword requirement relationships, where requirements may be related if there is a keyword match. Keywords were selected by reviewing and interpreting the semantics of the requirements rather than simply the syntactical information. Though different engineers or designers may select different keywords, a statistical analysis was performed to validate the consistency of keyword selection, as shown in Section 5.1.4. The keywords are selected by studying the requirements document and understanding how each requirement specifically affected the system design. Though it is a manual process, it allows for the selection of words not automatically selected by a syntactical POS tagger. This added flexibility in the relater selection process.

To separate between requirements which are potential to requirements propagation, relationship ranking is developed. The Pierburg requirements document contained 214 requirements and this ranking was developed to compare the results of change propagation between the multiple requirements highlighted. The ranking gives insight as to the strength of relationship, based on the path count, compared to other requirements. Second order relationships are important here as it was identified through the initial study (Toho subject based manual approach) that second order relationships are a contributor to change propagation.

Four requirements changed during the first engineering change, ECN01. Two requirements changed during ECN07, requirement 2.1.14 and 2.2.6. The propagated requirement, 2.2.6, was ranked as the 13th highest relationship path count. With this propagation approach a designer would have to review at least 13 of the 214 requirements to address the propagated requirements, a depth of 6%.

To propagate to the final engineering change, ECN11, all previous requirement change is evaluated to identify if change could have predicted. The final requirement which changes is requirement 2.7. Due to previous changes, requirement 2.7 has the 16th highest path count from previous changes. With this approach, to identify the propagated change in Requirement 2.7, an engineer or designer would have to review at least 16 requirements. This requires searching a depth of 7.5% of the requirements document.

Immediate advantages were realized through the use of keywords. Multiple syntactical elements could be used to develop relationships between requirements and a ranking is introduced. This approach was able to predict requirement change propagation at a maximum search depth of 7.5%. The enhancements of the Pierburg study made for significant improvements from the initial Toho study. However, limitations existed which could only be addressed through an automated system.

10.1.3. Advantages and Limitations

There are four primary takeaways from the manual approach that were used in the development of the syntactical approach. The takeaways are:

1. A requirement change management tool (higher order DSMs) can be used to propagate requirement change with a high degree of certainty.
2. The importance of second order propagation within requirements is critical to the success of effectively predicting requirement change
3. The recognized importance of requirement relationship types on the ability to propagate requirement change.
4. The needed ability to weight, rank or narrow the list of propagated requirements into a list of high potential requirements the designer is able to use.

It was identified that words could be used to relate requirements in a manner conducive to predicting requirement change propagation. Further, it was identified that second order propagation, though it may not be as strong as first order propagation, must be considered in as they go unaddressed. Because two types and varying quantities of relaters are used here, the manual study provided insight to the development of subsequent relaters. The ranking used in the Pierburg study also yielded a lower search depth for engineers or designers.

A limitation of the manual approach was the sheer number of requirements that an engineer or designer would have to review. As a result, the syntactical approach needed a means for scoring and/or ranking the requirements to narrow down the list of reviewed requirements and provide a sequence of review based on those most likely to change. To reduce the number of relations, the quantity of relaters must also be addressed. As a result, a thorough investigation of relaters was needed in the syntactical approach as such only those relaters which hare significant to requirement change propagation are used. This is performed to avoid superfluous relationships.

A limitation to this tool is it cannot predict self-propagation, which is an instance when a requirement continues to change at a later time, as seen with Requirement 2.1.14 in the Pierburg industry study. The inability to self-propagate is addressed in the syntactical approach.

10.2. Syntactical

A syntactical approach was developed which made use of requirement POS and used the nouns and keywords. The syntactical approach also incorporated a propagation scoring algorithm to add granularity between requirements and develop a means for ranking requirements based on the scoring outcome. This method was generalized through both the Toho and Pierburg industry projects and validated against a volatile third industry project, EVRAZ. Toho and Pierburg were also evaluated against the syntactical approach to provide a means for comparison between both approaches.

The syntactical approach was developed through a thorough investigation of POS tagging and graph theoretic metrics. The development of the tool is detailed in section 0. Three combinations were found to be accurate based on the filters described in section

5.2.2. The combinations were combined into a combination set to relate requirements. The syntactical approach also incorporated a scoring metric, described in section 5.2.3. Each propagated requirement was scored with consideration to the all the previous requirement changes. The score was developed using an RMS approach to add significance to those requirements related in the first while incorporating second order relationships. This score was ranked amongst the rest of the requirements document similarly scored. A search depth, based on the score ranking, is given to evaluate the capability of the approach to predict change propagation.

10.2.1. Toho

The syntactical approach was capable of predicting the change propagation experienced in the Toho industry project. This approach could have saved the corporation the time and monetary losses suffered due to the unanticipated change propagation.

The syntactical approach in the Toho industry study identified change propagation to ECN03 within a search depth of 10% and ECN04 at a search depth of 3.1%. The syntactical approach was able to make use of a score ranking, a feature not available in the subject based manual approach. Using the two data points in this project, a maximum search depth of 10% is needed to evaluate the requirement change propagation.

The syntactical approach was able to outperform the manual approach through the results and the input of data. The manual approach was time consuming and required an engineer or designer review the document to identify subjects. The syntactical approach made use of the combination set to develop relationships between requirements.

10.2.2. Pierburg

The syntactical approach was used to analyze the Pierburg industry project and was capable of predicting all engineering changes. Pierburg made use of the same relaters as the Toho project in the syntactical analysis. Keywords were still used in the syntactical approach of the Pierburg industry study. However, nouns were included in the relater combination set. The results of the study indicated that at most, 11% of the requirements document would need to be reviewed to predict change propagation.

The performance of the Pierburg industry study on the known propagation was very high. Predicting the search depth for requirement 2.1.14 of ECN07 was only 3.3% of the requirements document. This was the 7th highest scoring requirement based on the scoring algorithm. The second requirement change in ECN07, requirement 2.2.6 had the highest ranking propagation score. This propagation was also noted through the corporation and the syntactical tool was able to identify this change with high strength. ECN11, which affected requirement 2.1.14, was a questionable propagation that was included in the study to identify if this propagation could be one that went unnoticed by the client and customer. This caused the performance of the Pierburg industry study to weaken as the score ranked as the 23rd highest. As a result, the depth of search was 10.7% of the requirements document.

The syntactical approach performed admirably on the Pierburg project as it was a very strong relater in those change propagations that were sure to be propagations and did well in those changes that are believed to be propagating with low certainty.

10.2.3. EVRAZ

The EVRAZ industry study syntactical analysis was the first analysis performed through the syntactical approach after its development. The Toho and Pierburg syntactical analysis were formed after the tool was generalized. The EVRAZ industry study was a suitable study for the syntactical approach as it was a volatile project with many instances of change of which the tool could be evaluated upon. A total of five requirement change propagations are predicted within EVRAZ, with a requirement changing twice. This is of interest because the syntactical approach allows for self-propagation and this was an opportunity to test this capability.

The first change occurred during instance03 to requirement 70. The results of the study indicated a depth of 3.8% would need to be reviewed to identify the requirement change propagated. The next change occurred to requirement 133 at change instance07 which yielded a search depth of 8.6%. During change instance08, two changes took place and are both recorded as separate ECNs, ECN03 and ECN04. Both could be found within a search depth of 2.2%, with one ranking as the highest score. Up to this point in the analysis, the maximum search depth is 8.6%. This was an admirable search depth and the tool revealed its possible utility in industry practice.

The final change is of great interest here as it is a self-propagation, an added feature of the syntactical approach. Though self-propagation operates by relating the requirement to itself, hence inflating its score, it does not rank the requirement as high as it should. This is both a positive and negative attribute. The positive here is that the ranking will not always rank the previous requirement changes ahead of all other requirements. The negative to this is that self-propagation, while a beneficial utility, may be difficult to predict. As noted in the EVRAZ industry study, the self-propagation was found at a rank of twenty four, yielding a search depth of 13%.

10.2.4. Advantages and Limitation

The advantages of the syntactical analysis are the immediate benefits of predicting change propagation. The existing requirements modeling and managing tools in industry and research were presented in this dissertation. The tool developed in this research offers a distinct advantage in its ability to manage requirement change propagation in an automated manner. The tool is accurate, as shown through the multiple studies, and can address various forms of change propagation. Based on the studies performed and the existing resources, the tool provides an innovative and unmatched approach to predicting requirement change propagation.

A consideration when using such a system is the time and effort required to maintain the tool. An associate, usually the designer or engineer, at all times must maintain such a system to ensure all requirements, and their relationships are up to date. This requires time as requirements are continuously changing and new requirements are introduced thereby resulting in changes in relationships. However, the cost benefit of using and maintaining a requirements change propagation prediction tool could prove to be financially sound as it may reduce losses resulting from mismanagement of changes. A specific example of this is seen in the initial study where the corporation lost nearly a month of time and thousands of dollars due to their inability to predict requirement change, based on interviews. The president of the corporation stated "this tool could have saved us a $100,000 because of unanticipated changes," indicating its industrial potential.

10.3. Comparison and Findings

A tabulated comparison of both approaches is shown in Table 10.2. The manual approach yields a search depth of 27% and 7.5% for the Toho and Pierburg industry studies respectively. The syntactical approach performed more consistently through all three industry studies, yielding a 10%, 11% and 13.0% search depth for the Toho, Pierburg, and EVRAZ industry projects respectively.

A comparison of the Toho industry studies for both approaches reveals the syntactical approach is able to predict requirement change propagation at an earlier search depth. This is a significant difference between the manual creation. The comparison of the Pierburg industry study for the manual and syntactical is of interest because both made use of multiple keywords. As seen, the manual approach is slightly better in predicting change propagation. However, the manual keyword approach is not suitable for the self-propagation experienced with requirement 2.1.14. As seen in Table

8.2, this requirement could not be analyzed for propagation. Nonetheless, the slight increase in performance of the manual over the syntactical approach does not offset the amount of time and effort needed to develop manual keywords for the manual based approach. Though the syntactical approach makes use of keywords, they are not always needed and keywords may be deactivated.

Table 10.2: Comparison of Results

	Project		
Relationship Approach	**Toho**	**Pierburg**	**EVRAZ**
Manual	27%	7.5%	-
Syntactical	10%	11%	13.0%

A total of ten requirements are predicted through the syntactical approach. The results of the studies indicate that all can be identified within 13% search depth. A graphical representation of the search depth vs. all ten changes predicted I shown in Figure 10.1. As seen from the results, an engineer or designer may not have to review all the requirement change propagation. Reviewing 13% of the requirement text will ensure all of the requirements propagated are predicted. However, as seen in the graph, 60% of the requirements predicted were predicted within a search depth of 4%. If an engineer or designer is satisfied with reviewing most of the change propagation that may occur, a search depth of 4% is suitable.

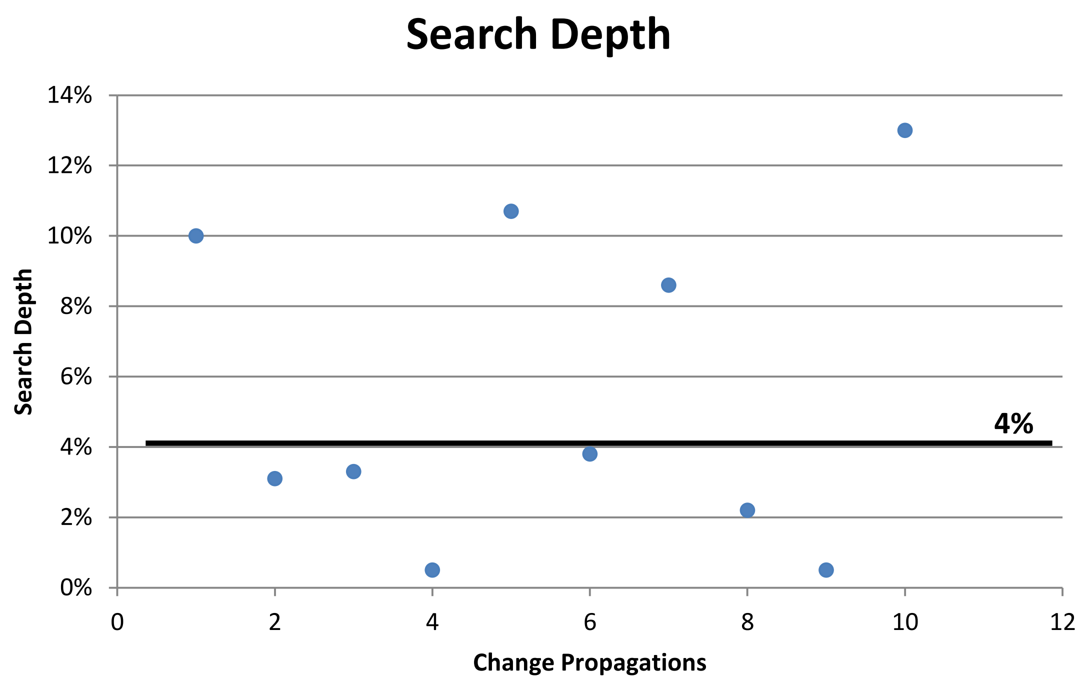


Figure 10.1: Search Depth of Syntactical Studies

From an effort perspective, the manual approach requires great effort into developing the relationships. This requires reviewing the document, gaining familiarity with the word frequency and significance of the words within each sentence and developing relaters based on this. Further, the syntactical approach was developed on MATLAB, an efficient program in processing the code script of the propagation. The manual approach is performed in Microsoft Excel, which is not efficient for processing large quantities of data.

Though keywords are needed in the syntactical approach, their use may be deactivated to make it a fully automated system. However, the deactivation of keywords has not been investigated. It can be hypothesized that the deactivation of keywords will have a small impact on the propagation capability of the system because the keywords selected in the optimal combinations were keywords 3 and 5. Since the keywords were listed in order of frequency in the keyword bank, keywords three and five may not possess many relationships with other requirement text. As seen in the keyword analysis shown in section 5.1.3, the fifth keyword only adds 6% more relationships past that of 3 keywords.

The syntactical approach showed more consistency in results than the manual based approach. Though the manual approach varied in relaters, the use of a manual based approach indicated much deviation in the propagation results. The syntactical approach results indicated a search depth within relative range of one another.

The syntactical analysis is also preferred here because it allows for cumulative propagation analysis. Both manual based approaches only consider the propagation due to a single previous requirement change. The syntactical approach is able to address a dynamic of change propagation that the manual approach is not. This is a significant benefit of the syntactical approach that is needed to function as a change propagation tool.

There were two important findings in the studies. It was found that second order relationships appear to be the key to predicting requirement change propagation. While first order relationships have the greatest impact on change propagation, they are often anticipated and addressed early. Second order relationships are of great interest because most designers and engineers cannot predict the propagation of changes to such length. In feedback with the corporation clients, the first order subject based relationships were recognized as generally easy to predict. For example, if a change is made to the spindle on the machine, the machine spindle may affect the bearing holding the spindle. However, a second order relation arises when the bearing, in turn, has an effect on an adjacent snap ring. As a result, while the spindle is not directly related to the snap ring, there is a second order relationship there that may cause a change in the snap ring. It is difficult for designers to intuitively predict changes in the second order, especially for complex systems which may have hundreds or thousands of requirements. Rarely was unforeseen propagation occurring in first order form, rather it was occurring in second order, as seen in some of the study results. This introduces an interesting dynamic to propagating requirements that cannot be recognized by simple designer attentiveness to change; rather the use of a requirement change propagation prediction tools is needed.

In all three studies, a third order relationship was deemed inadequate. At the third order, nearly 95% of all the requirements are related, making it difficult to accurately predict changes due to the superfluous relationships. This was also identified during the iterations weighting each relationship order, when a 9-3-0 weighting was selected. If weighted, third order relationships distorted the results because of the number of requirements related at that path length.

10.4. Statistical Validation

Though the syntactical tool was capable of predicting requirement change propagation in every instance, statistical validation is needed to ensure this test is of significance. A binomial test to measure the statistical significance of the finding is presented. The purpose of performing a binomial test is to identify what is the probability the syntactical tool predicted propagated requirements by chance. As a result, the hypothesis tested here are:

H_0 = The tool cannot predict the requirements which propagated (p-value = normal probability)
H_A = The tool is can predict the requirements which propagated (p-value < normal probability)

The hypothesis tests if the tool is capable, to an $\alpha = 0.001$, predict change propagation. As noted previously, though $\alpha = 0.001$ is a stringent alpha for rejecting the null hypothesis, it is used here to ensure quality research is performed. Further, it is important in this research to avoid Type I statistical errors, which is reduced when alpha is low. A Type I statistical error occurs when the null hypothesis is wrongfully rejected, effectively resulting in a false positive test. It is important to show that this tool is capable of predicting requirement change propagation and as a result, a very low alpha is used to support this claim and reduce the chance of a Type I error. The chance of a Type I error here is simply alpha, which is 0.1%.

The statistical validation on the syntactical analysis for each of the requirement change propagation prediction instance is presented in Table 10.3. This hypothesis is tested against the normal probability which is effectively the chance of selecting the correct requirement from the number of trials needed. The trials here are the ranking of the requirement during the scoring. For instance, the prediction of requirement 9.3.7 from the Toho project was the 16th highest requirement scored. The designer or engineer must review at least sixteen requirements to identify the propagated requirement. In this instance, there are sixteen trials needed before the successful change is identified.

Predicting ECN07, which affected two requirements, is predicted through seven trials in which two successes are identified.

Each prediction has an individual p-value for that prediction. A cumulative prediction is noted which identifies the probability of all the propagations to be predicted. Since this is tested against an α = 0.001, the cumulative p-value is 2.2e-16, which rejects the null hypothesis. The p-value of 2.2e-16 is not exact but is used because the lowest tabulated p-value available for this test is 2.2e-16. ***The conclusion from this statistical analysis is there is sufficient statistical evidence that the tool is able to predict requirement change propagation***.

Table 10.3: Binomial p-value of Prediction

Industry Project	**Requirement Predicted**	**Trials (Ranking)**	**P-value**
Toho	9.3.7	16	0.09545
	9.3.10	5	0.03086
Pierburg	2.1.14 and 2.2.6	7	0.00045
	2.7	23	0.10210
EVRAZ	70	7	0.03703
	133	16	0.08264
	98	4	0.02133
	89	1	0.00538
	70	24	0.12140
Cumulative p-value			**2.2e-16**

10.5. Implications and Impact of Tool

The motivation for the development of a requirement change propagation tool is the possible advantages it can offer both industry and research interest in the field of requirements and change management. The tool was developed and validated against industry studies, to prove its potential to serve as a viable design tool for designers and engineers to use throughout the design process. Further, important findings relating requirements, POS tagging, and graph theoretic are made to enhance the field of requirements research.

10.5.1. Industrial Design and Engineering Practice

This study identified that it is possible for an automated tool to predict engineering change through requirement change propagation. Such a tool could have an immediate impact on how engineering changes are handled, modeled, and analyzed. Many monetary and time losses are involved with change propagation, as was identified when performing the study. Currently, no tool exists to mitigating the issues involved with change propagation. An automated tool could have an immediate impact on change practice and how system requirements are handled throughout the design process.

Pierburg was a project of interest as many requirements where changed throughout the process. One of the documents of the Pierburg project specifically stated:

"Beginning of project very bumpy, many ECNs, proposals and confusion up front"

This is an indication of the volatile nature of projects such as this. A requirements change propagation prediction tool can assist in nearly all domains of design. On another project, a document stated:

"Automotive job, changes will happen. Lots of ECNs"

The possibilities for this tool in industry are broad and can assist in many of the facets described in Section 1.2. Feedback from the corporation has indicated that this tool could have assisted in many of the changes propagated. Many of the changes resulted in monetary losses due to inaccurate cost estimations of the change.

10.5.2. Engineering Design Research

Requirements play the most critical role within the design process, yet they are the least utilized for their ability to manage the design process. While arguably the most important document within a design process and late in the design process to ensure satisfaction, but are generally not used outside of the initial design phases. While they are one of the first documents generated, they are also one of the last documents completed. Throughout this evolution, it is analyzed that more than half of a systems requirements will be changed before it is completed [58,59]. This change is important as it can have significance on the cost of the final part. Engineering changes occur frequently and can at times determine as much as 70 to 80% of the final cost of a product [195].

Design research focuses on finding and developing tools that aid the designer, increase collaboration, or make the design process more efficient. This research will assist in understanding how requirements, specifically their change, can be managed through an automated tool and assist designers and engineers make management decisions. Additionally, a resource which predicts change propagation can be an invaluable tool to any design process. This will increase research in the domain of requirements, requirements management, and change management. From an industrial standpoint, this research will benefit designers by minimize the effect of requirement change on the design and product costs.

A contribution of this research is revealing how requirements can be related. A major requirement research is known as requirements linking, where requirements are related to other requirements or elements of the design process. While this research makes use of relaters for change propagation prediction purposes, it introduces an innovative means for relating requirements. This innovative means can also be automated, reducing the effort and expertise required to develop relationship.

It is important to note the DSMs created here are intentionally not created by system experts who worked on the design. The aim of this tool is to be robust enough to operate with less experienced designers and engineers, making it usable by anyone. Research has been performed to view comparison between a requirement based DSM and an expert generated DSM and revealed those developed by experts have less relations between requirements, however additional relations do not adversely affect the approach [196].

CHAPTER 11.
CONCLUSION AND FUTURE WORK

The broader impact of this research exists within the research and industrial application domain. Requirement research within the electromechanical domain is limited and is a relatively new area of research that has recently introduced itself into the realm of engineering design. Further requirement change has been recognized as an area of potential need, yet few methods exists to address many of the issues of mismanaged requirements. Change management has been addressed through models and tools which require, at the minimum, component architecture developed. As a result, designers and engineers are left to manage change propagation with minimal assistance, vulnerability in the design process.

The purpose of this research was to develop a requirements change management tool designers can use to assist managing the volatile nature of requirements. Further, the ability of predicting changes, both within the physical and requirements domain, was explored through the use of requirement relaters. Requirements were related through the use of specific syntactical elements (subjects, keywords, and POS). Two studies were performed to identify if subjects and keywords could be used to relate requirements in a manner conducive to predicting change propagation. The results indicated that requirements could be used; however there were many limitations in a manual approach. A syntactical approach was developed, generalized through the Toho and Pierburg industry studies, and validated against an EVRAZ industry study.

The tool was developed by investigating 32,768 permutations of relater combinations of nouns, verbs, and manually selected keywords, referred to as a syntactical approach. Three combinations where selected and overlaid to develop an optimal combination set. Using the combination set, requirement relationship order, which were found to be of significance during the manual approach studies, are weighted. An RMS approach is taken to add significance to first order relationships. Using the scoring algorithm, all forms of requirements propagation can be propagated forward. This includes propagation from a single requirement or cumulative changes of requirements which happen at different instances.

The manual creation approach yielded results that indicated an engineer would have to search a depth of 27% if no ranking system is used and all first and second order relationships are addressed. When a scoring system is introduced, ranking the path count for each requirement, the Pierburg keyword based study search depth is decreased to 7.5%.

The syntactical approach makes use of a more intricate scoring system which amplifies first order relationships without sacrificing second order relationships. Further, an automatic approach is presented for developing requirement relationships, with the option of incorporating manually selected keywords. Using this approach, favorable results are achieved as the maximum search depth for the Toho, Pierburg, and EVRAZ projects are 10%, 11%, and 13% respectively.

This research develops a means of using requirements documentation, regardless of the phase of the design process to assess requirement change propagation. The benefits of such a tool will provide the designer to ability to assess the value of making design changes by viewing its potential propagation effect. When change is needed, a requirements change propagation tool can assist in determining the impact of this change. The designer is able to view what other requirements must be revisited to ensure the change does not adversely affect other requirements or facets within the design. External to propagation, this tool will help alerting the designer of the critical requirements within the design of a system. These requirements are those which have the greatest sensitivity due to their number or strength of relationships with other requirements. By viewing this, a designer can continuously ensure such requirements are satisfied. The benefit of this tool is to assist the designer throughout the design process, specifically during times of change.

Change will occur, as with any design process, and the ability to properly anticipate and manage this change can relieve the designer of monetary losses and inefficiencies within the design process. Designers and engineers are not equipped to handle such change due to their limited ability to reason change propagation. It was noted that most change propagation that occurred were due to two reasons: (1) too many first order propagations or (2) unforeseen second order propagations. A designer or engineer can only reason so many elements of a design in parallel. When a change is made to a single or multiple, in complex changes, requirements it is difficult to cognitively reason all the possible path of propagation and the depth at which this propagation will travel.

The development of a computational tool to address and mitigate the risks of requirement change propagation can be immediately implemented into design practice. A computational tool allows for the use of this tool with minimal user effort as relationships are automatically developed and does not require a specific level of expertise. The tool is also robust to accept any form of requirement change documents. A tool such as this could be an add-on feature on other requirement management systems to incorporate a change propagation prediction component or function as a standalone system. For this reason, AEC is considering adoption of the predictive strategy to help guide the engineers during the ECN proposal process so that accurate costs can be comprehensively predicted.

REFERENCES

[1] G Pahl and W Beitz, *Engineering Design: A Systematic Approach*, 2nd ed.: Springer, 1998.

[2] Karl T Ulrich and Steven D Eppinger, *Product Design and Development*, 1st ed.: McGraw Hill, 1995.

[3] David G. Ullman, *The Mechanical Design Process*, 3rd ed.: McGraw-Hill, 2003.

[4] Kevin Otto and Kristin Wood, *Product Design*.: Prentice Hall, 2000.

[5] Elizabeth Hull, Ken Jackson, and Jeremy Dick, *Requirements Engineering*, 2nd ed.: British Library Cataloguing, 2005.

[6] Zhen Yu Chen, Shengji Yao, Jian Qiang Lin, Yong Zeng, and Armin Eberlein, "Formalisation of product requirements: from natural language descriptions to formal specifications," *International Journal of Manufacturing Research*, vol. 2, no. 3, pp. 362-387, 2007.

[7] Tal Cohen and Robert E. Fulton, "A Data Approach to Tracking and Evaluating Engineering Changes," in *Design Engineering Technical Conferences*, Atlanta, 1998.

[8] Clive L Dym, *Engineering Design a Synthesis of Views*.: Cambridge University Press, 1994.

[9] "Managing Engineering Design Information," in *Aircraft Design, Systems & Operations Conference*, 1988.

[10] K.M. Marchek M. Kannapan, "A Schema for Negotiation Between Intelligent Design Agents in Concurrent Engineering," *Intelligent Computer Aided Design*, 1992.

[11] Andreas S Andreou, Andreas C Zographos, and George A Papadopoulos, "A Three-Dimensional Requirements Elicitation and Management Decision-Making Scheme For The Development Of New Software Components".

[12] Stig Ottosson, "Dynamic Product Development: Findings from Participating Action Research in a Fast New Product Development Process," *Journal of Engineering Design*, vol. 7, no. 2, pp. 151-169, 1996.

[13] Beshoy Morkos et al., "Conceptual Development Of Automotive Forward Lighting System Using White Light Emitting Diodes (Led)," *SAE International Journal of Passenger Cars- Electronic and Electrical Systems*, vol. 2, pp. 201-211, October 2009.

[14] Timothy Hess, Beshoy Morkos, Mark Bowman, and Joshua Summers, "Cross Analysis of Metal Foam Design Parameter for Achieving Desired Fluid Flow," in *ASME International Mechanical Engineering Congress & Exposition*, Denver, Colorado, 2011.

[15] W. Lam and V. Shankararaman, "Requirements Change: A Dissection of Management Issues," in *25th Euromicro Conference*, 1999, p. 2244.

[16] M. Lehman, "Software Future: Managing Evolution," *Software, IEEE*, vol. 15, no. 1, pp. 41-44, 1998.

[17] M R Cutkosky and J M Tenenbaum, "Toward a Computational Framework for Concurrent Engineering," in *IECON Industrial Electronics Society*, 1990.

[18] P John Clarkson, Caroline Simons, and Claudia Eckert, "Predicting Change Propagation in Complex Design," in *20th IEEE International Conference on Software Maintenance*, vol. 126, Sept 2004.

[19] M. Lehman and L Belady, *Program evolution: processes of software change*.: Academic Press, 1985.

[20] Yuh-Min Chen, Wei-Shin Shir, and Chung-Yen Shen, "Distributed engineering change management for allied concurrent engineering," *International Journal of Computer Integrated Manufacturing*, vol. 15, no. 2, pp. 127-151, 2002.

[21] Gary S. Palmer, Beshoy Morkos, and Joshua D. Summers, "Investigation of Design Tools as Complexity Management Techniques," in *International Design Engineering Technical Conference*, Montreal, Canada, 2010.

[22] Zhen Yu Chen and Yong Zeng, "Classification of Product Requirements Based on Product Environment," *Concurrent Engineering: Research and Applications*, vol. 14, no. 3, pp. 219-230, September 2006.

[23] Beshoy Morkos, Shraddha Joshi, Joshua D. Summers, and Greg G. Mocko, "Evaluation Of Requirements And Data Content Within Industry In-House Developed Data Management System," in *International Design Engineering Technical Conference*, Montreal, Canada, 2010.

[24] Beshoy Morkos and Joshua D. Summers, "Requirement Change Propagation Prediction Appraoch: Results From an Industry Case Study," in *International Design Engineering Technical Conference*, Montreal, Canada, 2010.

[25] G.A., Stahovich, T.F. Ollinger, "REDESIGNIT - A Constraint-based Tool for Managing Design Changes," in *International Design Engineering Technical Conference*, Pittsburgh, 2001.

[26] R., Strens, M. Sugden, "Strategies, tactics and methods for andling change," in *IEEE Symposium and Workshop o n Engineering of Computer-Based Systems ECBS*, 1996.

[27] S., Eason, K., Dobson, J., Harker, "The change and evolution of requirements as a challenge to the practice of software engineering," in *IEEE international symposium on requirements engineering*, San Diego, California, 1993.

[28] Hongxing Cheng, Yuanqiang Xia, and Xia Hu, "Requirements Change Management of Information System Based on Keyword Mapping," in *The Sixth Wuhan International Conference on E-Business*, pp. 135-140.

[29] G.Kotonya and I.Summerville, "Viewpoints for Requirement Definition," *Software Engineering Journal*, vol. 7, no. 6, pp. 375-387, 1992.

[30] Yong Zeng, "Environment-Based Formulation of Design Problem," *Journal of Integrated Design and Process Science*, vol. 8, no. 4, pp. 45-63, December 2004.

[31] James McLellan, Beshoy Morkos, Mocko G. Greg, and Joshua D. Summers, "Requirements Modeling Systems for Mechanical Design: A Systematic Method for Evaluating Requirement Management Tools and Languages," in *International Design Engineering Technical Conference*, Montreal, Canada, 2010.

[32] Beshoy Morkos, Prabhu Shankar, and Joshua Summers, "Predicting Requirement Change Propagation Using Higher Order Design Structure Matrices: An Industry Case Study," *Journal of Engineering Design*, Submitted and Accepted 2011.

[33] Beshoy Morkos, James Mathieson, and Joshua Summers, "Analysis of Requirements Change Propagation Dependencies Through Graph Theoretic Metrics," *Journal of Engineering Design*, vol. Submitted, 2011.

[34] Beshoy Morkos, James Mathieson, Joshua D. Summers, and Jaret Mathews, "Development of NASA Endurance Testing Apparatus Simulating Wheel Dynamics and Environment on Lunar Terrain," in *SAE World Congress and Exhibition, Tire and Wheel Technology*, Detroit, MI, 2010.

[35] Darrell Barker, "Requirements Modeling Technology a Vision for Better, Faster, and Cheaper Systems," in *VHDL International Users Forum Fall Workshop*, 2000, pp. 3 - 6.

[36] Rawling R., Hall A. Hammond J., "Will it Work?," in *Fifth IEEE Int. Symp. on Requirements Engineering*, Toronto, Canada, 2001, pp. 102-109.

[37] Sergey Diev, "Requirements Development As a Modeling Activity," *ACM SIGSOFT Software Engineering Notes*, vol. 32, no. 2, March 2007.

[38] Karl E. Wiegers, Software Requirements: Practical Techniques for Gathering and Managing , 2nd ed.: Microsoft Press, 2003.

[39] INCOSE, Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, no. TP-2003-002-03 in Version 3, Ed., 2006.

[40] Ralph R Young, *Effective Requirements Practices*. Boston: Addison-Wesley, 2001.

[41] J.O. Grady, System Verification: Proving the Design Solution Satisfies the Requirements. New York: Elsevier, 2007.

[42] Beshoy Morkos and Joshua Summers, "Elicitation and Development of Requirements Through Integrated Methods," in *International Design Engineering Technical Conference*, San Diego, 2009.

[43] N.P. Suh, "The Principles of Design," Oxford University Press, 1990.

[44] M Williams, "Enabling Schoolteachers to Participate in the Design of Educational Software," in *Proc. Participatory Design Conf*, 1994, pp. 153-157.

[45] H Holbrook, "A scenario-based methodology for conducting requirements elicitation.," *SIGSOFT Softw. Eng. Notes*, vol. 15, no. 1, pp. 94-104, January 1990.

[46] Suranjan Chakraborty, Saonee Sarker, and Joseph S Valacich, "Understanding Analyst Effectiveness In Requirements Elicitation: A Gestalt Fit Perspective".

[47] Ljerka Beus-Dukic and Ian Alexander, "Learning How To Discover Requirements," in *Requirements Engineering Education and Training*, 2008, pp. 12-14.

[48] Griffin, Abbie, and Hauser, "The Voice of the Customer," *Marketing Science*, vol. 12, no. 1, Winter 1993.

[49] Deepti Mishra, Alok Mishra, and Ali Yazici, "Successful Requirement Elicitation by Combining Requirement Engineering Techniques," in *International Conference on the Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008*, 2008, pp. 258-263.

[50] Beshoy Morkos and Joshua D. Summers, "Development and Representation of Requirements to Support Collaborative Design: An Industry OEM Case Study," *International Journal of Product Development*, *In Preparation*, 2011.

[51] Pazos-Arias J., Garcia-Duque J., Barragans-Martinez B Lopez-Nores M., "An agile approach to support Incremental Development of Requirements Specifications," in *Proceeding of the Australian Software Engineering Conference*, 2006, pp. 258-263.

[52] Anne Bruseberg and Deana McDonagh-Philp, "New Product Development by Eliciting User Experience and Aspirations," *International Journal Human-Computer Studies*, vol. 55, pp. 435-452, 2001.

[53] Evan W. Duggan, "Generating System Requirements With Facilitated Group Techniques," *Human-Computer Interaction*, vol. 18, pp. 373-394, 2003.

[54] L. Lertiz-Higgins, "Practicing Persona Development: An In-House Case Study," *Usability and Information Design*, pp. 350-355, 2004.

[55] Beshoy Morkos and Joshua D. Summers, "Implementing Design Tools in Capstone Design Projects: Requirements Elicitation Through Use of Personas," in *Capstone Conference*, Boulder, CO, 2010.

[56] R. J. Barnes, D. C. Gause, and E. C. Way, "Teaching the Unknown and the Unknowable in Requirements Engineering," in *Requirements Engineering Education and Training REET '08.* , 2008, pp. 30-37.

[57] N Nurmuliani, Didar Zowghi, and Susan P. Williams, "Requirements Volatility and Its Impact on Change Effort: Evidence-based Research in Software Development Projects," in *AWRE*, Adelaide, Australia, 2006.

[58] Atsushi Kobayashi and Mamaro Maekawa, "Needs Based Requirements Change Management," in *IEEE International Conference and Workshop on the Engineering of Computer Based Systems ECBS*, 2001.

[59] Saffena Ramzan and Naveed Jkram, "Making Decision in Requirement Change Management," in *First International Conference on Information and Communication Technologies* , 2005, pp. 309-312.

[60] Monica Giffin et al., "Change Propagation Analysis in Complex Technical Systems," *Journal of Mechanical Design*, vol. 131, Aug 2009.

[61] Eddie Smith, "Re-Engineering a Trash/Recycling Collection Vehicle - Based on Challengine Customer Requirements," Clemson University, Masters Thesis 2010.

[62] Beshoy Morkos and Gary S Palmer, "A Study Of Designer Familiarity With Product And User During Requirement Elicitation," in *Eighth International Symposium on Tools and Methods of Competitive Engineering*, Ancona, Italy, 2010.

[63] Prabhu Shankar, Beshoy Morkos, and Joshua D. Summers, "A Hierarchical Modeling Scheme With Non Functional Requirements ," in *ASME Design Engineering Technical Conference*, Montreal, Canada, 2010.

[64] Christos Spitas, "Analysis of Systematic Engineering Design Paradigms in Industrial Practice: A Survey," *Journal of Engineering Design*, vol. 22, no. 6, pp. 427-445, 2011.

[65] Sándor Vajna, Steffen Clement, André Jordan, and Tibor Bercsey, "The Autogenetic Design Theory: An Evolutionary View of The Design Process," *Journal of Engineering Design*, vol. 16, no. 4, pp. 423-440, 2005.

[66] Ottosson Stig, "Dynamic Product Development — DPD," *Technovation*, vol. 24, no. 3, pp. 207-217, March 2004.

[67] J. Brown, "Managing Product Relationships Enabling Iteration and Innovation in Design," Aberdeen Group, Boston, 2006.

[68] C., Clarkson, P., Zanker, W., 2004 Eckert, "Change and Customization in Complex Engineering Domains," in *Research in Engineering Design*., 2004.

[69] Stig Ottosson and Evastina Björk, "Research on Dynamic Systems—Some Considerations," *Technovation*, vol. 24, no. 11, pp. 863-869, November 2004.

[70] I C Wright, "A review of research into engineering change management: implications for product design," *Design Studies*, vol. 18, no. 1, pp. 33-42, January 1997.

[71] P., Malmqvist, J. Pikosz, "A Comparitive Study of Engineering Change Management in Three Swedish Engineering Companies," in *Design Engineering Technical Conference*, 1998, pp. 78-85.

[72] S.R., Tomer, A Schach, "A Maintenance-oriented Approach to Software Construction," *Journal of Software Maintenance Research and Practice*, vol. 12, no. 1, pp. 25-45, 2000.

[73] V. Ravlich, "Modeling Software Evolution by Evolving Interoperation Graphs," *Annals of Software Engineering*, vol. 9, pp. 235-248, 2000.

[74] Prabhu Shankar, Beshoy Morkos, and Joshua D Summer, "Reasons for change propagation: A case study in an automotive OEM," *Research in Engineering Design*, In Press 2010.

[75] Yutaka Matsuo and Mitsuru Ishizuka, "Keyword Extraction from a Single Document Using Word Co-occurrence Statistical Information," *International Journal on Artificial Intelligence Tools*, vol. 13, 2004.

[76] Beshoy Morkos, "Computational Representation and Reasoning Support for Requirements Change Management in Complex System Design," Clemson University, Clemson, SC, Ph.D. Dissertation 2011.

[77] John R. Hauser and Don Clausing, *The House of Quality*.: Harvard Business Review, 1988.

[78] Andrew Olewnik and Kemper Lewis, "Can a House Without a Foundation Support Design?," in *International Design Engineering Technical Conference*, Long Beach, CA, 2005.

[79] Beshoy Morkos and Joshua D. Summers, "A Survey of Requirement Change Types," in *ASME International Design Engineering Technical Conference*, Chicago, IL, 2012.

[80] Bashar Nuseibeh, Jeff Kramer, and Anthony Finkelstein, "Expressing the Relationships Between Multiple Views in Requirements Specifications," in *International Conference on Software Engineering*, Baltimore, MD, 1993.

[81] P. Zave and M. Jackson, "Conjuectoin as Composition," in *ACM Transactions on Software Engineering and Methodology*, 1993.

[82] Bashar Nuseibeh and Steve Easterbrook, "Requirements Engineering: A Roadmap," in *Proceedings of the Conference on The Future of Software Engineering* , Limerick, Ireland, 2000.

[83] S.R. Schach, Classical and Object-Oriented Software Engineering, 4th ed. London: McGraw-Hill, 1999.

[84] Grady Booch, James Rumbaugh, and Ivar Jacobson, *The Unified Modeling Language User Guide*, 1st ed.: Addison Wesley, ISBN 0-201-30998, 1998.

[85] S. Gnesi, D. Latella, and M. Massink, "A stochastic extension of a behavioural subset of UML statechart diagrams," in *Fifth IEEE International Symposim on High Assurance Systems Engineering*, 2000, pp. 55-64.

[86] Dr. Il-Yeol Song. (2003) Object Oriented Analysis and Design Using UML. [Online]. http://www.pages.drexel.edu/~ap62/pages/uc_3.html

[87] M. Fowler and Scott K., UML Distilled. Applying the Standard Object Modeling Language.: Addison-Wesley, 1997, 1997.

[88] Martin Glinz, "Problems and Deficiencies of UML as a Requirements Specification Language," *IEEE*, pp. 11-22, 2000.

[89] Feng Guo and Zhang Meng, "Translating UML statechart diagrams to X-nets," in Proceedings of the 2003 IEEE Symposium on Human Centric Computing Languages and Environments (IEEE Cat. No.03TH8722), 2003, pp. 198-200.

[90] Liu Jiufu and Yang Zhenxing, "Test generation from statechart and B method for flight control software of unmanned aerial vehicle," in *2009 4th International Conference on Computer Science & Education (ICCSE 2009)*, 2009, pp. 851-6.

[91] Stefan Kramer, Robert Kacsich Hermann Kaindl, "A Case Study of Decomposing Functional Requirements Using Scenarios," in *IEEE International Conference on Requirements Engineering, Third International Conference on Requirements Engineering (ICRE'98)*, 1998, p. 0156.

[92] Andrew Gemino and Drew Parke, "Use Case Diagrams in Support of Use Case Modeling Deriving Understanding from the Picture," *Journal of Database Management*, vol. 20.1, p. 1(24), Jan-Mar 2009.

[93] M. Griss, P. Johnson I. Jacobson, *Software Reuse: Architecture, Process and Organization*.: Addison-Wesley Professional, 1997.

[94] A. Fantechi, S. Gnesi, G. Lami, A. Maccari A. Bertolino, "Use Case Description of Requirements for Product Lines," in *Proceedings of the International Workshop on Requirements Engineering for Product Lines 2002 - REPL '02*, 2002, pp. 12-18.

[95] D. Batra, "Unified Modeling Language (UML) topics: The past, the problems, and the prospects.," *Journal of Database Management*, vol. 19, no. 1, pp. i-vii, 2008.

[96] "UML Use Case Diagrams," Engineering Notebook, 1998.

[97] D. Rosenberg and K. Scott, Use case driven object modeling with UML: a practical approach. (Addison-Wesley Object Technology Series).: Addison-Wesley, 1999.

[98] Liwu Li, "Use Case Patterns," International Journal of Software Engineering and Knowledge Engineering, vol. 12, no. 1, pp. 19-40, 2002.

[99] D. Kulak and E. Guiney, *Use cases - requirements in context.*, 1st ed.: Addison-Wesley Professional, 2000.

[100] J. A. Cruz-Lemus, M. Genero, M. Piattini, and M. Toval, "An empirical study of the nesting level of composite states within UML statechart diagrams," in *Perspectives in Conceptual Modeling. ER 2005 Workshop AOIS*, Klagenfurt, Germany, 2005.

[101] D. Harel, "Statecharts:a visual formalism for complex systems," *Science of Computer Programming*, vol. 8, pp. 231-274, 1987.

[102] D. Harel and Pnueli A., "On the development of reactive systems," in *Logics and Models of Concurrent Systems*.: Springer, 1985.

[103] R.J. Wieringa, Design Methods for Reactive Systems: Yourdon, Statemate and the UML (The Morgan Kaufmann Series in Software Engineering and Programming), 1st ed.: Morgan Kaufmann, 2003.

[104] Rik Eshuis, "Reconciling statechart semantics," *Science of Computer Programming*, vol. 74, no. 3, pp. 65-99, January 2009.

[105] D. Harel, "Statecharts in the making: A personal account," in Proceedings of the rd ACM SIGPLAN Conference on History of Programming Languages, 2007.

[106] Alexander Egyed and Dave Wile, "Statechart Simulator for Modeling Architectural Dynamics," in *Proceedings Working IEEE/IFIP Conference on Software Architecture*, 2001, pp. 87-96.

[107] J. Holt and S. Perry, "SysML: describing the system [Notebook: modelling languages]," *Information Professional*, vol. 3, no. 4, pp. 35-37, 2006.

[108] M. Hause, F. Thom, and A. Moore, "Inside SysML," *Journal of Computing & Control Engineering*, vol. 16, no. 4, pp. 10-15, 2005.

[109] OMG Inc., "OMG Systems Modeling Language (OMG SysML™)," 2010.

[110] T. Costa, A. Sampaio, and G. Alves, "Using SysML in Systems Design," in International Conference on Information Management, Innovation Management and Industrial Engineering, Porto, Portugal, 2009.

[111] M. Grecki, Z. Geng, G. Ayvazyan, S. Simrock, and B. Aminov, "Applicatoin of SysMLto design of ATCA based LLRF control system," in *Nuclear Science Symposium Conference. NSS'08 IEEE*, Hamburg, Germany , 2008.

[112] Chih-Hung Chang et al., "A SysML-Based Requirement Supporting Tool for Embedded Software," in *5th International Conference on Secure Software Integration & Reliability Improvement Companion*, 2011.

[113] Beshoy Morkos, "Analysis of DOORS," 2009.

[114] Noraini Ibrahim, Wan M. Nasir, W. Kadir, and Safaai Deris, "An Experimental Design Method For Evaluating Usability Factor Of ReChaP Process Model," *International Journal of Innovative Computing*, vol. 1, no. 1, 2011.

[115] Noraini Ibrahim et al., "ReChaP Prototype: A Tool for Simplifying Requirement Change Propagation to Software Design," in *The 3rd International Conference on Software Technology and Engineering*, Kuala Lumpur, Malaysia, 2011.

[116] Noraini Ibrahim, Wan M. Nasir, W. Kadir, and Safaai Deris, "Simplifying Requirement Change Propagation to Software Design," in *5th International Conference on Information & Communication Technology and Systems*, Surabaya, Indonesia, 2009.

[117] Noraini Ibrahim, Wan M. Nasir, W. Kadir, and Safaai Deris, "Propagating Requirement Change into Software Designs to Resilient Software Evolution," in *Asia-Pacific Software Engineering Conference*, Penang Malaysia, 2009.

[118] Yan Chen, Ping Cheng, and Jing Yin, "hange propagation analysis of trustworthy requirements based on dependency relations," in *Information Management and Engineering*, 2010.

[119] Simon Lock , Awais Rashid, Peter Sawyer, and Gerald Kotonya, "Systematic Change Impact Determination in Complex Object Database Schemata," in *Workshop on Object-Oriented Technology* , 1999.

[120] "An Integrated Framework for Requirement Change Impact Analysis," in *Australian Conference on Requirements Engineering*, 1999.

[121] S. Lock and G. Kotonya, "Requirement Level Change Management and Impact Analysis," Lancaster University Computing Department, Technical report CSEG/21/98 1998.

[122] D.V. Steward, "The Design Structure System: A Method for Managing the Design of Complex Systems," in *IEEE Transactions on Engineering Management*, 1981, pp. 71-74.

[123] T.R. Browning, "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions," in *IEEE Transactions on Engineering Management*, 2001, pp. 292-306.

[124] S., Whitney, D., smith, R.P., Gebala, D.A. Eppinger, "A Model Based Method for Organizing Tasks in Product Development," in *Research in Engineering Design*., 1994.

[125] R., de Weck, O. L Smaling, "Assessing Risks and Opportunities of Technology Infusion in System Design," *Journal of Systems Engineering*, vol. 10, no. 1, pp. 1-25, 2007.

[126] M.M. Lehman, "Software Evolution," *Encyclopedia of Software Engineering*, vol. 2, pp. 507–1513 , 2002.

[127] K. H. Bennett and V. T. Rajlich, "Software maintenance and evolution: a roadmap. ," in *Proceedings of the conference on The future of Software engineering*, 2000.

[128] Noraini Ibrahim, W.M. Nasir W. Kadir, Shahliza Abd Halim, Safaai Deris, and Maslina A. Aziz, "Synthetic Experiment in Evaluating the Usability Factor of the Requirement Change Propagation Process Model," in *Communications in Computer and Information Science*.: Springer LNCS, ISI Thomson Indexing, 2011.

[129] Ibrahim Bin Suhaimi, "A Document-Based Software Traceability to Support Change Impact Analysis of Object-Oriented Software," 2006.

[130] Ahmed E. Hassan and Richard C. Holt, "Predicting Change Propagation in Software Systems," in *Proceedings of the IEEE International Conference on Software Maintenance (ICSM)*, 2004.

[131] Ahmed E. Hassan and Richard C. Holt, "Replaying development history to assess the effectiveness of change propagation tools," *Journal of Empirical Software Engineering*, vol. 11, no. 3, pp. 335-367, 2006.

[132] Noraini Ibrahim, Safaai Deris, Wan Mohamad, and Nasir Wan Kadir, "A Review on Change Propagation Approaches in Evolvable Software," 2007.

[133] Henri Basson, "A Change Propagation Model and Platform For Multi-Database Applications," in *Proceeding ICSM '01 Proceedings of the IEEE International Conference on Software Maintenance* , Washington, DC, 2001.

[134] N. Ibrahim, W.M.N.W. Kadir, and S Deris, "Propagating Requirement Change into Software High Level Designs towards Resilient Software Evolution," in *Software Engineering Conference APSEC '09*, Asia-Pacific, 2009, pp. 347-354.

[135] M. Lindvall and K. Sandahl, "Traceability aspects of impact analysis in object-oriented systems," *Journal of Software Maintenance: Research and Practice, ,* vol. 10, no. 1, pp. 37-57, 1998.

[136] M. Glinz, "On Non-Functional Requirements," in *IEEE International Requirements Engineering Conference*, 2007.

[137] Wilhelm Hasselbring and Ralf Reussner, "Toward Trustworthy Software Systems," *Computer*, vol. 39, no. 4, pp. 91-92, 2006.

[138] S. Ajila, "Software Maintenance: An Approach to Impact Analysis of Objects Change," *Software-Practice and Experience*, vol. 25, no. 10, pp. 1155-1181, 1995.

[139] T. Goradia, "Dynamic Impact Analysis: A costeffective Technique to Enforce Error Propagation," in *Proceedings of the International Symposium on Software Testing and Analysis*, 1993, pp. 171-181.

[140] L. Li and A. J. Offutt, "Algorithmic Analysis of the Impact of Changes to Object-Oriented Software," in *Proceedings of the International Conference on Software Maintenance*, 1996, pp. 171-184.

[141] J. Han, "Supporting Impact Analysis and Change, Propagation in Software Engineering Environments," in *Proceedings of the IEEE International Workshop on Software Technology and Engineering Practice*, 1997, pp. 172-182.

[142] S. D. McCrickard and G. D. Abowd, "Assessing the Impact of Changes at the Architectural level: A case study on Graphical Debuggers," in *Proceedings of the International Conference on Software Maintenance*, 1996, pp. 59-67.

[143] Moriconi M. and T. C. Winkley, "Approximate reasoning about the effects of program changes," *IEEE Transactions on Software Engineering*, vol. 16, no. 9, pp. 980-992, 1990.

[144] R. Moreton, "A Process Model for Software Maintenance Software," in *Change Impact Analysis*, 1996, pp. 29-33.

[145] R. Moreton, "A process model for software maintenance," *Journal of Information Technology*, vol. 5, pp. 100-104, 1990.

[146] N. H. Madhavji, "Environment Evolution: The Prism Model of Changes," *IEEE Transactions on Software Engineering*, vol. 18, no. 5, pp. 380-392, 1992.

[147] J. Thompson, "What you need to manage requirements," in *IEEE Software*, 1994, pp. 115-118.

[148] Maria Carrascosa, Steven D. Eppinger, and Daniel E. Whitney, "Using The Design Structure Matrix to Estimate Product Development Time," in *ASME Design Engineering Technical Conference*, Atlanta, GA, 1998, pp. 13–16.

[149] A. A. Yassine, D. E. Whitney, and T. Zambito, "Assessment of rework probabilities for simulating product development processes using the design structure matrix (DSM)," *Engineering Conference*, no. 617, pp. 1-9, 2001.

[150] Josh King, "Design structure matrix-based Engineering Change management for product development," *International Journal of Internet Manufacturing and Services*, vol. 1, no. 3, p. 231, 2008.

[151] P. T. Helo, "Product configuration analysis with design structure matrix," *Industrial Management Data Systems*, vol. 106, no. 7, pp. 997-1011, 2006.

[152] John N. Warfield, "Binary Matrices in System Modeling," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, no. 5, pp. 441-449, September 1973.

[153] R. Keller, T. Eger, C. M. Eckert, and P. J. Clarkson, "Matrices Or Node-Link Diagrams: Which Visual Representation Is Better For Visualising Connectivity Models," *Information Visualization*, vol. 5, pp. 62-76, 2006.

[154] H. Simon, "The Sciences of the Artificial," MIT Press, Boston, 1996.

[155] G. Auriol, C. Baron, and J-Y. Fourniols, "Teaching requirements skills within the context of a physical engineering project," in *Requirements Engineering Education and Training (REET'08)*, 2008, pp. 6-11.

[156] Frank Smadja, "Retrieving Collocations from Text: Xtract," *Computational Linguistics*, vol. 19, no. 1, pp. 143-177, 1993.

[157] Rekha Bhowmik, "Keyword Extraction from Abstracts and Titles," *IEEE Southeastcon*, pp. 610-617, 2008.

[158] Li-Chen Tsai, Sheue-Ling Hwang, and Kuo-Hao Tang, "Analysis of Keyword-Based Tagging Behaviors of Experts and Novices," *Online Information Review*, vol. 35, no. 2, pp. 272-290, 2011.

[159] Inderjeet Mani and Mark T. Maybury, *Advances in Automatic Text Summarization*. London: MIT Press, 1999.

[160] Richard Alterman, "Text Summarisation," in *Encyclopedia of Artificial Intelligence*. New York: John Wiley, 1992, pp. 1579-1587.

[161] A. P. Shashkin, "On Newman's Central Limit Theorem," *Theory of Probability and its Applications*, vol. 50, no. 2, pp. 330-337, 2006.

[162] Hans Fischer, "History of the Central Limit Theorem," in *Sources and Studies in the History of Mathematics and Physical Sciences*.: Springer, 2011, p. 402.

[163] Iddo Eliazar and Joseph Klafter, "A Randomized Central Limit Theorem," *Chemical Physics*, vol. 370, no. 1-3, pp. 290-293, 2010.

[164] H. B. Mann and D. R. Whitney, "On A Test Of Whether One Of Two Random Variables Is Stochasticaly Larger Than The Other," *Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50-60, 1947.

[165] Frank Wilcoxon, "Individual Comparisons By Ranking Methods," *International Biometrics Society*, vol. 1, no. 6, pp. 80-83, 1945.

[166] Anthony Almudevar, "Selection of Statistical Thresholds in GraphicalModels," *Journal on Bioinformatics and Systems Biology*, vol. 2003, no. 1, 2003.

[167] Raghib Abu-Saris and Mowaffaq Hajja, "Quadratic Means," *Journal of Mathematical Analysis and Applications*, vol. 288, no. 1, pp. 299-313, 2003.

[168] Maurice Kendall and William R. Buckland, *Dictionary of Statistical Terms*.: Prentice Hall Press, 1975.

[169] Robert O. Curtis and David D. Marshall, "Why Quadratic Mean Diameter?," *Western Journal of Applied Forestry*, vol. 15, no. 3, pp. 137-139, 2000.

[170] Kim Iles and Lester J. Wilson, "A Further Neglected Mean," *Mathematics Teacher*, vol. 70, no. 1, 1977.

[171] Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer, "Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network," in *Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT)*, 2003, pp. 252-259.

[172] Kristina Toutanova and Christopher D. Manning, "Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger," in *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000),* 2000, pp. 63-70.

[173] Carl Lamar and Gregory M. Mocko, "Linguistic Analysis of Natural Language Engineering Requirements Statements," in *Tools and Methods for Competitive Engineering*, Ancona, Italy, 2010.

[174] Jefferey O. Grady, System verification: proving the design solution satisfies the requirements.: Academic Press, 2007.

[175] C Berge, *Graphs and Hypergraphs*. Amsterdam, Netherlands: North-Holland Publishing Company, 1973.

[176] J. Mathieson and J. Summers, "Relational DSMs in Connectivity Compexity Measurement," , Greenville, SC, 2009.

[177] I. Pramanick and H. Ali, "Analysis and experiments for a parallel solution to the all pairs shortest path problem," in *1994 IEEE International Symposium on Circuits and Systems*, vol. 1, New York, NY, USA, 1994, pp. 479-82.

[178] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum flow problem," in *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, New York, 1986, pp. 136-146.

[179] Tao Fang, "Visibility Problems and Their Resolution for Software Requirement Change Management," *Ziran Zazhi*, vol. 25, no. 6, pp. 332-335, 2003.

[180] S.J Andriole, Manage Systems: Requirements, Methods, Tools and Cases.: McGraw-Hill., 1996.

[181] S., Summers, J., Mocko, G Teegavarapu, "Case Study Methods for Design Research," in *ASME Design Engineering Technical Conference, DTM-4998*, Brooklyn, NY, 2008.

[182] R.K. Yin, Case study research: Design and methods. Applied social research methods series, 5th ed. London: : Sage Publications., 2003.

[183] R.E. Stake, "The case study method in social inquiry," *Educational researcher*, vol. 7, no. 2, pp. 5-8, 1978.

[184] R. Yin, *Case Study Research: Design and Methods*. Thousand Oaks, CA: Sage Publishing, 1994.

[185] Arch Woodside, *Case Study Research*.: Emerald Group Publishing, 2010.

[186] Robert E. Stake, *The Art of Case Study Research*.: Sage Publications, 1995.

[187] Glynis Cousin, "Case Study Research," *Journal of Geography in Higher Education*, vol. 29, no. 3, pp. 421-427, 2005.

[188] Joe R. Feagin and Anthony M. Orum, *A Case for the Case Study*.: The University of North Carolina Press, 1991.

[189] E., et al. Fricke, "Coping with changes: Causes, findings , and strategies," *Systems Engineering*, vol. 3, no. 4, pp. 169-179, 2000.

[190] (2011, July) How Textiles Are Made: What is the Creel? [Online]. http://brahmsmount.com/wordpress/wp-content/uploads/2011/07/IMG_1970.jpg

[191] [Online]. http://i00.i.aliimg.com/photo/v0/441538754/STABLE_CONSTANT_TENSION_YARN_CREEL.jpg

[192] Pierburg Exhaust Door. [Online]. http://seekpart24.com/pierburg/exhaust-gas-door-700509030

[193] Pierburg, "Exhaust Gas Flaps: For Emission Control and Increased Comfort," Product Information. [Online]. http://art.rexbo.net/pierburg/pdf/pg-pi-1004-en-web.pdf

[194] EVRAZ Calgary. [Online]. http://www.google.com/imgres?q=evraz+red+deer&um=1&hl=en&client=firefox-a&sa=X&rls=org.mozilla:en-US:official&biw=1280&bih=960&tbm=isch&tbnid=LVG_Sf3gvEvSsM:&imgrefurl=http://www.evrazna.com/LocationsFacilities/TubularOperations/CalgaryWorks/tabid/139/Def

[195] K.G., McIntosh, Engineering Data Management: A Guide to Successful Implementation. New York: McGraw Hill, 1995.

[196] Qi Dong, "Predicting and Managing System Interaction at Early Phase of Product Development Process," Massachusetts Institute of Technology, Cambridge, MA, Ph.D. Dissertation 2002.

[197] Michel Chaudron, "Requirements Engineering," Leiden University, Presentation 2007.

[198] Colette Rolland, Camille Salinesi, and Anne Etien, "Eliciting Gaps In Requirements Change," *Requirements Engineering*, vol. 9, pp. 1-15, Eng (2004).

[199] "ISO11442-6 (1996),".

[200] George E. Stark, Paul Oman, Alan Skillicorn, and Alan Ameele, "An examination of the effects of requirements changes on software maintenance releases," *Journal of Software Maintenance: Research and Practice*, vol. 11, no. 5, pp. 293 - 309, Oct 1999.

[201] Lan Lin and Jesse H. Poore, "Pushing Requirements Changes through to Changes in Specifications," in *2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering*, pp. 289-296.

[202] Richard Brooksby, "Requirements and Change," Ravenbrook Limited, 2003.

[203] Zhu Jiayi, Liang Yunjuan, and Gu Yuesheng, "The Requirements change Analysis for Different level users," in *International Symposium on Intelligent Information Technology Application Workshops,* 2008.

[204] Ross Ihaka. R Documentation: Level (Contour) Plots. [Online]. http://stat.ethz.ch/R-manual/R-patched/library/graphics/html/filled.contour.html

APPENDIX A
LOCALIZED REQUIREMENT CHANGE REPORT

Specifically, different types of requirements change are identified to acknowledge the different types of requirement changes that may be encountered. This is performed by surveying research performed in the realm of requirement change to construct a unique, all-encompassing taxonomy of types of changes. These requirement changes will be termed localized changes as the specific change within a single requirement statement is of interest.

Preliminary research has been performed in identifying types of change within individual requirements. This addresses the localized changes requirements experience. After identifying the types of requirement changes, a syntactical representation is developed so every specific syntax change may be correlated with its associated requirement change. The localized changes introduced here are preliminary in nature and identified through research surveying. It is outside the scope of this research to validate these types of changes; rather the purpose is to identify what types of changes may be experienced. The core contribution of the research lies within the global requirement change propagation prediction tool.

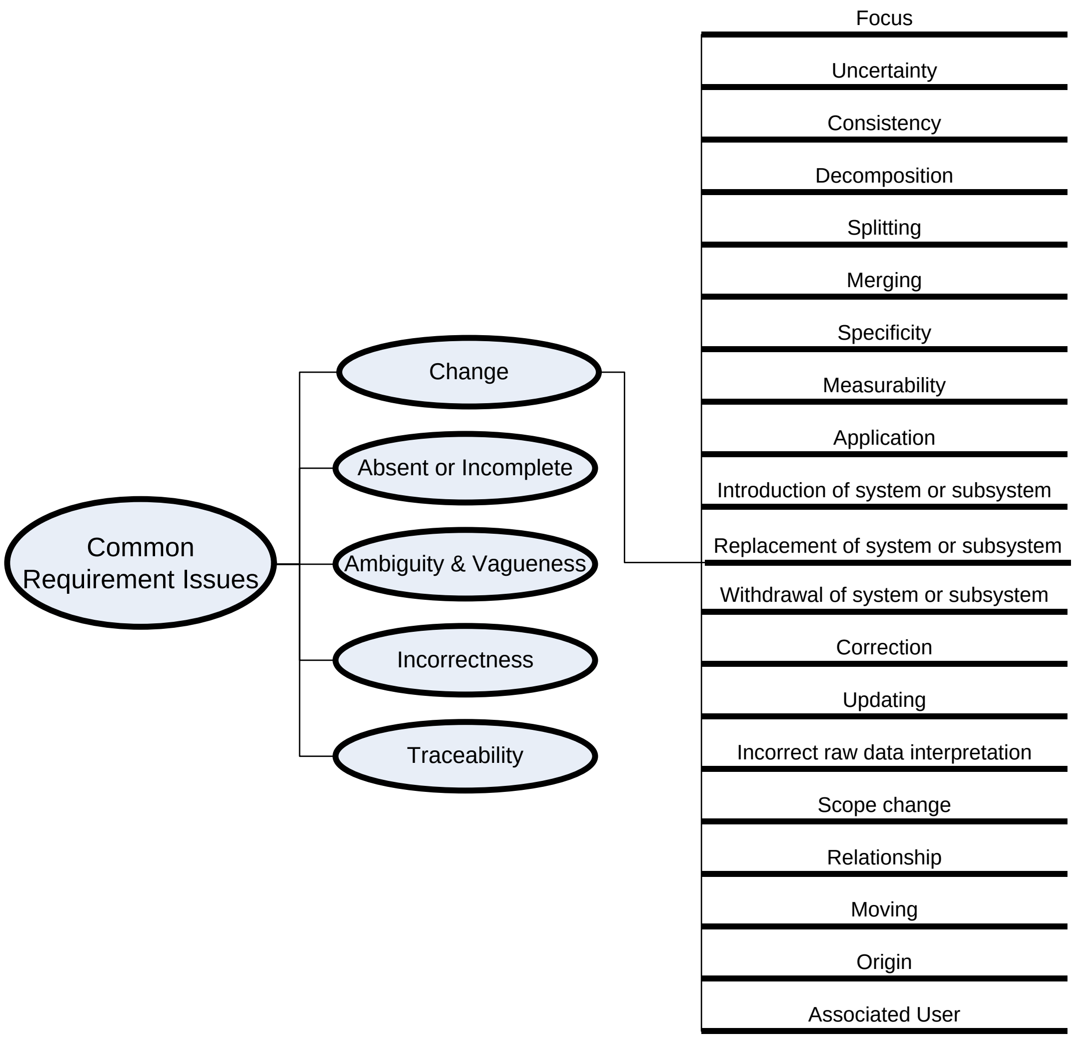


Figure 11.1: Common Localized Requirement Changes

The types of change encountered during with requirements evolution must be investigated so that such occurrence is understood and identified. This is a critical stage within the research as this will be the foundation for future investigation of requirements change, as described in the future works section of this dissertation (Chapter 11). Future work will entail developing correlation between requirement localized and global requirement changes in an attempt to identify if localized changes can assist in the accuracy of global requirement change prediction.

Localized changes will be evaluated by surveying requirements research, both in electromechanical and software systems. A taxonomy of requirement change types will be listed and the specific POS that change during each localized change will be determined. This will be used to enhance the understanding of requirement changes, within the syntactical domain, for single changes. Though this dissertation will not address correlations between localized changes and global changes, it is recognized as a future work.

Localized requirement changes occur specifically to a single requirement, investigating the POS changes within the requirement. This includes changes to a requirements noun such as system, subsystem, or component and changes to its verb such as function or behavior. These changes are important as they help in determining the type of changes that occur and how we these types of changes can be used at the global level. For example, it may be found that a specific type of localized changes results in specific global changes characteristics. The localized requirement changes presented in this dissertation will be represented by the POS changes of the sentence. This is the finest resolution of requirement change possible within requirements, evaluating the POS change of each requirement. Once the change encountered to a single requirement is well understood, this may lend itself to understanding how requirements, at globally change. This will be discussed in the future works chapter of this dissertation, Chapter 11.

A.1 - Identifying Types of Changes

The majority of the work performed with regard to engineering change has been to define and characterize engineering change propagation [68]. However, much work is needed in order to fully understand the mechanics of requirement change. Specifically, their localized changes must be analyzed. The first task in anticipating and evaluating change consequences is to represent them [7]. In representing requirement change, the localized changes are first examined. This is performed by identifying the types of changes encountered within design practice. Table 11.1 details the types of design changes that have been identified through research. The table lists the specific columns:

- ***Change***: The change that is encountered with a requirement. For example, "*Focus*" refers to a change in the focus of the requirement.
- ***Ref***: Reference where this requirements change was retrieved.
- ***Example***: Example providing this change in requirement. Examples are accompanied with before and after requirements viewing the occurrence of the specific change.
- ***Change Issue***: Issue that arises due to this type of micro level requirement change.
- ***Syntax Change***: The specific change in syntax during the occurrence of the requirement change. The syntax change can occur to the *System, Behavior/Characteristic,* and/or *Condition.* The change may occur to multiple syntax elements.

The information contained within Table 11.1 was populated through a survey of research on requirement change. Though this table may not be complete, few researchers have attempted to formally define all types of requirement changes. Further, none attempt to identify this change in the syntactical domain. The changes listed in the table were noted with their specific reference. An example, rooting from experience, of each type of change was recorded in the table. A description of the change issue is also presented to explain the phenomenon of that type of change. As this research also supports the creation of a representation, a syntax change is identified for each type of requirement change.

A specific syntax representation is shown in Table 11.2. This is shown to indicate specifically what changes within the syntactical structure of a sentence. This is the first step in defining and formally identifying localized requirement changes through their syntactical structure. By identifying the syntactical changes, any change that can occur to a requirement can be identified as a specific change type. Table 11.2 is color coded to identify the specific syntax which changed. Yellow shading indicates the syntax which changed while orange shadings represent those syntactical parts which change to support the primary, syntactical change depending on the nature of the change.

Table 11.1: Types of Micro Level Requirement Changes

Change	**Ref**	**Example**	**Change Issue**	**Syntax Change**
Uncertainty	[197]	***Before:*** *Changes in system excitement must stay within +/- 15% of operational frequency* ***After:*** *Changes in system excitement must stay within +/- 10 of operational frequency*	*Changes that occur in requirements ambiguity or uncertainty must be noted.*	*Condition*
Consistency	[197,198]	**Before:** Headlamp must output 2000 lumen when in high beam **After:** Headlight must output 2000 lumen when in high beams	Changes in vocabulary, units or terminology must be addressed.	System Behavior / Characteristic Condition
Decomposition (Atomic)	[197]	**Before**: Drill must operate, without failure, for 5 hours **After:** Motor must operate for 5 hours. Battery must last 5 hours. Transmission must operate for 5 hours.	Changes in system composition or decomposition must be accounted for.	System Behavior / Characteristic Condition
Splitting	[198]	**Before**: Power train must output at least 350 horsepower at a consumption not less than 20mpg **After:** **Req.1:** Engine must output at least 350 horsepower **Req.2:** Engine must provide gas consumption of at least 20mpg **Req.3:** Transmission must be capable of delivering 350hp	Allow for the merging of requirements into a single requirement.	System Behavior / Characteristic Condition

Table 11.1: Types of Micro Level Requirement Changes (continued)

Merging	[198]	**Before**: **Req.1:** Phone must input audible data **Req.2:** Phone must output audible data **After:** Phone must transfer audible data	Allow for the merging of requirements into a single requirement.	System Behavior / Characteristic
Specificity	[197]	**Before:** Weight of wheelchair must not exceed 30 lbs **After:** Weight of wheelchair must not exceed 30.5 lbs	Specificity changes that occur within requirements must be illustrated.	System Behavior / Characteristic
Measurability / Testing	[197]	**Before:** Beam weight must not exceed 50lbs **After**: Beam must have strength to weight ratio greater than 100	Changes in measurability of a requirement must be noted.	Behavior / Characteristic Condition
Application	[199]	**Before:** Metal foam must be used as a heat transferring medium **After**: Metal foam must be used as a structural member	Any changes with the application of a part must be noted.	Behavior / Characteristic Condition
Introduction of a new system	[198,199,200,201]	**Before:** Device must extract customer information **After**: Device must use secondary scanner to extract customer information	The introduction of a component or part within the requirement must be accounted for.	System
Replacement of a system	[198,199]	**Before:** Vehicle must be powered by internal combustion engine **After**: Vehicle must be powered by electric motor	Any replacement in component or part within requirement must be accounted for	System Condition

Table 11.1: Types of Micro Level Requirement Changes (continued)

Withdrawal of system	[198,199,200,201]	**Before:** Headrest must provide passenger head support **After**: System must provide passenger head support	Any withdrawal or removal in component or part within requirements must be addressed	System
Correction	[199]	**Before:** Structure must support 10 kpsi at 120 degrees C **After**: Structure must support 10 kpsi at 248 degrees F	Any corrections must be noted and commented.	System Behavior / Characteristic Condition
Updating	[202,199]	**Before:** Testing mechanism must be built no later than 5/30/08 **After**: Testing mechanism must be built no later than 4/30/08	Any updates that result in a changed requirement must be noted.	System Behavior / Characteristic Condition
Incorrect raw data interpretation	[199]	**Before:** Trunk must have automatic closing mechanism **After**: Trunk must be light enough to close with one hand	While the raw data of a requirement may not change, the requirement interpretation may change. This must be accounted for.	System Behavior / Characteristic Condition
Focus / Scope change / Source	[198,200]	**Before:** All devices must go through nondestructive testing before release **After**: At least 10% of all devices must go through nondestructive testing.	The scope of any requirement must be noted if changed.	System Behavior / Characteristic Condition

Table 11.1: Types of Micro Level Requirement Changes (continued)

Relationship	[198]	*Requirement: Vehicle must make use a manual or automatic transmission.* **Before:** If automatic transmission, vehicle must achieve a 0-60 acceleration within 7 seconds	If a relationship exists, where a requirement shows dependency on another requirement, this relationship must be monitored. If relationship is dispelled, it must be monitored.	System
		After: Vehicle must achieve a 0-60 acceleration within 7 seconds		
Moving	[198]	**Before:** Engine must be seated forward of center of gravity.	Any movement of a requirement's subject environment must be accounted for.	Condition
		After: Engine must be seated behind center of gravity		
Associated User	[203]	**Before:** Casting must be inspected for defects after cooling	Changes in the individuals associated with a requirement must be noted.	System Behavior / Characteristic Condition
		After: Casting must be inspected for defects after secondary operations		
Importance	[197]	**Before:** System must allow for multiple users. (Importance Rating: 2/5)	Changes in importance rating of each requirement must be noted.	
		After: System must allow for multiple users. (Importance Rating: 4/5)		

Table 11.2: Micro Level Requirement Change Analysis

Change	Subject	Modifier	Verb (modal)	Verb (transitive)	Condition (adjunct)	Object	Condition 2
Uncertainty	*System*	*Excitement*	*Must*	*Stay within*		*+/- 15% of operational frequency*	
	System	*Excitement*	*Must*	*Stay within*		*+/- 10% of operational frequency*	
Consistency (Type 1)	Headlamp		Must	Output	When in high beam	2000 lumen	
	Headlight		Must	Output	When in high beam	2000 lumen	
Consistency (Type 2)	Tweel		Must	Support		200 kgs	
	Tweel		Must	Support		450 lb-f	
Decomposition (Atomic) / Splitting (Type 1)	Drill		Must	Operate	Without failure	For 5 hours	
	Motor		Must	Rotate	Without failure	For 5 hours	
	Battery		Must	Supply	Without failure	For 5 hours	
	Transmission		Must	Transmit	Without failure	For 5 hours	

Table 11.2: Micro Level Requirement Change Analysis (continued)

Decomposition (Atomic) / Splitting (Type 2)	Power train		Must	Output	At least	350 horsepower	At consumption less than 20mgp
	Engine		Must	Output	At least	350 horsepower	
	Engine		Must	Consume	Less than	20mpg	
	Transmission		Must	Deliver		350 horsepower	
Merging	Microphone		Must	Input		Audible data	
	Speaker		Must	Output		Audible data	
	Phone		Must	Cancel		Noise Distortion	
	Phone		Must	Transfer	Without distortion	Audible data	
Specificity (Type 1)	Wheelchair	Weight	Must	Exceed	Not	30lbs	
	Wheelchair	Weight	Must	Exceed	Not	30.55 lbs	
Specificity (Type 2)	Wheelchair	Weight	Must	Exceed	Not	30lbs	
	Wheelchair Aluminum Subframe	Weight	Must	Exceed	Not	30lbs	
Measurability (Type 1)	Support Beam	Weight	Must	Exceed	Not	50lbs	
	Support Beam	Strength to weight ratio	Must	Exceed	Not	100	

Table 11.2: Micro Level Requirement Change Analysis (continued)

Measurability (Type 2)	Support Beam	Weight	Must	Exceed	Not	50lbs	
	Support Beam	Weight	Must	Less than		50lbs	
Application	Metal foam		Must	Transfer		Thermal energy	
	Metal foam		Must	Support		LEDs	
Introduction of new system	Wheel		Must	Withstand	During testing	Cryogenic conditions	
	Wheel and Test Frame		Must	Withstand	During Testing	Cryogenic conditions	
Replacement of new system (Type 1)	Device		Must	Extract		Customer information	
	Device		Must	Use	For extracting customer information	Secondary Scanner	
Replacement of new system (Type 2)	Strain Gauge		Must	Measure		Tire Deformation	
	Loading Cables		Must	Measure		Tire Deformation	
Withdrawal of system (Type 1)	Headrest and Side posts		Must	Provide	In case of crash	Body support	
	System		Must	Provide	In case of crash	Body support	

Table 11.2: Micro Level Requirement Change Analysis (continued)

Correction	Structure		Must	Support	At 120 degrees C	10kpsi	
	Support Member AF		Must	Support	At 250 degrees C	10kpsi	Without Bending
Updating	Testing Mechanism		Must	Built	No later	5/30/08	
	Merry-go-round		Must	Designed	No later	4/30/08	
Incorrect data interpretation	Trunk		Must	Have		Automatic closing mechanism	
	Trunk		Must	Be	Light enough	Close with one hand	
Focus (Type 1)	LED	Maximum temperature	Must	Exceed	Not	90 degrees C	
	LED Substrate	temperature	Must	Exceed	Not	90 degrees C	
Focus (Type 2)	Vehicle		Must	Produce		65 horsepower	
	Engine		Must	Produce		90 horsepower	
Focus (Type 3)	All devices		Must	Go through		Nondestructive testing	
	All devices	10% of	Must	Go through		Nondestructive testing	
Relationship (Type 1)	*Vehicle*	*Automatic transmission*	*Must*	*Achieve*	*0-60 acceleration*	*Within 7 second*	
	Vehicle		*Must*	*Achieve*	*0-60 acceleration*	*Within 7 second*	

Table 11.2: Micro Level Requirement Change Analysis (continued)

Relationship (Type 2)	Engine		Must	Seat	Forward	Center of Gravity	
	Engine		Must	Seat	Behind	Center of Gravity	
Associated User	Casting		Must	Inspected	After cooling	For defects	
	Casting		Must	Inspected	After secondary operation	For defects	
Importance (Type 1)	Headlamp reflector	Refraction	Must	Not exceed		10%	
	Headlamp reflector	Refraction	Should	Not exceed		10%	

A.2 - Localized Changes

In summarizing the results of the review, Table 11.3 details each type of change identified in Table 11.1 and segments its syntactical representation so the exact change may be identified. This is used to identify the localized change of requirements when a change occurs. Additionally, some of the changes identified were further separated into different forms of that change. For instance, a focus change may occur in multiple manners, a modifier or subject change.

The results of this review could contribute to the future work of this research by developing correlation between localized and global changes. The localized changes assisted in understanding how requirements change and if this change could be represented. The ability to represent requirement change through its syntax lends itself for evaluation of localized changes at the global level. For instance, if a subject is changed during a Specificity change, it could be hypothesized that all other requirements pertaining to this subject are vulnerable for change propagation. As a result, global change propagation is thoroughly investigated to identify if (1) change propagation can be predicted and (2) if this can be performed through the syntax of a requirement, such as its POS. This research identifies that this is in fact possible and an accurate way to relate requirements, however much analysis is performed to identify how requirements may be related through their POS and if a manual element (such as manually selected keywords) is of benefit. Though not in the scope of this research dissertation, there may be a causality that is realized between the change type and the type of global change experience. If this causality exists, localized changed may be used to further enhance the global requirement change propagation predictive abilities.

Table 11.3: Syntactical Change of Micro level Changes

Change	Specific Syntactical - Change
Focus (Type 1)	Modifier
Focus (Type 2)	Subject
Uncertainty	Object
Consistency	Subject
Decomposition	Subject
Splitting	Subject, T-verb, Object, Condition
Merging	T-verb
Specificity	Object
Measurability	Modifier, Object
Application	Object
Introduction of new system	T-verb, Object, Condition
Replacement of system (Type 1)	Object
Replacement of system (Type 2)	Subject
Withdrawal of system	Subject, Condition
Correction	Condition
Updating	T-verb, Object
Incorrect data interpretation	T-verb, Object, Condition
Scope	Modifier
Relationship	Modifier
Moving	Condition
Associated User	Condition

A.3 - Relating Localized to Global Changes

By first understanding the changes at the localized level, this research will assist in determining what specifically occurs to a requirement at the syntactical level during change. The end goal is to develop a means for correlating these micro level changes to macro level changes. Understanding the macro level changes will assist in determining requirements that may change due to propagation. The micro level changes tie in by further assisting in determining the type of change that may be encountered. For instance, if a requirement experience a micro level change in the condition it must satisfy, from supporting 500lbs to 750lbs, this requirement change will be propagated at the macro scale to identify potential requirement change. After identifying potential requirement change and further narrowing that selection down to a manageable amount through use of weightings, the micro level changes are once again related to the requirement. The condition on a potential requirement may require changing to support the condition change on the initially changed requirement.

.

APPENDIX B
COMBINATION CODE

B.1 - Morkos DSM.m

```
clc
clear all
%%
% %Just in Case you're using the bigbox - Optimize Parallel Configuration
% cpus = str2num(getenv('NUMBER_OF_PROCESSORS'));
% if MATLABpool('size') ~= (cpus-1)
%     MATLABpool(cpus-1)
% end

%%
progressbar %opens progress bar
%Loading Requirements
[~,Req]= xlsread('PierburgReq.xlsx');
[~,KW] = xlsread('Keywords.xlsx');
load('Pierburg_ECN.mat')

%Preparing the DSM
r = length(Req); %number of requirements
DSM = zeros(r);
kw = length(KW(1,:)); % number of Keywords

%Running the needed functions
[Nouns,Verbs,numnouns,numverbs,wordlength] = POS(r); %Runs the Part of Speech
function
[ComMat,relators]=ComMat(kw,numnouns,numverbs); %Runs the Combination Matrix
function
c= length(ComMat);
Results = zeros(c,44);

%%
%This is the large loop covering all the possible Combination Matrix
%Series.  This will create a Degrees of Freedom DSM, Binary DSM,
%Relationship Order DSM, and Flowpath DSM (Optional)
for a=1:length(c) %Typically a=1:c
z=0;
for i=1:r
   R=Req{i,1}; %i is the requirement number (single column requirement list)
   %For requirement R, check against every other requirement j
   for j=1:r %typically j=1:r
   z=0;
```

```
  %NOUNS
  for nn = 1:length(Nouns{j})
     if ComMat(a,nn)== 1
     x=findstr(R,Nouns{j}{nn}); %requirement number i tested for KW in j,k
        if x>=1;
        x=1;
        else
        x=0;
        end
     z=z+x;
     x=0;
     end
  end

  %VERBS
  for vv = 1:length(Verbs{j})
     if ComMat(a,vv+numnouns)== 1
     x=findstr(R,Verbs{j}{vv}); %requirement number i tested for KW in j,k
        if x>=1;
        x=1;
        else
        x=0;
        end
     z=z+x;
     x=0;
     end
   end

  %KEYWORDS
  for kk = 1:length(KW(j,:))
     if ComMat(a,kk+numnouns+numverbs)== 1
     x=findstr(R,KW{j,kk}); %requirement number i tested for KW in j,k
        if x>=1;
        x=1;
        else
        x=0;
        end
     z=z+x;
     x=0;
     end
     DSM(j,i)=z;
  end

end
```

```
end


%%
%Creates the Binary DSM. Identifies if a relationship exists.
BiDSM = DSM;
BiDSM(BiDSM>0)=1;

% Determines which Order Relationship Exists in (Shortest Path)
RelOrderDSM=iterative_spath(BiDSM); %you can try this one.
RelOrderDSM(find(eye(r)))=1;
% OrderDSM = all_shortest_paths(sparse(BiDSM));
PierburgSet{a} = RelOrderDSM;

%Finding the Postives Ratios
[results] = changePerf2(RelOrderDSM,Pierburg_ECN);
Results(a,45) = results.true_positive;
Results(a,46) = results.false_positive;
Results(a,47) = Results(a,45)/Results(a,46);

% Determines number of possible relationships to the nth level
% MultiOrder_Rel=par_max_flow(sparse(BiDSM),r);

%Results for Propgatoin of ECN01 to ECN07 (Req 2.1.14)
   %Degrees of Freedom
   Results(a,1) = DSM(76,17);
   Results(a,2) = DSM(22,17);
   Results(a,3) = DSM(2,17);
   Results(a,4) = DSM(138,17);
   Results(a,5) = DSM(176,17);

   %Relationship Order
   Results(a,19) = RelOrderDSM(76,17);
   Results(a,20) = RelOrderDSM(22,17);
   Results(a,21) = RelOrderDSM(2,17);
   Results(a,22) = RelOrderDSM(138,17);
   Results(a,23) = RelOrderDSM(176,17);

%Results for Propgatoin of ECN01 to ECN07 (Req 2.2.6)
   %Degrees of Freedom
   Results(a,6) = DSM(76,25);
   Results(a,7) = DSM(22,25);
   Results(a,8) = DSM(2,25);
   Results(a,9) = DSM(138,25);
```

```
    Results(a,10) = DSM(176,25);
    Results(a,11) = DSM(17,25);

    %Relationship Order
    Results(a,24) = RelOrderDSM(76,25);
    Results(a,25) = RelOrderDSM(22,25);
    Results(a,26) = RelOrderDSM(2,25);
    Results(a,27) = RelOrderDSM(138,25);
    Results(a,28) = RelOrderDSM(176,25);
    Results(a,29) = RelOrderDSM(17,25);

%Results for Propgatoin of ECN01 and ECN07 to ECN11(Req 2.5.13.3.3)
    %Degrees of Freedom
    Results(a,12) = DSM(76,85);
    Results(a,13) = DSM(22,85);
    Results(a,14) = DSM(2,85);
    Results(a,15) = DSM(138,85);
    Results(a,16) = DSM(176,85);
    Results(a,17) = DSM(17,85);
    Results(a,18) = DSM(25,85);

    %Relationship Order
    Results(a,30) = RelOrderDSM(76,85);
    Results(a,31) = RelOrderDSM(22,85);
    Results(a,32) = RelOrderDSM(2,85);
    Results(a,33) = RelOrderDSM(138,85);
    Results(a,34) = RelOrderDSM(176,85);
    Results(a,35) = RelOrderDSM(17,85);
    Results(a,36) = RelOrderDSM(25,85);

%Total Relationships
    TotalRelationships = numel(BiDSM(BiDSM==1));
    Results(a,37) = TotalRelationships;

    Results(a,38) = numel(DSM(DSM(76,:)>0));
    Results(a,39) = numel(DSM(DSM(22,:)>0));
    Results(a,40) = numel(DSM(DSM(2,:)>0));
    Results(a,41) = numel(DSM(DSM(138,:)>0));
    Results(a,42) = numel(DSM(DSM(176,:)>0));
    Results(a,43) = numel(DSM(DSM(17,:)>0));
    Results(a,44) = numel(DSM(DSM(25,:)>0));

%Stores it in a .mat file
save('Results','Results')
```

```
%For the stop bar
stopBar= progressbar(a/c);
if (stopBar) break; end

end
```

B.2 - POS.m function

```
function[Nouns,Verbs,numnouns,numverbs,wordlength]=POS(r)
[~,Words] = xlsread('WordsPOS2.xlsx');
numnouns = 5; %maximum of nouns I want to evaluate after cleaning
numverbs = 5; %maximum of nouns I want to evaluate after cleaning
Nouns=cell(r,1); %typically this is set to r since it will be pulled up in the
Morkos_DSM.m file
Verbs=cell(r,1); %typically this is set to r since it will be pulled up in the
Morkos_DSM.m file
wordlength = length(Words);

%Develop a list of nouns
for a=1:r %typically 1:number of requirements.  col 1 is word, col 2 is tag
   k=1;
   for i=1:length(Words(:,1))
   x = strfind(Words{i,a*2},'nn'); %Because WordsPOS2 is all in lower case, we have to
use 'nn'
       if x>0
       Nouns{a}{k} = Words{i,a*2-1};
       k = k+1;
       else
       end
   end
end

%Clean the nouns for duplicates and only show me a max of numnouns
Nouns = cellfun(@(c) unique(c),Nouns,'UniformOutput',false);
Nouns = cellfun(@(c) c(1:min(length(c),numnouns)),Nouns,'UniformOutput',false);

%Develop a list of verbs
for a=1:r %typically 1:number of requirements.  col 1 is word, col 2 is tag
   k=1;
   for i=1:length(Words(:,1))
   x = strfind(Words{i,a*2},'vb'); %Because WordsPOS2 is all in lower case, we have to
use 'vb'
       if x>0
       Verbs{a}{k} = Words{i,a*2-1};
       k = k+1;
       else
       end
   end
end
```

```
%Clean the verbs for duplicates and only show me a max of numvers
Verbs = cellfun(@(c) unique(c),Verbs,'UniformOutput',false);
Verbs = cellfun(@(c) c(1:min(length(c),numverbs)),Verbs,'UniformOutput',false);
```

APPENDIX C
OPTIMAL COMBINATION EVALUATION CODE

C.1 - Results.V2.m

```
clear all
clc

%%
% Just in Case you're using the bigbox - Optimize Parallel Configuration
cpus = str2num(getenv('NUMBER_OF_PROCESSORS'));
if MATLABpool('size') ~= (cpus-1)
   MATLABpool(cpus-1)
end
%%
%Threshold and number of iteration passes values
%these may not be needed however.  Depending on the filtration
order = 5; %cut off relationship order
%%
%opens progress bar
progressbar
%Loading the results file
Pierburg = xlsread('Result.Pierburg.xlsx');
Toho = xlsread('Result.Toho.xlsx');
EVRAZ = xlsread('Result.EVRAZ.xlsx');

%% FIRST FILTER
%Runs the filter1 function
%functions remove all false negative combinations
[Toho, Pierburg] = filter1(Toho,Pierburg ,order);

%% SECOND FILTER
%Finding the difference between TP and FP
%Removing all differences greater than set values
[Toho, Pierburg, EVRAZ] = filter2(Toho, Pierburg , EVRAZ);

%% THIRD FILTER
%Runs the filter3 function
%Removing all TP/FP ratios above a set value
[Toho, Pierburg, EVRAZ] = filter3(Toho, Pierburg, EVRAZ);

%%
%Sorting and Plotting
Toho=sortrows(Toho,53);
Pierburg=sortrows(Pierburg,48);
EVRAZ=sortrows(EVRAZ,5);
```

```
        % %Plot After Sorting
        plot(Toho(:,53),'DisplayName','Toho(:,53)','YDataSource','Toho(:,53)');figure(gcf
)
        figure
    plot(Pierburg(:,48),'DisplayName','Pierburg(:,48)','YDataSource','Pierburg(:,48)');figu
    re(gcf)
        figure
    plot(EVRAZ(:,5),'DisplayName',EVRAZ(:,5)','YDataSource','EVRAZ(:,5)');figure(gc
    f)
        figure

        %%
        %MATCHING COMBINATIONS
        Matches1 = []
        k=1
        for i=1:length(Pierburg)
            if any(Toho(:,1)==Pierburg(i,1)) == 1
                Matches1(k,1)=Pierburg(i,1)
                k=k+1
            else
            end
        end

        Matches2 = []
        k=1
        for i=1:length(EVRAZ)
            if any(Pierburg(:,1)==EVRAZ(i,1)) == 1
                Matches2(k,1)= EVRAZ(i,1)
                k=k+1
            else
            end
        end

        MatchesFinal = []
        k=1
        for i=1:length(Matches2)
            if any(Matches1(:,1)==Matches2(i,1)) == 1
                MatchesFinal(k,1)= Matches2(i,1)
                k=k+1
            else
            end
        end
```

C.2 - filter1.m function

```
function[Toho, Pierburg]=filter1(Toho,Pierburg, order)

%First Filter - Remove all which don't have Order Relations

%Toho
k=0;
for i=1:length(Toho)
   z=0;
   for j=23:43 %checking to see if there is a Relation Order between the
requirements
      if Toho(i-k,j)== 0 | Toho(i-k,j) >= order; %first column is the row heading
      z=z+1;
      else
      end
   end
   if z>0
   Toho(i-k,:)=[];
   k=k+1;
   end
end

%Pierburg
k=0;
for i=1:length(Pierburg)
   z=0;
   for j=20:37 %checking to see if there is a Relation Order between the
requirements
      if Pierburg(i-k,j)== 0; %first column is the row heading
      z=z+1;
      else
      end
   end
   if z>0
   Pierburg(i-k,:)=[];
   k=k+1;
   end
end
```

C.3 - filter2.m function

```
        function[Toho, Pierburg, EVRAZ]=filter2(Toho, Pierburg, EVRAZ)

    %FILTER 3 DEALS WITH THE DIFFERENCES BETWEEN TRUE POSITIVE
    AND FALSE
        %POSITIVE

        %Toho
        %Identifying the Difference between True Positive and False Positive
        for i=1:length(Toho);
        Toho(i,54)=Toho(i,51)-Toho(i,52); %True Positive - False Positive
        end

        k=0;
        for i=1:length(Toho);
           if Toho(i-k,54)<.1; %if TP - FP is less than 0
           z=1;
           else
           z=0;
           end
           if z>0
           Toho(i-k,:)=[];
           k=k+1;
           else
           end
        end

        %Pierburg
        %Identifying the Difference between True Positive and False Positive
        for i=1:length(Pierburg);
        Pierburg(i,49)=Pierburg(i,46)-Pierburg(i,47); %True Positive - False Positive
        end

        k=0;
        for i=1:length(Pierburg);
           if Pierburg(i-k,49)<.15; %if TP - FP is less than 0
           z=1;
           else
           z=0;
           end
           if z>0
           Pierburg(i-k,:)=[];
           k=k+1;
    else
    end
end

%EVRAZ
%Identifying the Difference between True Positive and False Positive
for i=1:length(EVRAZ);
EVRAZ(i,6)=EVRAZ(i,3)-EVRAZ(i,4); %True Positive - False Positive
end

k=0;
for i=1:length(EVRAZ);
    if EVRAZ(i-k,6)<0.17; %if TP - FP is less than 0
    z=1;
    else
    z=0;
    end
    if z>0
    EVRAZ(i-k,:)=[];
    k=k+1;
    else
    end
end

end
```

C.4 - filter3.m function

```
    function[Toho, Pierburg, EVRAZ]=filter3(Toho,Pierburg, EVRAZ)

%FILTER 3 DEALS WITH THE RATIO OF TRUE POSITIVE TO FALSE
POSITIVE

    %Toho
    k=0;
    for i=1:length(Toho);
       if Toho(i-k,53)<1.1; %if TP/FP is less than 1
       z=1;
       else
       z=0;
       end
       if z>0
       Toho(i-k,:)=[];
       k=k+1;
       else
       end
    end

    %Pierburg
    k=0;
    for i=1:length(Pierburg);
       if Pierburg(i-k,48)<1.15; %if TP/FP is less than 1
       z=1;
       else
       z=0;
       end
       if z>0
       Pierburg(i-k,:)=[];
       k=k+1;
       else
       end
    end

    %EVRAZ
    k=0;
    for i=1:length(EVRAZ);
       if EVRAZ(i-k,5)<1.3; %if TP/FP is less than 1
       z=1;
       else
       z=0;
    end
    if z>0
    EVRAZ(i-k,:)=[];
    k=k+1;
    else
    end
end

end
```

APPENDIX D
PROPAGATION ANALYSIS & PREDICTION CODE

D.1 - Weightings.m

```
clc
clear all

FO = 9  %First Order Scoring
SO = 3   %Second Order Scoring
TO = 0   %Third Order Scoring

[TohoAvg, PierburgAvg, EVRAZAvg] = Overlay(FO, SO, TO);
[TReview, PReview, EReview] = Depth(TohoAvg, PierburgAvg, EVRAZAvg)

TDepth = max(TReview(:,1))/159
PDepth = max(max(PReview))/214
EDepth                                                                          =
max([EReview(2,1),EReview(3,1),EReview(6,2),EReview(7,5),EReview(8,1),EReview(
8,2),EReview(9,1)])/187
MaxDepth = max([TDepth,PDepth,EDepth])

% %To Instance04
% for i = 1:length(EVRAZAvg);
% EVRAZAvgEC(1,i) = EVRAZAvg(70,i);
% EVRAZAvgEC(2,i) = EVRAZAvg(39,i);
% EVRAZAvgEC(3,i) = EVRAZAvg(75,i);
% EVRAZAvgEC(4,i) = sum(EVRAZAvgEC(:,i));
% end
% EScores = EVRAZAvgEC(4,:);
% EReview(2,1) = numel(EScores(EScores>=EScores(70)));
% EReview(3,1) = numel(EScores(EScores>=EScores(70)));
% EReview(4,1) = numel(EScores(EScores>=EScores(88)));
% EReview(4,2) = numel(EScores(EScores>=EScores(89)));
```

D.2 - Overlay.m function

```
function[TohoAvg, PierburgAvg, EVRAZAvg] = Overlay(FO, SO, TO)

load('PierburgSet.mat')
load('TohoSet.mat')
load('EVRAZSet.mat')

%% TOHO
for a= 1:length(TohoSet);
TohoSet{a}(TohoSet{a} >= 4) = 0;
TohoSet{a}(TohoSet{a} == 3) = TO;
TohoSet{a}(TohoSet{a} == 2) = SO;
TohoSet{a}(TohoSet{a} == 1) = FO;
end

%Overlay and Sum all the Rel Order Scores DSMs from each combination
TohoAvg = [];
for i=1:length(TohoSet{1});
   for j=1:length(TohoSet{1});
      A = [TohoSet{1}(i,j) TohoSet{2}(i,j) TohoSet{3}(i,j)];
      A(A==0)=[];
      TohoAvg(i,j)=sum(A);
   end
end

%% PIERBURG
for a= 1:length(PierburgSet);
PierburgSet{a}(PierburgSet{a} >= 4) = 0;
PierburgSet{a}(PierburgSet{a} == 3) = TO;
PierburgSet{a}(PierburgSet{a} == 2) = SO;
PierburgSet{a}(PierburgSet{a} == 1) = FO;
end

%Overlay and Sum all the Rel Order DSMs from each combination
PierburgAvg = [];
for i=1:length(PierburgSet{1});
   for j=1:length(PierburgSet{1});
      A = [PierburgSet{1}(i,j) PierburgSet{2}(i,j) PierburgSet{3}(i,j)];
      A(A==0)=[];
      PierburgAvg(i,j)=sum(A);
   end
end
```

```
%% EVRAZ
for a= 1:length(EVRAZSet);
EVRAZSet{a}(EVRAZSet{a} >= 4) = 0;
EVRAZSet{a}(EVRAZSet{a} == 3) = TO;
EVRAZSet{a}(EVRAZSet{a} == 2) = SO;
EVRAZSet{a}(EVRAZSet{a} == 1) = FO;
end

%Overlay and Sum all the Rel Order DSMs from each combination
EVRAZAvg = [];
for i=1:length(EVRAZSet{1});
    for j=1:length(EVRAZSet{1});
        A = [EVRAZSet{1}(i,j) EVRAZSet{2}(i,j) EVRAZSet{3}(i,j)];
        A(A==0)=[];
        EVRAZAvg(i,j)=sum(A);
    end
end
end
```

D.3 - Depth.m function

```
function [TReview, PReview, EReview] = Depth(TohoAvg, PierburgAvg,
EVRAZAvg)
    %% TOHO
    %ECN01 to ECN03 Change in 91 caused 110 and 111 to change
    for i = 1:length(TohoAvg);
    TohoAvgEC13(1,i) = TohoAvg(91,i);
    end
    TReview(1,1) = numel(TohoAvgEC13(TohoAvgEC13>=TohoAvgEC13(110)));
    TReview(1,2) = numel(TohoAvgEC13(TohoAvgEC13>=TohoAvgEC13(111)));
    TReview(1,3) = numel(TohoAvgEC13(TohoAvgEC13>=TohoAvgEC13(108)));

    %ECN03 to ECN04 Change 110 and 111 cause 113 to change
    for i = 1:length(TohoAvg);
    TohoAvgEC34(1,i) = TohoAvg(110,i);
    TohoAvgEC34(2,i) = TohoAvg(111,i);
    TohoAvgEC34(3,i) = TohoAvg(108,i);
TohoAvgEC34(4,i) = sqrt((TohoAvgEC34(1,i)^2 + TohoAvgEC34(2,i)^2 +
TohoAvgEC34(3,i)^2)/3);
    end
    TScores = TohoAvgEC34(3,:);
    TReview(2,1) = numel(TScores(TScores>=TScores(113)));

    %% PIERBURG
    %ECN01 to ECN07 Change in 2,17,76,138 cause 25 to change
    for i = 1:length(PierburgAvg);
    PierburgAvgEC(1,i) = PierburgAvg(2,i);
    PierburgAvgEC(2,i) = PierburgAvg(17,i);
    PierburgAvgEC(3,i) = PierburgAvg(76,i);
    PierburgAvgEC(4,i) = PierburgAvg(138,i);
PierburgAvgEC(5,i) =
sqrt((PierburgAvgEC(1,i)^2+PierburgAvgEC(2,i)^2+PierburgAvgEC(3,i)^2+Pierbur
gAvgEC(4,i)^2)/4);
    end

    PScores = PierburgAvgEC(5,:);
    PReview(1,1) = numel(PScores(PScores>=PScores(17)));
    PReview(1,2) = numel(PScores(PScores>=PScores(25)));

    %ECN01 to ECN07 Change in 2,17,76,138 cause 25 to change
    for i = 1:length(PierburgAvg);
    PierburgAvgEC(5,i) = PierburgAvg(17,i);
    PierburgAvgEC(6,i) = PierburgAvg(25,i);
    PierburgAvgEC(7,i) = sqrt((PierburgAvgEC(1,i)^2+PierburgAvgEC(2,i)^2+...
                         PierburgAvgEC(3,i)^2+PierburgAvgEC(4,i)^2+...
                         PierburgAvgEC(5,i)^2+PierburgAvgEC(6,i)^2)/6);
    end

    PScores = PierburgAvgEC(7,:);
    PReview(2,1) = numel(PScores(PScores>=PScores(97)));

    %% EVRAZ
    %% EVRAZ
    %To Instance04
    for i = 1:length(EVRAZAvg);
    EVRAZAvgEC(1,i) = EVRAZAvg(70,i);
    EVRAZAvgEC(2,i) = EVRAZAvg(39,i);
    EVRAZAvgEC(3,i) = EVRAZAvg(75,i);
EVRAZAvgEC(4,i)                                                      =
sqrt((EVRAZAvgEC(1,i)^2+EVRAZAvgEC(2,i)^2+EVRAZAvgEC(3,i)^2)/3);
    end
    EScores = EVRAZAvgEC(4,:);
    EReview(2,1) = numel(EScores(EScores>=EScores(70)));
    EReview(3,1) = numel(EScores(EScores>=EScores(70)));
    EReview(4,1) = numel(EScores(EScores>=EScores(88)));
    EReview(4,2) = numel(EScores(EScores>=EScores(89)));

    %To Instance05
    for i = 1:length(EVRAZAvg);
    EVRAZAvgEC(4,i) = EVRAZAvg(88,i);
    EVRAZAvgEC(5,i) = EVRAZAvg(89,i);
EVRAZAvgEC(6,i)                                                      =
sqrt((3*EVRAZAvgEC(1,i)^2+EVRAZAvgEC(2,i)^2+EVRAZAvgEC(3,i)^2+...
                        EVRAZAvgEC(4,i)^2+EVRAZAvgEC(5,i)^2)/7);
    end
    EScores = EVRAZAvgEC(6,:);
    EReview(5,1) = numel(EScores(EScores>=EScores(34)));

    %To Instance06
    for i = 1:length(EVRAZAvg);
    EVRAZAvgEC(6,i) = EVRAZAvg(34,i);
EVRAZAvgEC(7,i)                                                      =
sqrt((3*EVRAZAvgEC(1,i)^2+EVRAZAvgEC(2,i)^2+EVRAZAvgEC(3,i)^2+...

EVRAZAvgEC(4,i)^2+EVRAZAvgEC(5,i)^2+EVRAZAvgEC(6,i)^2)/8);
    end
    EScores = EVRAZAvgEC(7,:);
    EReview(6,1) = numel(EScores(EScores>=EScores(22)));
    EReview(6,2) = numel(EScores(EScores>=EScores(39)));

    %To Instance07
    for i = 1:length(EVRAZAvg);
    EVRAZAvgEC(7,i) = EVRAZAvg(22,i);
EVRAZAvgEC(8,i)                                                                    =
sqrt((3*EVRAZAvgEC(1,i)^2+EVRAZAvgEC(2,i)^2+EVRAZAvgEC(3,i)^2+...

EVRAZAvgEC(4,i)^2+EVRAZAvgEC(5,i)^2+EVRAZAvgEC(6,i)^2)+...
                    EVRAZAvgEC(7,i)^2/9);
    end
    EScores = EVRAZAvgEC(8,:);
    EReview(7,1) = numel(EScores(EScores>=EScores(6)));
    EReview(7,2) = numel(EScores(EScores>=EScores(10)));
    EReview(7,3) = numel(EScores(EScores>=EScores(173)));
    EReview(7,4) = numel(EScores(EScores>=EScores(119)));
    EReview(7,5) = numel(EScores(EScores>=EScores(133)));


    %To Instance08
    for i = 1:length(EVRAZAvg);
    EVRAZAvgEC(8,i) = EVRAZAvg(6,i);
    EVRAZAvgEC(9,i) = EVRAZAvg(10,i);
    EVRAZAvgEC(10,i) = EVRAZAvg(173,i);
    EVRAZAvgEC(11,i) = EVRAZAvg(119,i);
    EVRAZAvgEC(12,i) = EVRAZAvg(133,i);
EVRAZAvgEC(13,i)                                                                   =
sqrt((3*EVRAZAvgEC(1,i)^2+EVRAZAvgEC(2,i)^2+EVRAZAvgEC(3,i)^2+...

EVRAZAvgEC(4,i)^2+EVRAZAvgEC(5,i)^2+EVRAZAvgEC(6,i)^2+...

EVRAZAvgEC(7,i)^2+EVRAZAvgEC(8,i)^2+EVRAZAvgEC(9,i)^2+...

EVRAZAvgEC(10,i)^2+EVRAZAvgEC(11,i)^2+EVRAZAvgEC(12,i)^2)/14);
    end
    EScores = EVRAZAvgEC(13,:);
    EReview(8,1) = numel(EScores(EScores>=EScores(89)));
    EReview(8,2) = numel(EScores(EScores>=EScores(98)));

    %To Instance09 and Instance10
    for i = 1:length(EVRAZAvg);
    EVRAZAvgEC(13,i) = EVRAZAvg(98,i);
EVRAZAvgEC(14,i)                                                           =
sqrt((3*EVRAZAvgEC(1,i)^2+EVRAZAvgEC(2,i)^2+EVRAZAvgEC(3,i)^2+...

EVRAZAvgEC(4,i)^2+EVRAZAvgEC(5,i)^2+EVRAZAvgEC(6,i)^2+...

EVRAZAvgEC(7,i)^2+EVRAZAvgEC(8,i)^2+EVRAZAvgEC(9,i)^2+...

EVRAZAvgEC(10,i)^2+EVRAZAvgEC(11,i)^2+EVRAZAvgEC(12,i)^2+...
            EVRAZAvgEC(13,i)^2)/15);
  end
  EScores = EVRAZAvgEC(14,:);
  EReview(9,1) = numel(EScores(EScores>=EScores(70)));
  EReview(10,1) = numel(EScores(EScores>=EScores(131)));

  for i = 1:length(EVRAZAvg);
  EVRAZAvgEC(14,i) = EVRAZAvg(131,i);
EVRAZAvgEC(15,i)                                                           =
sqrt((3*EVRAZAvgEC(1,i)^2+EVRAZAvgEC(2,i)^2+EVRAZAvgEC(3,i)^2+...

EVRAZAvgEC(4,i)^2+EVRAZAvgEC(5,i)^2+EVRAZAvgEC(6,i)^2+...

EVRAZAvgEC(7,i)^2+EVRAZAvgEC(8,i)^2+EVRAZAvgEC(9,i)^2+...

EVRAZAvgEC(10,i)^2+EVRAZAvgEC(11,i)^2+EVRAZAvgEC(12,i)^2+...
            EVRAZAvgEC(13,i)^2+EVRAZAvg(14,i)^2)/16);
  end
  EScores = EVRAZAvgEC(15,:);
  EReview(11,1) = numel(EScores(EScores>=EScores(127)));
```

APPENDIX E
FINAL PROGRAM

```
clc
clear all
%%
%Just in Case you're using the bigbox - Optimize Parallel Configuration
cpus = str2num(getenv('NUMBER_OF_PROCESSORS'));
if MATLABpool('size') ~= (cpus-1)
   MATLABpool(cpus-1)
end

%Loading Requirements
[~,Req]= xlsread('PierburgReq.xlsx');
[~,KW] = xlsread('Keywords.xlsx');
load('Pierburg_ECN.mat')

%Preparing the DSM
r = length(Req); %number of requirements
DSM = zeros(r);
kw = length(KW(1,:)); % number of Keywords

%Running the needed functions
[Nouns,Verbs,numnouns,numverbs,wordlength] = POS(r); %Runs the Part of Speech function
[ComMat,relators]=ComMat(kw,numnouns,numverbs); %Runs the Combination Matrix function
c= length(ComMat);
Results = zeros(c,44);

%%
%This is the large loop covering all the possible Combination Matrix
%Series.  This will create a Degrees of Freedom DSM, Binary DSM,
%Relationship Order DSM, and Flowpath DSM (Optional)
Set = [5,8197,24578]
for a=1:length(Set) %Typically a=1:c
z=0;
for i=1:r
   R=Req{i,1}; %i is the requirement number (single column requirement list)
   %For requirement R, check against every other requirement j
   for j=1:r %typically j=1:r
   z=0;

      %NOUNS
      for nn = 1:length(Nouns{j})
         if ComMat(Set(a),nn)== 1
         x=findstr(R,Nouns{j}{nn}); %requirement number i tested for KW in j,k
```

```
            if x>=1;
            x=1;
            else
            x=0;
            end
        z=z+x;
        x=0;
        end
    end

    %VERBS
    for vv = 1:length(Verbs{j})
        if ComMat(Set(a),vv+numnouns)== 1
        x=findstr(R,Verbs{j}{vv}); %requirement number i tested for KW in j,k
            if x>=1;
            x=1;
            else
            x=0;
            end
        z=z+x;
        x=0;
        end
     end

    %KEYWORDS
    for kk = 1:length(KW(j,:))
        if ComMat(Set(a),kk+numnouns+numverbs)== 1
        x=findstr(R,KW{j,kk}); %requirement number i tested for KW in j,k
            if x>=1;
            x=1;
            else
            x=0;
            end
        z=z+x;
        x=0;
        end
        DSM(j,i)=z;
    end

   end
end


%%
```

```
%Creates the Binary DSM. Identifies if a relationship exists.
BiDSM = DSM;
BiDSM(BiDSM>0)=1;

% Determines which Order Relationship Exists in (Shortest Path)
RelOrderDSM=iterative_spath(BiDSM); %you can try this one.
RelOrderDSM(find(eye(r)))=1;

% OrderDSM = all_shortest_paths(sparse(BiDSM));
PierburgSet{a} = RelOrderDSM;

%Finding the Postives Ratios
[results] = changePerf2(RelOrderDSM,Pierburg_ECN);
Results(a,45) = results.true_positive;
Results(a,46) = results.false_positive;
Results(a,47) = Results(a,45)/Results(a,46);

% Determines number of possible relationships to the nth level
MultiOrder_Rel=par_max_flow(sparse(BiDSM),r);
PierburgMultiSet{a} = MultiOrder_Rel


%Stores it in a .mat file
save('Results', 'Results')


end
```

APPENDIX F
TOHO MANUAL – SUBJECT DSM

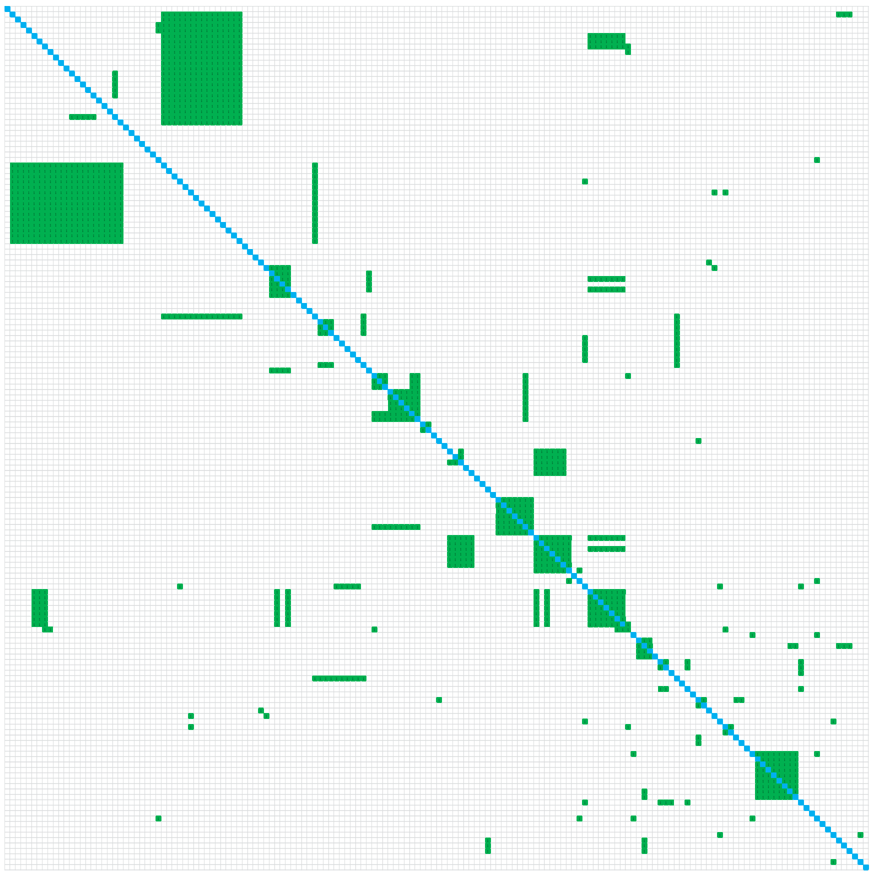


Figure 11.2: Toho Subject DSM – Base DSM

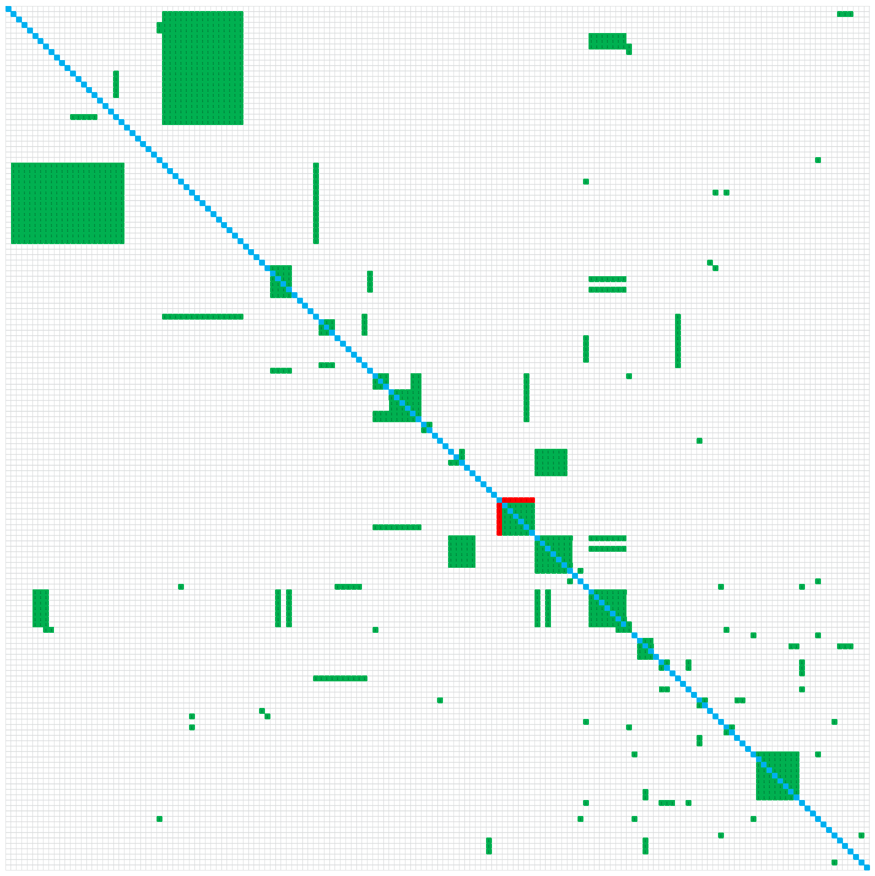


Figure 11.3: Toho Subject DSM – Requirement 9.2.3.1 First Order

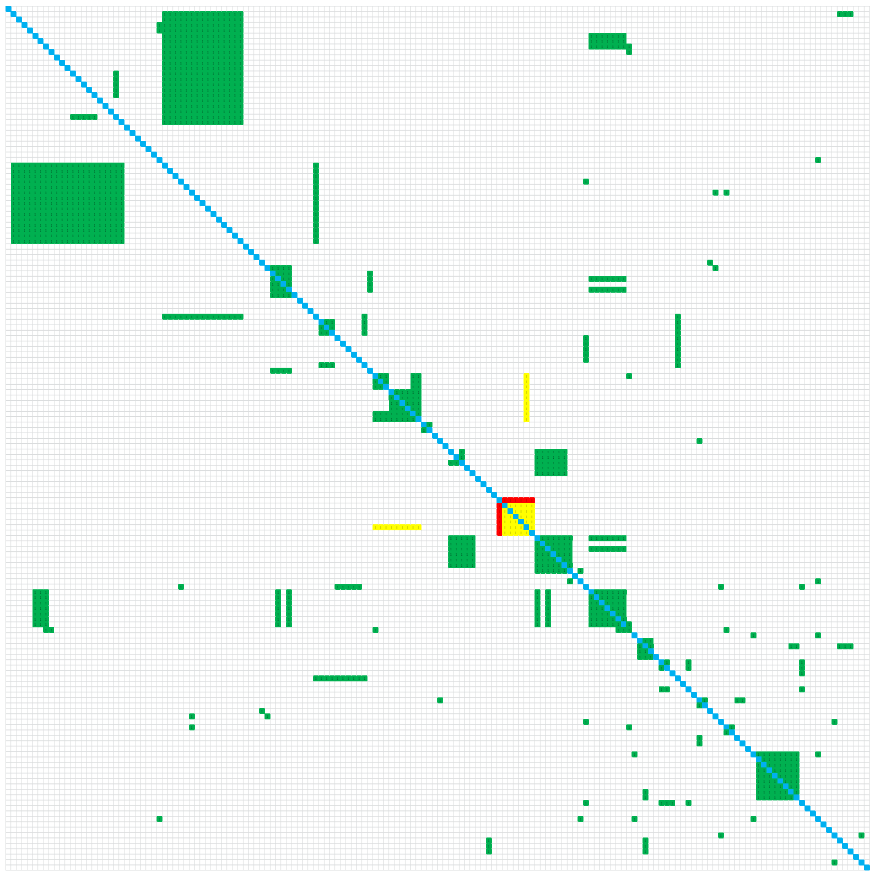


Figure 11.4: Toho Subject DSM – Requirement 9.2.3.1 Second Order

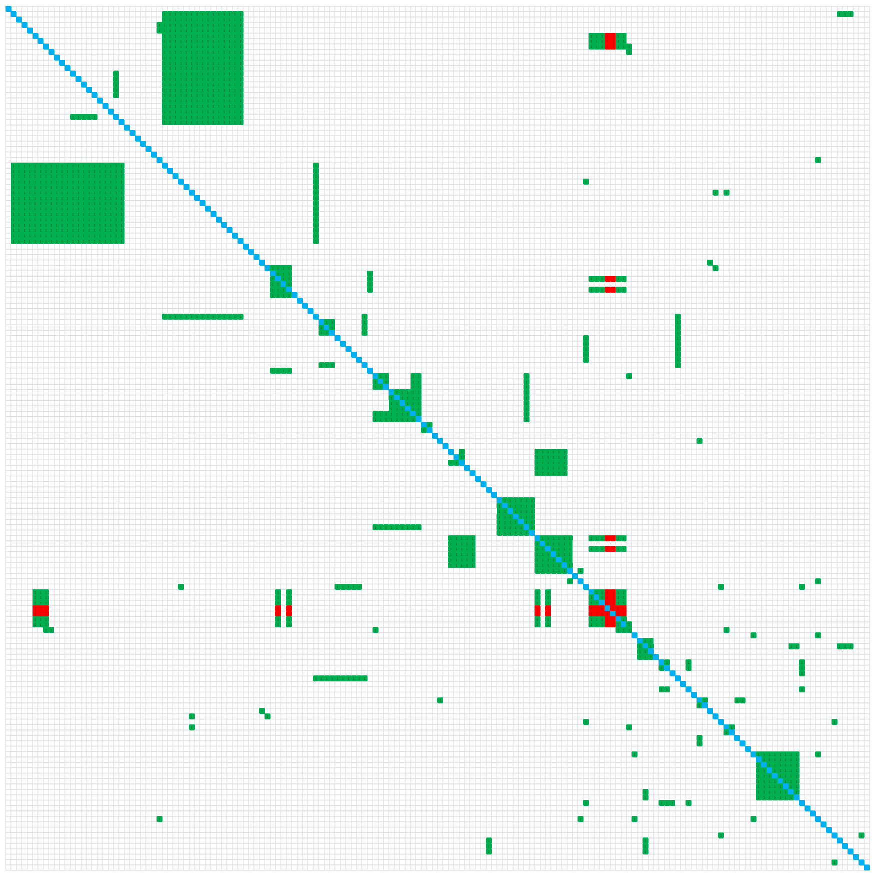


Figure 11.5: Toho Subject DSM – Requirement 9.3.7 First Order

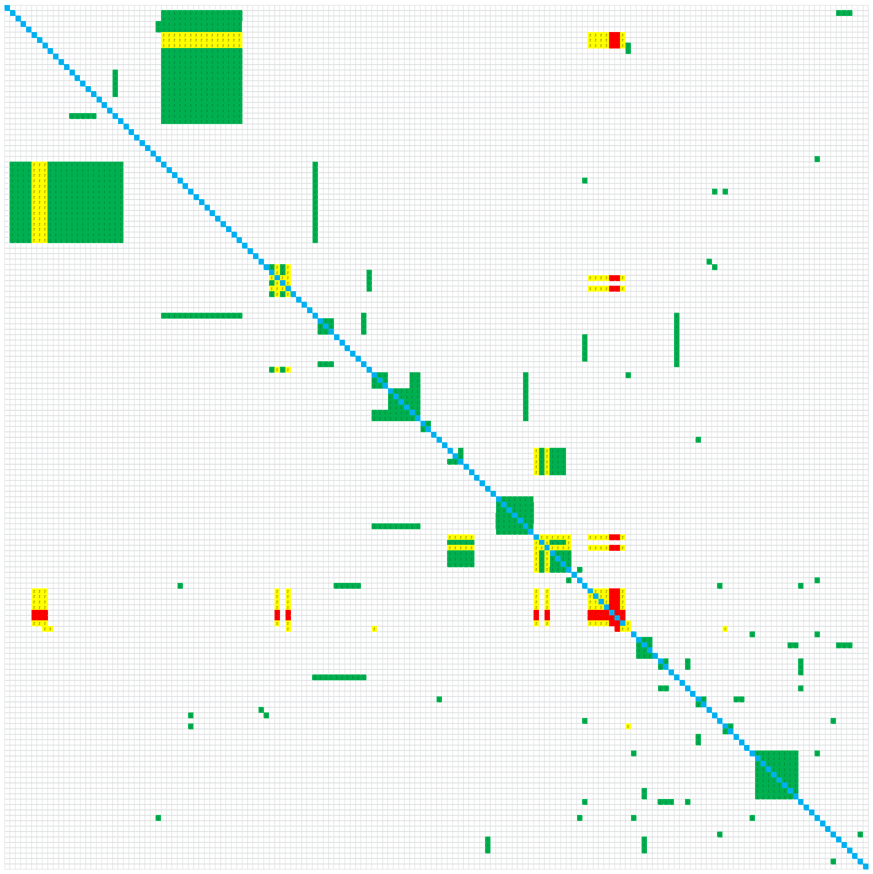


Figure 11.6: Toho Subject DSM – Requirement 9.3.7 Second Order

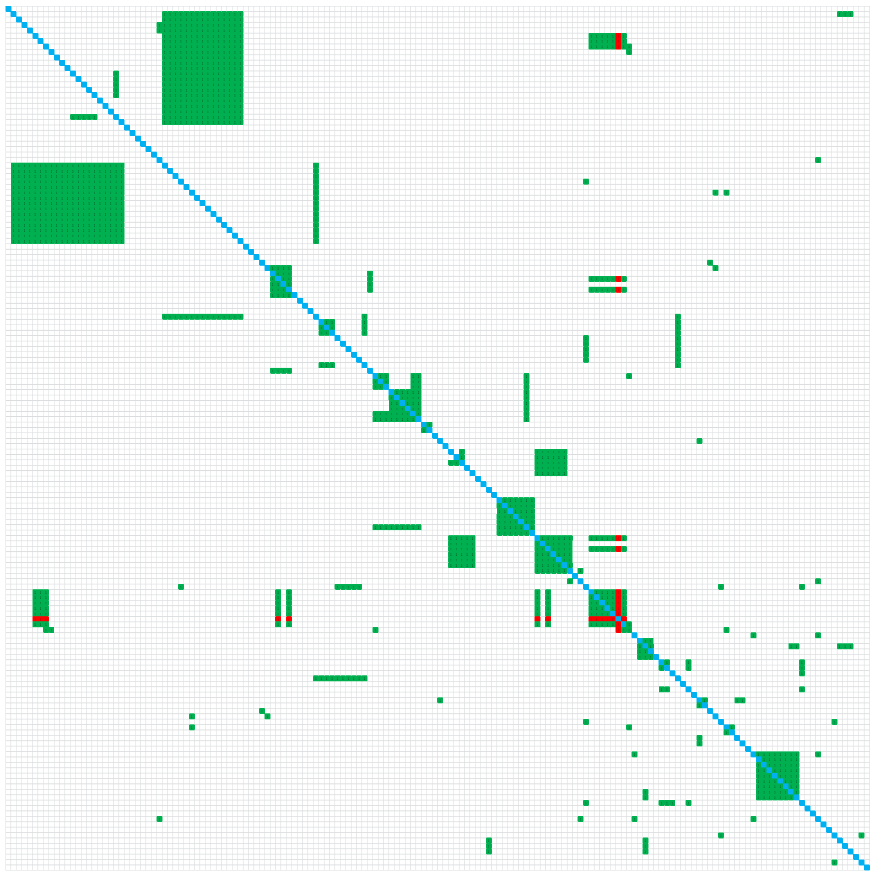


Figure 11.7: Toho Subject DSM – Requirement 9.3.10 First Order

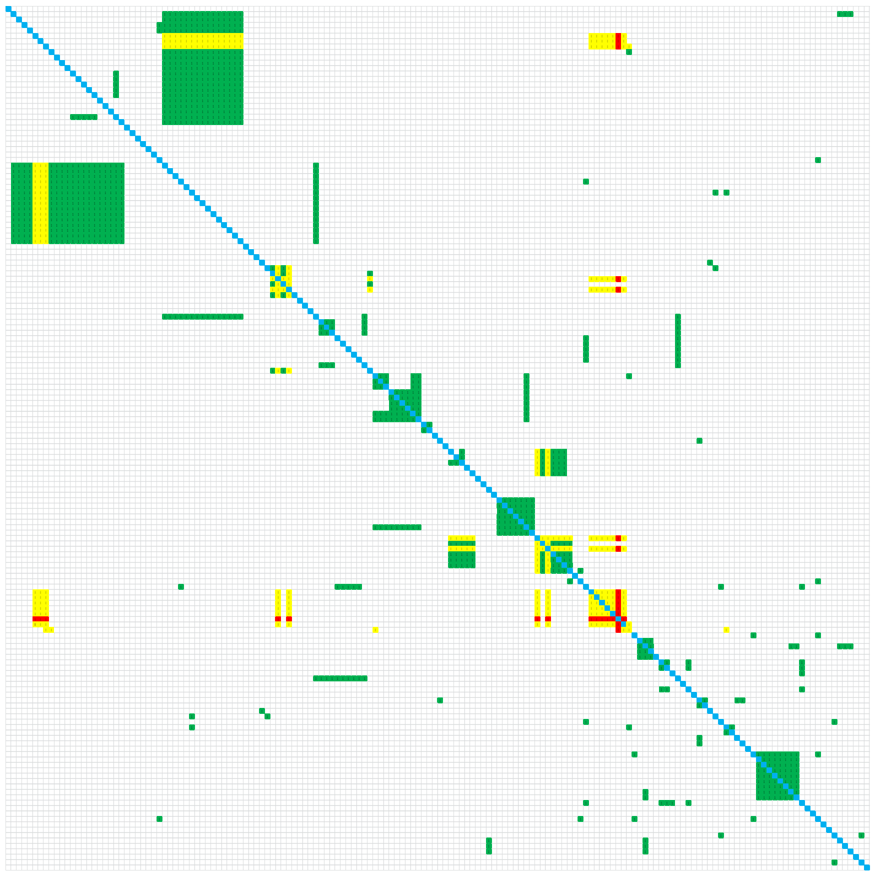


Figure 11.8: Toho Subject DSM – Requirement 9.3.10 Second Order

APPENDIX G
PIERBURG MANUAL – KEYWORD DSMS

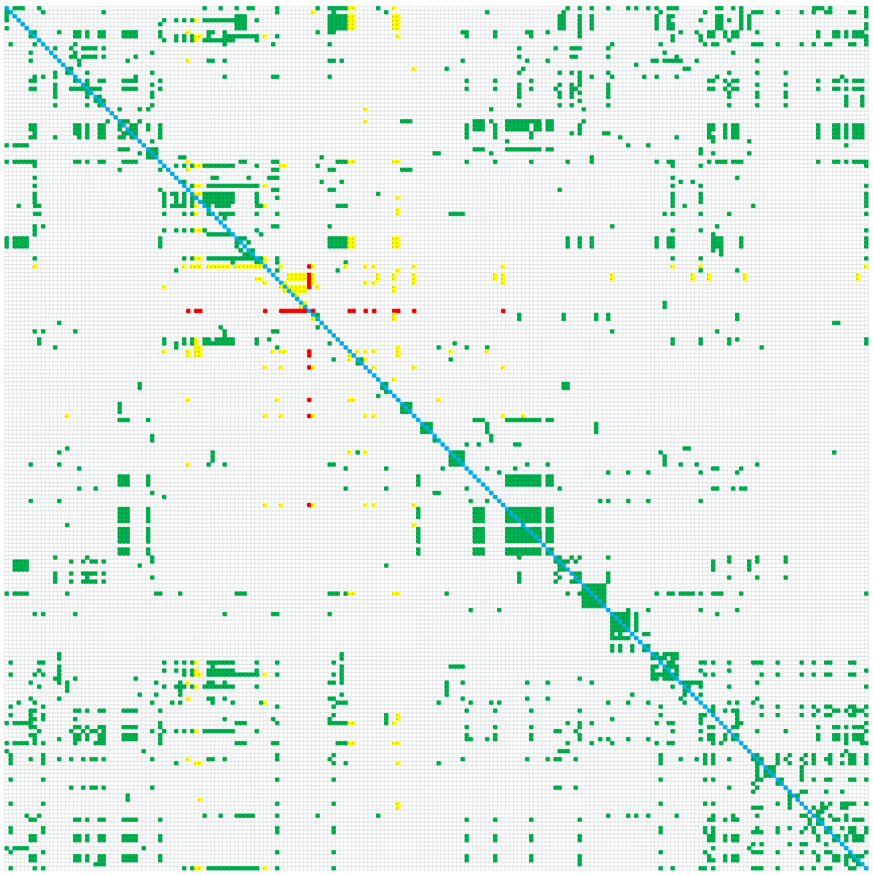


Figure 11.9: Pierburg Keyword DSM – Requirement 2.5.8

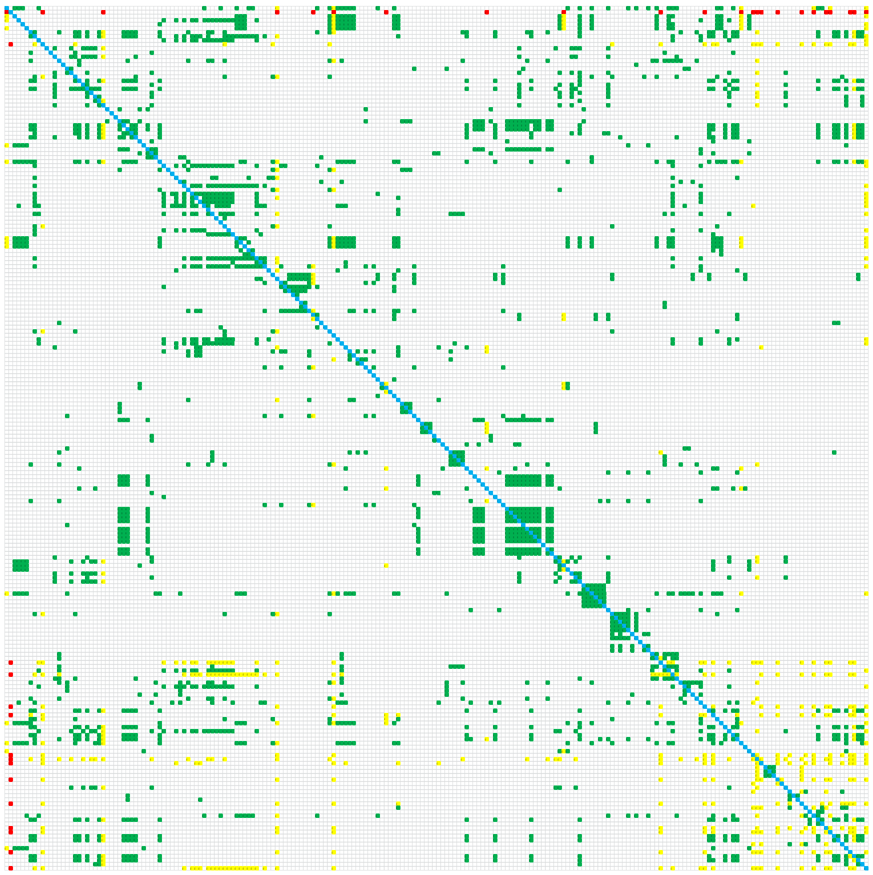


Figure 11.10: Pierburg Keyword DSM – Requirement 2.1.2

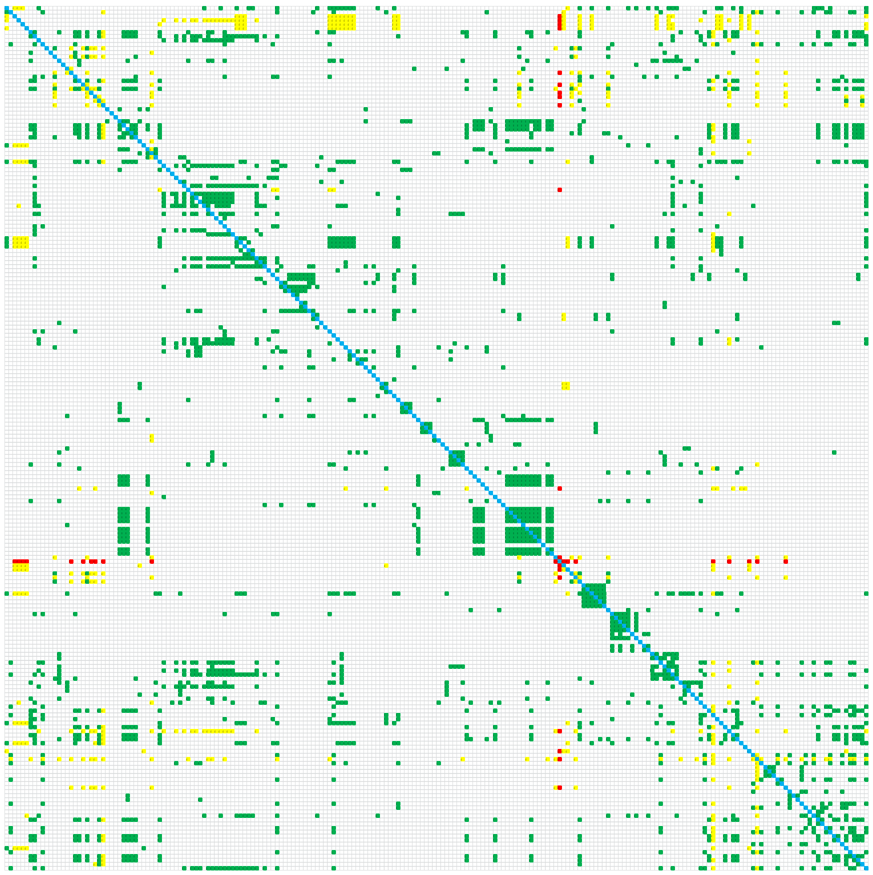


Figure 11.11: Pierburg Keyword DSM – Requirement 2.9.2

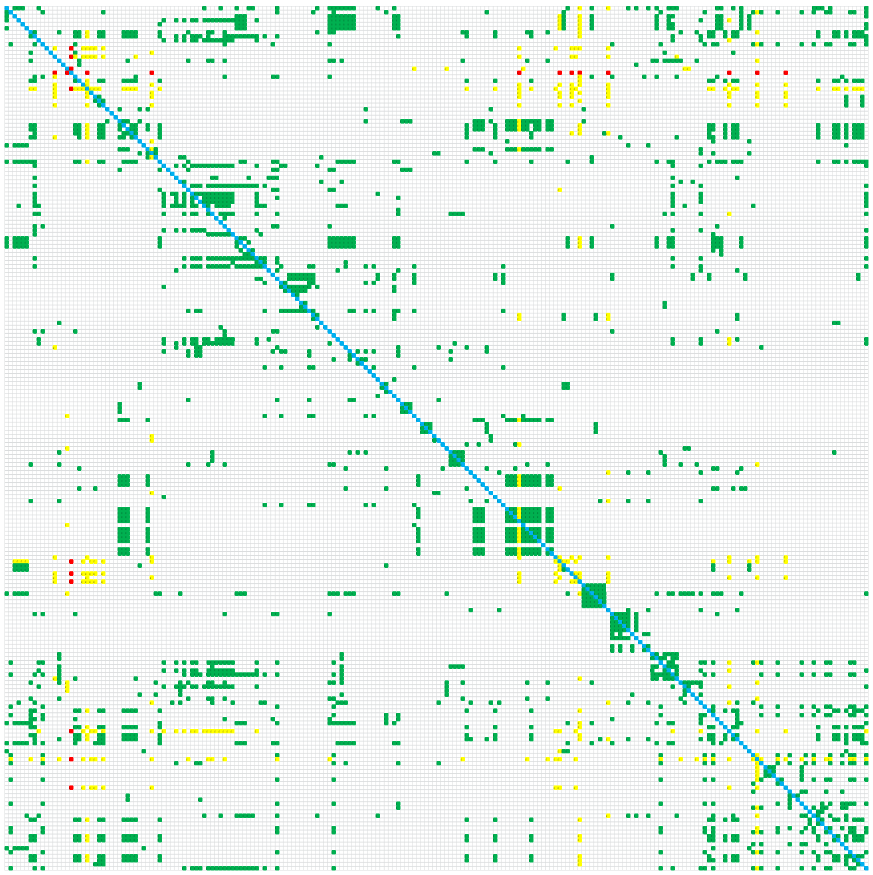


Figure 11.12: Pierburg Keyword DSM – Requirement 2.1.14

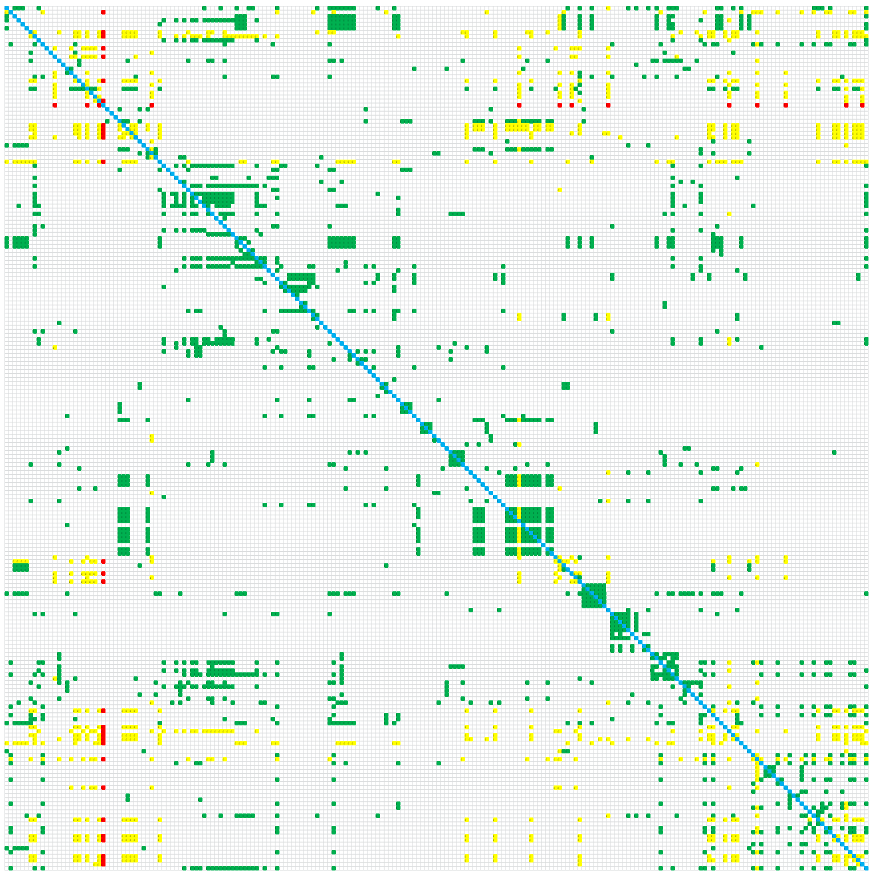


Figure 11.13: Pierburg Keyword DSM – Requirement 2.2.6

APPENDIX H
EZRAZ SYNTACTICAL DSMS

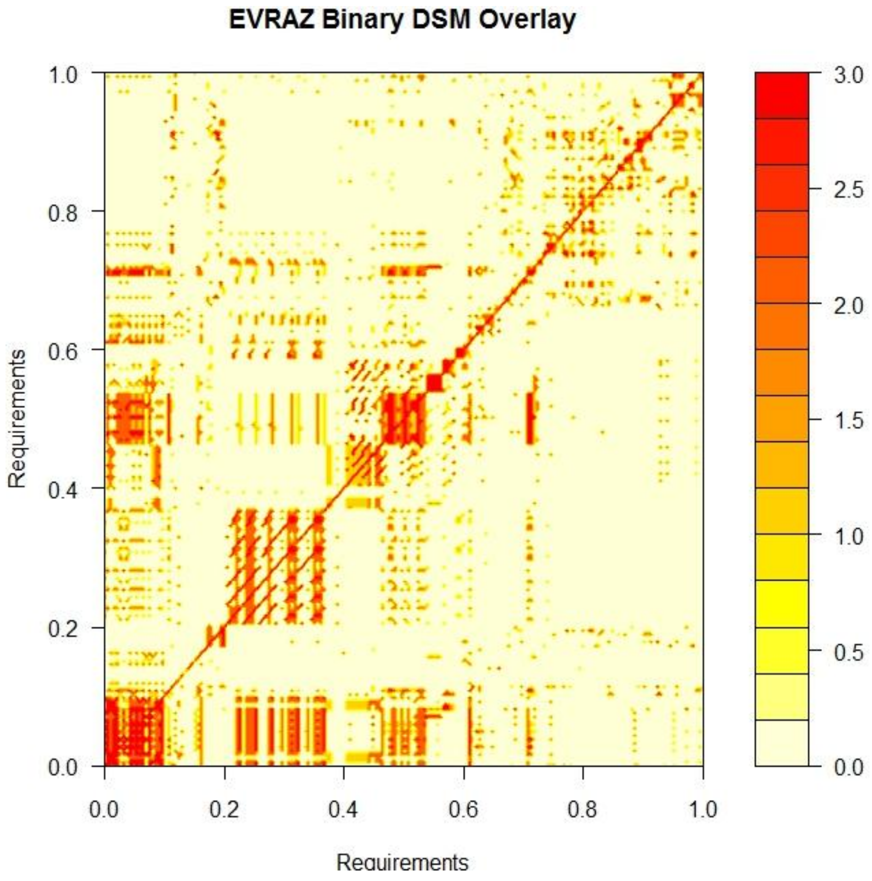


Figure 11.14: EVRAZ Binary DSM Overlay

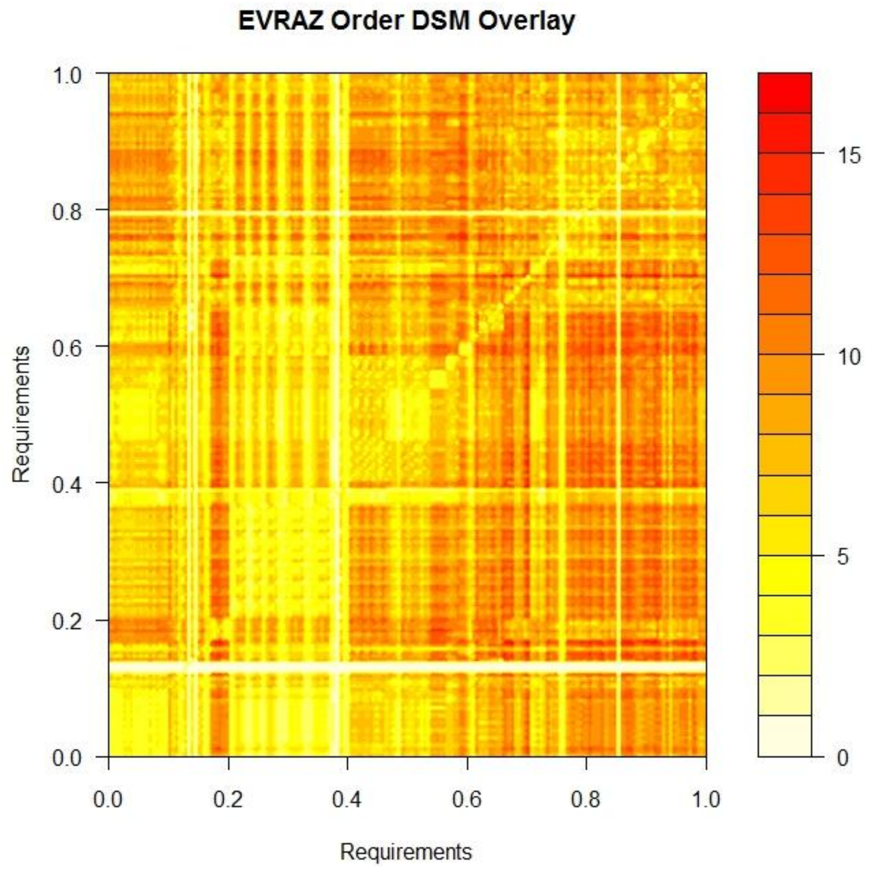


Figure 11.15: EVRAZ Relationship Order DSM Overlay

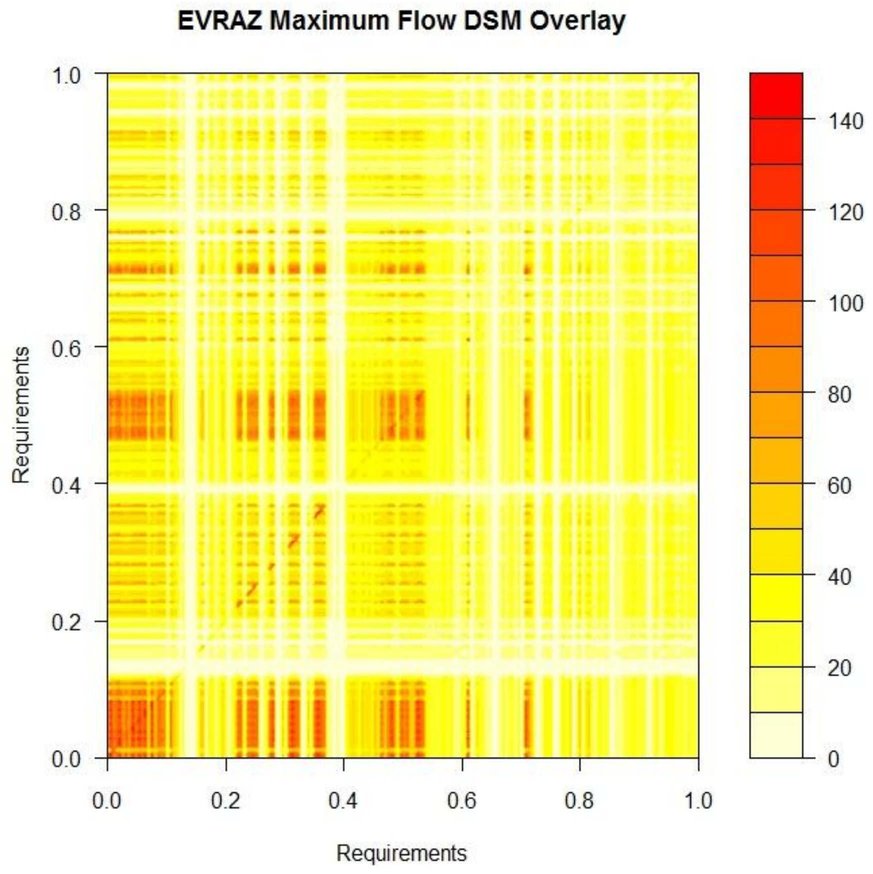


Figure 11.16: EVRAZ Maximum Flow DSM Overlay

APPENDIX I
MATRIX REPRESENTATION IN R

This report details the use of the statistical software R in developing matrix graphical representations. A matrix here is any dataset that has an m *x* n (or n *x* n) configuration between elements. The matrix is used to develop relationship between the elements and can be symmetric. An example matrix is shown in Figure 11.17.

As seen in the figure, the use of a numerical representation for matrices is difficult to comprehend. Effectively, the benefit of Figure 11.17 is to identify where relationship between elements exist, as it is binary. Nonetheless, it is apparent for such a simple set of data, its complexity is amplified when matrices are of larger size and dimension. The matrices plotted in this report are two dimensional binary matrices and three dimensional matrices (drawn in two dimensions). It is important to note the use of three dimensional contours are not explored here as they are not conducive plotting the element to element relationship matrices used here.

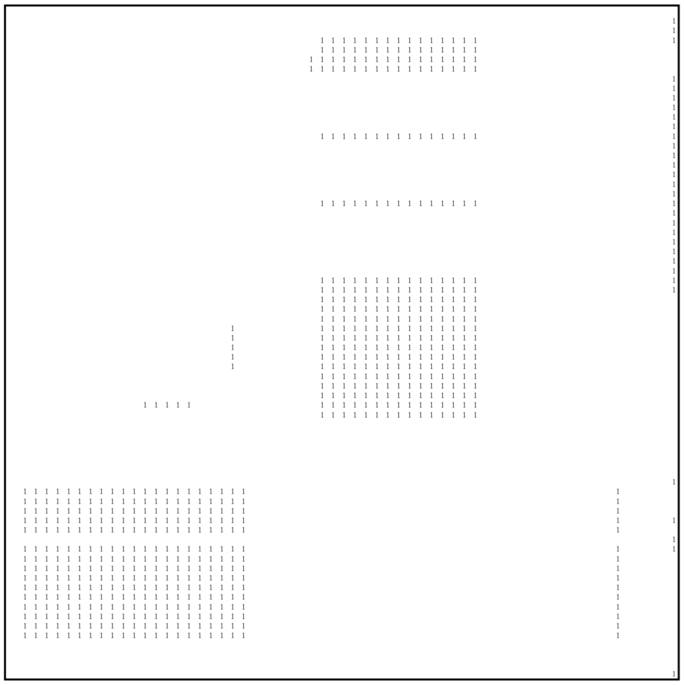


Figure 11.17: Example Binary DSM

The typical matrices used here are Design Structure Matrices (DSMs) which are matrices used to identify relationships between elements in a system. In some instances, these DSMs may be symmetrical. Alongside the relationships identified in each cell of the matrix, another dimension may be added to add granularity between cells. The strength of relationship may be used, adding another element of graphical need in the matrix representation. An example matrix using three dimensional data is shown in Figure 11.18. From a comprehension standpoint, such a figure does not add value. It is difficult for a reader to view this type of information numerically. Though each cell with a numeric value indicates a relationship exists, it is difficult to differentiate between other cells. As a result, a graphical means for representing the numerical data is needed. This is the motivation for using R as its advanced graphics features may be able to mitigate some of the issues in viewing large, complex matrices.

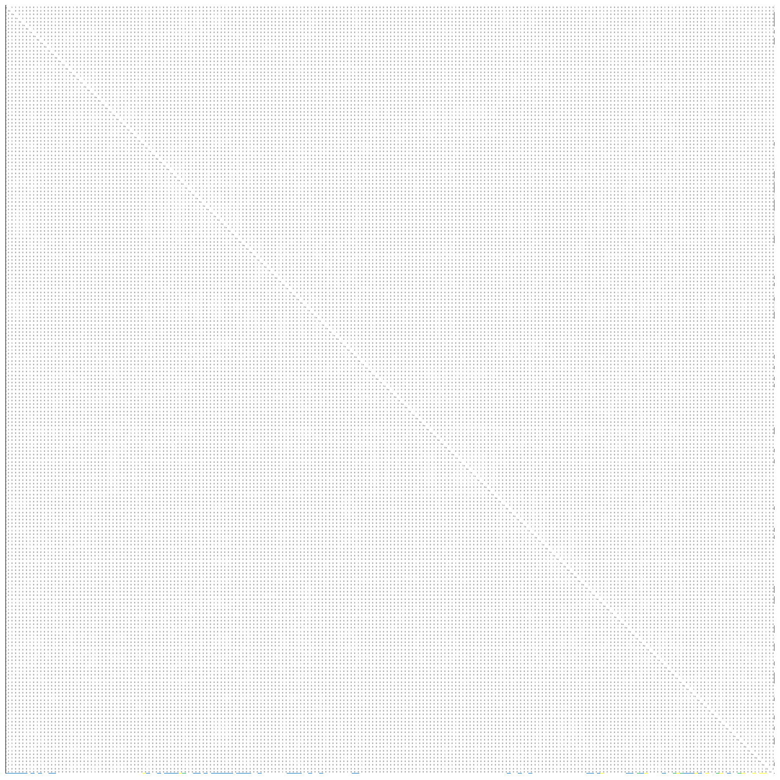


Figure 11.18: Three Dimensional Matrix

Plotting on R

Plotting matrices are possible on R making use of the contour plotting function. The data must be ready for plotting as sometimes R will not interpret a matrix like excel and MATLAB will. Basic procedures are needed before plotting the data and it is advised to convert all the data in the matrix from variables to numeric values, binding them as columns and rows. The specific function used to perform this in R is data.matrix. The code used to generate the matrices is as follows:

```
DSM <-data.matrix(DSM, rownames.force = NA)
BiDSM <-data.matrix(BiDSM, rownames.force = NA)
```

This is a recommended operation even if the data in the matrix are already numeric. It is unknown why R does not review the data as a matrix array unless the data.matrix function is performed. Rownames.force is used to for row and column names of the data. If set to NA, it sets rownames to NULL. Since the purpose of this is to plot the data, it is recommended to set rownames.force = NA as x and y axis may be labeled during plotting. With the data in matrix form, and numerically populated, a contour is used to develop the matrix graphic. The code used to generate the plots for both the binary and three dimensional (which incorporates strength of relationship in addition to the existence of a relationship) matrices is shown here:

```
#Matrix for the Binary Graph
filled.contour(BiDSM, color = terrain.colors,
   plot.title = title(main = "Binary DSM",
   xlab = "Requirements", ylab = "Requirements"))
#Strength of relationship Matrix
filled.contour(DSM, color = terrain.colors,
   plot.title = title(main = "DSM Overlay",
   xlab = "Requirements", ylab = "Requirements"))
```

The code used develops two plots, one for a binary matrix, making use of the data shown in Figure 11.17, and another for a three dimensional matrix (numerical, unplotted version shown in Figure 11.18). The generated matrix representation is shown in Figure 11.19 depicts a binary matrix of element relationships. As seen from the graphic representation, it is easier to interpret this model than a base numeric model. Though it is binary, R provides a legend for the color gradient. This is typical for any contour plot developed on R. This legend may be removed from the plotting feature if needed. As seen, plotting makes it very easy to view elements of high relationship concentrations (indicated by horizontal or vertical lines). Such plotting is superior to basic numerical representation. As previously noted, the plot will incorporate the x and y axis needed.

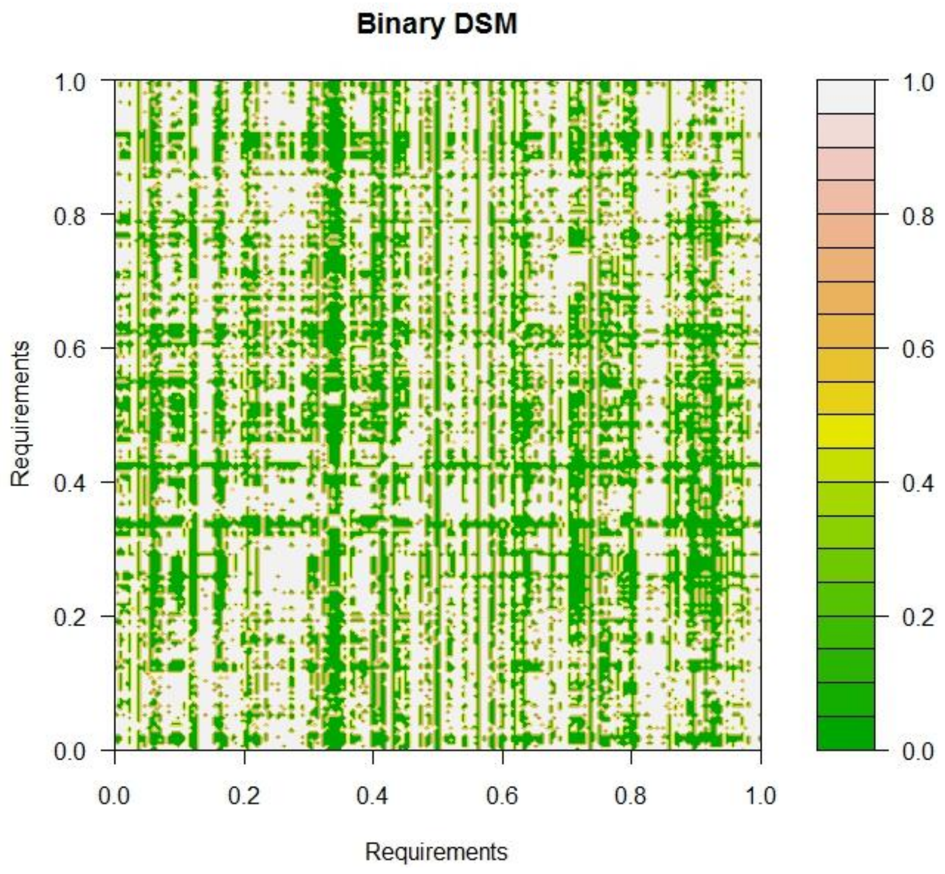


Figure 11.19: Binary Matrix

Figure 11.20 illustrates the output for the strength of relationship matrix, which was a three dimensional matrix with granularity between elements. Though it is termed a three dimensional matrix, it is illustrated as a two dimensional matrix with varying color within the color gradient selected by R. An advantage of using R over other tools, such as excel or MATLAB, is its ability to “smooth” the graph and colors. This will be discussed in greater detail in a later section with examples. Though the smoothing capability does not explicitly add value to the data, it does make the figure appear much more aesthetically pleasing, making it a presentable (for papers, reports, presentations) graphic. Different colors more conducive to granularity are available on R such as rainbow (shown in Figure 11.21). To develop the matrix plotting, a contour function termed filled.countour is used.

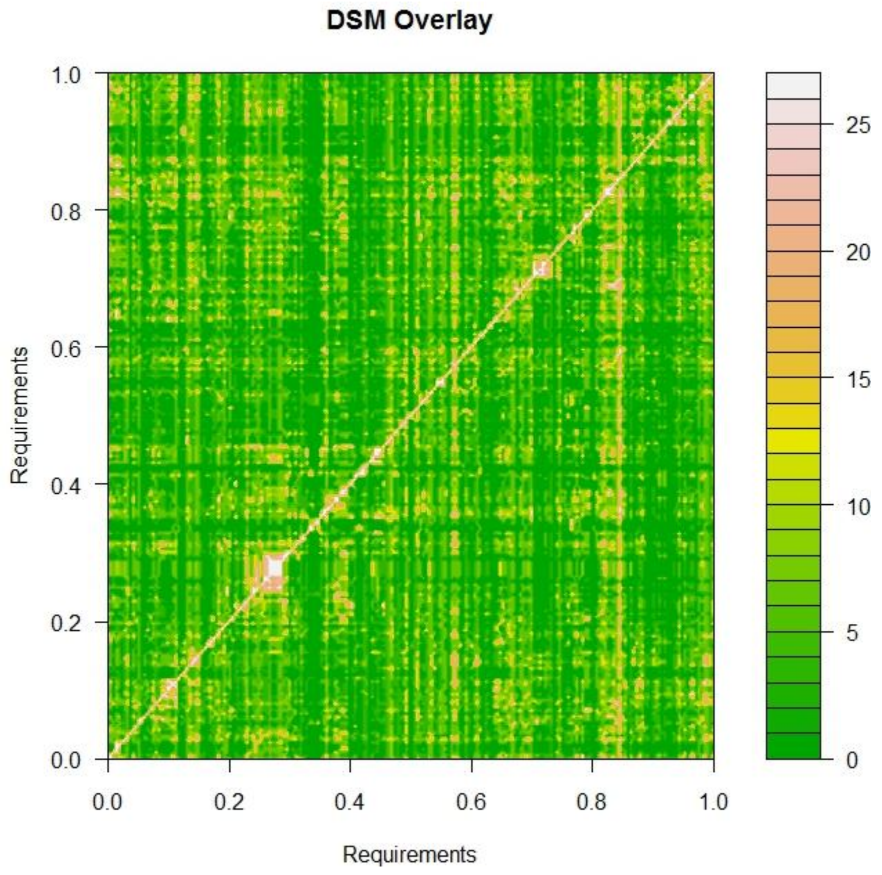


Figure 11.20: Three Dimensional Matrix

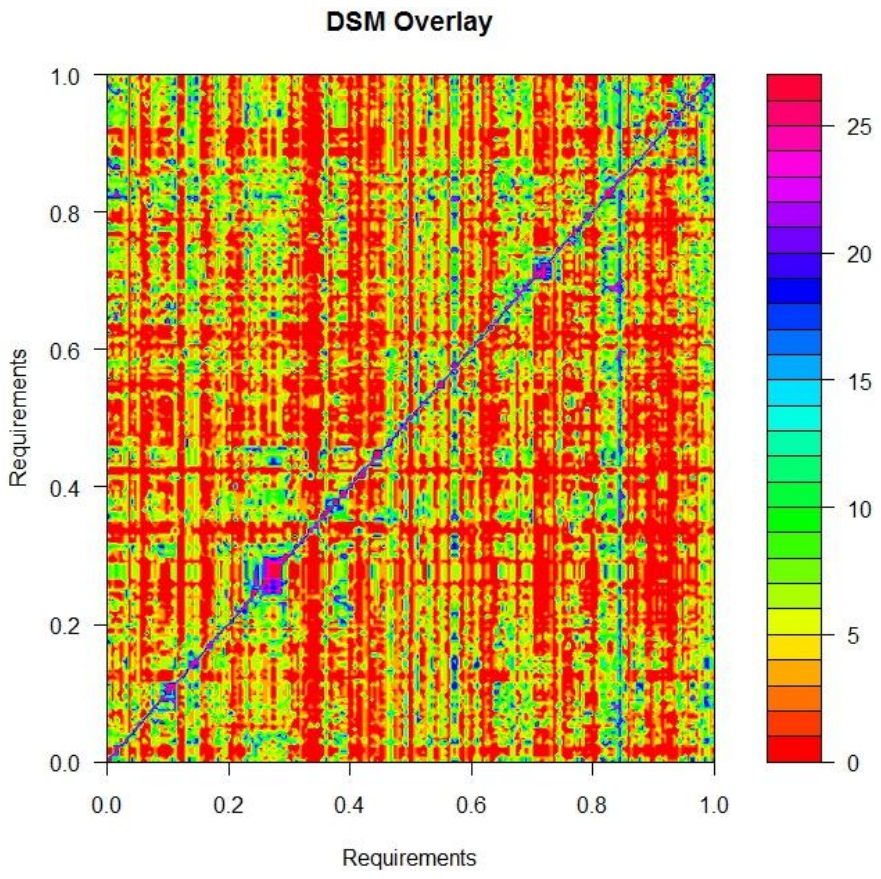


Figure 11.21: Three Dimensional Matrix with Rainbow color.palette

A filled.countour function is used to develop the matrix graphic. This function requires the graphic package to operate. The function has multiple arguments used to customize the plot. To demonstrate the ability to customize the filled.contour function, the help file code will be used and detailed. The help file of the filled.contour shows the following function with all the arguments included [204]:

```
filled.contour(x = seq(0, 1, length.out = nrow(z)),
        y = seq(0, 1, length.out = ncol(z)),
        z,
        xlim = range(x, finite=TRUE),
        ylim = range(y, finite=TRUE),
        zlim = range(z, finite=TRUE),
        levels = pretty(zlim, nlevels), nlevels = 20,
        color.palette = cm.colors,
```

col = color.palette(length(levels) - 1),
plot.title, plot.axes, key.title, key.axes,
asp = NA, xaxs = "i", yaxs = "i", las = 1,
axes = TRUE, frame.plot = axes, ...)

The arguments of the function are of relative ease to understand, making it easy to customize. The *x* and *y* arguments are the locations for which the matrix will be plotted. This can be considered an empty matrix array. The *z* values are the data points, located within each point on the *x* and *y* array which will be plotted. As performed with the DSM examples used above, a completed array of *x*, *y*, and *z* can be given. A limit for each dimension is given if the user wishes to produce an extract of the total matrix. This is useful if the user wants to visualize only a segment of the overall matrix. This is very useful for large, complex matrices with multiple sets of data clustered into one single, large matrix. The *levels* argument controls the gradient between the ranges of *z* for which the color will be plotted. The larger the levels, the more sensitive the gradient is to variation in the *z* of each cell. If *levels* is not specific, the *nlevels* argument may be used to develop a gradient for the color. The *color.palette* argument assigns the color used for the plot or the *col* argument. A single color may be used, with different shades or a specific color gradient, such as the terrain or rainbow used in the DSM examples. The *plot.title* simply states the titles of the plot while the *plot.axes* will control the axis on the plot. The *key.title* and *key.axes* add titles and axis on the plot as well. Because of the matrix illustrated, an argument for *asp* is available to control the aspect ratio of the plot, which defaults at 1 if not specified. The *xaxs* and *yaxs* control the axis style of the horizontal and vertical axis respectively. The default of these arguments makes use of the internal labeling offered by the graphics package. The *las* controls the type of styling to be used. The *axes* and *frame.plot* arguments indicate if the axis and a box should be drawn.

There are many key advantages to using R for plotting matrices in a graphical manner. To properly justify the use of R for such plotting, it is important to compare it to other tools which provide this functionality. Excel and MATLAB will be explored for their ability to develop matrix plotting and compared to R.

Comparison to Excel and MATLAB

As seen from the example, R is capable of providing plotting functionality equal to that of excel and MATLAB and in some capacities, it is better. A typical excel output of a matrix, example shown in Figure 11.22, provides a fine colored cell assortment based on the data in each cell. Though excel is capable of plotting matrices, there is no matrix specific means for performing this. This is performed through a cell management where the user must define a conditional formatting for the matrix. There is no default matrix generated plot default as there is with R and MATLAB. Though excel is capable of outputting such graphs, it was not designed to explicitly do so and is left for the user to identify how. As a result, the multiple arguments presented in R cannot be duplicated in excel, rendering it as a limited tool for developing graphical matrix representation.

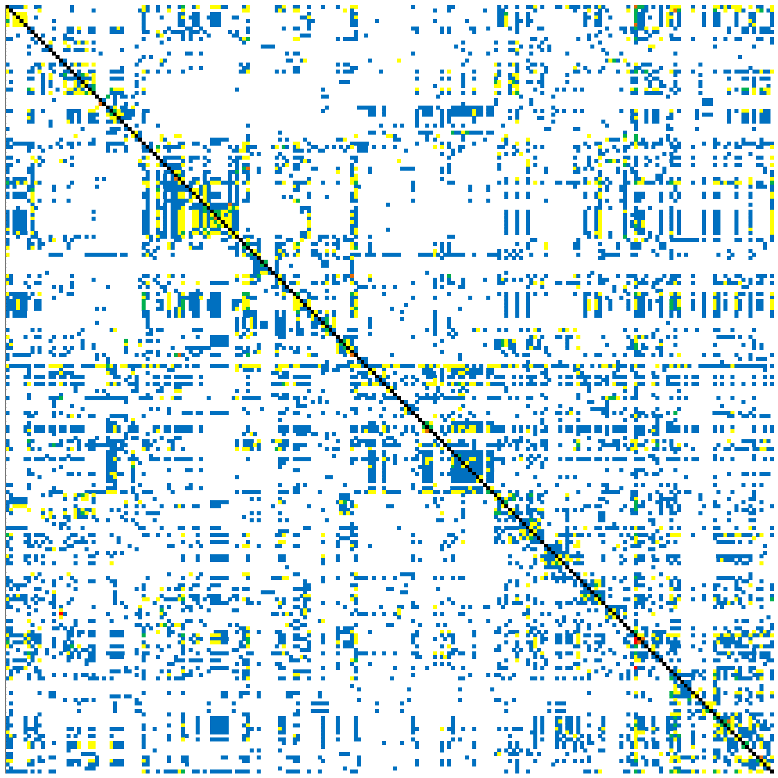


Figure 11.22: Excel Matrix Graphic

MATLAB offers a means for plotting matrices through a contour plotting, examples shown in Figure 11.23 and Figure 11.24. Alongside the plotting capability, MATLAB does offer the argument customization offered in R. In some regards, it does offer greater customization because of its plotting capability. Though MATLAB offers specific matrix plotting, its greatest limitation is it does not offer the plot smoothing which R does, making for a very pixilated graphic.

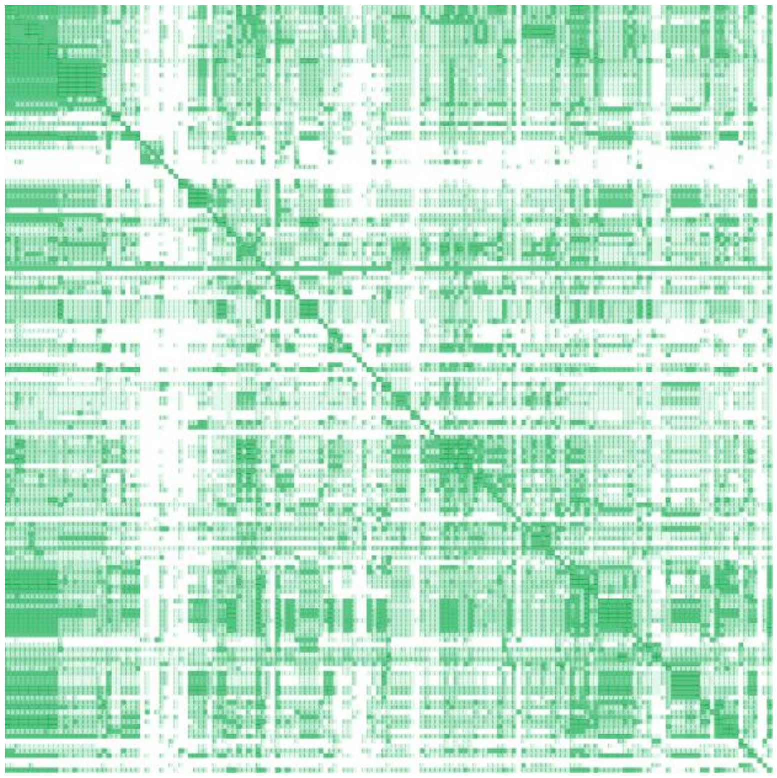


Figure 11.23: MATLAB Matrix Graphic

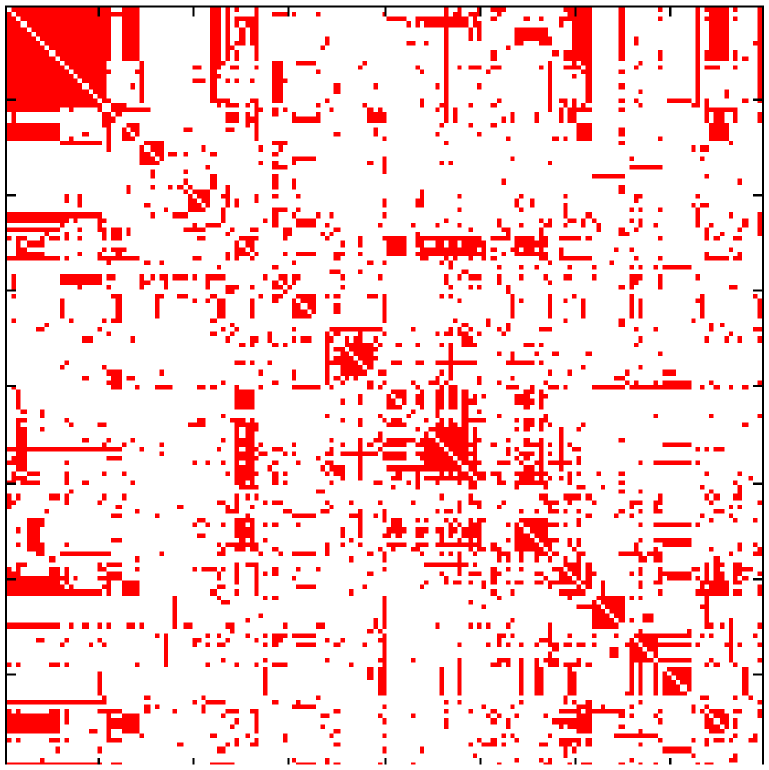


Figure 11.24: MATLAB Matrix Graphic

Takeaway

From the evaluation and comparison performed it is evident R is suitable and proficient in plotting matrices. It offers users the capability to plot any numeric matrices, in a two dimensional manner, while providing the user with a host of different function arguments to customize the plot. It is vastly superior to plotting in excel and provides customization capability comparable to that of other programs such as MATLAB. A key advantage to R is its graphing feature enhancement through plot smoothing, a feature very difficult to find in most matrix plotting tools and not available in excel nor MATLAB.

APPENDIX J
R CODE

```
setwd("C:/Users/Beshoy/Desktop")

library(graphics)
library(ellipse)
library(lattice)
library(cluster)
library(MASS)
library(gplots)
library(maps)
library(RColorBrewer)
library(cluster)
library(onion)
library(plotrix)
library(color.palette)
library(stats)

################################
#Comparing populations

# Load up the data
data <- read.csv(file="TPTester.csv",head=T)
save(data,file="data.rda")
load("data.rda")
TohoF2 <-data[1]
TohoF3 <-data[2]
PierburgF2 <- data[3]
PierburgF3 <- data[4]

summary(PierburgF2)
hist(TohoF3$TohoF3, prob=T, breaks=100, col="blue", xlim=c(0.75,1.3),xlab="F3
Scores", ylab="frequency",main="Toho F3 Scores")
lines(density(TohoF3$TohoF3),lwd=3)

hist(PierburgF3$PierburgF3, prob=T, breaks=450, xlim = c(0.75,1.5), col="blue",
xlab="F3 Scores", ylab="frequency",main="Pierburg F3 Scores")
lines(density(PierburgF3$PierburgF3),lwd=3)


######################################################################
######################################################################
### PLOTS A CONTOUR
# Load up the DSM!
DSM <- read.csv(file="EVRAZMaxOverlay.csv",head=F)
save(DSM,file="DSM.rda")
```

```
load("DSM.rda")

DSM <-data.matrix(DSM, rownames.force = NA)
filled.contour(DSM, color = function(x)rev(heat.colors(x)),
    plot.title = title(main = "EVRAZ Maximum Flow DSM Overlay",
    xlab = "Requirements", ylab = "Requirements"))

########################################################################
########################################################################
### Binomial Test

x = 9 #number of successes
n = 104 #number of trials
p = 1/1892 #propability of sucess
binom.test(x, n, p, conf.level = 0.95)
```