Knowledge Management of System Interfaces and Interactions for Product Development Processes

by

Ronnie E. Thebeau

B.S. Electrical Engineering, Worcester Polytechnic Institute, 1992

Submitted to the System Design & Management Program
In Partial Fulfillment of the Requirements for the Degree of

Master of Science in Engineering and Management

at the

Massachusetts Institute of Technology

February 2001

Signature of Author_ ________________________
Author: Ronnie E. Thebeau
System Design and Management
January 12, 2001

Corporate Sponsor ________________________
Corporate Sponsor: Patrick Hale
Director Global Systems and Controls
Otis Elevator Company

Certified By____________ ________________________
Thesis Advisor: Dr. Daniel E. Whitney
Senior Research Scientist

Accepted By ____
LFM/SDM Co-Director: Dr. Paul A. Lagace
Professor of Aeronautics & Astronautics and Engineering Systems

Accepted By ____________________ ________
LFM/SDM Co-Director: Dr. Stephen C. Graves
Abraham Siegel Professor of Management

Knowledge Management of System Interfaces and Interactions for Product Development Processes

by

Ronnie E. Thebeau

Submitted to the System Design & Management Program
on December 21,2000 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Engineering and Management

Abstract

A system architecture was developed and analyzed for a basic elevator system using a limited number of system level components. A Design Structure Matrix was created which represented the complex interactions of the system components. These components were derived from a decomposition of system requirements, code and safety requirements, and evaluation of scenario operational requirements. Clustering routines using cost assignment of interactions aided in optimizing the cluster assignment of components. These cost assignments reflect cost and time associated with managing interactions inside and outside of subsystems. Management and optimization of the interfaces between the clustered components leads to an architecture that minimizes complexity and will hopefully lead to quicker and less costly product development cycles.

Using this approach, near-optimal architectures can be analyzed and alternatives can be evaluated for system level impact. As was observed with this test case, highly complex or integrative systems are difficult to analyze, even with the tools utilized. These tools provided a structured approach that utilizes an objective process. This approach provides documentation and analysis of the architecture that is normally managed on the fly as product development progresses. The results of the analysis can provide a framework for an organizational structure of the product development process, provide an avenue for dialogue between design teams responsible for different subsystems, provide a process for evaluation of architecture alternatives, and identify the interactions between subsystems that must be managed carefully.

Thesis Supervisor:	Dr. Daniel E. Whitney
Title:	Senior Research Scientist
	Center for Technology, Policy, and Industrial Development

Acknowledgements

There are certainly many people I would like to thank as I finish the last part of my program at MIT and complete this thesis.

First, I must thank those at Otis who have given me this unique and rewarding opportunity to enrich my experience and knowledge. I'd like to thank the Otis senior management team for supporting the program that opened up the opportunity. I'd especially like to thank Pat Hale for sparking my interests in Systems Engineering, systems thinking, and for guiding me through the process and believing in my abilities. I must also thank Dave Lanesey and Rudi Steger for being extremely supportive of the extensive time required to complete this degree. They allowed the school requirements to almost always come first. Their understanding and support allowed me to give this program my full attention.

I'd like to thank my advisor, Dan Whitney, for his guidance as I worked through the thesis process. I didn't always know where we were going with this, but he kept me on track. Thanks for the attention and guidance. I'd also like to thank Qi Dong. Even at a distance, Qi provided valuable guidance and tips as I got started with the DSM.

I'd also like to thank the staff of the SDM program and my fellow students for making this an enriching program. We sometimes had fun too.

I must also thank my friends and family for their understanding and support. I have missed many events over the past two years. I must thank my fiancée's family for their support and understanding as I was only able make occasional visits.

I must send a special thanks to my Mom (Norma), and my brothers Ricky and Mark. They have been there for me as needed even though their schedules were also busy. Many times I had to miss family events but they always understood. They made trips to visit me so that I had more time to work. It's great to have a family that you always know will be there. Thank you Mom, Ricky, Mark. Hey Mom – look what I've done now!!

Most of all, I must send a HUGE thank you to my fiancée, Robin. Getting engaged during this program wasn't easy. Robin has delayed wedding plans, moved vacations, and put many other plans on hold while this program progressed. Robin, thank you for waiting for me. We can now begin the next stage.

Table of Contents

CHAPTER 1 INTRODUCTION **7**

MOTIVATION 7
ORGANIZATION OF THE THESIS 8

CHAPTER 2 BACKGROUND **11**

ELEVATORS 11
Brief History *11*
How elevators work *12*
THE ELEVATOR SYSTEM DESIGN PROCESS 14
THE ELEVATOR SYSTEM DESIGN PROCESS 15
THE DESIGN STRUCTURE MATRIX 17
PREVIOUS WORK 19

CHAPTER 3 CREATION OF THE DSM **20**

INTRODUCTION 20
FUNCTIONAL DECOMPOSITION 21
SCENARIO ANALYSIS 21
FUNCTIONAL TO PHYSICAL MAPPING 22
ENTER THE DSM INTERACTIONS 24

CHAPTER 4 ORIGINAL CLUSTERING ALGORITHM **27**

ORIGINAL ALGORITHM BACKGROUND 27
MATLAB® TOOLS 29
PARAMETER APPLICATION TO ELEVATOR DSM 30
RESULTS OF ORIGINAL ALGORITHM 40
CHAPTER SUMMARY 42

CHAPTER 5 MODIFIED CLUSTERING ALGORITHM **43**

INTRODUCTION 43
PENALIZE COST OF MULTIPLE CLUSTER MEMBERSHIPS 43
SAVE THE BEST SOLUTION 49
CHAPTER SUMMARY 55

CHAPTER 6 CLUSTER ANALYSIS **56**

MULTIPLE RUNS 56
FINDING LIKE CLUSTERS 57
VARIED INTERACTION LEVELS 62
EXTRACTION OF INTEGRATIVE ELEMENTS 63
CHAPTER SUMMARY 69

CHAPTER 7 RESULTS SUMMARY **70**

CLUSTER ANALYSIS 70
SYSTEM ELEMENTS 71
CLUSTERED ELEMENTS 73
INTERACTIONS OUTSIDE THE CLUSTERS 76
CHAPTER SUMMARY 77

CHAPTER 8 CONCLUSIONS **78**

RESEARCH CONCLUSIONS 78

FUTURE WORK 81

CHAPTER 9 APPENDIX **83**

APPENDIX A FUNCTIONAL DECOMPOSITION EXAMPLE 83
APPENDIX B SCENARIO TO FUNCTIONAL MAPPING MATRIX 85
APPENDIX C FUNCTIONAL TO PHYSICAL MAPPING 86
APPENDIX D BASE DESIGN STRUCTURE MATRIX 87
APPENDIX E FINAL CLUSTERING RESULTS 89
APPENDIX F MATLAB MODIFIED CLUSTERING ROUTINES 92

BIBLIOGRAPHY **149**

List of Figures

Figure 1: Elevator System Diagram 14
Figure 2: Design Structure Matrix Interactions 35
Figure 3: Foundation Algorithm Cluster Matrix - Element to Cluster Mapping 36
Figure 4: Foundation Algorithm - Clustered DSM 37
Figure 5: Foundation Algorithm- Cluster List Example 38
Figure 6: Foundation Algorithm Cost History Example 39
Figure 7: Cluster Matrix with Multiple Membership Penalties 46
Figure 8: Clustered DSM with Multiple Cluster Membership Penalty 47
Figure 9: Cost History with Penalty for Multiple Cluster Membership 48
Figure 10: Elevator DSM with Varied Interaction Strengths 51
Figure 11: Cluster Matrix of DSM with Varied Interaction Strengths 52
Figure 12: Clustered DSM with Varied Interaction Strengths 53
Figure 13: Cost History, New Algorithm, DSM with Varied Interaction Strengths 54
Figure 14: Cluster Likeness Example 58
Figure 15: Example of Cluster Average Likeness Calculation 59
Figure 16: Cluster Likeness Example Plot; Standard Interactions 61
Figure 17: Clusters driven by Integrative Elements 64
Figure 18: Cluster DSM with System Parameters on the Outside 67
Figure 19: P&W Distributed and Modular Parameters 68
Figure 20: Interactions outside of the Cluster 77

List of Tables

Table 1: Example DSM Interactions 18
Table 2 : Clustering Parameter Values 31
Table 3: Clustering Analysis Results 70
Table 4: Cluster Defined System Tree 75

Chapter 1 Introduction

Motivation

Product development of today's complex systems often requires managing large numbers of interfaces, both physical and organizational. Physical systems and their architecture continue to become more complex as the number of interfaces grows and "increasing complexity is at the heart of the most difficult problems" (Rechtin & Maier 8). To reduce complexity, it is often desirable to minimize the number of actual interfaces close to what may be considered an optimal or essential number of interfaces that produce the desired functionality and meet product requirements. Managing these interfaces requires an understanding of functional requirements, system interactions, and the system implications of local design changes. Engineers and product development managers usually have a good understanding of local design requirements, but may frequently under-estimate the system effect of interface architecture or the effect of local design changes.

It is believed that insufficient knowledge management of system interfaces may result in an excess number of interfaces, design induced rework, or insufficient knowledge of implications of design changes. A knowledge management tool to document, track, and analyze system, subsystem, and component interactions may minimize the number of interfaces, reduce cycle-time by minimizing rework, and provide an essential systems understanding for engineers and managers. Efficient interface design will also allow for technology and design updates of product components while minimizing the impact on other components and the system level architecture. Ideally, the objective is to define proper interfaces that allow efficient upgrades/changes to product components without causing rework that is not functionally required by the upgrade/change. It is highly desirable to have a process for developing an optimized interface architecture that is independent of current technology, company history and organizational structure, and one that is able to support efficient field service.

The management of complexity related to interactions is equally important to product development organizations. As the number of interactions between development teams increases, delays are incurred from waiting for information and errors are more likely as designs are more reliant on information from other teams. Therefore it is equally important to optimize team interactions as it is for a system's physical interactions.

Organization of the Thesis

To create a process and method for evaluating interface architectures, a test case will be developed for a roped elevator system. The test case will be somewhat generic but will carry enough detail to evaluate the requirements across different engineering disciplines, interface types, and functional requirements. This test case will also demonstrate the potential for using this method on actual products.

There are several stages of this work. The first set of activities will be focused around the creation of a Design Structure Matrix (DSM) and will be covered in Chapter 3. The DSM will represent the interactions between the elements and will be used to document and evaluate the interface architecture. The DSM will be created through a functional decomposition of system level requirements. These functional requirements will then be used to develop the physical system that will represent the functional requirements. The physical elements will be mapped to the functional requirements through a matrix implementation. Then finally, the DSM will be extracted from the functional to physical mapping.

The second set of activities will be focused on evaluating the example DSM with a clustering algorithm. The "first step in structuring is usually aggregating – collecting or clustering closely related functions or requirements together" (Rechtin 39). The clustering algorithm will be used to optimize the architecture by grouping like elements. The objective will be to contain the majority of element-to-element interactions within clusters and minimize the interactions outside of the clusters while keeping the size each cluster reasonable. "Choosing the appropriate aggregation of functions is critical in the design of systems." (Rechtin 39) The implementation and evaluation of a foundation clustering algorithm will be covered in Chapter 4. The foundation clustering algorithm is based on work by Carlos Iñaki Gutierrez Fernandez in his thesis "Integration Analysis of Product Architecture to Support Effective Team Co-location". This initial algorithm will be used to evaluate clustering, its application to the elevator problem, and initially get appropriate clustering parameters. This algorithm will also be used to get an understanding of the clustering procedure.

The objective is to produce a system architecture, and possibly an organizational structure, by assigning system elements to groups. With this, we can learn more about the elevator system and how its components can be optimally clustered to minimize unnecessary interfaces. The end result provides some innovative ways to cluster the elements to produce an architecture and it confirms some of the design choices for the current architecture. Th results will show that some elements produce well-defined clusters while other elements hinder optimal clustering. When the elements hindering effective clustering are removed from the clustering calculations, they provide valuable information if redefined as a system element that is managed differently than the clustered elements.

The example problems in Fernadez's work were much simpler than the elevator problem, and were mainly focused on effective co-location of team members. The clustering algorithm in Fernandez's thesis will be extended to make it more applicable to the elevator problem. The changes will reflect the needs of complex real-world problems where there is a large number of elements. Several modifications to the algorithm will be evaluated. The final clustering algorithm and parameters will be documented in Chapter 5.

All findings of the clustering routines, including how well it evaluated and proposed interface architectures, will be discussed in Chapter 6. The results and their implications on architecture and product development organizations will be covered in Chapter 7. Finally, conclusions and proposed future work will be covered in Chapter 8.

Chapter 2 Background

Elevators

The interface issues and clustering algorithms for this work have been applied to a basic roped elevator system. The elevator system presents some issues that make clustering and interface design nontrivial. The elevator system does not provide any natural grouping that spans the system. Instead the elevator is made up of many subsystems that represent different types of interfaces. These interfaces may not always be spatially close and can be highly integrative. There are control, communication, power, structural, and safety functions that are all integrated into an automated system that must communicate and take commands from the lay person, meet performance requirements, and maintain safe operating conditions for the passengers and service personnel.

Brief History[i]

Prior to 1852 elevators had largely been used as a hoisting device for freight with the largest advances coming from steam and hydraulic power. In 1852, Elisha Graves Otis introduced the world's first safety system that would protect the passenger in case of a breakage in the ropes. In 1857 the first passenger elevators were installed in New York and by 1873 over 2,000 elevators were installed across America. In 1903 Otis Elevator introduced the first elevators that were driven directly by an electric motor, without a gearbox, and it cleared the way for the era of the

[i] Dates and statistics obtained from Otis Elevator Company public web page at http://www.otis.com/aboutotis/elevatorsinfo/

skyscrapers. Since then improvements and innovations have included automatic controls, microprocessor based controls, improved ride quality, and improved performance times.

How elevators work

The basic roped elevator (Figure 1) consists of a cab or containment to hold the passenger or load, a counterweight to balance the weight of the cab and an electric motor to provide the motion. Hoist ropes (usually steel cables) connect the cab and counterweight and are wrapped over a rotating sheave that is connected to the motor to provide motion. Compensation ropes are connected to the bottom of the cab and bottom of the counterweight to counterbalance the weight of the hoist ropes as the cab and counterweight travel up and down.

The cab and counterweight travel in a shaft called the hoistway that runs the length of the travel distance for the system. Steel rails are located on the side of the hoistway for guidance of the cab up and down the hoistway. The guidance system maintains the orientation of the cab with respect to the building. A similar set of rails guide the counterweight. The motor and control systems are usually located in a room at the top of the hoistway called the machine room. The combined hanging weight of the cab and counterweight provide the necessary traction for the ropes so that they don't slip on the motor sheave.

The control system is usually located in the machine room and receives signals from different sensors and systems located on the motor, in the building hallway, within the hoistway, in the machine room, and in the cab. Motion commands and requests can originate from the hallway, the cab, or other systems (group control, building systems control, etc.). A multiple wire cable (travel cable) that connects the machine room to the cab supplies power and communication signals for the cab. One end of the travel cable is anchored at the machine room and the other travels with the cab.

The mechanical safety system consists of mechanical jaws located beneath the cab to stop it in case of rope breakage or an uncontrolled falling motion in excess of maximum speed. The mechanical jaws (called safeties) are activated by a mechanical governor system located at the top of the hoistway. The governor is connected to the safeties by a rope and when the cab speed exceeds the maximum, the governor system clamps the governor rope and pulls on the safeties, causing them to drop and lock onto the steel rails. There is also a buffer, usually a hydraulic piston, located at the bottom of the hoistway, to cushion the landing of the car in case it travels below the bottom floor.

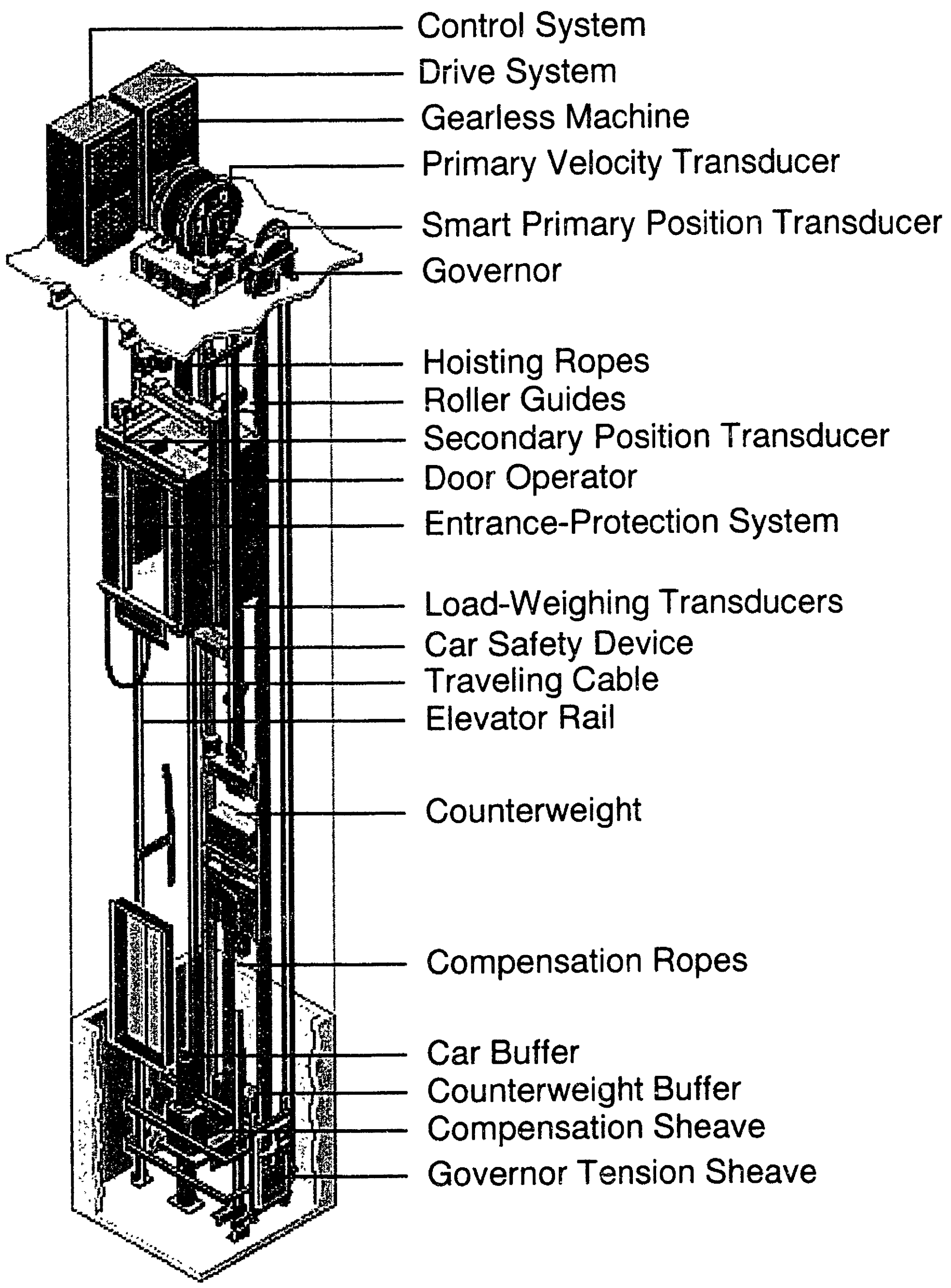


Figure 1: Elevator System Diagram[ii]

[ii] Otis Impact Resource CD. Copyright Otis Elevator Company 2000

The Elevator System Design Process

The design process for an elevator system typically takes 2-5 years for new product development and 1-2 years for product modification. The process usually begins with a marketing request to meet some new market requirements for product performance, price, or features. When the market requirements have been defined, engineering must evaluate the request for feasibility and resource availability. This part of the process usually includes discussions with manufacturing, field service, and product strategy experts. After negotiations, a document is produced to define the system requirements. Sometimes this takes the form of a modification to an existing product or it can initiate a new product development.

During the development phase that defines the system requirements, there are often many requirements that must be traded off. Product performance usually comes at a price, and the end user of an elevator system (riding passenger) is usually not the purchaser of the product (building owner, construction company). Therefore performance for the passenger must be weighed against the cost requirements of the purchaser. Every installation of an elevator is a unique system as all buildings vary in height, number of floors, floor height, number of door openings, etc. The performance parameters such as maximum load weight, maximum velocity, maximum acceleration also varies from building to building. Even the system inertia (total mass) varies with the height of the building and the purchaser's decisions on the style and features of the passenger cab. Because it is impractical to design a system for each building, the system design must be able to operate in a range of conditions, as each installation is custom built. It is also impractical to design many different systems because of the relatively small volume and high cost of engineering and manufacturing for each system. This usually results in some type of platform for a range of products. These platforms cover a range of duties. The platforms usually fall into one of three categories- hydraulic systems, geared systems (systems using a gearbox), and gearless systems (system whiteout a gearbox).

One of the difficulties with the platforms is they usually address very different segments of the market (performance, price, and options) and can create very different architectures. The architectures are not usually interchangeable between platforms. This can cause significant price or performance variations between market segments and can force a customer into a platform that does not meet their particular needs. This can be especially true if a customer has a need that borders two platforms and all decisions and concessions can make one of the available products infeasible. It can be very difficult to define the design parameters, as the exact operational requirements will vary from building to building.

The design decisions must also consider the many systems and people for which it interacts. This can include the riding passengers, freight, traveling robots (common in hospitals), other elevators (group control), building system managers (software/hardware systems to monitor & control building systems), service personnel, the building, power supplies, emergency equipment, emergency personnel (firefighters, medical, etc.), and others, as well as the interfaces that occur with the elevator system itself. Each of these brings separate requirements and additional considerations for the system architect

Therefore, it is highly desirable to create a system architecture that meets system performance requirements, but for which the interfaces are standard and optimal. This can create a more versatile architecture for which components of platforms are interchangeable as the needs arise. It can also create a situation where the design groups can design sub-systems that, if they maintain the interface architecture, can be designed to replace or update technology without having to make changes to other parts of the system.

The Design Structure Matrix

The Design Structure Matrix is a useful tool for representing the interactions between different elements. These elements can take the form of physical components, design teams, systems, design parameters or any other items where an interaction or interface occurs. The interaction typically takes the form of energy, spatial, material, or information (Pimmler 7). The DSM also shows direction of flow. Interactions below the diagonal indicate feed-forward interactions and above the diagonal interactions indicate feedback. Feedback and feed-forward are especially important for time or decisions based sequences. For more detailed information on the DSM see the Pimmler and Eppinger paper or go to the Massachusetts Institute of Technology (MIT) DSM web page[iii].

As an example DSM is shown in Table 1. This example will discuss sequencing but it is not used in this research project. If the DSM contained design parameters then sequencing would be important, as the order of completion would be important. For the purposes of this paper sequencing is not considered as crucial as is the knowledge of where the interfaces should be optimally designed. The locations of interfaces are the primary parameters regardless of the direction of the interaction at the interface.

There are several types of interactions shown in this example. The diagonals do not have any significance, except that an element interacts with itself. All diagonals usually have an entry. Interaction BC represents an interaction where Element-B provides something to Element-C. If sequencing were being considered, Interaction-BC would be feed-forward. Interaction-DB represents an interaction where Element-D provides something to Element-B and Interaction-BA represents an interaction where Element-B provides something to Element A. If sequencing were being considered, Interaction-DB and Interaction-BA would represent feedback. Interaction EA represents the condition where Element-E passes something to Element-A and interaction AE represents the condition where Element-A passes something to Element-E. If sequencing were considered, they represent feedback and feed-forward respectively. Together they may represent a coupled interaction where the two elements are dependent on or interact with each other.

		From				
		A	B	C	D	E
To	A	X	BA			EA
	B		X		DB	
	C		BC	X		
	D				X	
	E	AE				X

Table 1: Example DSM Interactions

The entries of the DSM can represent the strength of the interaction. For example, for each of the interactions in Table 1 (AE, BC, DB, EA), the value placed in the DSM could be 1 if we

[iii] http://web.mit.edu/dsm/

want to weight each interaction equally. An option to vary the interaction strength could possibly use 0.5 for a weak interaction, 1 for a normal interaction, and 2 for a strong interaction. The choice depends on what will be done with the DSM and how it affects the analysis. The DSM created for the elevator system started with 1's for all interactions, but later the interaction strengths were modified to represent weak and strong interactions. The details for the elevator DSM will be discussed later.

Previous Work

The methodology of creating a DSM is based on an explanations published on MIT's DSM web page and a MIT Master of Science Thesis by Qi Dong, "Representing Information Flow and Knowledge Management in Product Design Using the Design Structure Matrix". The method for creating a DSM from a functional-to-physical mapping is based on an idea generated by Qi Dong. The base mathematical clustering algorithm is developed from the clustering algorithm in the Fernandez thesis. Pimmler and Eppinger have suggested some additional ideas for clustering and varying interaction strengths in their paper. These previous works developed the foundation for the process that was followed to analyze system interface architecture and product development process.

Chapter 3 Creation of the DSM

Introduction

This chapter will cover the creation of the Design Structure Matrix from listing the elements to be included to entering the DSM interactions. It will include the different methodologies used to list the elements; a functional decomposition of the system, a check of the list of elements using scenarios, and a functional requirements to physical elements mapping. It will also cover the method for identifying the DSM interactions and then entering them into the DSM.

To create the DSM that was used in this process, an example system was chosen that represented a basic roped elevator system. This particular system was chosen because is was basically generic and was most representative of any elevator system. The DSM that was created is not complete, nor is it completely correct. Instead, this is an example DSM with an understanding that more work can be done to complete the DSM or make it more representative of an actual or desired system. This author took some liberty in creating the entries. Some functional requirements and physical implementations are generic and not representative of any particular system while other entries are representative of an actual elevator system. The idea was to create an example DSM that could represent all of the problems that we would look to analyze with the clustering routines. Because of this, some of the DSM entries are high-level system requirements and others are more detailed. If an actual system was being analyzed or developed it is expected that much more work would be dedicated to the creation of the DSM and its entries.

To create the DSM, a 5-step process was followed as listed below.

1. Functional Decomposition of high-level elevator requirements.
2. Scenario Analysis to verify and complete the functional requirements listing.
3. Functional-to-Physical Mapping
4. DSM Extraction from Functional-to-Physical Mapping
5. Enter the DSM Entries representing the interactions

Functional Decomposition

Assembling functional requirements through a basic decomposition of the high-level system requirements started the process of creating the data for the DSM. Due to time constraints not all requirements have been captured, but instead, an important subset was developed for the purposes of testing the feasibility of using clustering algorithms to aid in interface design. The process yielded the basic and most important functional requirements as observed in the outline presented in Appendix A. Other functional requirements were derived from analyzing possible operational scenarios. Existing architecture was also examined in an effort to capture some of the current interface designs.

Scenario Analysis

Examining operational scenarios is one approach for ensuring that all functional requirements have been captured. This approach was used to supplement the functional requirements listed through functional decomposition of the high-level system requirements. Several scenarios for normal operating modes and abnormal or failure modes were listed. Then the functional requirements that were obtained through functional decomposition were mapped to the operational scenarios. If the functional requirements did not adequately address the operational scenario, additional functional requirements were added so that it adequately covered the operational mode. Many of the additional functional requirements that were found through this process mainly dealt with code issues, failure issues, and some requirements that were derived from design decisions. A complete listing of the functional requirements and the scenario mapping can be found in Appendix B.

Another motivation behind the mapping of the scenarios was the possibility of examining how the clusters address the different scenarios. For example, it may not be optimal for a cluster to be built if only one element of the cluster was needed to address any scenario. If that were to occur there may be some cost associated with applying a cluster to an operating condition when the cluster did not efficiently address the scenario. Using the scenario to functional mapping and the functional to physical mapping, we may have been able to extract clusters for each of the scenarios. Therefore there may be some additional procedures or algorithms that could be developed to address clustering to meet operational scenarios. Unfortunately, there wasn't enough time to complete this part of the analysis. This may be something to consider for future research.

Functional to Physical Mapping

The next step in the process was to create a mapping of the functional requirements to their physical implementation. This process closely followed research work by Qi Dong and she was consulted as this work progressed. The main idea behind this process is that every functional requirement is answered by some physical implementation. Therefore, for each functional requirement a single physical element was listed. Sometimes this was difficult, as it appeared that there might be several physical elements that implement the functional requirement. Upon closer examination, it was found that the functional requirements that appeared to have more than one physical element to implement it, was actually made up of more than one functional requirement. In other words, the functional requirement could be broken down into several functional requirements so that there was a one to one mapping of functional requirements to physical elements. Because of this, this process also uncovered other functional requirements. Usually these additional functional requirements were derived requirements from the physical implementation that was being modeled in the mapping. The additional functional requirements are second and third tier functional requirements that result from a zigzag process of defining functional requirements, mapping the functional requirements to physical elements, which then get mapped to additional functional requirements, that get mapped down to more functional requirements, etc. The functional requirements listed in Appendix B contain all functional requirements that were listed either through functional decomposition, scenario analysis, or function to physical mapping.

Once the functional to physical mapping had been completed, it was time to place the entry marks into the matrix to represent the interactions. Although it should have been possible to enter the interactions in a functional-to-functional mapping or a functional-to-physical mapping, I found it very difficult to correctly identify all of the interactions. Instead, I moved directly to the next step of creating the physical-to-physical mapping (physical DSM) and then entered the interactions. The mapping of the functional requirements to physical elements can be found in Appendix C. Although this mapping was created, there was not sufficient time to use the functional-to-physical mapping to evaluate the clustering or the resulting architecture. The entries developed in the physical-to-physical mapping were transferred to the functional-to-physical mapping since this process assumed that every functional requirement could be replaced by its physical implementation. If time had allowed, additional analysis of the clusters using the functional-to-physical mapping could have been observed.

The next step in this process was to replace all of the functional entries on the matrix with their corresponding physical element. This resulted in a matrix with the physical elements on both axes of the matrix. This then represented the DSM that was going to be used. After creating the DSM axes, the physical interactions were documented. A generic mapping of the DSM without interaction strengths can be found in Appendix D. This base DSM served as the foundation for entering the interaction strengths.

Enter the DSM Interactions

Initially in this process, all types and strengths of interactions were entered into the DSM with equal weight. The DSM interactions included power, communications, control, safety, and a category called other. These do not match the classical categories of energy, material exchange, and information[iv]. In any case, the DSM treated all interactions the same. The reasoning behind this is that an interface must be managed without much regard for the type of interaction that is

[iv] As specified in the Pimmler and Eppinger paper

taking place. A design team or a subsystem must manage all interactions regardless of the type. All interactions have cost associated with them.

To simplify the process, all interaction strengths were initially weighted equally. For example, all X's in the DSM in Appendix D were initially replaced with 1's. This allowed the analysis of the clustering algorithm and a check of the results without complicating the process because of varying interaction strengths. Once the process and algorithm had been completed, the interaction values were then modified to acknowledge the different strengths of the interactions. A strong interaction received a 2 and a weak interaction received a 0.5. For the purposes of this research, a strong interaction was defined as an interaction that was highly important, critical to operation, or was not optional. Weak interactions were defined as interactions that were not important to the design of the system, interactions that were a result of design choices but may be modified, or interactions that were not critical to the operation of the system. As will be discussed in Chapter 6, the clustering algorithm yielded better results when the strength of the interaction was accounted for in the analysis.

As previously discussed, all types of interactions were treated the same. An energy interaction was treated the same as a communication interaction. If time had allowed, further analysis may have included clustering on different types of interactions or some type of combined clustering. The combined clustering would take into account all types of interactions and would recognize that an element could interact with another element in different ways. The cost of an interaction between two elements would be higher if they shared more than one interaction type rather than the current method of equal interaction weights regardless of the number of interactions taking place between the two elements. For this research project, all interaction types were treated equally.

Chapter 4 Original Clustering Algorithm

Original Algorithm Background

This analysis was begun with the basic clustering algorithm developed in the Fernandez thesis. For a detailed explanation of the algorithm, see the thesis by Fernandez. Developing co-located design teams was the primary purpose for the clustering algorithm developed in Fernandez's research. That is slightly different than the intent of this research, but the concepts are the same. The algorithm by Fernandez will be modified in Chapter 5 for real-world complex problems as applied to this elevator example.

The original algorithm was written in C code, compiled and linked to Microsoft® Excel. The original algorithm consisted of several steps.

1. Each element is initially placed in its own cluster
2. Calculate the Coordination Cost[v] of the Cluster Matrix
3. Randomly choose an element
4. Calculate bid from all clusters for the selected element
5. Randomly choose a number between 1 and rand_bid (algorithm parameter)
6. Calculate the total Coordination Cost if the selected element becomes a member of the cluster with highest bid (use second highest bid if step 5 is equal to rand_bid)
7. Randomly choose a number between 1 and rand_accept (algorithm parameter)
8. If new Coordination Cost is lower than the old coordination cost or the number chosen in step 7 is equal to rand_accept, make the change permanent otherwise make no changes
9. Go back to Step 3 until repeated a set number of times

There was also simulated annealing in the algorithm to avoid getting stuck in a local optimum when there may have been a better global optimum. The simulated annealing worked by making a change without using data to determine of the change is beneficial or not. This was accomplished in step 6 by randomly (1 out of N times[vi]) taking the second highest bid rather than the highest bid. The second part of the simulated annealing was to randomly (1 out of M[vii] times) accept the change even if the coordination cost was not improved in step 8. N and M were parameters of the clustering algorithm.

The bid was calculated with the following formula. Note that the bid is calculated for the randomly chosen element in step 3.

For the element chosen in step 3, get a bid from each cluster j such that

$$\text{ClusterBid}_j = \frac{(\text{inout})^{\text{powdep}}}{(\text{ClusterSize}_j)^{\text{powbid}}}$$

where:		
	j	= cluster number
	ClusterBid_j	= Bid from Cluster j for the chosen element
	inout	= sum of DSM interactions of the chosen element with each of the elements in cluster j
	powdep	= exponential to emphasize interactions
	powbid	=exponential to penalize size of the cluster

The Coordination cost was calculated with the following formula. The Coordination Cost is calculated using the DSM matrix and the Cluster matrix that defines which elements are in each of the clusters.

For an interaction between element j & k that occur <u>within a cluster</u>

$$\text{IntraClusterCost} = (\text{DSM}(j,k) + \text{DSM}(k,j)) * \text{ClusterSize}(y)^{\text{powcc}}$$

[v] Coordination Cost is the calculated value of the objective function of the optimization routine

For and interaction between element j & k that occurs <u>outside of a cluster</u>

$$\text{ExtraClusterCost} = (\text{DSM}(j,k) + \text{DSM}(k,j)) * \text{DSMSize}^{\text{powcc}}$$

$$\text{TotalCost} = \sum \text{IntraClusterCost} + \sum \text{ExtraClusterCost}$$

where:

TotalCost	=	Coordination Cost
IntraClusterCost	=	Cost of interaction occurring within a cluster
ExtraClusterCost	=	Cost of interaction occurring outside of any clusters
DSM(j,k),DSM(k,j)	=	DSM interaction between element j & k
ClusterSize(y)	=	Number of elements in the cluster y
DSMSize	=	Number of elements in the DSM
powcc	=	penalizes the size of clusters

These equations served as the basis and starting point for the analysis and clustering of the elevator DSM.

Matlab® Tools

As previously discussed, the clustering algorithms had been written in C-code, compiled and linked to Microsoft Excel. I have instead chosen to implement the algorithms in the mathematical package Matlab. I have chosen this package for several reasons. I have extensive experience with using Matlab and writing routines and analysis packages within the Matlab environment. There is no compilation of code; therefore changes to the algorithms could be made without having to recompile. Matlab is also designed to work with and manipulate matrices very easily. Matlab routines can work on any computing platform that has Matlab installed. Matlab also has capabilities to link with Microsoft Excel and it is possible to design

[vi] N was a settable parameter
[vii] M was a settable parameter in the algorithm

graphical user interfaces. Matlab also has extensive graphical capabilities that could be used in the analysis. For these reasons, all manipulation, calculations, and analysis have been implemented in the Matlab package.

In addition to the implementation of the clustering algorithms in Matlab, several Matlab routines were created that would graphically represent the data. This eased the task of analyzing and reporting the results.

Parameter Application to Elevator DSM

The clustering algorithm was then applied to the elevator DSM that had all interactions valued at 1. The initial run had the parameters set at their default values and then they were adjusted to get an appropriate level of clustering. The process of getting suitable parameters was done on a trial and error basis. This process of trial and error allowed the user to get a better understanding of the clustering parameters but it became frustrating as each run could take approximately 5-10 minutes[viii] to complete.

The parameters' default values and final values are shown in Table 2. The default values are those suggested by the Fernandez thesis. The final values are the parameters that appeared to give the best results for the elevator DSM. The final parameters were also used for the modified algorithm in Chapter 5.

[viii] Calculations performed on a Laptop with a 233MHz processor & 64M RAM

Parameter	Default	Final
pow_cc	1	1
pow_bid	0	1
pow_dep	1	4
max_cluster_size	61	61
rand_accept	30	122
rand_bid	30	122
times	2	2
stable_limit	2	2

Table 2 : Clustering Parameter Values

The parameter **pow_cc** penalized the size of the cluster in the cost calculation. When the value was increased from the default level of 1, it had a minimal effect on reducing the maximum cluster size. Therefore the best results were obtained with pow_cc set at 1.

The parameter **pow_bid** was initially set at zero, which resulted in the size of the cluster not being penalized during the bidding process. Therefore a large cluster has as much weight during the bidding process as a small cluster if the interaction values were equal. Because of the exponential nature of the bidding algorithm, if pow_bid was set higher than 1, the clustering algorithm produced many small clusters because the larger clusters were penalized so high. The best results were obtained with pow_bid set at 1.

The parameter **pow_dep** emphasized high interactions during the bidding process. When pow_dep was set at the default value of 1, there was marginal emphasis placed on the interactions. This became important as the size penalty (pow_bid) was increased. When pow_dep was set at lower values, the cluster algorithm was more likely to produce results with many small clusters (especially when the size of the cluster was penalized). The best results were produced when pow_dep was set to 4. For the elevator DSM, there didn't appear to be much of a difference in the results if the value was set higher than 4, but lower numbers produced small clusters when the size penalty was used. Therefore, the final value for pow_dep was set at 4.

The simulated annealing part of the algorithm used parameters (**rand_bid, rand_accept**) that specified how often the algorithm would make a less than optimal change. The parameter rand_bid specified how often to accept the bid from the second highest bidder instead of the highest bidder. The algorithm worked by randomly (approximately 1 out of rand_bid times) accepting the bid of the second highest bidder instead of the highest bidder. The parameter rand_accept worked similarly. The algorithm randomly (1 out of approximately rand_accept times) would make a change by making an element a member of the highest bidding cluster even if the change did not improve the calculated objective cost. I started the evaluation with the parameters set at ½ of the DSM size. This setting would cause the algorithm to make less than optimal changes after choosing approximately half of the elements in the DSM. After several runs it became apparent that such frequent changes resulted in the algorithm taking a long time to run and the final result was rarely better than when the changes were made less often. Because of the nature of this algorithm, it was possible for the simulated annealing changes to result in final solutions that were not as optimal as a solution that may have been found prior to the change. This problem will be discussed further in the next section, but the final outcome was that the clustering algorithm worked better if the simulated annealing occurred less often. Therefore, the final parameters were set to twice the size of the DSM.

Times and **Stable_limit** were set at 2. These parameters specified how long the algorithm would run before it reported the results. Times specified the number of times (times*DSM Size) the algorithm will pick a new element before checking for stability. Stable_limit specifies the number of times it must loop though the process without making a change. Therefore the algorithm will have to loop through the process at least Times*DSMsize*Stable_limit without making a change before it finishes. The value of 2 appeared to be sufficient to find an optimal solution. Setting these parameters higher than the default level caused the algorithm to take longer to run and didn't appear to produce better results. Therefore, the values were finally set at two.

To illustrate some of the problems, an intermediate trial run is contained in Figure 2 through Figure 6. The parameters for this run were as follows:

pow_cc	1
pow_bid	0
pow_dep	1
max_cluster_size	61
rand_accept	30
rand_bid	30
times	2
stable_limit	2

Figure 2 contains a graphical representation of the original DSM before clustering. Figure 3 contains a graphical representation of the cluster matrix after clustering. Figure 4 contains a graphical representation of the DSM after clustering has been applied. Figure 5 contains a textual list of the physical elements that are members of each of the clusters as well as information pertaining to their location in the original DSM and whether or not they members are members of more than one cluster. Figure 6 shows a graphical representation of the calculated coordination cost as the algorithm progressed to find an optimal clustering solution.

As can be seen by the run in Figure 4 there is a cluster that contains about a third of the DSM elements. For the elevator DSM, I was looking for cluster sizes that contained about 6-10 items. The desire is based on the size of the cluster versus the DSM size, knowledge of the system and its elements, and the desire to keep the number of elements to something that would be easy to understand. A cluster with 20 or more elements is difficult to manage and will contain elements that are not highly integrated but are only indirectly connected through other elements.

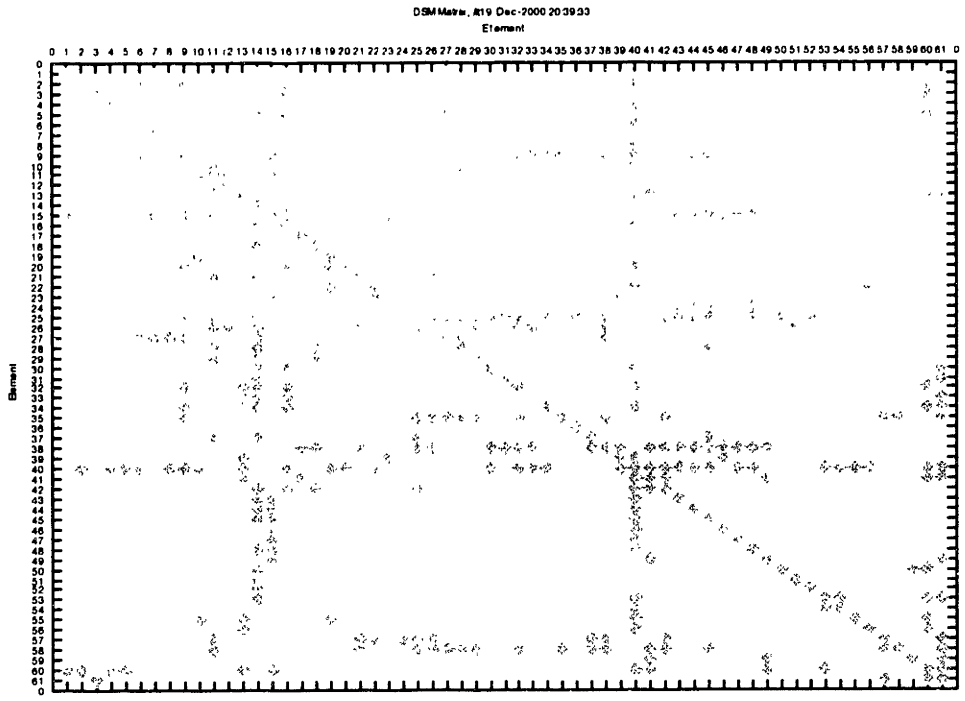
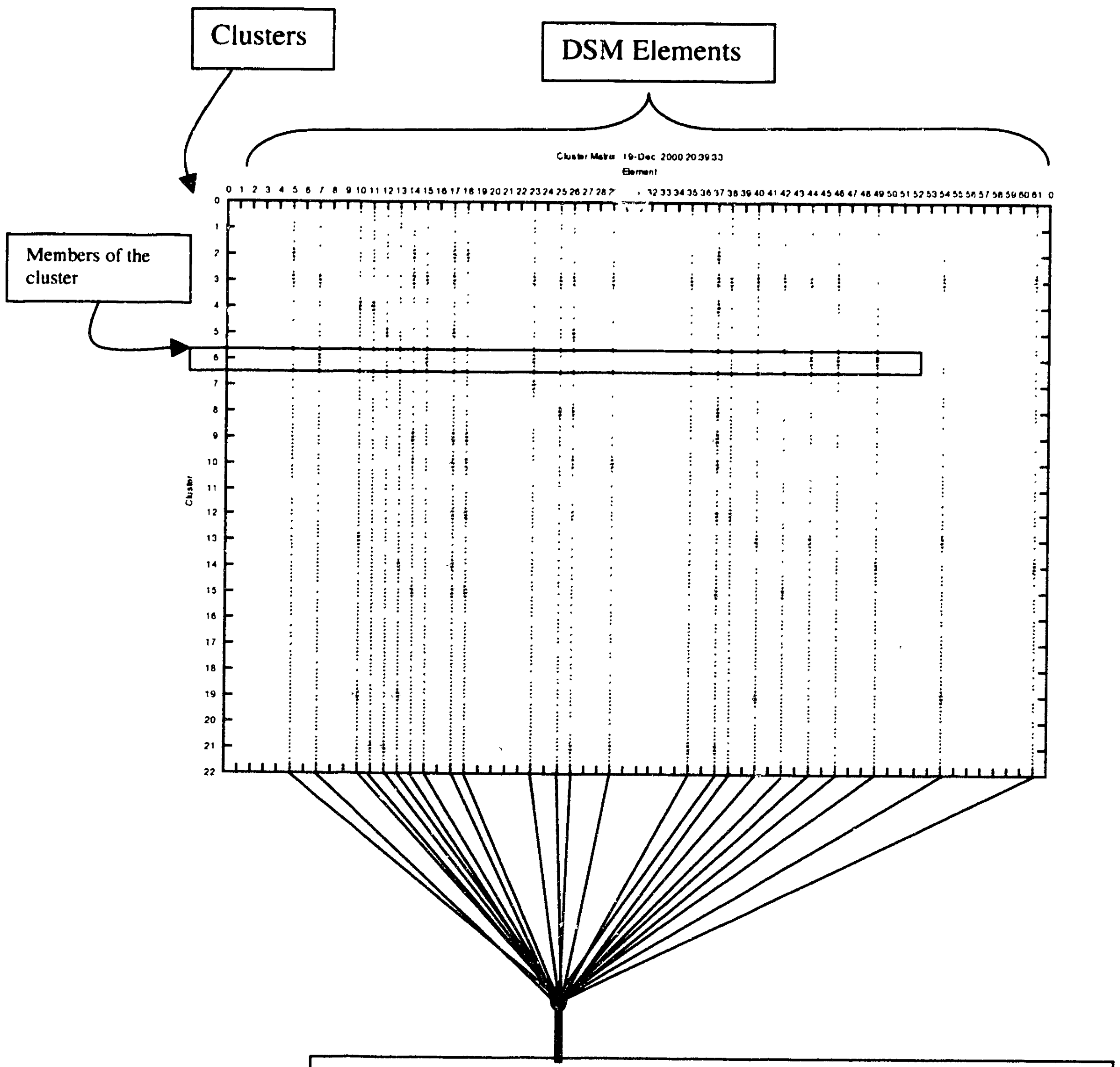


Figure 2: Design Structure Matrix Interactions



Vertical Lines indicate elements that appear in more than one cluster.

Many of the elements appear in more than one cluster and this was addressed in the modified clustering algorithm.

Figure 3: Foundation Algorithm Cluster Matrix - Element to Cluster Mapping

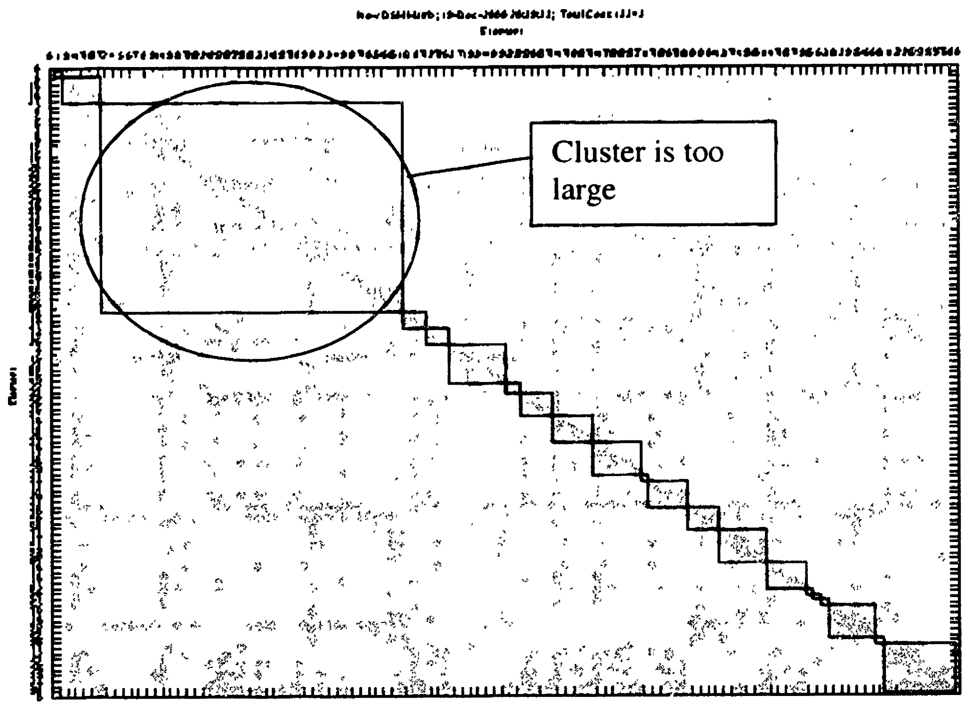
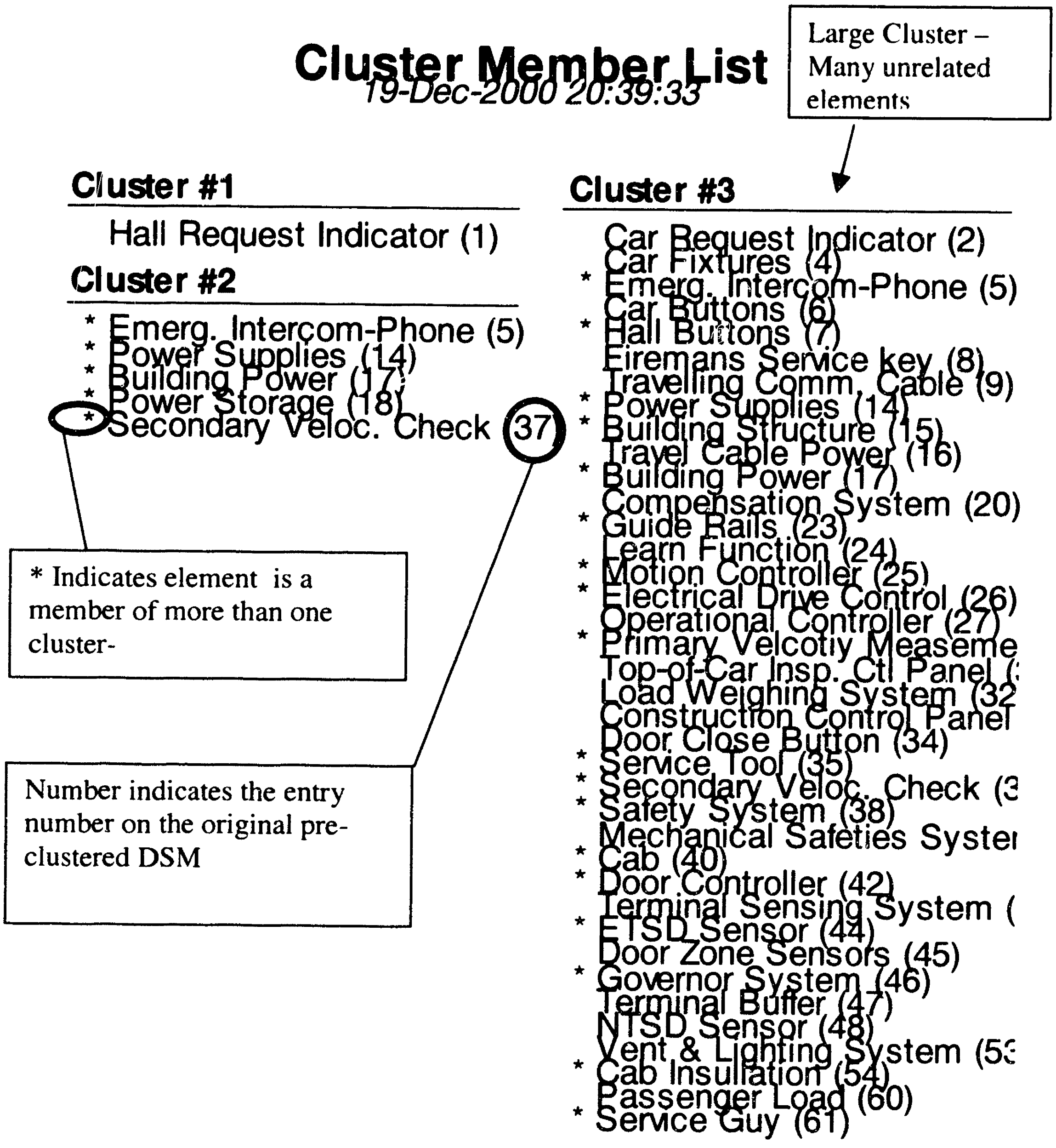


Figure 4: Foundation Algorithm - Clustered DSM



Cluster Member List
19-Dec-2000 20:39:33

Large Cluster – Many unrelated elements

Cluster #1

Hall Request Indicator (1)

Cluster #2

* Emerg. Intercom-Phone (5)
* Power Supplies (14)
* Building Power (17)
* Power Storage (18)
* Secondary Veloc. Check (37)

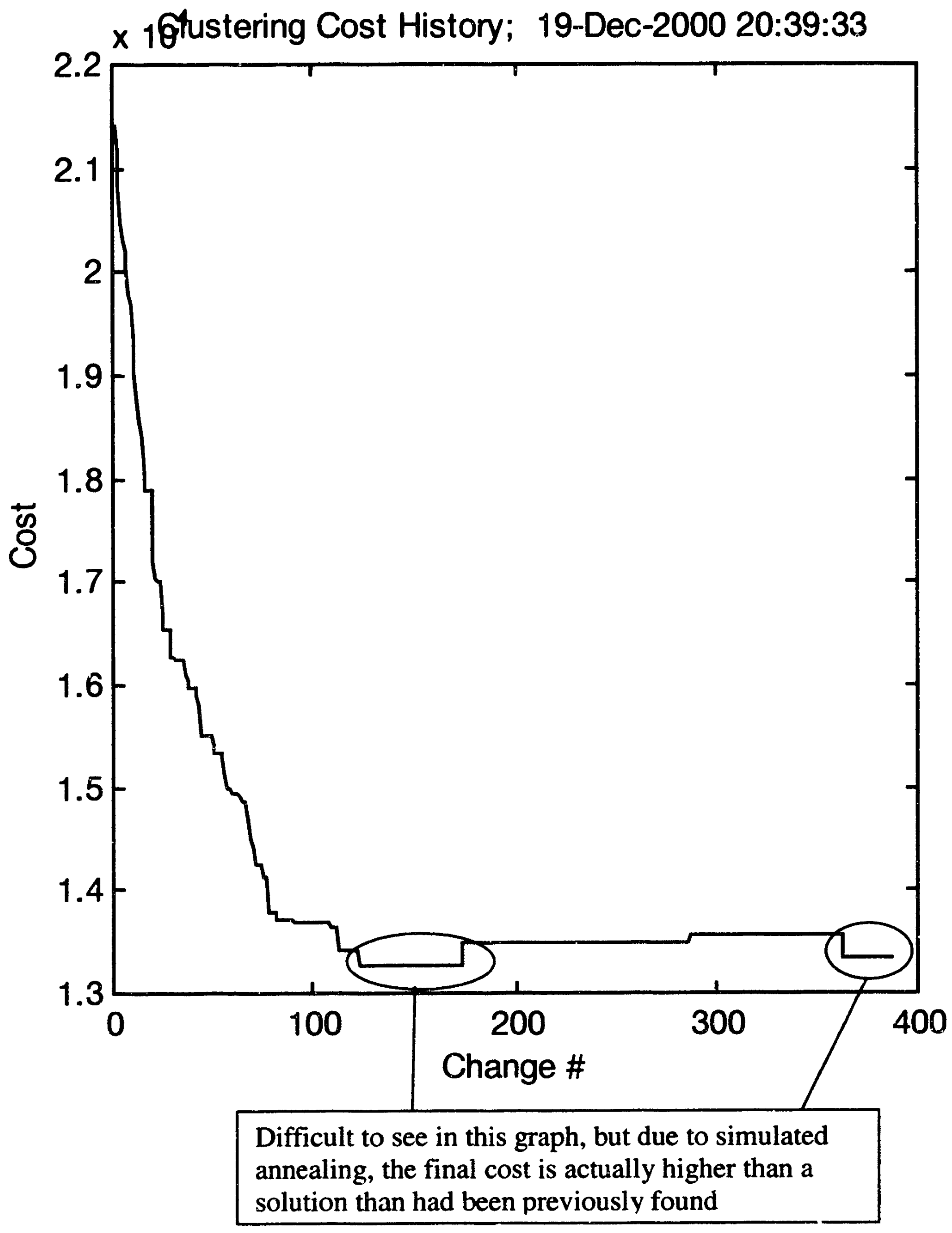
Cluster #3

Car Request Indicator (2)
Car Fixtures (4)
* Emerg. Intercom-Phone (5)
Car Buttons (6)
* Hall Buttons (7)
Firemans Service key (8)
Travelling Comm. Cable (9)
* Power Supplies (14)
* Building Structure (15)
Travel Cable Power (16)
* Building Power (17)
Compensation System (20)
* Guide Rails (23)
Learn Function (24)
* Motion Controller (25)
* Electrical Drive Control (26)
Operational Controller (27)
* Primary Velcotiy Measeme
Top-of-Car Insp. Ctl Panel (
Load Weighing System (32
Construction Control Panel
Door Close Button (34)
* Service Tool (35)
* Secondary Veloc. Check (3
* Safety System (38)
Mechanical Safeties Syste
* Cab (40)
* Door Controller (42)
Terminal Sensing System (
* ETSD Sensor (44)
Door Zone Sensors (45)
* Governor System (46)
Terminal Buffer (47)
NTSD Sensor (48)
Vent & Lighting System (5
* Cab Insulation (54)
Passenger Load (60)
* Service Guy (61)

* Indicates element is a member of more than one cluster-

Number indicates the entry number on the original pre-clustered DSM

Figure 5: Foundation Algorithm- Cluster List Example



Difficult to see in this graph, but due to simulated annealing, the final cost is actually higher than a solution than had been previously found

Figure 6: Foundation Algorithm Cost History Example

Results of Original Algorithm

This initial clustering algorithm appeared to have clustered like elements. The size of the clusters and the number of clusters depended largely on the values of the algorithm parameters. To get clustering appropriate to the system being analyzed, a lengthy trial and error process of changing parameters and running the algorithm had to be undertaken. Although the effect of the parameters is predefined, it is probable that the size of the DSM and how integrative the elements of the DSM are affect the results. Although time consuming, the process requires that the user become familiar with the clustering algorithm and the DSM under consideration.

There were some concerns with the results of the clustering algorithm and the fixes to these problems will be addressed in Chapter 5. Two major problems were observed at this point in the process: 1) many of DSM elements belonged to more than one cluster, 2) simulated annealing caused the final solution to be worse than a previously found solution.

When the clustering algorithm places many of the elements in multiple clusters, the meaning and usefulness of the cluster is diminished. If the element is in more than one cluster, it forces interfaces between all of these clusters on multiple levels. We would like an element to be placed only with other elements that it is most like. This is desirable for two reasons. First, by minimizing cluster membership, we minimize the interfaces between clusters in the physical context. Secondly we minimize the information flow between teams in the organizational context. Therefore it is highly desirable to minimize multiple cluster membership.

Multiple memberships also resulted in clusters that contained elements with very different needs and interface requirements. To evaluate the feasibility of a cluster, I made an attempt to name each cluster based on the members of the cluster. This became very difficult, as cluster members did not directly interface with each other. There were many cluster members that were from very different parts of the system with weak interconnectivity. Although it was possible to see the link between the members of a cluster, the identity of the cluster was weak. For example, a cluster may contain the power supply, the building structure, safety system and the hallway fixtures. The building structure and the power supply were linked to the hallway fixture even though they do not interface with each other. The safety system was only linked to the power supply and to no other member of this particular cluster. This made this cluster very difficult to manage as it had very different requirements and the cluster was sparsely connected between all of its members. In addition to this cluster, the power supply, the building structure, the safety system, and the hallway fixtures were all members of other clusters where their interface requirements were much stronger. This type of clustering result was significantly improved in the modified clustering algorithm. The modified algorithm forces clusters to contain members with a high degree of connectivity and it significantly reduces or eliminates multiple cluster membership.

The second significant problem was related to the simulated annealing. The simulated annealing caused the clustering algorithm to make less than optimal changes. In the bidding part of the algorithm, the routine would randomly select the second highest bid instead of the highest bidder. This would result in less than optimal changes if the second highest bid also improved the coordination cost.

The problem related to the simulated annealing actually resulted from the part of the algorithm that decides whether or not to make a change. In this part of the algorithm, a change would be made (1 out of rand_accept times) even if the change did not result in an improvement in the coordination cost. The purpose of this part of the algorithm was to prevent the algorithm from getting stuck in a local optimum. Unfortunately, the when the change was made, the algorithm did not check that the change finished at a coordination cost that was better than anything that had already been found. Therefore, the majority of the runs resulted in a final configuration that had a higher total coordination cost than a solution previously found.

Chapter Summary

The initial clustering algorithm developed in the Fernandez thesis was evaluated for use in developing the elevator system architecture. First, the algorithm was coded into the Matlab environment because of Matlab's capabilities of working with matrices and its significant data analysis capabilities. Then the algorithm parameters were developed to obtain clustering results that were appropriate for the elevator problem. The algorithm results were promising but did not provide consistent clusters. There were two significant problems. First, the DSM elements were often members of more than one cluster, which made it difficult to understand how the element could be managed or designed appropriately if it was a member of more than one cluster. Second, the final solution that was reported by the clustering algorithm was not usually the best solution that had been found by the algorithm. These two problems will be addressed and corrected in the next chapter.

Chapter 5 Modified Clustering Algorithm

Introduction

The results of the initial clustering algorithm developed in Chapter 4 did not provide an adequate answer for developing a system architecture. There were two main problems. First, the clustering results often assigned elements to more than one cluster. This can be confusing when developing the system architecture or assigning the element to a design group. This will be addressed by adding a cost penalty to the assignment of an element to more than one cluster. The second problem to be addressed is with the solution that is reported by the clustering algorithm. The algorithm would often report a solution that was not the most optimal solution that had been found. This is caused by the simulated annealing portion of the algorithm and will be address by forcing the algorithm to save the best solution that it has found at any time.

Penalize Cost of Multiple Cluster Memberships

The first modification to the clustering algorithm was to prevent the algorithm from placing a large number of elements in multiple clusters. A few different schemes were evaluated for modification of the algorithm. The first scheme was to force an element to be a member of only one cluster. In this scheme, after the bids had been made and calculated, the coordination cost would be calculated for two different configurations. In the first configuration, the coordination cost would be calculated for the configuration where the element was a member of its current cluster. Then the coordination cost would be calculated for the configuration where the element was a member of only the new high bidder. The two coordination costs would be compared and the lowest cost configuration would be chosen. In this scheme the element is only a member of its current cluster or the new cluster. This would ensure single membership in the clusters. After reviewing the alternatives, the first scheme was not implemented in order to pursue the following scheme.

The second scheme, which has been implemented, was to develop a modification that would remain consistent with how the algorithm evaluates changes and clusters. The modification would penalize, or add cost, to a solution that made an element a member of more than one cluster. This was accomplished by recreating the DSM after every modification. The DSM would have en entry for an element every time the element showed up in a cluster. This modification was also consistent with the method chosen to graphically represent the clustered DSM. For example, the DSM in Figure 2 is the original DSM. The clustered DSM in Figure 4 appears to be much more cluttered because every element has an entry for each cluster of which it is a member. If an element were a member of 3 clusters, the element would be represented in the new DSM 3 times, once for each cluster. Therefore, each additional entry of the element increases the size of the DSM and the number of entries within the DSM is increased. This increase in the DSM size, and the additional DSM entries, increase the cost calculation. With this new approach for costing multiple cluster memberships, it is still possible for an element to be a member of multiple clusters, but it does so with a cost penalty if it is not an optimal solution. With this modification to the algorithm, the routine is still trying to minimize the objective function but because we believe the multiple memberships is more costly to implement or manage, it has been represented as such in the cost calculations.

This change immediately improved the results of the clustering algorithm. The results shown in Figure 7 through Figure 9 have been run with the same DSM and with the parameters previously listed as the final parameters in Table 2. Notice that this has significantly reduced the number of multiple cluster memberships. In Figure 7, only one element is a member of multiple clusters. In Figure 8, we can notice that the clustered DSM is much cleaner than what was observed in Figure 4 before the modification. The cluster sizes are also closer to what was desired. This change was a significant improvement, but as can be seen in Figure 9, the final solution was not the run's best solution due to the simulated annealing. The simulated annealing caused the final solution to be less optimal than one of the intermediate solutions. This will be addressed in the next section.

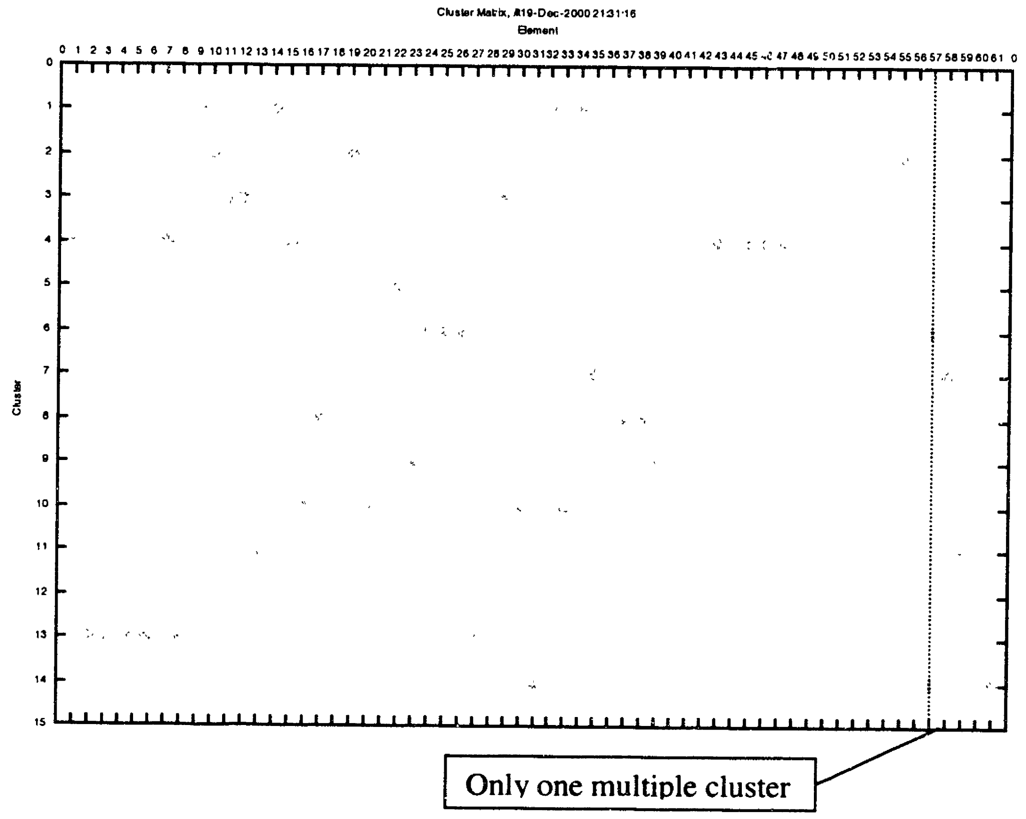


Figure 7: Cluster Matrix with Multiple Membership Penalties

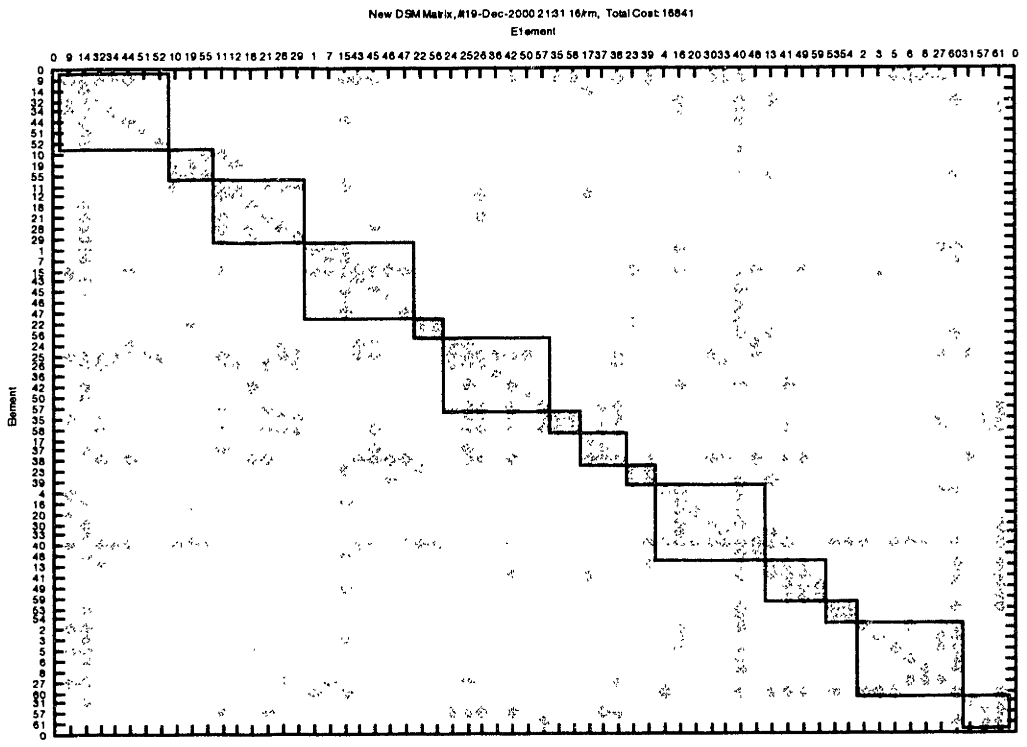


Figure 8: Clustered DSM with Multiple Cluster Membership Penalty

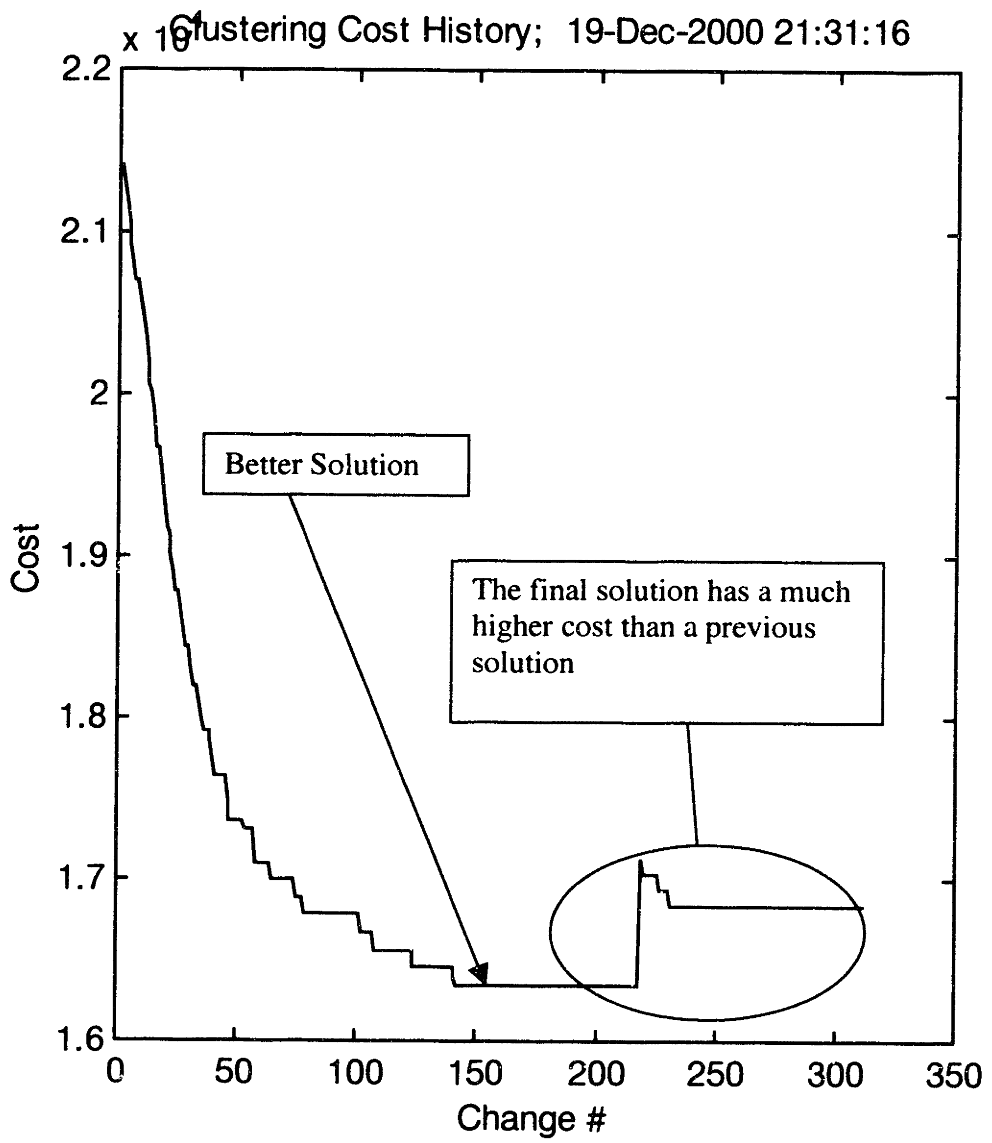


Figure 9: Cost History with Penalty for Multiple Cluster Membership

Save the Best Solution

As observed with the prior examples, the final clustering solution may not be the best solution. In both of the previous examples (Figure 6 & Figure 9), the final solution had a higher coordination cost than some other solution found during the run. The simulated annealing algorithm attempts to ensure that a solution has not gotten stuck in a local optimum by randomly making changes even if there is no improvement in the solution. It is ironic that because of the simulated annealing, the final solution is actually a solution that gets stuck at a local optimum that is worse than a solution that it has previously found.

Once the algorithm made the change that resulted in increasing the total cost, the information about that lower cost was lost and basically forgotten. This was a basic flaw in the flow of the algorithm. To solve this problem, the algorithm can save information about any best solution that has been found. Then, before any change is made, if the change is going to increase our total coordination cost, the algorithm saves two pieces of information. The first piece of information is the value of the lowest cost solution that has been discovered. The second piece of information is the cluster matrix that produced the best solution. When the algorithm thinks that a solution has been found, it then compares the cost of the current solution with the cost of the best solution. If the current solution cost is equal to or lower than the best solution cost then the run is over and it reports the current solution. If the current solution is greater than the best solution, then the algorithm replaces the current solution with the best solution and continues to try to optimize the clusters (beginning with the best solution). The reason for continuing is that the simulated annealing process may have interrupted the best solution before the optimization process had finished. Once the solution has a final solution that is less than or equal to any best solution, the run stops and reports the results.

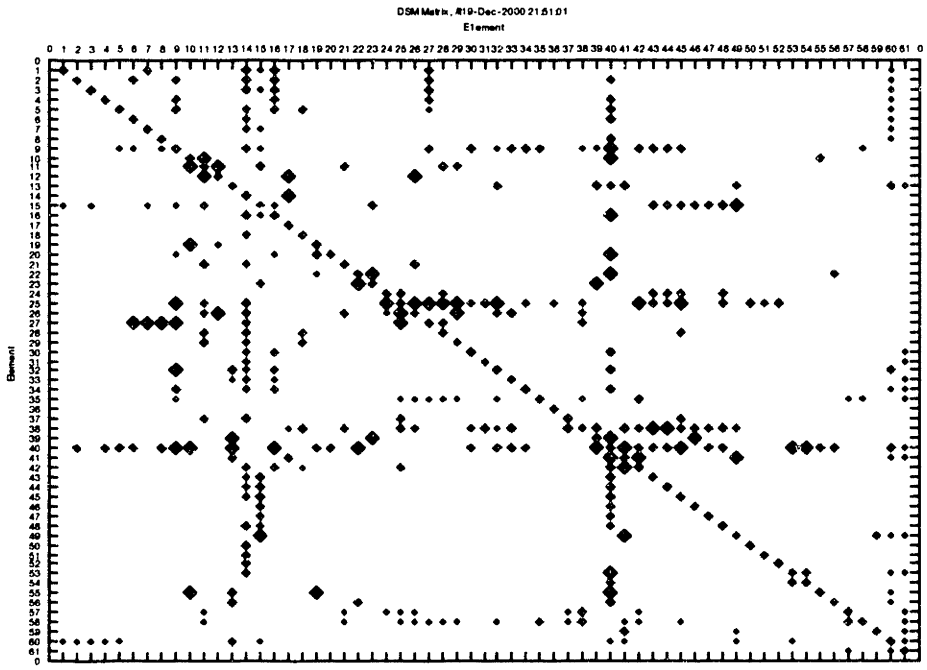
Also, because of the simulated annealing it may be possible to get into a loop where the algorithm goes back to the best solution and then jumps to a high cost solution, thinks it is done but has to return to the best solution again. Because of this possibility, the algorithm has a parameter to set how many times it can think it is done but have to loop back. If the algorithm reaches this limit, then the reported solution is the best solution that had been found up to that point.

This change did result in reporting the best solution that the algorithm was able to find. It also, however, increased the time to run the algorithm because it sometimes had to go back to an intermediate solution and continue the optimization process. The typical run time increased from 5-10 minutes[ix] per run to 10-30 minutes per run. Figure 13 contains an example that shows the cost history jumping up and then jumping back down because it did not find a better solution.

Once all the changes were made, the new clustering routine was run on the DSM with the varied interaction strengths (see Chapter 3, section on "Enter the DSM Interactions" for explanation of varied interaction strengths). The added value of the varied interaction strengths is that the clustering algorithm will weigh strong interactions significantly more than standard or weak interactions because of the exponential nature of the equations. Because the bidding and costing algorithms (discussed in Chapter 4) use the DSM entries (interactions strengths) in their

[ix] Calculations performed on a laptop with 233MHz processor

calculations, the interaction strengths affect the final cost totals calculated in these algorithms. As will be discussed in the next Chapter on the cluster analysis, the results of the clustering algorithm are better and more consistent with the varied interaction strengths. An example clustering result using the new algorithm is shown in Figure 10 to Figure 13.



The size of the entries in the DSM corresponds to the interaction strength.
Large =2, Medium = 1, Small = 0.5

Figure 10: Elevator DSM with Varied Interaction Strengths

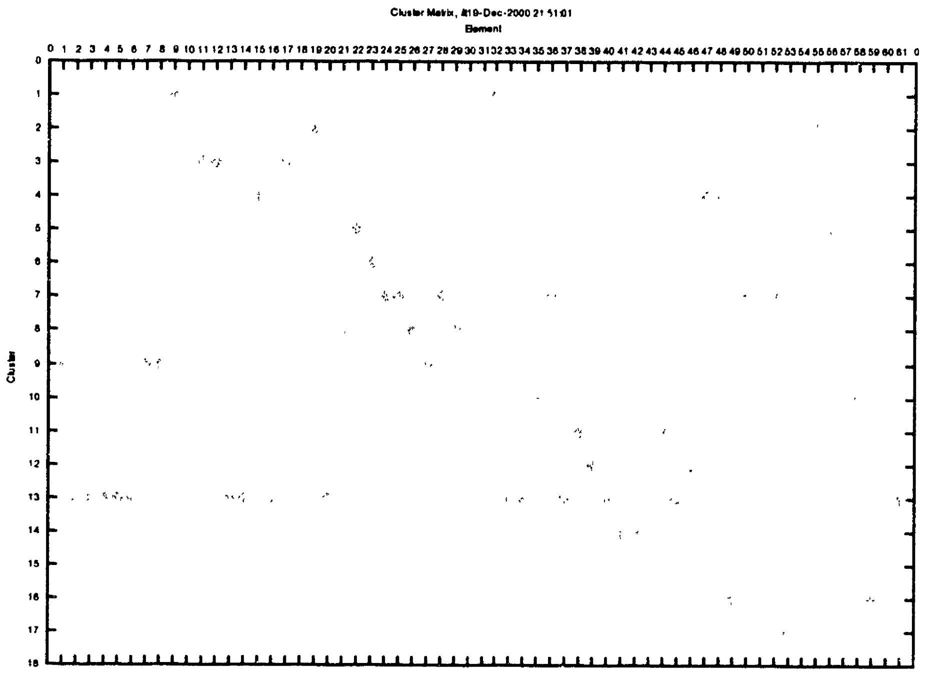


Figure 11: Cluster Matrix of DSM with Varied Interaction Strengths

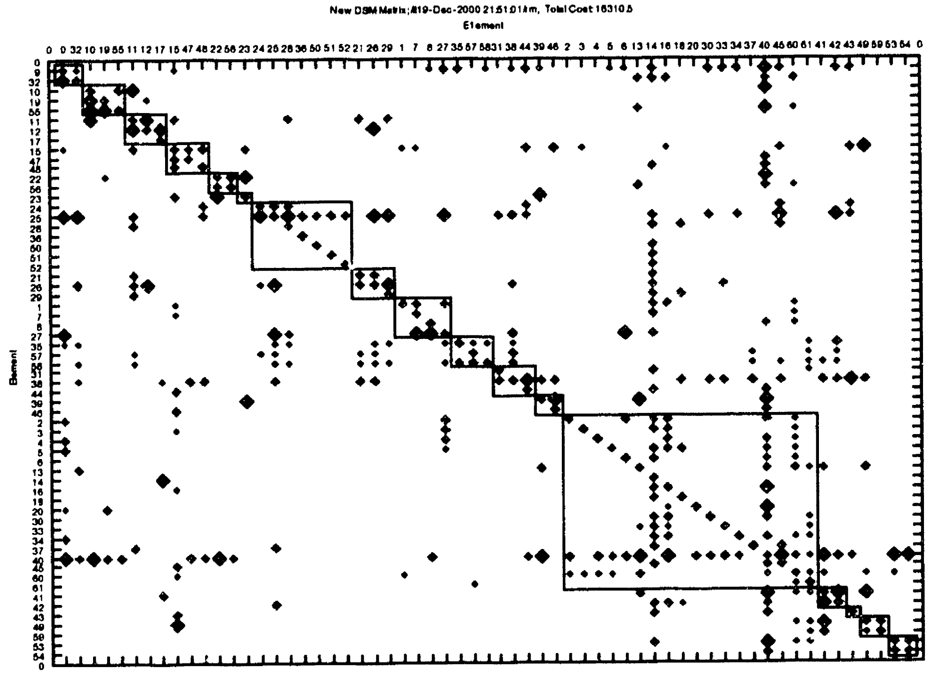


Figure 12: Clustered DSM with Varied Interaction Strengths

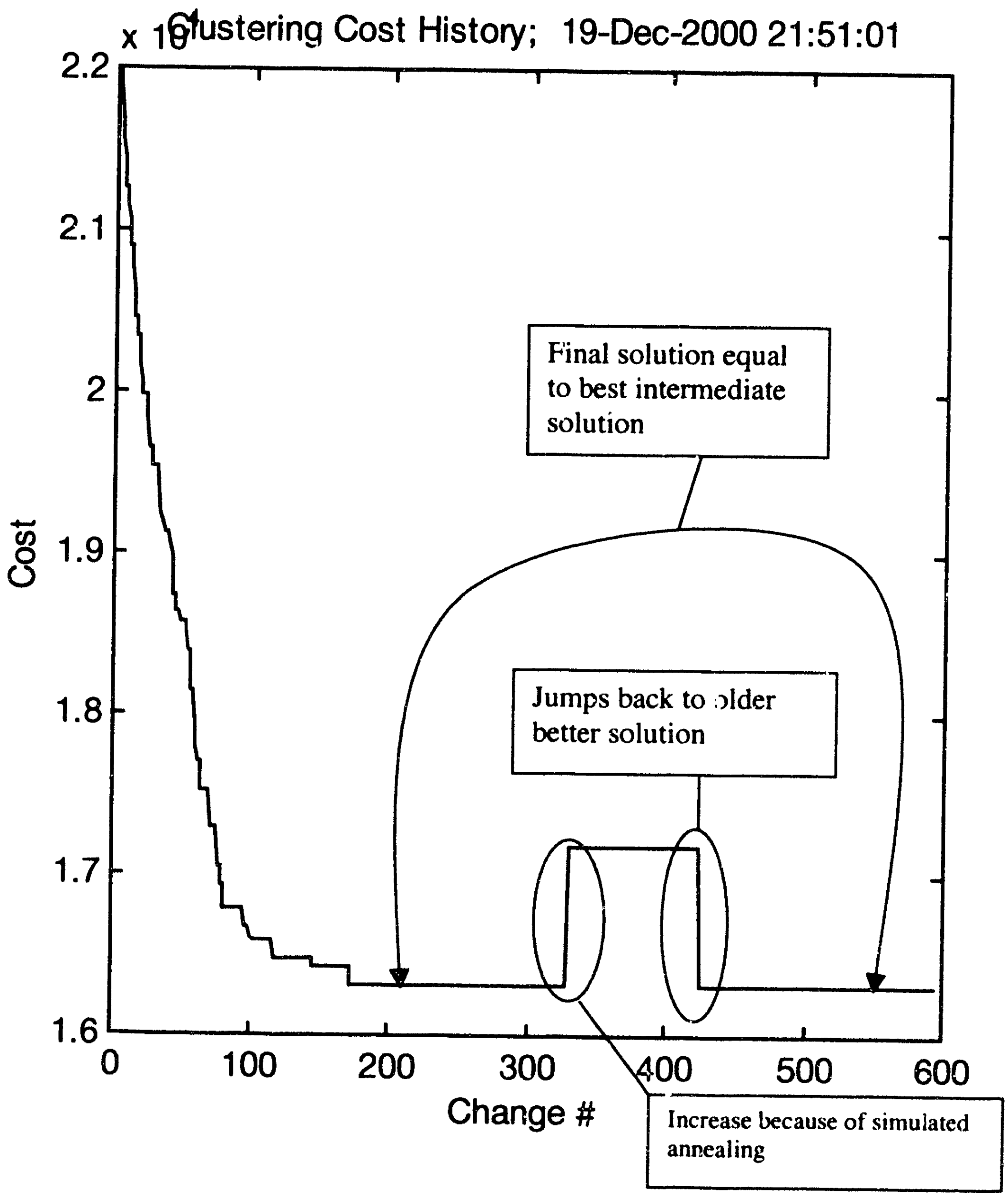


Figure 13: Cost History, New Algorithm, DSM with Varied Interaction Strengths

Chapter Summary

The problems encountered in Chapter 4 included the assignment of elements to multiple clusters and the reporting of a solution that was not the best solution found by the algorithm. Two changes were made to the algorithm to correct the problem. The first change added a cost penalty to elements that are members of more than one cluster. This significantly reduced or eliminated the multiple cluster assignment of elements. This also resulted in clusters that were much easier to recognize and analyze. The results greatly improved the possibility of creating a system architecture from the cluster elements. The second change forced the algorithm to keep track of the best solution found so that any non-optimal change resulting from the simulated annealing could be reversed if the final solution was not as good as any previous solution. This did cause the algorithm to always report the best solution that it was able to find. These two improvements significantly improved the results of the clustering algorithm. The next chapter will analyze the results of the clustering algorithm and its implications for the elevator problem.

Chapter 6 Cluster Analysis

The clustering algorithm randomly selects elements for bidding by clusters for membership. Due to this randomness, every run of the algorithm is slightly different as elements are selected in a different order and the clusters are built up differently. Therefore one run of the algorithm is not sufficient to provide an answer to the optimal clustering configuration. Several runs are required, and these runs will produce slightly different answers. Several steps have been taken to analyze the consistency of the answers produced by the algorithm. The clusters were also inspected for feasibility of implementation in design and management.

Multiple Runs

To analyze the consistency of the answers produced by the clustering algorithm, several runs were completed under identical conditions. The data including the DSM, the cluster matrix, cost history, and final coordination cost was stored from each of the 10 runs. Then each of the 10 runs was individually compared to the other 9 runs and scored for how well the cluster matched up with the clusters of the other runs. This provided analysis of the clustering algorithm under identical conditions. Then parameters or conditions could be changed and the consistency analysis completed again. This allowed for observation of what different parameters or algorithms produced more consistent answers. For the elevator DSM, the 10 run set took approximately 2 hours to complete.

Finding like Clusters

In order to measure the consistency of the answers, like clusters between runs had to be found. To get the like clusters, the clusters of one run were measured against the clusters of another run. The like elements between any two clusters were obtained by taking the dot product of the two cluster matrices. To measure the likeness of two clusters, twice sum of the like elements was divided by the sum of the total number of elements in the two clusters. The likeness measurement used two times the number of like elements because the like elements are members of both clusters.

$$Likeness(X,Y) = \frac{XinY + YinX}{TotalXY}$$

$0 \leq Likeness \leq 1$

X = Cluster X of Run 1

Y = Cluster Yof Run 2

XinY = Number of elements in Cluster X that can be found in Cluster Y

YinX = Number of Elements in Cluster Y that cab be found in Cluster X

XinY = YinX

TotalXY = Total number of elements in Cluster X and Cluster Y

for all X and Y of both Runs

For example take the two clusters in Figure 14. The cluster from Run-1 has 5 elements, the cluster from Run-2 has 4 elements. The have 3 elements in common. Therefore the likeness measure is $\frac{3+3}{4+5} = \frac{3*2}{4+5} = \frac{6}{9} = 0.66$. Therefore, the likeness score for these two clusters would be 0.66. This measure is done for every combination of clusters and runs.

Cluster	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	Total in Cluster
1	1	0	0	1	1	0	1	0	1	5
2	1	0	0	0	0	0	1	0	1	4
Dot Product	1	0	0	0	0	0	1	0	1	3
Likeness										0.66

Figure 14: Cluster Likeness Example

Once this likeness number has been attained for every combination of clusters, then we can get the average likeness for a cluster of one run against all runs. To get the number we compile a list for each cluster of a run. The list includes which cluster it was most like in the other runs and what the likeness scores were for those clusters. Then these numbers can be averaged to get an average likeness for the cluster. An example will make this more clear. Let's say we are compiling the list for Cluster-2 of Run-3. In the example in Figure 15, the average likeness was calculated to be 0.7455. This average is calculated for each of the clusters in every run. These averages were then graphed in a bar chart as shown in the example in Figure 16. Once the score for each cluster had been calculated, the average for the run and the average over all 10 runs were calculated. This represented how well a run matched other runs and how well the clustering algorithm did over-all. In the example plot (Figure 16), average for each cluster of a run is represented by the bar, the average for the run is listed below the subplot, and the average for all 10 runs is listed at the top of the sheet.

Run	Cluster # of highest likeness	Likeness with best matched cluster	Notes
1	2	0.8	
2	3	0.75	
3	not compared to clusters in same run		
4	5	0.9	
5	6	0.4	
6	7	1.0	Best Match = 1.0
7	2	0.86	
8	4	1.0	
9	5	0.7	
10	3	0.3	
Average		0.74555	Average score for Cluster-2 of Run-3

Figure 15: Example of Cluster Average Likeness Calculation

For the rest of the analysis all cluster parameters were held constant with the following values.

pow_cc	=	1
pow_bid	=	1
pow_dep	=	4
max_cluster_size	=	61
rand_accept	=	122
rand_bid	=	122
times	=	2
stable_limit	=	2

For the elevator DSM with all interaction strengths set at 1 (will be referred to as the standard interaction strength), the cluster comparison is shown in Figure 16. This figure also explains the different values that have been calculated. According to the calculations, Cluster-4 of Run-1 has an average likeness value of approximately 0.8. This means that according to the formula presented, there is an average of an 80% match between this cluster and its best match on the other runs. The graph also shows that this cluster has at least one exact match in the other runs.

Run-1 has an average likeness value of 0.70363. Therefore, the clusters of Run-1 have an average match of approximately 70% with all other runs. This value is important because it shows how well a run matched up with all other runs. If this number is lower than the averages for the other runs, then it may have been a bad run. This allows for removal of runs that may have resulted in significantly different results.

In the same manner, the likeness for all 10 runs is the average of the runs. In the example, the average for the standard interactions is 0.60649. This means that the average match across all clusters of all runs is approximately 60%. This number is important as changes are made to the clustering parameters, clustering algorithm, or DSM data. This number can be used to compare how consistent clusters were developing across multiple runs. The confidence in the results increases as this number increases, because it reflects how repeatable the outcomes are even with the randomness of the algorithm.

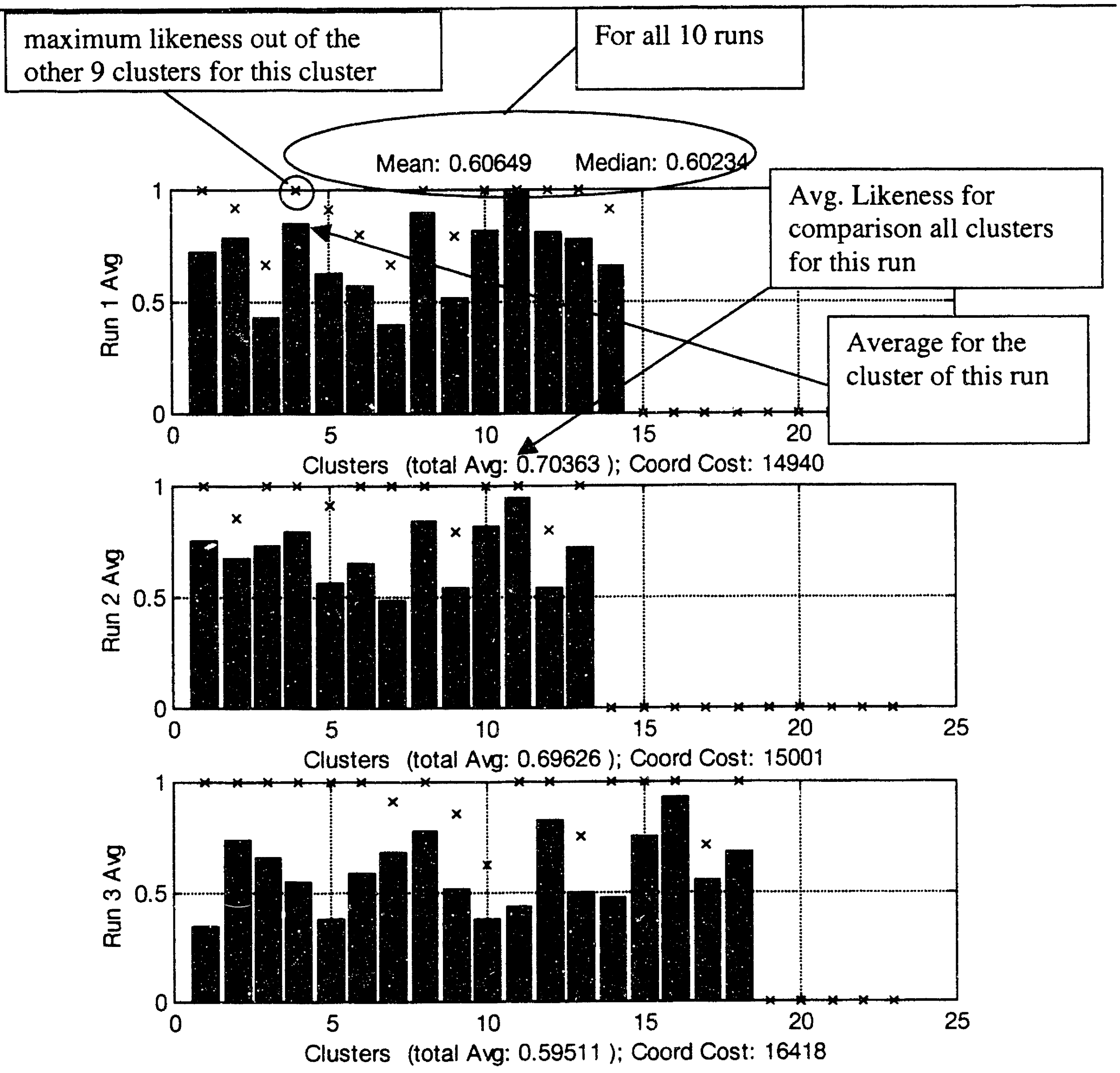


Figure 16: Cluster Likeness Example Plot; Standard Interactions

Varied Interaction Levels

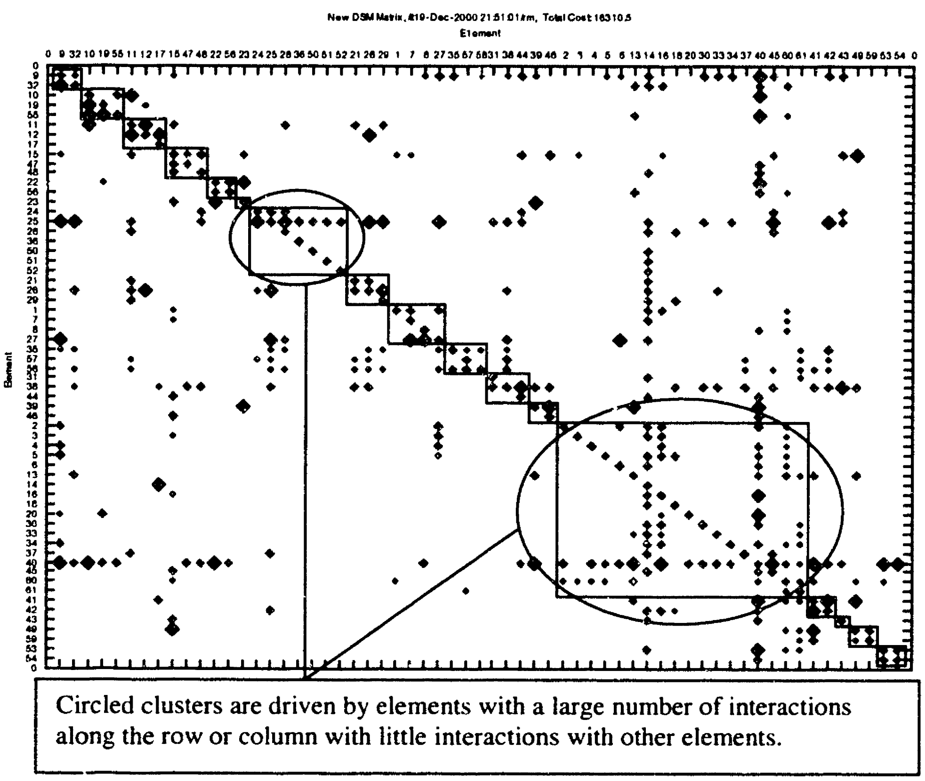
After reviewing the make up of the clusters that were being developed, it became apparent that the members of the clusters were not always highly similar nor did they always exhibit high interaction levels. Some cluster members appear to be connected to other members either through interactions that should not have been driving the design, or they were driven together through indirect connections.

In order to help the clustering algorithm identify which elements had a higher necessity of interconnectivity, the interaction levels in the DSM were modified. As previously discussed, weak interactions were represented with a value of 0.5 while strong interactions were represented with a 2. A strong interaction was defined as an interaction that was highly important, critical to operation, or was not optional. Weak interactions were defined as those that were not critical to the design of the system, that were a result of design choices but may be modified, or interactions that were not critical to the operation of the system. When these interactions were raised to a power in the clustering and bidding calculations, it gave strong interactions much more weight.

This change produced a more consistent clustering result. The strong interactions drive the clustering and the weak interactions have little value unless no strong interactions are present. By valuing the interactions in this manner, the clustering algorithm was able to understand the importance of an interaction and value it accordingly in the optimization calculations. This is especially important because there is no way to represent the interfaces or interactions that are prohibited or not desired.

Extraction of Integrative Elements

Another observation of the clustering results is that the DSM elements that were integrative across the DSM drove integration of some of the clusters. These elements either had a large number of interactions across the row, column, or both. A few of these elements were in the DSM for documentation purposes. For example, the service person and the load do not need to be considered for the clustering purpose, but they do need to be represented to show the interactions that they may have with the design. If we re-examine the clusters that were produced at the end of Chapter 5, there were two clusters that were driven by elements with many interactions. The cluster results have been reproduced in Figure 17 with the two integrative clusters circled. Upon closer examination of the clusters, it becomes clear that the clusters formed around the element with many interactions. The other elements within these clusters do not have many, if any, interactions with each other. In the large cluster, it can be noticed that several of these elements are highly integrative with the elements of the cluster. These results produced clusters that were mostly related through the integrative element. Once again, if the cluster was evaluated by attempting to give it a name based on the cluster members, it was difficult to do for these clusters because most of the elements appeared to be unrelated.



Circled clusters are driven by elements with a large number of interactions along the row or column with little interactions with other elements.

Figure 17: Clusters driven by Integrative Elements

To alleviate the problems caused by these elements, the algorithm was modified to allow removal of them. They were removed for clustering purposes only, but were re-introduced into the clustered DSM on the outside edges. This process also improved the results. The improvement can be shown in several areas. First the likeness measurement showed remarkable improvement in the scoring as these integrative elements were removed for clustering. This meant that the clustering results were becoming more repeatable with the removal of the integrative elements. Secondly, the graphical view of the clustered DSM (Figure 18, page 67) showed most interactions were residing within the clusters with the exception of the elements that had been removed. And lastly, it became much easier to name the cluster elements.

Some conclusions can be made about the elements that were removed. For instance, the travel cable is actually the physical implementation of the interface between the cab and controller. At this point, it's the knowledge that the cab and controller require an interface that is important, not the implementation of the interface. Also, by representing the interface between the components of the cab and the controller, while including the traveling cable in the DSM, the interface has essentially been represented twice. Therefore the travel cables were removed for clustering purposes.

The service person and passenger load elements were included in the DSM mainly for documentation purposes. It is important to know what elements these two will interface with, but because of the number of elements that they interface with, they do not add much information to the clustering routine. Once clustering has been completed it is actually more important to identify what clusters the passenger load and the service personal will have to interface with.

The cab already represents a cluster. The cab is made of individual components that interface with other components. Without decomposing the cab another level, we have forced the cab to interface with many items. This situation results in clusters that would cluster with the cab but wouldn't necessarily cluster with each other. If time allowed, it would be better to decompose the cab even further and run the clustering algorithm again.

The power supplies were represented as a single element with reason. In reality, most components or subsystems have their own power supplies or share a power supply with a few other items. By implementing the power supply as a single item, we can reconfigure the power supply to meet the clustering architecture. Therefore, once the clustering algorithm is completed the power supplies could be broken down to fit neatly within each of the clusters, or each cluster can have responsibility for a section of the power supply that interfaces with their cluster. Important information may be extracted by representing the power supply as a single element so it may be better that the power supply is not included in the clustering algorithm. The power supply was a frequent cause of clusters that didn't contain elements that were directly related.

The motion controller and safety system also caused some of these problems. In the last stage of removing elements before clustering, these components were also removed. These components interface with many of the elements in the group and they represented several functions. Again, they were removed for the purpose of analyzing how well the clustering performed in the absence of their influence. These components and their interfaces are important in the architecture of the system and management of the organization. If time allowed, the elements should have been decomposed even further to get at the functions that make up the motion controller and safety system.

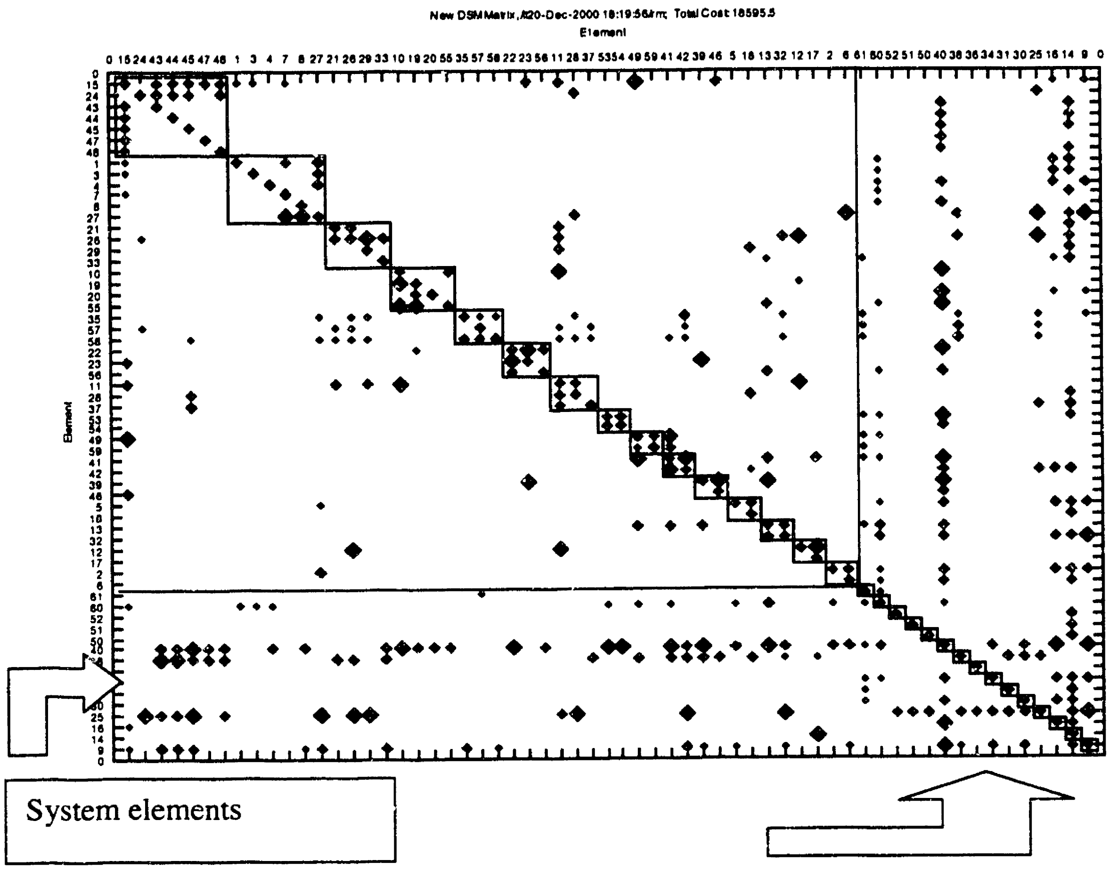


Figure 18: Cluster DSM with System Parameters on the Outside

The extraction of system or integrative parameters is not a new idea. It is common practice to move DSM elements that have many entries along a row, column or both to the outside of the DSM, while keeping the clustered elements in the middle. A discussion of this in the context of design parameters is discussed in detail in a thesis on the application of the DSM on Jet Engines (Bartkowski)[x] and the process is shown in Figure 19.

[x] Includes discussion on the importance of sequencing and knowledge management for System Level parameters

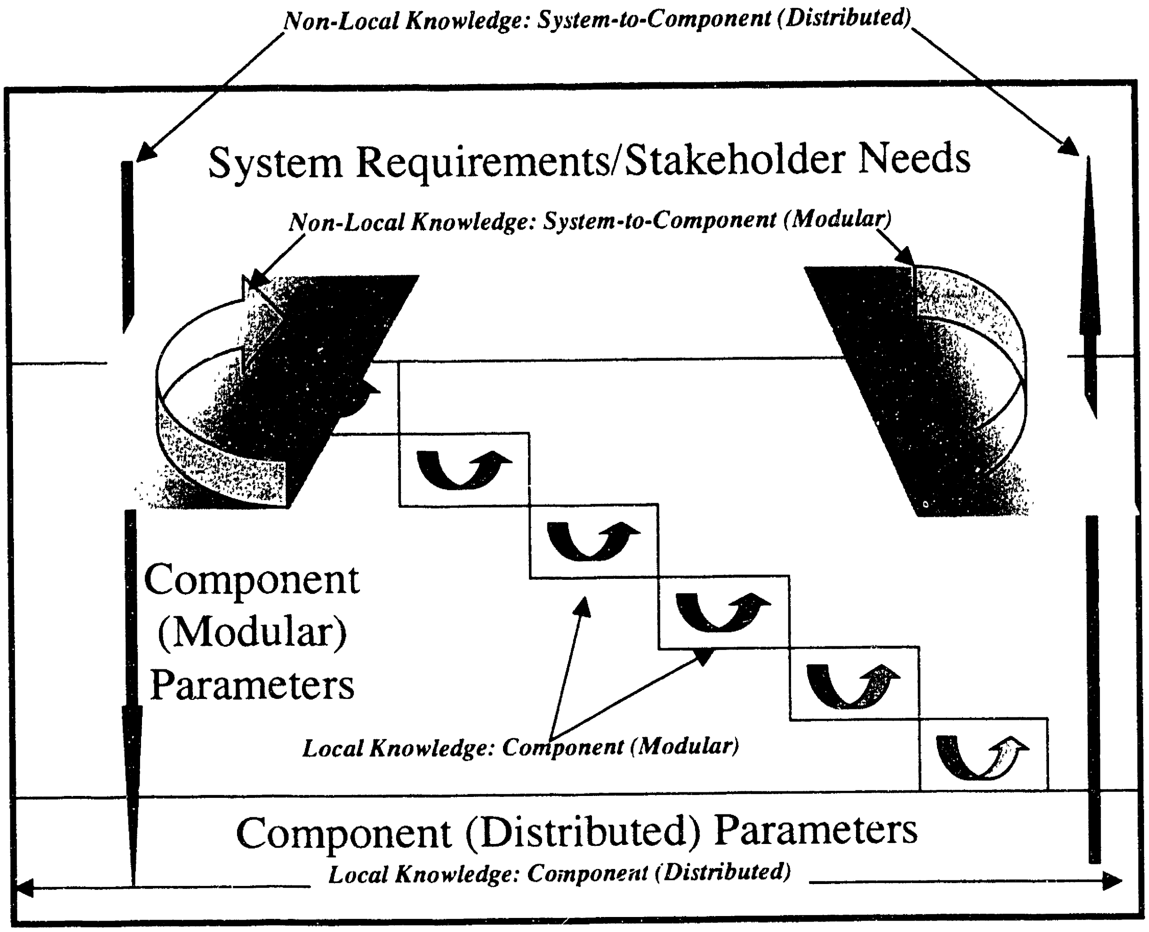

Non-Local Knowledge: System-to-Component (Distributed)
System Requirements/Stakeholder Needs
Non-Local Knowledge: System-to-Component (Modular)
Component (Modular) Parameters
Local Knowledge: Component (Modular)
Component (Distributed) Parameters
Local Knowledge: Component (Distributed)

Figure 19: P&W Distributed and Modular Parameters[xi]

The system elements are generally elements that have interactions throughout the system. In many cases, the system elements cannot be decomposed further and therefore they have a significant influence on the system and its components. Since their influence in the system is so great, it is desirable to have the elements managed by a systems organization that maintains contacts with all of the design groups and has no stake in any of the components (modules). It would be difficult for any group or component to own the system elements since they require so much information and require interactions with so many other parts of the system and development organization. This idea holds for both the design parameters as discussed in the Bartkowski thesis and the physical elements in this elevator problem. The system elements require special attention and are usually managed by a group that is void of component design responsibilities.

Chapter Summary

As previously discussed, the clustering algorithm randomly selects elements for bidding by clusters for membership. Due to this randomness, every run of the algorithm will produce slightly different answers as elements are selected in a different order and the clusters are formed differently. Therefore one run of the algorithm was not sufficient to provide an answer to the optimal clustering configuration. Several runs were made and the results were analyzed for consistency. A few measurements of consistency were developed and the likeness measurement was used to evaluate different clustering options and DSM configurations. The results showed that clustering is significantly improved if the DSM interaction strengths are varied according their importance in the creation of the architecture. Also, the elements with many interactions were removed as system elements because they would cause creation of sparsely populated clusters that were related only by the system element. These changes significantly improved the results of the algorithm and resulted in clusters that could be used to define an optimally configured interface architecture.

[xi] Bartkowski Thesis, pg 47

Chapter 7 Results Summary

Cluster Analysis

As outlined in the preceding chapter, the enhancements to the algorithm greatly improved the consistency of the results. The enhancements included adding the varied interaction strengths, and the removal of the integrative elements. As shown in Table 3, the results improved when the standard interaction strengths (all interactions valued at 1) were changed to represent their perceived strengths. The greatest improvement resulted from extracting the system parameters before performing the clustering routine. These system parameters interfaced with many of the elements within the DSM and confused the algorithm when it was looking at the interaction strengths. Trial #5 resulted in very consistent clusters throughout the 10 runs and an example of the clustering results are contained in Appendix E.

Table 3: Clustering Analysis Results

Trial #	Parameters	System Parameters Extracted Before Clustering	10 Run Likeness Average
1	Standard	-	0.60234
2	Varied Interaction Strengths	-	0.62796
3	Extracted elements to System Parameters	Travel cable comm. Travel cable power Service Personnel Pass. Load	0.72789
4	Extracted elements to System Parameters	Trial 3 removed elements plus Power Supplies Cab	0.7781
5	Extracted elements to System Parameters	Trial 3 & 4 removed elements plus Motion Controller Safety System	0.84319

System Elements

The extracted elements represent system parameters, already clustered elements, or design decisions. These will be discussed with an example from this analysis. The safety system is an example of a system element, the cab is an example of an already clustered element, and the power supply is an example of an element requiring further design decisions.

The Safety System parameter was extracted to the outside edges of the DSM and not considered for the clustering routine, it is an example of a system parameter. Many of the systems, components, and development teams must include some part of the safety system. The extraction says that the safety system should either be managed by a systems organization, or carefully decomposed to the interfaces that each of the clusters must interact with. This represents important information in the design of the architecture and management of the product development teams. The process of deciding to extract the parameter as a system parameter provides information itself. To extract the information, the user must understand that the parameter has many interfaces that can cause the erratic clustering. This understanding will force the user to make one of three decisions. The parameter can continue to be treated as a system parameter and it can be managed by a systems organization. Secondly, the parameter can be managed by a systems organization, but be will be broken down into the parts that each cluster will interface with and that part of the system parameter will be delegated to the cluster. The third alternative is to further decompose the parameter into its sub-elements and then attempt to re-cluster if the number of interfaces per element has been reduced.

The cab is an example of an element that already represents a cluster. The walls, frame, floor, aesthetics, and structural parts of the cab are all represented by the cab element. Therefore many elements interface with the cab. Again, the extraction of the element represents important information. Once it is recognized that the cab has many interactions that interfere with the clustering algorithm, some design decisions can be made. The cab can be further decomposed and then the clustering algorithm can be used again. This will allow the cab elements to either be clustered back together or it may be redistributed into different clusters if that produces a more optimal solution. This has the potential to reconfigure the architecture or create alternate thinking about how the cab should be developed. If the cab element remains integrated as a single unit, it may be useful to segregate cab design management into a cab manager and cab interface manager. In any instance, recognizing the cab as a system element provides valuable information.

The power supply was purposely represented as a single element in the DSM to understand how it would be treated in the clustering algorithm. As with the other system parameters, the power supply weakened the clustering results and needed to be extracted as a system parameter. This system parameter was valuable after the clustering had been performed. Now that the clusters have been developed, the clustered DSM can illustrate two pieces of information: 1) it identifies the clusters that the power supply interfaces with and 2) identifies the elements of the cluster that cause the interface. With this information there are several options to consider. The power supply can be separated and distributed to the clusters that interface with it. This can provide a design for the power supply system architecture. The power supply can also remain a system element and the architecture could be developed so that the new DSM has defined the interfaces between the power supply and the clusters.

Although the system elements have not been included in the clustering algorithm, they do provide valuable information. The process of extracting the elements causes the user to identify the elements as system parameters and understand the implications of the extraction. The system parameters also represent design decisions that can be evaluated once the clustering is complete.

Clustered Elements

The clusters represent groups of elements that should have a relatively high level of interactivity. The objective is to maintain interconnections within clusters so as to minimize the cross talk between clusters and between development groups. In the case of the elevator DSM, the clusters did represent recognizable groups. At this time, neither the DSM nor the results are at a stage where concrete architecture or organizations can be developed, but it is starting to show some results.

For instance, using the new algorithm, varied interaction strengths, and removal of system elements, the clustering produced 10 main clusters of elements and 15 clusters with only one element(which included the eight removed elements). An initial attempt to recognize the clusters and develop a system from the clusters is shown in Table 4. Highly interactive elements are contained within each cluster. The clusters can then be grouped together to represent groups that have similar functionality or requirements. This can be useful for system architecture or organization of product development teams.

The clustered elements in Table 4 create a system architecture that is similar to the current system architecture with a couple of exceptions. For example, the clustering produced a cluster that contained the operational controller, hallway fixtures, and car fixtures. At Otis Elevator Company, a group responsible for fixtures typically designs the fixtures and a separate design group designs the operational controller. The clustering algorithm recognized that there are significant amounts of information that must flow between the fixtures and the operational controller and it has grouped them together. By keeping these design tasks and systems separate, the teams must communicate often and efficiently. Any miscommunication could result in design errors or delays. This result does not mandate that the system and development teams become organized this way, but it does allow the systems architect, engineers and managers to make an objective analysis and informed choices. Once again, the elevator system DSM must be much more complete before concrete conclusions can be made about the architecture and results, but this process is providing promising information. Also, it is apparent that the clustering results are affected by the input, therefore careful consideration should be made about the DSM entries, the interaction strengths, and their meaning.

Grouped Sub-systems to create the system.

Grouped modules to make sub-systems.

Grouped clusters to make modules.

Name given cluster based on its members.

Each Cluster member box shows a cluster. Shaded boxes show system elements removed prior to clustering.

Table 4: Cluster Defined System Tree

Elevator System	Electrical Systems	Motion	Motion Sensing	Hallway Fixtures * Building Structure Learn Function Terminal Sensing System ETSD Sensor Door Zone Sensors Terminal Buffer NTSD Sensor
			Motion Control	**Motion Controller**
			Mode Limiting Sensors	Emerg. Gen Signal
				Building Sway Sensor
				Earthquake Sensor
				Top of car Insp. Panel
		Power Supply	Power Supply	**Power Supplies**
		Operations	Operational Control	Hall Request Indicator Car Request Indicator Car Fixtures Hall Buttons Car Buttons Fireman's Service Key Operation Controller
		Electrical Power	Electrical Drive	Drive Power Section Building Power Brake System Electrical Drive Control Primary Velocity Meas Construction Ctl Panel
		Health	System Health	Service Tool Secondary Vel Check Diagnostics Remote Monitor System
			Safety System	**Safety System**
	Hoistway Systems	Mechanical Power Control	Mechanical Motion	Ropes Motor Counterweight Compensation System Brake System Primary Position System Ropes/Hitch Springs
			Guidance	Cab Guidance System Guide Rails Guide Springs Alignment
		Load Containment System	Load Support	Cab Floor Plank Load Weighing System Mechanical Safeties Sys. Governor System *
			Load Contain.	**Cab**
			Doors	Cab Doors Door Controller Hoistway Doors Access Keying on Doors
				Door Obstruction Sensor
			Cab originated commands	Door Close Signal Remote Insp. panel
			Emergency Comm./Power	Emerg Intercom-Phone Power Storage
			Traveling Cable	**travel Cable Power**
				travel Cable Comm.
			Passenger Comfort	Vent & Lighting System Cab Insulation
	Human/ passenger Interface	Human/ passenger Interface	Human Interface	**Service Personnel**
				Passenger Load

Elevator System Developed from the Clusters

Interactions Outside the Clusters

The interactions outside of the clusters, as shown in Figure 20, represent interactions between clusters and interactions between system elements and they should get careful attention. These interactions represent possible architecture decisions where standard interfaces could be developed. Also, no one cluster owns these interactions. The interactions represent interfaces between two clusters or system elements. It is highly probable that the interactions can cause conflicts between clusters and should be managed by a systems organization that does not have a stake in the interface. The purpose of the clustering process was to minimize or eliminate the external interactions. Therefore, once the clustering has been completed, a careful analysis should be performed in order to understand the interactions and decide if further design improvements can be made to eliminate or move the interface within a cluster. Moving the interface inside a cluster essentially assigns ownership of the interface to that cluster. Because each situation can be different, each interaction outside of clusters should receive careful analysis and understanding.

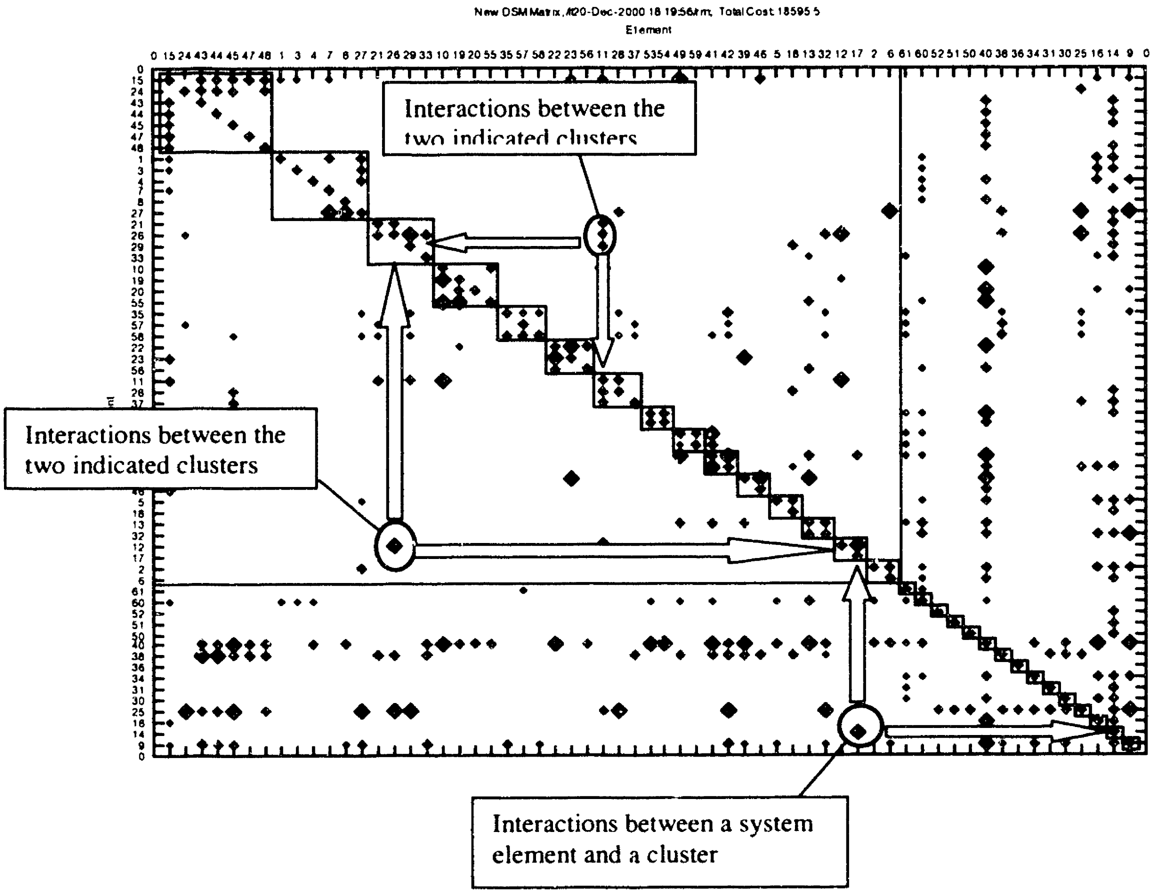

Interactions between the two indicated clusters
Interactions between the two indicated clusters
Interactions between a system element and a cluster

Figure 20: Interactions outside of the Cluster

Chapter Summary

The analysis showed that varying the strengths of the interactions within the DSM produced more consistent results. The results were further improved by removing system elements from the clustering process. The systems elements that were removed from the clustering process do provide valuable information and must reviewed and managed carefully. A review of the results of the clustering showed that interactions within clusters belong to the cluster and interactions outside of the cluster should be managed by a systems engineering effort since they cannot be effectively assigned to any single cluster because of the exchange with other clusters.

Chapter 8 Conclusions

Research Conclusions

In today's environment of changing product requirements, rapidly advancing technology, and decreasing product development times, products are becoming more complex to meet the requirements. As the complexity increases it becomes more difficult to understand all of the interactions and interfaces that develop within the product and product development teams. Minimizing the interfaces between systems, components and development teams can help reduce rework caused by misinformation, reduce complexity by minimizing the amount of information exchanged and decrease the product development cycle time by standardizing interfaces.

To reduce interfaces and interactions, a clustering algorithm was developed and evaluated for its effectiveness at organizing the interactions of a complex system. A process was created whereby the system and interactions are represented in a matrix format called the Design Structure Matrix. The Design Structure Matrix was created from functional requirements derived from a decomposition of system requirements. The mapping of the functional requirements to physical elements provided information of how the physical elements were derived. Once the mapping was developed, and the DSM was created, the DSM was reorganized by clustering elements with a high degree of interactions.

The developed clustering algorithm provided valuable information that can be used to develop, evaluate, and manage the system architecture. Although the algorithm produced satisfactory results, it cannot be used without an understanding of the algorithm, the desired results, and the system being evaluated. Use of the clustering algorithm requires proper setup of the parameters that control it. To get the proper parameters, the algorithm is run on the intended DSM and the parameters are adjusted until the results produce the desired cluster size and number of clusters. Therefore, the user must understand the algorithm well enough to be able to adjust the proper parameters to get the desired results. The user must also understand the system well enough to know the number of clusters that are desired. Although it would be relatively easy to modify the algorithm to produce the exact number of clusters that are desired, it is much more informative to let the algorithm produce the optimal configuration and analyze it from that point. This process allows the user to gain an understanding of the system, understand how the algorithm wants to cluster elements, and what drives the clustering. This learning process can provide valuable knowledge about the system.

To get consistent and meaningful clusters, some elements of the DSM had to be excluded from the clustering algorithm. These elements represent system elements or elements that should be decomposed further. These elements have interactions with many of the other elements within the DSM. They can be recognized by the large number of entries that appear along their row or column of the DSM. Exclusion of these parameters from the clustering algorithm produces more consistent and meaningful clusters, but the exclusion itself provides valuable information. The system architect or manager should understand why the parameters have a high number of interactions. Then decisions can be made about whether or not to further decompose the element and re-cluster, assign parts of the element to the clusters with which it interacts, or manage the element from the systems level of the organization. These considerations provide valuable information about the system architecture and allow for informed choices.

Finally the clusters themselves represent a way to organize the DSM elements or information in a way to minimize the amount of information exchanged. Information flow within a cluster provides an efficient way to transfer information within a subsystem or within a design team. This is usually the least costly option. Interactions outside of the clusters represent information or interactions that are not owned by any of the clusters. The clustering algorithm should have minimized the number of these that occur. These interactions can be costly and time consuming. Conflicts may occur and design parameters can be unclear if no strong decisions are made. Therefore, the interactions outside of the clusters should be managed or assigned by a systems level organization.

This process requires a significant knowledge of the system and its interactions. It is difficult to produce and manage unless the user has a clear understanding of the process and the system. Therefore it would be highly advantageous for this to be completed by a team of experts or systems people that have knowledge of the entire system and can solicit information from required experts. In this simplified elevator example, the process forced the questioning of many of the design decisions as the DSM was created and the interactions were entered in. It also led to a new understanding of the elements, their interactions with each other and conditions for which some elements may not provide easy clustering solutions. The process provided possible future direction for system development and architecture changes. It will be highly advantageous to reapply this process to a complete representation of the elevator system.

Future Work

This research provided valuable information with a limited set of data. Although the DSM represented the entire elevator system, it did so with a subset of the available items that could have been used. It would be extremely valuable to continue or recreate this process on a more complete representation of the system

Some conclusions were reached about what can be done with the system parameters that were excluded from the clustering algorithm. It would be interesting to continue the process by making some of the changes in the system elements and analyzing the results that are produced. Can system elements be decomposed and successfully re-entered into the clustering algorithm? In the power supply example, can a true power supply architecture be developed by distributing the architecture to the clusters? These and other questions about the system parameters would be important considerations in future analysis.

When this research began, it was the intention to analyze the clusters in the context of the scenarios that were produced. Because scenarios were mapped to functional requirements, and functional requirements were mapped to physical elements, the clusters can be mapped back to the scenarios through these matrices. How well do the clusters address the needs of the scenarios? Can alternate clustering based on scenarios be accomplished? This type of clustering would be useful so that the architecture neatly addresses different scenarios. The hypothesis is that it may be desirable to have clusters match scenario needs so that pieces of clusters are not used when a small subset of requirements are needed for a scenario.

This process created a one-to-one mapping of functional requirements to physical elements. Most of the analysis was done using the physical elements as test data. It would be interesting to know if the same conclusions can be made if we applied the clusters to the functional requirements. It should be similar, but what conclusions can be drawn from the clustering of functional requirements?

It would also be interesting to cluster on different types of interactions or on multiple types of interactions. For example, several matrices could be developed to represent the communication or power interfaces. Then clustering could be performed to produce architectures that represent the electrical system or the mechanical system. It is also possible that one type of interaction may be more important than another type. For example, a process may value electrical interactions much higher than mechanical interactions. Therefore the algorithm could be modified to cluster based on a combination of interaction types or set to value some type differently than others. There are many possibilities if the needs exist.

Chapter 9 Appendix

Appendix A Functional Decomposition Example

Provide Safe Vertical Transportation for Passengers and Freight

1 Communications
 1.1 Accept user requests
  1.1.1 Input Car Passenger Destination Request
  1.1.2 Input Hall Service Request
 1.2 Provide Status to user
  1.2.1 Communicate Hall Passenger
   1.2.1.1 Communicate Elevator Status
   1.2.1.2 Acknowledge Hall Service request
  1.2.2 Communicate Car Passenger
   1.2.2.1 Communicate Elevator Status
   1.2.2.2 Acknowledge Car Destination Request
   1.2.2.3 Provide communication link to cab
2 Move the Load
 2.1 Energy Conversion
  2.1.1 Provide Energy for Motion
  2.1.2 Energy Transformation to Force
  2.1.3 Transmit force to containment
 2.2 Guidance
  2.2.1 Provide Path for Motion in Hoistway
  2.2.2 Maintain Cab Guidance in Hoistway
 2.3 Support
  2.3.1 Support Load for movement
 2.4 Hold the Load at Destination
3 Control
 3.1 Control trajectory
  3.1.1 Compute Trajectory
  3.1.2 Control Motion to Trajectory
 3.2 Measure Motion Parameters
  3.2.1 Measure Trajectory (velocity)
  3.2.2 Measure Position
 3.3 Input Destination Target
  3.3.1 Process Service Request
4 Load Containment
 4.1 Support the Load
5 Maintain Safe Environment
 5.1 Isolate User from Hazards of Elevator motion
 5.2 Protect User/Load

- 5.2.1 Determine if safe to move
- 5.3 Prevent uncontrolled motion
  - 5.3.1 Detect uncontrolled motion
  - 5.3.2 Protect from freefall condition
    - 5.3.2.1 Detect freefall condition
    - 5.3.2.2 Stop freefall condition
  - 5.3.3

6 Service

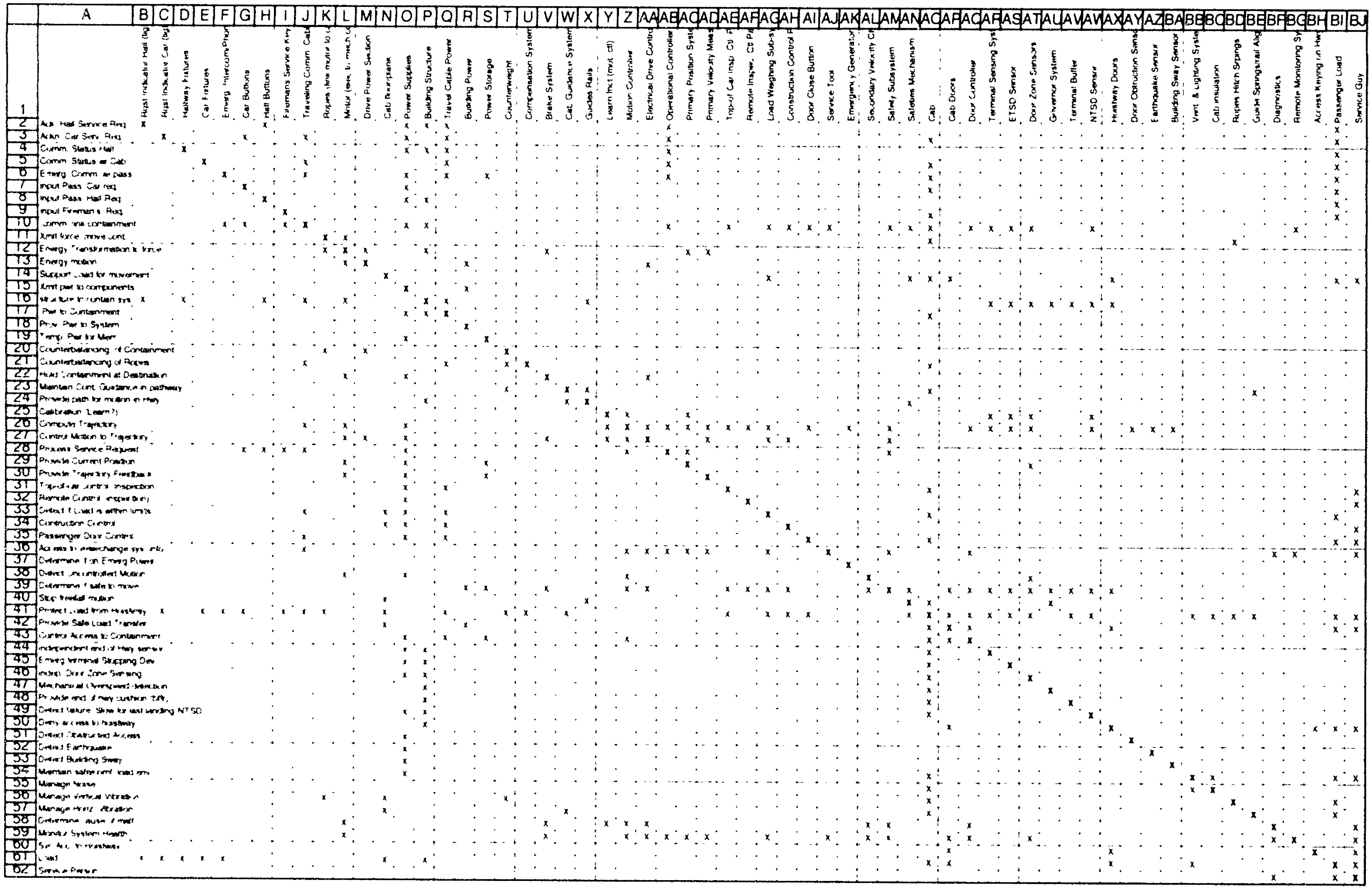
- 6.1

Hall passenger:	Passenger/Load in Building hallway waiting for service
Car passenger:	Passenger in Elevator Car

Appendix B Scenario to Functional Mapping Matrix

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1		Normal Operation	Remote Inspection	Top-of-Car Inspection	Installation (contruction mod	Start-up (calibration)	Re-leveling	Fireman's Service	Freefall (>110% vel dn)	Overspeed Condition	Overload Condition	Emergency Stop	Obtructed Containment Acc	Landing Overshoot	Overshoot Terminal Landing	Fail to slow for last landing	Loss of Power Supply	Position Meas Failure	Velocity Meas Failure	Automatic Rescue (Recover	Earthquake	Building Sway	Emergency Power	Manual Rescue	Stop and Shutdown
2	Acknowledge Hall Service Request	X						X														X	X		
3	Acknowledge Car Service Request	X						X														X	X		
4	Comm Status w/ Hall Pass	X																				X	X		X
5	Comm. Status w/ Cab Pass	X						X			X		X									X	X		X
6	Emerg Comm w/ pass								X	X		X					X				X			X	X
7	Input Passenger Car request	X						X										X	X			X	X		
8	Input Passenger Hall Request	X																				X	X		
9	Input Fireman's Service Request							X																	
10	Provide comm link to containment	X		X				X	X	X	X	X	X	X	X	X	X	X	X		X	X	X		X
11	Xmit force to move containment	X	X	X	X	X	X	X		X	X			X	X	X		X	X	X	X	X	X	X	X
12	Energy Transformation to force	X	X	X	X	X	X	X						X	X	X		X	X	X	X	X	X		X
13	Provide Energy for motion	X	X	X	X	X	X	X						X	X	X		X	X	X	X	X	X		X
14	Support Load for movement	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
15	Transmit power to components	X	X	X	X	X	X	X		X	X	X	X	X	X	X		X	X	X	X	X	X		X
16	Provide structure to contain sys	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
17	Provide Power to Containment	X	X	X			X	X		X	X	X	X	X	X	X		X	X	X	X	X	X		X
18	Provide Power to System	X	X	X	X	X	X	X		X	X	X	X	X	X	X		X	X	X	X	X	X		X
19	Temporary Power for Memory																X								
20	Provide Counterbalancing of Containmen	X	X	X	X	X	X	X		X	X	X	X	X	X	X		X	X	X	X	X	X	X	X
21	Provide Counterballancing of Ropes	X	X	X			X	X		X	X	X	X	X	X	X		X	X	X	X	X	X	X	X
22	Hold Containment at Destination	X	X	X	X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
23	Maintain Cont Guidance in pathway	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
24	Provide path for motion in Hwy	X	X	X	X	X	X	X	X	X		X		X	X	X	X	X	X	X	X	X	X	X	X
25	Calibration (Learn?)					X																			
26	Compute Trajectory	X	X	X	X	X	X	X						X	X	X		X	X	X	X	X	X		X
27	Control Motion to Trajectory	X	X	X	X	X	X	X						X	X	X		X	X	X	X	X	X		X
28	Process Service Request	X						X														X	X		
29	Provide Current Position	X				X	X	X		X	X	X		X	X	X	X	X	X	X	X	X	X		X
30	Provide Trajectory Feedback	X	X	X	X	X	X	X		X		X		X	X	X	X	X	X	X	X	X	X		X
31	Top-of-car control (inspection)			X																					
32	Remote Control (inspection)		X																						
33	Detect if Load is within limits	X					X	X			X											X	X		
34	Contruction Control				X																				
35	Passenger Door Control	X						X													X	X	X		
36	Access to view/change sys info					X																		X	X
37	Determine if on Emerg Power																						X		
38	Detect Uncontrolled Motion								X	X				X	X	X		X	X						
39	Determine if safe to move	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X	X	X		X
40	Stop freefall motion								X																
41	Protect Load from Hoistway	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
42	Provide Safe Load Transfer	X					X	X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X
43	Control Access to Containment	X					X	X			X		X			X				X	X	X	X		X
44	independent end of Hwy sensor		X	X	X	X									X										
45	Emerg terminal Stopping Dev					X																			
46	Indep Door Zone Sensing	X				X	X	X						X	X			X	X	X	X	X	X		X
47	Mechanical Overspeed detection								X	X									X						
48	Provide end of hwy cushion (bffr)								X						X										
49	Detect failure Slow for last landing-NTSD					X										X									
50	Deny access to hoistway	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
51	Detect Obstructed Access												X												
52	Detect Earthquake																				X				
53	Detect Building Sway																					X			
54	Maintain safe/comf load env	X	X	X			X	X	X	X	X	X	X	X	X	X		X	X	X	X	X	X		X
55	Manage Noise	X					X				X		X	X	X	X				X	X	X	X		X
56	Manage Vertical Vibration	X					X				X		X	X	X	X				X	X	X	X		X
57	Manage Horizontal Vibration	X					X				X		X	X	X	X				X	X	X	X		X
58	Determine cause of malfunction																	X	X						X
59	Monitor System Health																								X
60	provide Service Access to Hoistway			X																				X	X

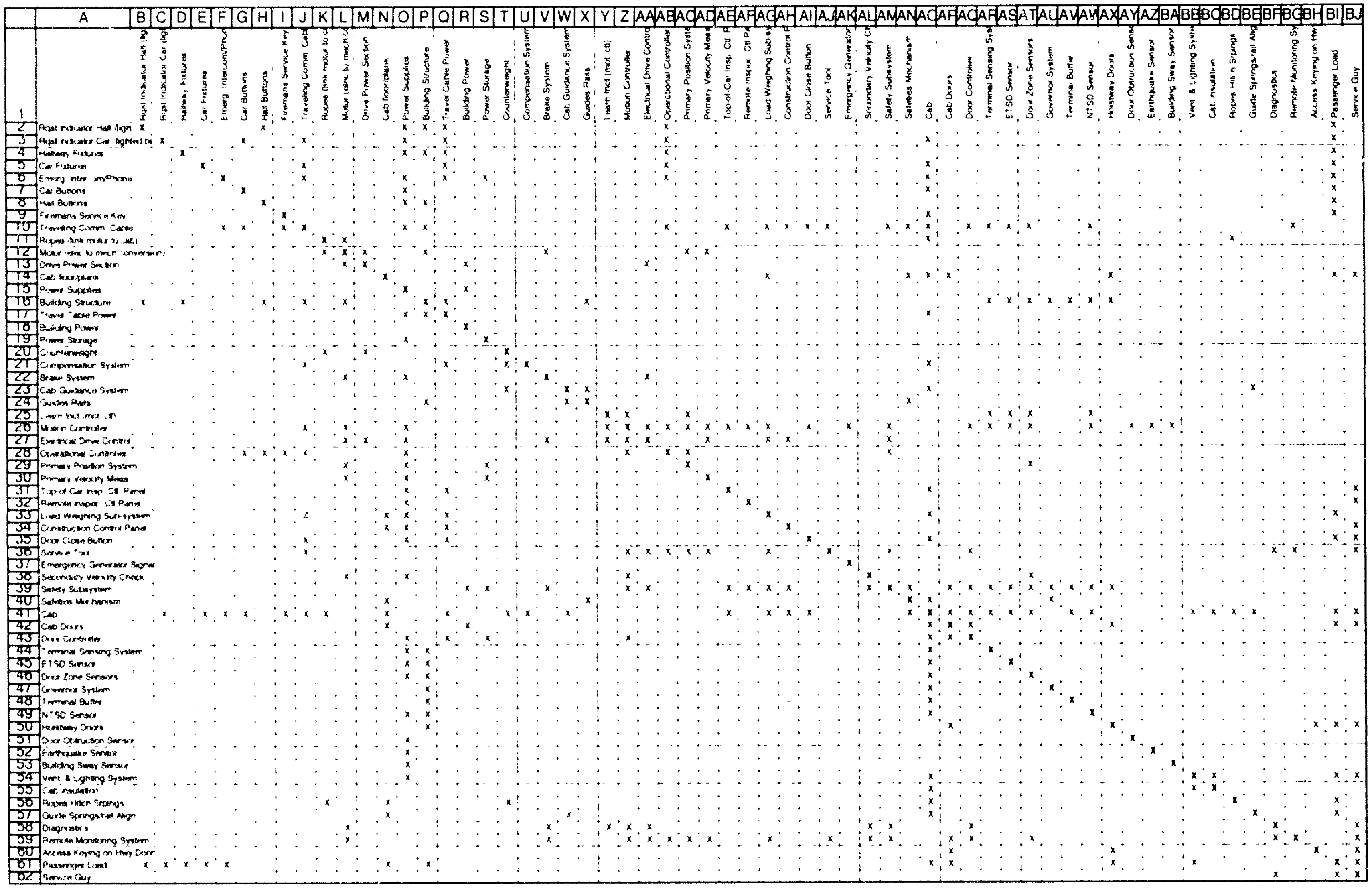
Appendix C Functional to Physical Mapping



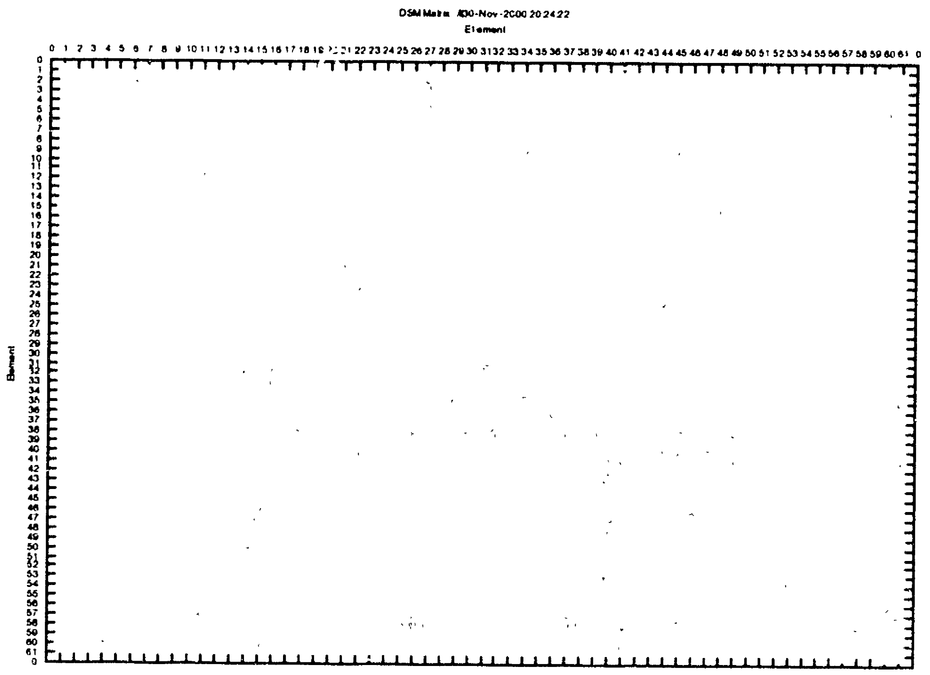
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ	BA	BB	BC	BD	BE	BF	BG	BH	BI	BJ
1		Rqst Indicator Hall (bg	Rqst Indicator Car (bg	Hallway Fixtures	Car Fixtures	Emerg. IntercomPhon	Car Buttons	Hall Buttons	Firemans Service Key	Traveling Comm Cab	Ropes (link motor to c	Motor (elec. to mech c	Drive Power Section	Cab Backplank	Power Supplies	Building Structure	Travel Cable Power	Building Power	Power Storage	Counterweight	Compensation System	Brake System	Car Guidance System	Guides Rails	Learn Inct (mot. ctl)	Motion Controller	Electrical Drive Contro	Operational Controller	Primary Position Syste	Primary Velocity Meas	Top-of Car Insp. Ctl F	Remote Inspec. Ctl Pa	Load Weighing Sub-sy	Construction Control F	Door Close Button	Service Tool	Emergency Generator	Secondary Velocity Ch	Safety Subsystem	Safeties Mechanism	Cab	Cab Doors	Door Controller	Terminal Sensing Sys	ETSD Sensor	Door Zone Sensors	Governor System	Terminal Buffer	NTSD Sensor	Hoistway Doors	Door Obstruction Sens	Earthquake Sensor	Building Sway Sensor	Vent. & Lighting Syste	Cab insulation	Ropes Hitch Springs	Guide Springs/rail Alig	Diagnostics	Remote Monitoring Sy	Access Keying on Hwy	Passenger Load	Service Guy
2	Actr. Hall Service Req	[illegible]																																																											
3	Actn. Car Serv. Req																																																												
4	Comm. Status Hall																																																												
5	Comm. Status at Cab																																																												
6	Emerg. Comm. w/ pass																																																												
7	Input Pass. Car req																																																												
8	Input Pass. Hall Req																																																												
9	Input Fireman's Req																																																												
10	Comm. link containment																																																												
11	Xmit force move cont.																																																												
12	Energy Transformation to force																																																												
13	Energy motion																																																												
14	Support Load for movement																																																												
15	Xmit pwr to components																																																												
16	structure to contain sys.																																																												
17	Pwr to Containment																																																												
18	Prov. Pwr to System																																																												
19	Temp. Pwr for Main																																																												
20	Counterbalancing of Containment																																																												
21	Counterbalancing of Ropes																																																												
22	Hold Containment at Destination																																																												
23	Maintain Cont. Guidance in pathway																																																												
24	Provide path for motion in Hwy																																																												
25	Calibration (Learn?)																																																												
26	Compute Trajectory																																																												
27	Control Motion to Trajectory																																																												
28	Process Service Request																																																												
29	Provide Current Position																																																												
30	Provide Trajectory Feedback																																																												
31	Top-of-car control inspection																																																												
32	Remote Control inspection																																																												
33	Detect if Load is within limits																																																												
34	Construction Control																																																												
35	Passenger Door Control																																																												
36	Access to interchange sys. info																																																												
37	Determine if on Emerg Power																																																												
38	Detect Uncontrolled Motion																																																												
39	Determine if safe to move																																																												
40	Stop freefall motion																																																												
41	Protect Load from Hoistway																																																												
42	Provide Safe Load Transfer																																																												
43	Control Access to Containment																																																												
44	independent end of Hwy sensor																																																												
45	Emerg terminal Stopping Dev																																																												
46	indep. Door Zone Sensing																																																												
47	Mechanical Overspeed detection																																																												
48	Provide end of Hwy cushion (bfr)																																																												
49	Detect failure. Slow for last landing NTSD																																																												
50	Deny access to hoistway																																																												
51	Detect Obstructed Access																																																												
52	Detect Earthquake																																																												
53	Detect Building Sway																																																												
54	Maintain safe cmf. load env.																																																												
55	Manage Noise																																																												
56	Manage Vertical Vibration																																																												
57	Manage Horiz. Vibration																																																												
58	Determine cause of malf.																																																												
59	Monitor System Health																																																												
60	Svc. Acc. to Hoistway																																																												
61	Load																																																												
62	Service Person																																																												

Appendix D Base Design Structure Matrix

Base DSM in Microsoft Excel

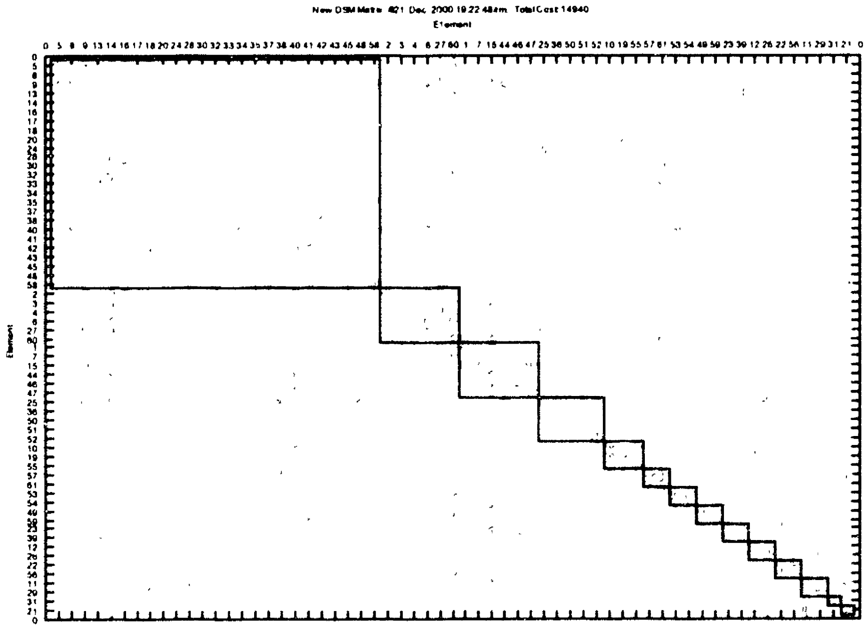


	A
1	
2	Rqst Indicator Hall (light)
3	Rqst Indicator Car (lighted bt)
4	Hallway Fixtures
5	Car Fixtures
6	Emerg. Intercom/Phone
7	Car Buttons
8	Hall Buttons
9	Firemans Service Key
10	Traveling Comm. Cable
11	Ropes (link motor to cab)
12	Motor (elec. to mech conversion)
13	Drive Power Section
14	Cab floor/plank
15	Power Supplies
16	Building Structure
17	Travel Cable Power
18	Building Power
19	Power Storage
20	Counterweight
21	Compensation System
22	Brake System
23	Cab Guidance System
24	Guides Rails
25	Learn Inct (mot. ctl)
26	Motion Controller
27	Electrical Drive Control
28	Operational Controller
29	Primary Position System
30	Primary Velocity Meas
31	Top-of-Car Insp. Ctl Panel
32	Remote Inspec. Ctl Panel
33	Load Weighing Sub-system
34	Construction Control Panel
35	Door Close Button
36	Service Tool
37	Emergency Generator Signal
38	Secondary Velocity Check
39	Safety Subsystem
40	Safeties Mechanism
41	Cab
42	Cab Doors
43	Door Controller
44	Terminal Sensing System
45	ETSD Sensor
46	Door Zone Sensors
47	Governor System
48	Terminal Buffer
49	NTSD Sensor
50	Hoistway Doors
51	Door Obstruction Sensor
52	Earthquake Sensor
53	Building Sway Sensor
54	Vent & Lighting System
55	Cab Insulation
56	Ropes Hitch Springs
57	Guide Springs/rail Align
58	Diagnostics
59	Remote Monitoring System
60	Access Keying on Hwy Door
61	Passenger Load
62	Service Guy

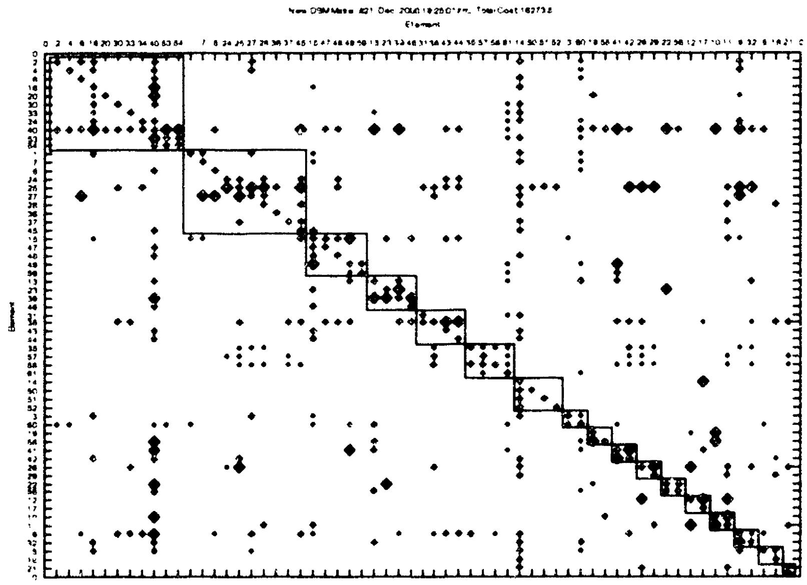


Graphical Representation of Base DSM using Matlab

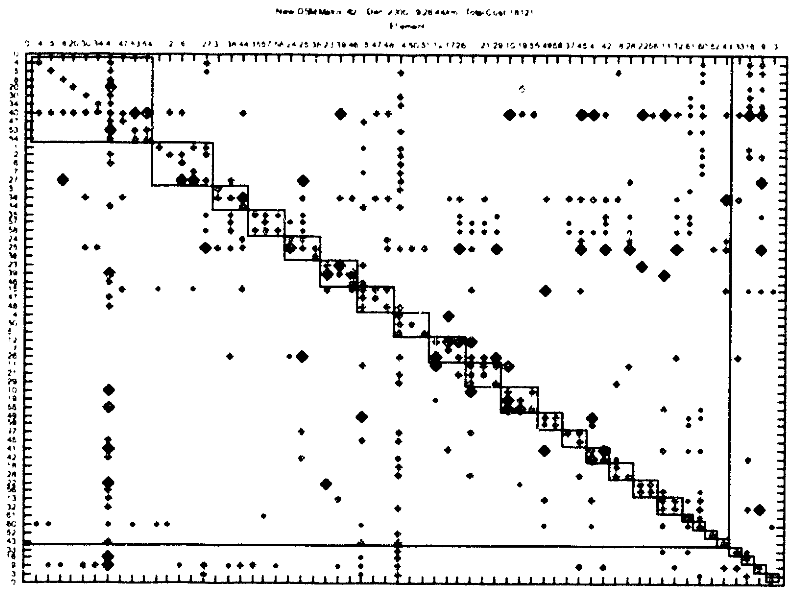
Appendix E Final Clustering Results



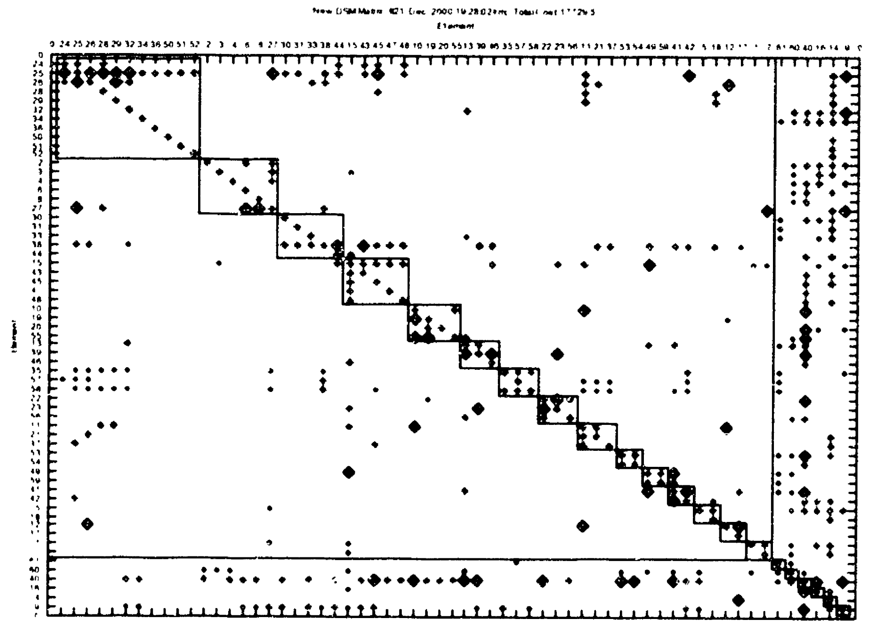
Clustered DSM using all elements.



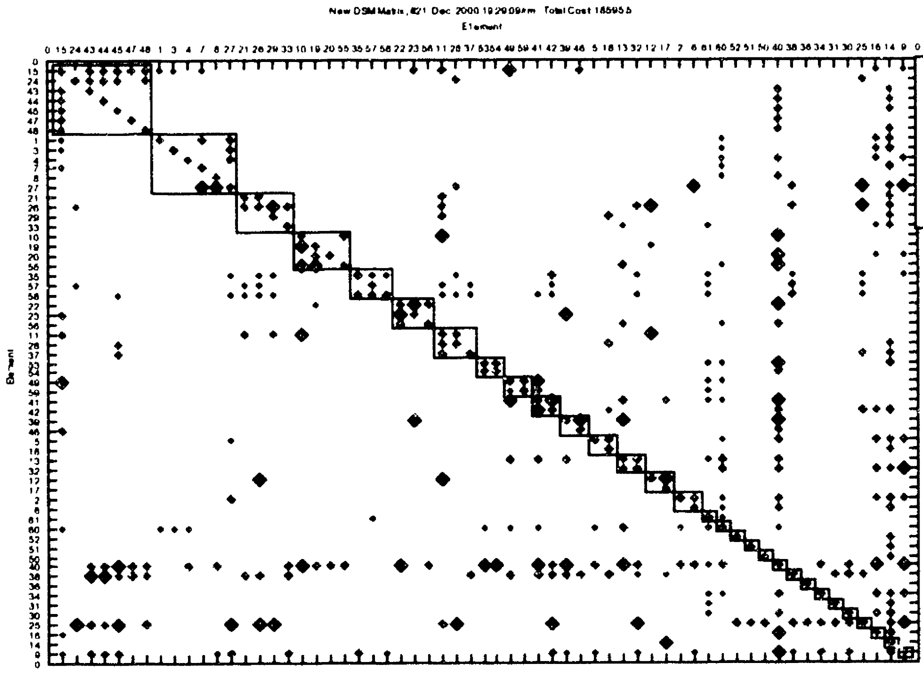
Clustered DSM. Using all parameters but varied interaction strengths.



Clustered DSM.
Varied Interaction Levels
4 System parameters excluded from cluster algorithm.



Clustered DSM.
Varied Interaction Levels
6 System parameters excluded from cluster algorithm.



Clustered DSM.
Varied Interaction Levels
8 System parameters excluded from cluster algorithm.

Appendix F Matlab Modified Clustering Routines

FileName	Page	Descriptions
Cluster Algorithm Functions and Scripts		
run_cluster_A	93	Master file to load DSM variables and run the clustering algorithm and graph results
Elevator_DSM	97	file to define the DSM with all interactions valued at 1
Elevator_DSM_varied	108	File to define the DSM with varied interaction strengths
bid	119	calculates bid for from each of the clusters for the chosen element
cluster	121	Clustering Algorithm that uses the costing routine to penalize multiple cluster membership
coord_cost	127	Calculates the coordination cost using the penalty for multiple cluster memberships
delete_clusters	133	deletes duplicate clusters
Cluster Graphing Functions		
reorder_dsm_bycluster	130	reorders the DSM according to the cluster matrix – also renters elements for every cluster that they are a member of
Reorder_cluster	132	Sort the clusters by cluster size
dsm_autolabel	135	assign numbers to the DSM entries for the graphing routines
place_diag	136	place 1's along the diagonals of the matrix
graph_matrix	137	graph the DSM matrix or cluster matrix
line_mult_cluster	139	place lines for duplicate clusters in the cluster matrix graph
plot_cluster_list	140	print list of cluster members
Cluster Analysis Calculations and Graphing		
likeness_calc	143	Script file to get the average match between clusters of several runs of the clustering algorithm.
find_cluster_matches	144	Function to find matching clusters from different runs of the clustering algorithm
get_match_avg	146	Function calculate the average of cluster matches between two runs of cluster calculations

run_cluster_A

```
% ************************************************************************
% ************************************************************************
% ************************************************************************
% ************************************************************************
% ************************************************************************
%                                                                        *
%  files:  run_cluster_A.m                                               *
%                                                                        *
%  Created by: Ronnie E  Thebeau                                         *
%             System Design and Management Program                      *
%             Massacusetts Institute of Technology                       *
%                                                                        *
%  Date: December 2000                                                   *
%                                                                        *
%                                                                        *
% Script file to load in the DSM for analysis, run the                   *
% clustering algorithm and graph the results                             *
%                                                                        *
% ************************************************************************
% ************************************************************************
% ************************************************************************
% ************************************************************************



% ************************************************************************
%                  SPECIFY & GET DSM MATRIX                                *
% ************************************************************************
% name of script file containing the DSM matrix and DSM labels           *
% must contain to variables                                              *
%                                                                        *
%    DSM           matrix defining the DSM interaction values            *
%                                                                        *
%    DSMLABEL         cell array containing strings for the labels         *
%                  of the DSM elements                                   *
%                                                                        *
%    DSMFunc_LABELcell array containg the strings which label            *
%                  the functional requirements that are mapped           *
%                  to the DSM elements                                   *
%                                                                        *

Elevator_DSM_varied; % enter name of DSM script file

% ******************** END GET DSM MATRIX ********************************



% ************************************************************************
%                         USER PARAMETERS                                *
% ************************************************************************
%    print flag       set to 1 to print the plotted figures              *
%                                                                        *
%    extract_elements   defines which elements will be considered          *
%                       as system elements and will not be used            *
%                       for the clustering purposes                        *
%                       The system parameters generally contain            *
%                       many entries along the row, column, or both        *
%                                                                          *
%    Cluster param      structure containing the parameters which          *
%                       control the clustering algorithm                   *
% **************************************************************************

print_flag = 0;
extract_elements = [9,16,60,61,40,14,38,25];

Cluster_param.pow_cc           = 1;   % penalty assigned to cluster size(2)
Cluster_param.pow_bid          = 1,   % high value penalizes large clusters (0-3,
2)
Cluster_param.pow_dep          = 4,   % high value emphasizes high interactions (0-
2, 2)
Cluster_param.max_cluster_size = 61;  % max size of cluster (DSM size)
Cluster_param.rand_accept      = 122, % proceed w/ 1 of N changes even if no
imp  (0.5-2 *DSM)
Cluster_param.rand_bid         = 122, % take second highest bid 1 out of N times
(0.5-2 * DSM)
Cluster_param.times            = 2;   % attept time*size before check sys.
stability (2)
Cluster_param.stable_limit     = 2,   % loop at least stable limit*times*size
(2)

% **************************************************************************
% *********************  END USER PARAMETERS *******************************
% **************************************************************************




% **************************************************************************
%                    EXTRACT SPECIFIED SYSTEM PARAMETERS                   *
% **************************************************************************

DSM_matrix = DSM;
DSM_matrix_original = DSM_matrix, % save a copy of the original DSM

% set system parameters to zero to remove their influence during clustering
for i = 1: length(extract_elements)
   DSM_matrix(extract_elements(i),:) = 0;
   DSM_matrix(:,extract_elements(i)) = 0;
end
% ******************  END EXTRACT SYSTEM PARAMETERS ************************




% **************************************************************************
%                           RUN CLUSTERING                                 *
% **************************************************************************

[Cluster_matrix, total_coord_cost, cost_history, old_data] =
Cluster(DSM_matrix, Cluster_param),

% ************************  END RUN CLUSTERING *****************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
%                                                                           *
%                          GRAPHING                                           *
%                                                                           *
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************

% sort the cluster matrix by cluster size
[Cluster_matrix] = reorder_cluster(Cluster_matrix);  % sort cluster by cluster
size

% Get label of axes for DSM and cluster graphs
[Cluster_label] = DSM_autolabel(Cluster_matrix);

% create generic number labels that correspond to the location
% in the original DSM
[DSM_labels] = DSM_autolabel(DSM_matrix);


% re-order the DSM matrix and labels according to the results of the cluster
matrix
[New_DSM_matrix, New_DSM_labels] = reorder_DSM_byCluster(DSM_matrix_original,
Cluster_matrix, DSM_labels);

% Place the value 1 along the diagonal of the DSM
[graph_DSM_matrix] = place_diag(DSM_matrix_original, 1);
[graph_New_DSM_matrix] = place_diag(New_DSM_matrix, 1);

% GRAPH THE MATRICES
get_date = now;    % get date and time for graphing
current_date = datestr(get_date,0);

% Create titles for the graphs
DSM_title = ['DSM Matrix;  ' current_date];
Cluster_title = ['Cluster Matrix;  '  current_date];
New_DSM_title = ['New DSM Matrix; ' current_date ';   Total Cost: '
num2str(total_coord_cost)];

% graph the original DSM Matrix
graph_matrix(graph_DSM_matrix,'Element','Element',DSM_title, DSM_labels,
DSM_labels, print_flag);

% graph the Cluster Matrix
graph_matrix(Cluster_matrix,'Element','Cluster',Cluster_title, DSM_labels,
Cluster_label, 0);

% add lines on the cluster matrix to identify elements that belong
% to more than one cluster
```

```
line_mult_cluster(Cluster_matrix,gcf, gca);
if print_flag==1
   print;
end

% graph the new DSM matrix reodered by the cluster assignments
graph_matrix(graph_New_DSM_matrix,'Element','Element',New_DSM_title,
New DSM_labels, New_DSM_labels, print_flag, Cluster_matrix);

% create a text list of the elements of each cluster
plot_cluster_list(Cluster_matrix, DSM_matrix, DSMLABEL, current_date,
print_flag);
plot_cluster_list(Cluster_matrix, DSM_matrix, DSMFunc_LABEL, current_date,
print_flag);


% get only the nonzero value in the cost history
% cutoff the end of the history array which was not filled in
[cost_g_zero, cg] = find(cost_history);
max_run = max(cost_g_.ero);

% plot the cost history
plot(cost_history(1:max_run)),
title(['Clustering Cost History;  ' current_date]),
xlabel('Change #'),
ylabel('Cost');
orient landscape;
if print_flag ==1
   print;
end
% ***************************************************************************
%                          END GRAPHING                                    *
% ***************************************************************************
```

Elevator_DSM

```
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
%                                                                           *
%  Files:  Elevator_DSM.m                                                   *
%                                                                           *
%  Created by: Ronnie E. Thebeau                                            *
%              System Design and Management Program                         *
%              Massacusetts Institute of Technology                         *
%                                                                           *
%  Date: December 2000                                                      *
%                                                                           *
% Entries indicate an interaction between two elements and the value        *
% represents the interaction strength                                       *
%                                                                           *
% This DSM represents a subset of elements and interactions  within a          *
% generic elevator system                                                   *
%                                                                           *
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************



% ***************************************************************************
%                        DSM TEMPLATE                                       *
% ***************************************************************************

DSM_size = 61;          % number of elements in the DSM
DSM = zeros(DSM_size);

% ***** DSM ENTRIES *****

DSM(1,1)   = 1;
DSM(15,1) = 1;
DSM(60,1) = 1;

DSM(2,2)   = 1;
DSM(40,2)  = 1;
DSM(60,2)  = 1;

DSM(3,3)   = 1;
DSM(15,3)  = 1;
DSM(60,3)  = 1;

DSM(4,4)   = 1;
DSM(40,4)  = 1;
DSM(60,4)  = 1;

DSM(5,5)   = 1;
DSM(9,5)   = 1;
DSM(40,5)  = 1;
DSM(60,5)  = 1;

DSM(2,6)   = 1;
DSM(6,6)   = 1;
DSM(9,6)   = 1;
DSM(27,6)  = 1;
DSM(40,6)  = 1;

DSM(1,7)   = 1;
DSM(7,7)   = 1;
DSM(15,7)  = 1;
DSM(27,7)  = 1;

DSM(8,8)   = 1;
DSM(9,8)   = 1;
DSM(27,8)  = 1;
DSM(40,8)  = 1;

DSM(2,9)   = 1;
DSM(4,9)   = 1;
DSM(5,9)   = 1;
DSM(9,9)   = 1;
DSM(15,9)  = 1;
DSM(20,9)  = 1;
DSM(25,9)  = 1;
DSM(27,9)  = 1;
DSM(32,9)  = 1;
DSM(34,9)  = 1;
DSM(35,9)  = 1;
DSM(40,9)  = 1;

DSM(10,10) = 1;
DSM(11,10) = 1;
DSM(19,10) = 1;
DSM(40,10) = 1;
DSM(55,10) = 1;

DSM(10,11) = 1;
DSM(11,11) = 1;
DSM(12,11) = 1;
DSM(15,11) = 1;
DSM(21,11) = 1;
DSM(25,11) = 1;
DSM(26,11) = 1;
DSM(28,11) = 1;
DSM(29,11) = 1;
DSM(37,11) = 1;
DSM(57,11) = 1;
DSM(58,11) = 1;

DSM(11,12) = 1;
DSM(12,12) = 1;
DSM(19,12) = 1;
DSM(26,12) = 1;

DSM(13,13) = 1;
DSM(32,13) = 1;
DSM(33,13) = 1;
DSM(39,13) = 1;
DSM(40,13) = 1;
DSM(41,13) = 1;
DSM(55,13) = 1;
DSM(56,13) = 1;
DSM(60,13) = 1;

DSM(1,14)  = 1;
DSM(2,14)  = 1;
DSM(3,14)  = 1;
DSM(5,14)  = 1;
DSM(6,14)  = 1;
DSM(7,14)  = 1;
DSM(9,14)  = 1;
DSM(14,14) = 1;
DSM(16,14) = 1;
DSM(18,14) = 1;
DSM(21,14) = 1;
DSM(25,14) = 1;
DSM(26,14) = 1;
DSM(27,14) = 1;
DSM(28,14) = 1;
DSM(29,14) = 1;
DSM(30,14) = 1;
DSM(31,14) = 1;
DSM(32,14) = 1;
DSM(33,14) = 1;
DSM(34,14) = 1;
DSM(37,14) = 1;
DSM(42,14) = 1;
DSM(43,14) = 1;
DSM(44,14) = 1;
DSM(45,14) = 1;
DSM(48,14) = 1;
DSM(50,14) = 1;
DSM(51,14) = 1;
DSM(52,14) = 1;
DSM(53,14) = 1;

DSM(1,15)  = 1;
DSM(3,15)  = 1;
DSM(7,15)  = 1;
DSM(9,15)  = 1;
DSM(11,15) = 1;
DSM(15,15) = 1;
DSM(16,15) = 1;
DSM(23,15) = 1;
DSM(43,15) = 1;
DSM(44,15) = 1;
DSM(45,15) = 1;
DSM(46,15) = 1;
DSM(47,15) = 1;
DSM(48,15) = 1;
DSM(49,15) = 1;
DSM(60,15) = 1;

DSM(1,16)  = 1;
DSM(2,16)  = 1;
DSM(3,16)  = 1;
DSM(4,16)  = 1;
DSM(5,16)  = 1;
DSM(15,16) = 1;
DSM(16,16) = 1;
DSM(20,16) = 1;
DSM(30,16) = 1;
DSM(32,16) = 1;
DSM(33,16) = 1;
DSM(34,16) = 1;
DSM(40,16) = 1;
DSM(42,16) = 1;
DSM(12,17) = 1;
DSM(14,17) = 1;
DSM(17,17) = 1;
DSM(38,17) = 1;
DSM(41,17) = 1;

DSM(5,18)  = 1;
DSM(18,18) = 1;
DSM(28,18) = 1;
DSM(29,18) = 1;
DSM(38,18) = 1;
DSM(42,18) = 1;

DSM(19,19) = 1;
DSM(20,19) = 1;
DSM(22,19) = 1;
DSM(40,19) = 1;
DSM(55,19) = 1;

DSM(20,20) = 1;
DSM(40,20) = 1;

DSM(11,21) = 1;
DSM(21,21) = 1;
DSM(26,21) = 1;
DSM(38,21) = 1;
DSM(57,21) = 1;
DSM(58,21) = 1;

DSM(22,22) = 1;
DSM(23,22) = 1;
DSM(40,22) = 1;
DSM(56,22) = 1;

DSM(15,23) = 1;
DSM(22,23) = 1;
DSM(23,23) = 1;
DSM(39,23) = 1;

DSM(24,24) = 1;
DSM(25,24) = 1;
DSM(26,24) = 1;
DSM(57,24) = 1;

DSM(24,25) = 1;
DSM(25,25) = 1;
DSM(26,25) = 1;
DSM(27,25) = 1;
DSM(35,25) = 1;
DSM(37,25) = 1;
DSM(38,25) = 1;
DSM(42,25) = 1;
DSM(57,25) = 1;
DSM(58,25) = 1;

DSM(12,26) = 1;
DSM(21,26) = 1;
DSM(25,26) = 1;
DSM(26,26) = 1;
DSM(35,26) = 1;
DSM(38,26) = 1;
DSM(57,26) = 1;
DSM(58,26) = 1;
DSM(1,27)  = 1;
DSM(2,27)  = 1;
DSM(3,27)  = 1;
DSM(4,27)  = 1;
DSM(5,27)  = 1;
DSM(9,27)  = 1;
DSM(25,27) = 1;
DSM(27,27) = 1;
DSM(35,27) = 1;
DSM(58,27) = 1;

DSM(11,28) = 1;
DSM(24,28) = 1;
DSM(25,28) = 1;
DSM(27,28) = 1;
DSM(28,28) = 1;
DSM(35,28) = 1;
DSM(58,28) = 1;

DSM(11,29) = 1;
DSM(25,29) = 1;
DSM(26,29) = 1;
DSM(29,29) = 1;
DSM(35,29) = 1;
DSM(58,29) = 1;

DSM(9,30)  = 1;
DSM(25,30) = 1;
DSM(30,30) = 1;
DSM(38,30) = 1;
DSM(40,30) = 1;

DSM(25,31) = 1;
DSM(31,31) = 1;
DSM(38,31) = 1;

DSM(9,32)  = 1;
DSM(13,32) = 1;
DSM(25,32) = 1;
DSM(26,32) = 1;
DSM(32,32) = 1;
DSM(35,32) = 1;
DSM(38,32) = 1;
DSM(40,32) = 1;
DSM(58,32) = 1;

DSM(9,33)  = 1;
DSM(26,33) = 1;
DSM(33,33) = 1;
DSM(38,33) = 1;
DSM(40,33) = 1;

DSM(9,34)  = 1;
DSM(25,34) = 1;
DSM(34,34) = 1;
DSM(40,34) = 1;

DSM(9,35)  = 1;
DSM(35,35) = 1;
DSM(58,35) = 1;

DSM(25,36) = 1;
DSM(36,36) = 1;

DSM(37,37) = 1;
DSM(38,37) = 1;
DSM(57,37) = 1;
DSM(58,37) = 1;

DSM(9,38)  = 1;
DSM(25,38) = 1;
DSM(26,38) = 1;
DSM(27,38) = 1;
DSM(35,38) = 1;
DSM(38,38) = 1;
DSM(57,38) = 1;
DSM(58,38) = 1;

DSM(9,39)  = 1;
DSM(13,39) = 1;
DSM(23,39) = 1;
DSM(38,39) = 1;
DSM(39,39) = 1;
DSM(40,39) = 1;

DSM(2,40)  = 1;
DSM(4,40)  = 1;
DSM(5,40)  = 1;
DSM(6,40)  = 1;
DSM(8,40)  = 1;
DSM(9,40)  = 1;
DSM(10,40) = 1;
DSM(13,40) = 1;
DSM(16,40) = 1;
DSM(20,40) = 1;
DSM(22,40) = 1;
DSM(30,40) = 1;
DSM(32,40) = 1;
DSM(34,40) = 1;
DSM(39,40) = 1;
DSM(40,40) = 1;
DSM(41,40) = 1;
DSM(42,40) = 1;
DSM(43,40) = 1;
DSM(44,40) = 1;
DSM(45,40) = 1;
DSM(46,40) = 1;
DSM(47,40) = 1;
DSM(48,40) = 1;
DSM(53,40) = 1;
DSM(54,40) = 1;
DSM(55,40) = 1;
DSM(56,40) = 1;
DSM(60,40) = 1;

DSM(13,41) = 1;
DSM(38,41) = 1;
DSM(40,41) = 1;
DSM(41,41) = 1;
DSM(42,41) = 1;
DSM(49,41) = 1;
DSM(58,41) = 1;
DSM(59,41) = 1;
DSM(60,41) = 1;
DSM(9,42)  = 1;
DSM(25,42) = 1;
DSM(35,42) = 1;
DSM(38,42) = 1;
DSM(40,42) = 1;
DSM(41,42) = 1;
DSM(42,42) = 1;
DSM(57,42) = 1;
DSM(58,42) = 1;

DSM(9,43)  = 1;
DSM(15,43) = 1;
DSM(24,43) = 1;
DSM(25,43) = 1;
DSM(38,43) = 1;
DSM(40,43) = 1;
DSM(43,43) = 1;

DSM(9,44)  = 1;
DSM(15,44) = 1;
DSM(24,44) = 1;
DSM(25,44) = 1;
DSM(38,44) = 1;
DSM(40,44) = 1;
DSM(44,44) = 1;

DSM(9,45)  = 1;
DSM(15,45) = 1;
DSM(24,45) = 1;
DSM(25,45) = 1;
DSM(28,45) = 1;
DSM(37,45) = 1;
DSM(38,45) = 1;
DSM(40,45) = 1;
DSM(45,45) = 1;
DSM(58,45) = 1;

DSM(15,46) = 1;
DSM(38,46) = 1;
DSM(39,46) = 1;
DSM(46,46) = 1;

DSM(15,47) = 1;
DSM(38,47) = 1;
DSM(40,47) = 1;
DSM(47,47) = 1;

DSM(15,48) = 1;
DSM(24,48) = 1;
DSM(25,48) = 1;
DSM(38,48) = 1;
DSM(40,48) = 1;
DSM(48,48) = 1;

DSM(13,49) = 1;
DSM(15,49) = 1;
DSM(38,49) = 1;
DSM(41,49) = 1;
DSM(49,49) = 1;
DSM(59,49) = 1;
DSM(60,49) = 1;

DSM(25,50) = 1;
DSM(50,50) = 1;

DSM(25,51) = 1;
DSM(51,51) = 1;

DSM(25,52) = 1;
DSM(52,52) = 1;

DSM(40,53) = 1;
DSM(53,53) = 1;
DSM(54,53) = 1;
DSM(60,53) = 1;

DSM(40,54) = 1;
DSM(53,54) = 1;
DSM(54,54) = 1;

DSM(10,55) = 1;
DSM(40,55) = 1;
DSM(55,55) = 1;

DSM(22,56) = 1;
DSM(40,56) = 1;
DSM(56,56) = 1;

DSM(35,57) = 1;
DSM(57,57) = 1;
DSM(58,57) = 1;
DSM(61,57) = 1;

DSM(9,58)  = 1;
DSM(35,58) = 1;
DSM(58,58) = 1;

DSM(49,59) = 1;
DSM(59,59) = 1;

DSM(1,60)  = 1;
DSM(2,60)  = 1;
DSM(3,60)  = 1;
DSM(4,60)  = 1;
DSM(5,60)  = 1;
DSM(6,60)  = 1;
DSM(7,60)  = 1;
DSM(8,60)  = 1;
DSM(13,60) = 1;
DSM(32,60) = 1;
DSM(34,60) = 1;
DSM(40,60) = 1;
DSM(41,60) = 1;
DSM(49,60) = 1;
DSM(53,60) = 1;
DSM(55,60) = 1;
DSM(56,60) = 1;
DSM(60,60) = 1;
DSM(61,60) = 1;

DSM(13,61) = 1;
DSM(30,61) = 1;
DSM(31,61) = 1;
DSM(33,61) = 1;
DSM(34,61) = 1;
DSM(35,61) = 1;
DSM(40,61) = 1;
DSM(41,61) = 1;
DSM(49,61) = 1;
DSM(53,61) = 1;
DSM(57,61) = 1;
DSM(58,61) = 1;
DSM(59,61) = 1;
DSM(60,61) = 1;
DSM(61,61) = 1;


% **************************************************************************
%                         DSM Elements Labels                               *
% **************************************************************************

DSMLABEL = cell(DSM_size,1);

DSMLABEL{1,1} = 'Hall Request Indicator';
DSMLABEL{2,1} = 'Car Request Indicator';
DSMLABEL{3,1} = 'Hallway Fixtures';
DSMLABEL{4,1} = 'Car Fixtures';
DSMLABEL{5,1} = 'Emerg. Intercom-Phone';
DSMLABEL{6,1} = 'Car Buttons';
DSMLABEL{7,1} = 'Hall Buttons';
DSMLABEL{8,1} = 'Firemans Service key';
DSMLABEL{9,1} = 'Travelling Comm. Cable';

DSMLABEL{10,1} = 'Ropes';
DSMLABEL{11,1} = 'Motor';
DSMLABEL{12,1} = 'Drive Power Section';
DSMLABEL{13,1} = 'Cab Floor/Plank';
DSMLABEL{14,1} = 'Power Supplies';
DSMLABEL{15,1} = 'Building Structure';
DSMLABEL{16,1} = 'Travel Cable Power';
DSMLABEL{17,1} = 'Building Power';
DSMLABEL{18,1} = 'Power Storage';
DSMLABEL{19,1} = 'Counterweight';

DSMLABEL{20,1} = 'Compensation System';
DSMLABEL{21,1} = 'Brake System';
DSMLABEL{22,1} = 'Cab Guidance System';
DSMLABEL{23,1} = 'Guide Rails';
DSMLABEL{24,1} = 'Learn Function';
DSMLABEL{25,1} = 'Motion Controller';
DSMLABEL{26,1} = 'Electrical Drive Control';
DSMLABEL{27,1} = 'Operational Controller';
DSMLABEL{28,1} = 'Primary Position System';
DSMLABEL{29,1} = 'Primary Velcotiy Measement';

DSMLABEL{30,1} = 'Top-of-Car Insp. Ctl Panel';
DSMLABEL{31,1} = 'Remote Insp. Ctl. Panel';
DSMLABEL{32,1} = 'Load Weighing System';
DSMLABEL{33,1} = 'Construction Control Panel';
DSMLABEL{34,1} = 'Door Close Button';
DSMLABEL{35,1} = 'Service Tool';
DSMLABEL{36,1} = 'Emerg. Gen. Signal';
DSMLABEL{37,1} = 'Secondary Veloc. Check';
DSMLABEL{38,1} = 'Safety System';
DSMLABEL{39,1} = 'Mechanical Safeties System';

DSMLABEL{40,1} = 'Cab';
DSMLABEL{41,1} = 'Cab Doors';
```

```
DSMLABEL{42,1} = 'Door Controller';
DSMLABEL{43,1} = 'Terminal Sensing System';
DSMLABEL{44,1} = 'ETSD Sensor';
DSMLABEL{45,1} = 'Door Zone Sensors';
DSMLABEL{46,1} = 'Governor System';
DSMLABEL{47,1} = 'Terminal Buffer';
DSMLABEL{48,1} = 'NTSD Sensor';
DSMLABEL{49,1} = 'Hoistway Doors';

DSMLABEL{50,1} = 'Door Obstruction Sensor';
DSMLABEL{51,1} = 'Earthquake Sensor';
DSMLABEL{52,1} = 'Building Sway Sensor';
DSMLABEL{53,1} = 'Vent & Lighting System';
DSMLABEL{54,1} = 'Cab Insullation';
DSMLABEL{55,1} = 'Ropes/Hitch Springs';
DSMLABEL{56,1} = 'Guide Springs/ Rail Align';
DSMLABEL{57,1} = 'Diagnostics';
DSMLABEL{58,1} = 'Remote Monitoring System';
DSMLABEL{59,1} = 'Acces keying on Hstway Doors';

DSMLABEL{60,1} = 'Passenger Load';
DSMLABEL{61,1} = 'Service Guy';


% ************************************************************************
%            Functional Mapping to Physical Elements                     *
% ************************************************************************

% Each of the functional labels represents the functional
% requirement for which the physcial DSM element represents
% Used to cross-reference the physical elemnts and
% functional requiremnts

DSMFunc_LABEL = cell(DSM_size,1);

DSMFunc_LABEL{1,1} = 'Acknowledge Hall Svc Rqst';
DSMFunc_LABEL{2,1} = 'Acknowledge Car Service Rqst';
DSMFunc_LABEL{3,1} = 'Comm. Status w/ Hall Pass.';
DSMFunc_LABEL{4,1} = 'Comm. Status w/ Car Pass.';
DSMFunc_LABEL{5,1} = 'Emerg. Comm. w/ car pass.';
DSMFunc_LABEL{6,1} = 'Input Car Rqst';
DSMFunc_LABEL{7,1} = 'Input Hall Rqst';
DSMFunc_LABEL{8,1} = 'Input Firemans Svc Rqst';
DSMFunc_LABEL{9,1} = 'Provide Comm. Link to Pass';

DSMFunc_LABEL{10,1} = 'Xmit force to containment';
DSMFunc_LABEL{11,1} = 'Energy Transformation to Force';
DSMFunc_LABEL{12,1} = 'Provide Energy for Motion';
DSMFunc_LABEL{13,1} = 'Support load for movement';
DSMFunc_LABEL{14,1} = 'Provide power to comp.';
DSMFunc_LABEL{15,1} = 'Provide structure to contain Sys.';
DSMFunc_LABEL{16,1} = 'Provide power to containment';
DSMFunc_LABEL{17,1} = 'Provide power for system';
DSMFunc_LABEL{18,1} = 'Temporary power for memory';
DSMFunc_LABEL{19,1} = 'Provide Counterbalancing to contain.';

DSMFunc_LABEL{20,1} = 'Provide coubterbalancing of ropes';
DSMFunc_LABEL{21,1} = 'Hold containment at destination';
DSMFunc_LABEL{22,1} = 'Maintain guidance in pathway';
DSMFunc_LABEL{23,1} = 'Provide path for motion in Hwy';
DSMFunc_LABEL{24,1} = 'System Calibration';
DSMFunc_LABEL{25,1} = 'Compute Trajectory';
DSMFunc_LABEL{26,1} = 'Control Motion to trajectory';
```

```
DSMFunc_LABEL{27,1} = 'Process Service Request';
DSMFunc_LABEL{28,1} = 'Provide current position';
DSMFunc_LABEL{29,1} = 'Provide Trajectory feedback';

DSMFunc_LABEL{30,1} = 'Top-of-car control (inspection)';
DSMFunc_LABEL{31,1} = 'Remote Control (inspection)';
DSMFunc_LABEL{32,1} = 'Detect if load is within limits';
DSMFunc_LABEL{33,1} = 'Contruction Control';
DSMFunc_LABEL{34,1} = 'Passenger Door control';
DSMFunc_LABEL{35,1} = 'Provide access to view/change sys. info.';
DSMFunc_LABEL{36,1} = 'Determine if on Emerg. Power';
DSMFunc_LABEL{37,1} = 'Detect Uncontrolled motion';
DSMFunc_LABEL{38,1} = 'Determine if safe to move';
DSMFunc_LABEL{39,1} = 'Stop free-fall motion';

DSMFunc_LABEL{40,1} = 'Protect load from hwy';
DSMFunc_LABEL{41,1} = 'Provide safe load transfer';
DSMFunc_LABEL{42,1} = 'Control acces to containment';
DSMFunc_LABEL{43,1} = 'independent end of hwy sensor';
DSMFunc_LABEL{44,1} = 'Emerg. Terminal Stopiing device';
DSMFunc_LABEL{45,1} = 'Indep. Door zone sensing';
DSMFunc_LABEL{46,1} = 'Mechanical Overspeed detection';
DSMFunc_LABEL{47,1} = 'Provide end of Hwy cushion (bffr)';
DSMFunc_LABEL{48,1} = 'Detect failure slow for last landing -NTSD';
DSMFunc_LABEL{49,1} = 'Deny access to hwy';

DSMFunc_LABEL{50,1} = 'Detect obstructed access';
DSMFunc_LABEL{51,1} = 'Respond to Earthquake';
DSMFunc_LABEL{52,1} = 'Detect Building sway';
DSMFunc_LABEL{53,1} = 'Maintainsafe/comf load env.';
DSMFunc_LABEL{54,1} = 'Manage noise';
DSMFunc_LABEL{55,1} = 'Manage vertical vibration';
DSMFunc_LABEL{56,1} = 'Manage horizontal vibration';
DSMFunc_LABEL{57,1} = 'Determine cause of malfunction';
DSMFunc_LABEL{58,1} = 'Monitor system health';
DSMFunc_LABEL{59,1} = 'provide service access to hwy';

DSMFunc_LABEL{60,1} = 'Passenger Load';
DSMFunc_LABEL{61,1} = 'Service Guy';
```

Elevator_DSM_varied

```
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
%                                                                           *
%  Files:  Elevator_DSM_varied.m                                            *
%                                                                           *
%  Created by: Ronnie E. Thebeau                                            *
%              System Design and Management Program                         *
%              Massacusetts Institute of Technology                         *
%                                                                           *
%  Date: December 2000                                                      *
%                                                                           *
% Script file to create the Design Structure Matrix                         *
%                                                                           *
% Entries indicate an interaction between two elements and the value        *
% represents the interaction strength                                       *
%                                                                           *
% This DSM represents a subset of elements and interactions  within a       *
% generic elevator system                                                   *
%                                                                           *
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************


DSM_size = 61;          % number of elements in the DSM
DSM = zeros(DSM_size);

% ***** DSM ENTRIES *****
% interactions between the elements of the DSM

DSM(1,1)   = 1;
DSM(15,1) = 0.5;
DSM(60,1) = 0.5;

DSM(2,2)   = 1;
DSM(40,2)  = 1;
DSM(60,2)  = 0.5;

DSM(3,3)   = 1;
DSM(15,3)  = 0.5;
DSM(60,3)  = 0.5;

DSM(4,4)   = 1;
DSM(40,4)  = 1;
DSM(60,4)  = 0.5;

DSM(5,5)   = 1;
DSM(9,5)   = 0.5;
DSM(40,5)  = 1;
DSM(60,5)  = 0.5;

DSM(2,6)   = 1;
DSM(6,6)   = 1;
DSM(9,6)   = 0.5;
DSM(27,6)  = 2;
DSM(40,6)  = 1;

DSM(1,7)   = 1;
DSM(7,7)   = 1;
DSM(15,7)  = 0.5;
DSM(27,7)  = 2;

DSM(8,8)   = 1;
DSM(9,8)   = 0.5;
DSM(27,8)  = 2;
DSM(40,8)  = 1;

DSM(2,9)   = 1;
DSM(4,9)   = 1;
DSM(5,9)   = 1;
DSM(9,9)   = 1;
DSM(15,9)  = 0.5;
DSM(20,9)  = 0.5;
DSM(25,9)  = 2;
DSM(27,9)  = 2;
DSM(32,9)  = 2;
DSM(34,9)  = 1;
DSM(35,9)  = 0.5;
DSM(40,9)  = 2;

DSM(10,10) = 1;
DSM(11,10) = 2;
DSM(19,10) = 2;
DSM(40,10) = 2;
DSM(55,10) = 2;

DSM(10,11) = 2;
DSM(11,11) = 1;
DSM(12,11) = 2;
DSM(15,11) = 1;
DSM(21,11) = 1;
DSM(25,11) = 1;
DSM(26,11) = 1;
DSM(28,11) = 1;
DSM(29,11) = 1;
DSM(37,11) = 1;
DSM(57,11) = 0.5;
DSM(58,11) = 0.5;

DSM(11,12) = 2;
DSM(12,12) = 1;
DSM(19,12) = 0.5;
DSM(26,12) = 2;

DSM(13,13) = 1;
DSM(32,13) = 1;
DSM(33,13) = 0.5;
DSM(39,13) = 2;
DSM(40,13) = 2;
DSM(41,13) = 1;
DSM(55,13) = 1;
DSM(56,13) = 1;
DSM(60,13) = 1;

DSM(1,14)  = 1;
DSM(2,14)  = 1;
DSM(3,14)  = 1;
DSM(5,14)  = 1;
DSM(6,14)  = 1;
DSM(7,14)  = 1;
DSM(9,14)  = 1;
DSM(14,14) = 1;
DSM(16,14) = 1;
DSM(18,14) = 1;
DSM(21,14) = 1;
DSM(25,14) = 1;
DSM(26,14) = 1;
DSM(27,14) = 1;
DSM(28,14) = 1;
DSM(29,14) = 1;
DSM(30,14) = 1;
DSM(31,14) = 1;
DSM(32,14) = 1;
DSM(33,14) = 1;
DSM(34,14) = 1;
DSM(37,14) = 1;
DSM(42,14) = 1;
DSM(43,14) = 1;
DSM(44,14) = 1;
DSM(45,14) = 1;
DSM(48,14) = 1;
DSM(50,14) = 1;
DSM(51,14) = 1;
DSM(52,14) = 1;
DSM(53,14) = 1;

DSM(1,15)  = 0.5;
DSM(3,15)  = 0.5;
DSM(7,15)  = 0.5;
DSM(9,15)  = 0.5;
DSM(11,15) = 1;
DSM(15,15) = 1;
DSM(16,15) = 0.5;
DSM(23,15) = 1;
DSM(43,15) = 1;
DSM(44,15) = 1;
DSM(45,15) = 1;
DSM(46,15) = 1;
DSM(47,15) = 1;
DSM(48,15) = 1;
DSM(49,15) = 2;
DSM(60,15) = 0.5;

DSM(1,16)  = 1;
DSM(2,16)  = 1;
DSM(3,16)  = 1;
DSM(4,16)  = 1;
DSM(5,16)  = 1;
DSM(15,16) = 0.5;
DSM(16,16) = 1;
DSM(20,16) = 0.5;
DSM(30,16) = 1;
DSM(32,16) = 1;
DSM(33,16) = 0.5;
DSM(34,16) = 1;
DSM(40,16) = 2;
DSM(42,16) = 1;

DSM(12,17) = 2;
DSM(14,17) = 2;
DSM(17,17) = 1;
DSM(38,17) = 0.5;
DSM(41,17) = 1;

DSM(5,18)  = 1;
DSM(18,18) = 1;
DSM(28,18) = 1;
DSM(29,18) = 1;
DSM(38,18) = 1;
DSM(42,18) = 0.5;

DSM(19,19) = 1;
DSM(20,19) = 1;
DSM(22,19) = 0.5;
DSM(40,19) = 1;
DSM(55,19) = 2;

DSM(20,20) = 1;
DSM(40,20) = 1;

DSM(11,21) = 1;
DSM(21,21) = 1;
DSM(26,21) = 1;
DSM(38,21) = 1;
DSM(57,21) = 0.5;
DSM(58,21) = 0.5;

DSM(22,22) = 1;
DSM(23,22) = 2;
DSM(40,22) = 2;
DSM(56,22) = 1;

DSM(15,23) = 1;
DSM(22,23) = 2;
DSM(23,23) = 1;
DSM(39,23) = 2;

DSM(24,24) = 1;
DSM(25,24) = 2;
DSM(26,24) = 0.5;
DSM(57,24) = 0.5;

DSM(24,25) = 1;
DSM(25,25) = 1;
DSM(26,25) = 2;
DSM(27,25) = 2;
DSM(35,25) = 0.5;
DSM(37,25) = 1;
DSM(38,25) = 1;
DSM(42,25) = 1;
DSM(57,25) = 0.5;
DSM(58,25) = 0.5;

DSM(12,26) = 2;
DSM(21,26) = 1;
DSM(25,26) = 2;
DSM(26,26) = 1;
DSM(35,26) = 0.5;
DSM(38,26) = 1;
DSM(57,26) = 0.5;
DSM(58,26) = 0.5;

DSM(1,27)  = 1;
DSM(2,27)  = 1;
DSM(3,27)  = 1;
DSM(4,27)  = 1;
DSM(5,27)  = 0.5;
DSM(9,27)  = 1;
DSM(25,27) = 2;
DSM(27,27) = 1;
DSM(35,27) = 0.5;
DSM(58,27) = 0.5;

DSM(11,28) = 1;
DSM(24,28) = 1;
DSM(25,28) = 2;
DSM(27,28) = 1;
DSM(28,28) = 1;
DSM(35,28) = 0.5;
DSM(58,28) = 0.5;

DSM(11,29) = 1;
DSM(25,29) = 2;
DSM(26,29) = 2;
DSM(29,29) = 1;
DSM(35,29) = 0.5;
DSM(58,29) = 0.5;

DSM(9,30)  = 1;
DSM(25,30) = 1;
DSM(30,30) = 1;
DSM(38,30) = 1;
DSM(40,30) = 1;

DSM(25,31) = 1;
DSM(31,31) = 1;
DSM(38,31) = 1;

DSM(9,32)  = 0.5;
DSM(13,32) = 1;
DSM(25,32) = 2;
DSM(26,32) = 1;
DSM(32,32) = 1;
DSM(35,32) = 0.5;
DSM(38,32) = 0.5;
DSM(40,32) = 1;
DSM(58,32) = 0.5;

DSM(9,33)  = 1;
DSM(26,33) = 1;
DSM(33,33) = 1;
DSM(38,33) = 1;
DSM(40,33) = 1;

DSM(9,34)  = 1;
DSM(25,34) = 1;
DSM(34,34) = 1;
DSM(40,34) = 1;

DSM(9,35)  = 1;

DSM(35,35) = 1;
DSM(58,35) = 1;

DSM(25,36) = 1;
DSM(36,36) = 1;
DSM(37,37) = 1;
DSM(38,37) = 1;
DSM(57,37) = 0.5;
DSM(58,37) = 0.5;

DSM(9,38)  = 0.5;
DSM(25,38) = 1;
DSM(26,38) = 1;
DSM(27,38) = 1;
DSM(35,38) = 0.5;
DSM(38,38) = 1;
DSM(57,38) = 1;
DSM(58,38) = 1;

DSM(9,39)  = 0.5;
DSM(13,39) = 1;
DSM(23,39) = 2;
DSM(38,39) = 1;
DSM(39,39) = 1;
DSM(40,39) = 2;

DSM(2,40)  = 1;
DSM(4,40)  = 1;
DSM(5,40)  = 1;
DSM(6,40)  = 1;
DSM(8,40)  = 1;
DSM(9,40)  = 2;
DSM(10,40) = 2;
DSM(13,40) = 1;
DSM(16,40) = 2;
DSM(20,40) = 2;
DSM(22,40) = 2;
DSM(30,40) = 1;
DSM(32,40) = 1;
DSM(34,40) = 1;
DSM(39,40) = 2;
DSM(40,40) = 1;
DSM(41,40) = 2;
DSM(42,40) = 1;
DSM(43,40) = 1;
DSM(44,40) = 1;
DSM(45,40) = 1;
DSM(46,40) = 1;
DSM(47,40) = 1;
DSM(48,40) = 1;
DSM(53,40) = 2;
DSM(54,40) = 1;
DSM(55,40) = 2;
DSM(56,40) = 1;
DSM(60,40) = 0.5;

DSM(13,41) = 1;
DSM(38,41) = 1;
DSM(40,41) = 2;
DSM(41,41) = 1;
DSM(42,41) = 2;
DSM(49,41) = 2;
DSM(58,41) = 0.5;
DSM(59,41) = 1;
DSM(60,41) = 0.5;

DSM(9,42)  = 1;
DSM(25,42) = 2;
DSM(35,42) = 1;
DSM(38,42) = 1;
DSM(40,42) = 1;
DSM(41,42) = 2;
DSM(42,42) = 1;
DSM(57,42) = 0.5;
DSM(58,42) = 0.5;

DSM(9,43)  = 1;
DSM(15,43) = 1;
DSM(24,43) = 1;
DSM(25,43) = 1;
DSM(38,43) = 2;
DSM(40,43) = 1;
DSM(43,43) = 1;

DSM(9,44)  = 1;
DSM(15,44) = 1;
DSM(24,44) = 1;
DSM(25,44) = 1;
DSM(38,44) = 2;
DSM(40,44) = 1;
DSM(44,44) = 1;

DSM(9,45)  = 1;
DSM(15,45) = 1;
DSM(24,45) = 1;
DSM(25,45) = 2;
DSM(28,45) = 1;
DSM(37,45) = 1;
DSM(38,45) = 1;
DSM(40,45) = 2;
DSM(45,45) = 1;
DSM(58,45) = 0.5;

DSM(15,46) = 1;
DSM(38,46) = 1;
DSM(39,46) = 2;
DSM(46,46) = 1;

DSM(15,47) = 1;
DSM(38,47) = 1;
DSM(40,47) = 1;
DSM(47,47) = 1;

DSM(15,48) = 1;
DSM(24,48) = 1;
DSM(25,48) = 1;
DSM(38,48) = 1;
DSM(40,48) = 1;
DSM(48,48) = 1;

DSM(13,49) = 1;
DSM(15,49) = 2;
DSM(38,49) = 1;
DSM(41,49) = 2;
DSM(49,49) = 1;
DSM(59,49) = 0.5;
DSM(60,49) = 0.5;

DSM(25,50) = 1;
DSM(50,50) = 1;
DSM(25,51) = 1;
DSM(51,51) = 1;

DSM(25,52) = 1;
DSM(52,52) = 1;

DSM(40,53) = 2;
DSM(53,53) = 1;
DSM(54,53) = 1;
DSM(60,53) = 0.5;

DSM(40,54) = 2;
DSM(53,54) = 1;
DSM(54,54) = 1;

DSM(10,55) = 1;
DSM(40,55) = 1;
DSM(55,55) = 1;

DSM(22,56) = 1;
DSM(40,56) = 1;
DSM(56,56) = 1;

DSM(35,57) = 0.5;
DSM(57,57) = 1;
DSM(58,57) = 1;
DSM(61,57) = 0.5;

DSM(9,58)  = 0.5;
DSM(35,58) = 0.5;
DSM(58,58) = 1;

DSM(49,59) = 1;
DSM(59,59) = 1;

DSM(1,60)  = 0.5;
DSM(2,60)  = 0.5;
DSM(3,60)  = 0.5;
DSM(4,60)  = 0.5;
DSM(5,60)  = 0.5;
DSM(6,60)  = 0.5;
DSM(7,60)  = 0.5;
DSM(8,60)  = 0.5;
DSM(13,60) = 1;
DSM(32,60) = 1;
DSM(34,60) = 0.5;
DSM(40,60) = 1;
DSM(41,60) = 0.5;
DSM(49,60) = 0.5;
DSM(53,60) = 0.5;
DSM(55,60) = 0.5;
DSM(56,60) = 0.5;
DSM(60,60) = 1;
DSM(61,60) = 0.5;

DSM(13,61) = 0.5;
DSM(30,61) = 0.5;
DSM(31,61) = 0.5;
DSM(33,61) = 0.5;
DSM(34,61) = 0.5;
DSM(35,61) = 0.5;
DSM(40,61) = 0.5;
DSM(41,61) = 0.5;
DSM(49,61) = 0.5;
DSM(53,61) = 0.5;
DSM(57,61) = 0.5;
DSM(58,61) = 0.5;
DSM(59,61) = 0.5;
DSM(60,61) = 0.5;
DSM(61,61) = 0.5;




% ****************************************************************************
%                        DSM Elements Labels                             *
% ****************************************************************************

DSMLABEL = cell(DSM_size,1);

DSMLABEL{1,1} = 'Hall Request Indicator';
DSMLABEL{2,1} = 'Car Request Indicator';
DSMLABEL{3,1} = 'Hallway Fixtures';
DSMLABEL{4,1} = 'Car Fixtures';
DSMLABEL{5,1} = 'Emerg. Intercom-Phone';
DSMLABEL{6,1} = 'Car Buttons';
DSMLABEL{7,1} = 'Hall Buttons';
DSMLABEL{8,1} = 'Firemans Service key';
DSMLABEL{9,1} = 'Travelling Comm. Cable';

DSMLABEL{10,1} = 'Ropes';
DSMLABEL{11,1} = 'Motor';
DSMLABEL{12,1} = 'Drive Power Section';
DSMLABEL{13,1} = 'Cab Floor/Plank';
DSMLABEL{14,1} = 'Power Supplies';
DSMLABEL{15,1} = 'Building Structure';
DSMLABEL{16,1} = 'Travel Cable Power';
DSMLABEL{17,1} = 'Building Power';
DSMLABEL{18,1} = 'Power Storage';
DSMLABEL{19,1} = 'Counterweight';

DSMLABEL{20,1} = 'Compensation System';
DSMLABEL{21,1} = 'Brake System';
DSMLABEL{22,1} = 'Cab Guidance System';
DSMLABEL{23,1} = 'Guide Rails';
DSMLABEL{24,1} = 'Learn Function';
DSMLABEL{25,1} = 'Motion Controller';
DSMLABEL{26,1} = 'Electrical Drive Control';
DSMLABEL{27,1} = 'Operational Controller';
DSMLABEL{28,1} = 'Primary Position System';
DSMLABEL{29,1} = 'Primary Velcotiy Measement';

DSMLABEL{30,1} = 'Top-of-Car Insp. Ctl Panel';
DSMLABEL{31,1} = 'Remote Insp. Ctl. Panel';
DSMLABEL{32,1} = 'Load Weighing System';
DSMLABEL{33,1} = 'Construction Control Panel';
DSMLABEL{34,1} = 'Door Close Button';
DSMLABEL{35,1} = 'Service Tool';
DSMLABEL{36,1} = 'Emerg. Gen. Signal';
DSMLABEL{37,1} = 'Secondary Veloc. Check';
```

```
DSMLABEL{38,1} = 'Safety System';
DSMLABEL{39,1} = 'Mechanical Safeties System';

DSMLABEL{40,1} = 'Cab';
DSMLABEL{41,1} = 'Cab Doors';
DSMLABEL{42,1} = 'Door Controller';
DSMLABEL{43,1} = 'Terminal Sensing System';
DSMLABEL{44,1} = 'ETSD Sensor';
DSMLABEL{45,1} = 'Door Zone Sensors';
DSMLABEL{46,1} = 'Governor System';
DSMLABEL{47,1} = 'Terminal Buffer';
DSMLABEL{48,1} = 'NTSD Sensor';
DSMLABEL{49,1} = 'Hoistway Doors';

DSMLABEL{50,1} = 'Door Obstruction Sensor';
DSMLABEL{51,1} = 'Earthquake Sensor';
DSMLABEL{52,1} = 'Building Sway Sensor';
DSMLABEL{53,1} = 'Vent & Lighting System';
DSMLABEL{54,1} = 'Cab Insullation';
DSMLABEL{55,1} = 'Ropes/Hitch Springs';
DSMLABEL{56,1} = 'Guide Springs/ Rail Align';
DSMLABEL{57,1} = 'Diagnostics';
DSMLABEL{58,1} = 'Remote Monitoring System';
DSMLABEL{59,1} = 'Acces keying on Hstway Doors';

DSMLABEL{60,1} = 'Passenger Load';
DSMLABEL{61,1} = 'Service Guy';




% ***************************************************************************
%                  Functional Mapping to Physical Elements              *
% ***************************************************************************

% Each of the functional labels represents the functional
% requirement for which the physcial DSM element represents
% Used to cross-reference the physical elemnts and
% functional requiremnts

DSMFunc_LABEL = cell(DSM_size,1);

DSMFunc_LABEL{1,1} = 'Acknowledge Hall Svc Rqst';
DSMFunc_LABEL{2,1} = 'Acknowledge Car Service Rqst';
DSMFunc_LABEL{3,1} = 'Comm. Status w/ Hall Pass.';
DSMFunc_LABEL{4,1} = 'Comm. Status w/ Car Pass.';
DSMFunc_LABEL{5,1} = 'Emerg. Comm. w/ car pass.';
DSMFunc_LABEL{6,1} = 'Input Car Rqst';
DSMFunc_LABEL{7,1} = 'Input Hall Rqst';
DSMFunc_LABEL{8,1} = 'Input Firemans Svc Rqst';
DSMFunc_LABEL{9,1} = 'Provide Comm. Link to Pass';

DSMFunc_LABEL{10,1} = 'Xmit force to containment';
DSMFunc_LABEL{11,1} = 'Energy Transformation to Force';
DSMFunc_LABEL{12,1} = 'Provide Energy for Motion';
```

```
DSMFunc_LABEL{13,1} = 'Support load for movement';
DSMFunc_LABEL{14,1} = 'Provide power to comp.';
DSMFunc_LABEL{15,1} = 'Provide structure to contain Sys.';
DSMFunc_LABEL{16,1} = 'Provide power to containment';
DSMFunc_LABEL{17,1} = 'Provide power for system';
DSMFunc_LABEL{18,1} = 'Temporary power for memory';
DSMFunc_LABEL{19,1} = 'Provide Counterbalancing to contain.';

DSMFunc_LABEL{20,1} = 'Provide coubterbalancing of ropes';
DSMFunc_LABEL{21,1} = 'Hold containment at destination';
DSMFunc_LABEL{22,1} = 'Maintain guidance in pathway';
DSMFunc_LABEL{23,1}   'Provide path for motion in Hwy';
DSMFunc_LABEL{24,1} = 'System Calibration';
DSMFunc_LABEL{25,1} = 'Compute Trajectory';
DSMFunc_LABEL{26,1} = 'Control Motion to trajectory';
DSMFunc_LABEL{27,1} = 'Process Service Request';
DSMFunc_LABEL{28,1} = 'Provide current position';
DSMFunc_LABEL{29,1} = 'Provide Trajectory feedback';

DSMFunc_LABEL{30,1} = 'Top-of-car control (inspection)';
DSMFunc_LABEL{31,1} = 'Remote Control (inspection)';
DSMFunc_LABEL{32,1} = 'Detect if load is within limits';
DSMFunc_LABEL{33,1} = 'Contruction Control';
DSMFunc_LABEL{34,1} = 'Passenger Door control';
DSMFunc_LABEL{35,1} = 'Provide access to view/change sys. info.';
DSMFunc_LABEL{36,1} = 'Determine if on Emerg. Power';
DSMFunc_LABEL{37,1} = 'Detect Uncontrolled motion';
DSMFunc_LABEL{38,1} = 'Determine if safe to move';
DSMFunc_LABEL{39,1} = 'Stop free-fall motion';

DSMFunc_LABEL{40,1} = 'Protect load from hwy';
DSMFunc_LABEL{41,1} = 'Provide safe load transfer';
DSMFunc_LABEL{42,1} = 'Control acces to containment';
DSMFunc_LABEL{43,1} = 'independent end of hwy sensor';
DSMFunc_LABEL{44,1} = 'Emerg. Terminal Stopiing device';
DSMFunc_LABEL{45,1} = 'Indep. Door zone sensing';
DSMFunc_LABEL{46,1} = 'Mechanical Overspeed detection';
DSMFunc_LABEL{47,1} = 'Provide end of Hwy cushion (bffr)';
DSMFunc_LABEL{48,1} = 'Detect failure slow for last landing -NTSD';
DSMFunc_LABEL{49,1} = 'Deny access to hwy';

DSMFunc_LABEL{50,1} = 'Detect obstructed access';
DSMFunc_LABEL{51,1} = 'Respond to Earthquake';
DSMFunc_LABEL{52,1} = 'Detect Building sway';
DSMFunc_LABEL{53,1} = 'Maintainsafe/comf load env.';
DSMFunc_LABEL{54,1} = 'Manage noise';
DSMFunc_LABEL{55,1} = 'Manage vertical vibration';
DSMFunc_LABEL{56,1} = 'Manage horizontal vibration';
DSMFunc_LABEL{57,1} = 'Determine cause of malfunction';
DSMFunc_LABEL{58,1} = 'Monitor system health';
DSMFunc_LABEL{59,1} = 'provide service access to hwy';

DSMFunc_LABEL{60,1} = 'Passenger Load';
DSMFunc_LABEL{61,1} = 'Service Guy';
```

Bid

```
function [cluster_bid]=bid(elmt, DSM_matrix, cluster_matrix, max_cluster_size,
pow_dep, pow_bid, cluster_size);
%[cluster_bid]=bid(elmt, DSM_matrix, cluster_matrix, max_cluster_size, pow_dep,
pow_bid, cluster_size);
%
%
%
% Function to calculate the bids from an element to each cluster of elements
%
%
%
%  Inputs:
%          elmt            The DSM element number to receive bids from clusters
%          DSM_matrix      The DSM matrix
%          Cluster_matrix     The Cluster Matrix (cluster,element)
%                                   1 = element in cluster, 0 = not in cluster
%          max_cluster_size   maximum cluster size
%          pow_dep         parameter for weighting interactions between elements
%          pow_bid         parameter for penalizing cluster sizes
%          cluster_size    array of cluster sizes
%
%
%  Outputs:
%          cluster_bid        array of the cluster bids

% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
%                                                                           *
%  File:   bid.m                                                            *
%                                                                           *
%  Created by: Ronnie E. Thebeau                                            *
%              System Design and Management Program                         *
%              Massacusetts Institute of Technology                         *
%                                                                           *
%  Date: December 2000                                                      *
%                                                                           *
%  Function to calculate the bids from clusters for the selected element.   *
%  Each cluster makes a bid for the selected element based on the           *
%  bidding parameters.                                                      *
%                                                                           *
%  This algorithm is based on work by Carlos Fernandez.                     *
%                                                                           *
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************


% get the number of clusters and number of elements
[n_clusters, DSM_size] = size(cluster_matrix);

% intialize the bidding array
cluster_bid = zeros(n_clusters,1);

% For each Cluster, if any element in the cluster has an interaction with
% the selected element then add the number of interactions with the selected
% element.  Then use the number of interactions to calculate the bid.

for i=1:n_clusters
   in=0;
   out=0;
   for j=1:DSM_size
      % if element J is in Cluster i, need j not equal to element to avoid
diagnal
      if((cluster_matrix(i,j)==1)&(j~=elmt))
         if (DSM_matrix(j,elmt)>0)
            in = in + DSM_matrix(j,elmt);
         end
         if (DSM_matrix(elmt,j) > 0)
            out = out + DSM_matrix(elmt,j);
         end
      end
   end
   % if there were any interactions with members of the clusters, make a bid
   if ( (in>0) | (out>0))
      if (cluster_size(i) == max_cluster_size)
         cluster_bid(i) = 0;
      else
         % calculate the cluster bid
         cluster_bid(i)=((in+out)^pow_dep)/((cluster_size(i)^pow_bid));
      end
   end
end
```

Cluster

```
function [Cluster_matrix, total_coord_cost, cost_history, old_data] =
Cluster(DSM_matrix, Cluster_param);
% [Cluster_matrix, total_coord_cost, cost_history, old_data] =
Cluster(DSM_matrix, Cluster_param);
%
% Inputs:
%         DSM_matrix      the DSM matrix
%         Cluster_param   structure containing the clustering control
parameters
%
% Outputs:
%         Cluster_matrix    cluster matrix based on the clustering algorithm
%         total_coord_cost  calculated cost of the cluster matrix solution
%         cost_history    record of the coordination cost as the algorith
%                         searched for a solution
%         old_data           intermediate solutions to the clustering routine
%
%
% Parameters of the structure "Cluster_param" include:
%    Cluster_param.pow_cc                penalty assigned to cluster size during
costing
%    Cluster_param.pow_bid            penalty assigned to cluster size during
bidding
%    Cluster_param.pow_dep            emphasizes high interactions
%    Cluster_param.max_cluster_size      max size of cluster
%    Cluster_param.rand_accept           randomly asccep change 1 of N times w/
no improvement
%    Cluster_param.rand_bid              randomly accept second highest bid 1 of
N times
%    Cluster_param.times                 attept "times" changes before checking
for stability
%    Cluster_param.stable_limit          loop at least stable_limit*times*size
before ending
%
%

% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
%                                                                           *
%  File:   Cluster.m                                                        *
%                                                                           *
% Created by: Ronnie E. Thebeau                                             *
%             System Design and Management Program                          *
%             Massacusetts Institute of Technology                          *
%                                                                           *
% Date: December 2000                                                       *
%                                                                           *
% Function to cluster the elements of a matrix                              *
% Algorithm based on work developed by Carlos Fernandez                     *
%                                                                           *
% This function runs a clustering algorithm then calculates the cost of     *
% the proposed solution.  The objective is to find the solution that        *
% in the lowest cost solution.                                              *
%                                                                           *
% There is a higher cost for interactions that occur outside of clusters    *
% and lower costs for interactions within clusters.  There are also         *
%  penalties assigned to the size of clusters to avoid a solution where    *
%  all elements are members of a single cluster.                           *
%                                                                          *
%  There results are highly dependant on the parameters passed to the      *
%  algoreithm.                                                             *
%                                                                          *
% *************************************************************************
% *************************************************************************
% *************************************************************************
% *************************************************************************




% *************************************************************************
%                   Clustering Control Parameters                          *
% *************************************************************************
%    Extract clustering control parameters
pow_cc        = Cluster_param.pow_cc;
pow_bid         = Cluster_param.pow_bid;
pow_dep         = Cluster_param.pow_dep;
max_cluster_size   = Cluster_param.max_cluster_size;
rand_accept     = Cluster_param.rand_accept;
rand_bid        = Cluster_param.rand_bid;
times         = Cluster_param.times;
stable_limit  = Cluster_param.stable_limit;
% ***************** END GETTING CLUSTERING PARAMETERS *********************




% *************************************************************************
%                   Initialize Matices and arrays                  *
% *************************************************************************

DSM_size   =  size(DSM_matrix,2);   % number of elements in the DSM
n_clusters =  DSM_size;           % initial # clusters = # elements
max_repeat = 10;                  % maximum # of times to run through calc

coordination_cost        =  zeros(DSM_size,1);
cluster_size           =  zeros(DSM_size,1);
new_cluster_matrix     =  zeros(DSM_size,DSM_size);
new_cluster_size         =  zeros(DSM_size,1);
cluster_bid              =  zeros(DSM_size,1);
new_cluster_bid        =  zeros(DSM_size,1);
new_coordination_cost =  zeros(DSM_size,1);
rnd_elmt_arr           =  zeros(DSM_size,1);
cluster_list           =  zeros(DSM_size,1);
% ****************** END INTIALIZE MATRICES ******************************


% *************************************************************************
%                         Initialize Clustering                          *
% *************************************************************************
% **** initialize the cluster matrix along the diagonals****
%    Initial clustering: nth cluster contains the nth element
cluster_diagonals = ones(1,n_clusters);
Cluster_matrix = diag(cluster_diagonals);
cluster_size = ones(n_clusters,1);

% calculate the initial starting coordination cost of the clustering
[total_coord_cost] = Coord_Cost(DSM_matrix, Cluster_matrix, cluster_size,
pow_cc);
best_coord_cost = total_coord_cost;

% create an empty array to hold the cost history
cost_history = zeros(10e3,1);
history_index = 0;

% create matrices to store the best cost and the corrresponding
% cluster size and clustrix that have been found so far
best_curr_cost = total_coord_cost;
best_curr_Clust_mat = Cluster_matrix;
best_cluster_size = cluster_size;
% ********************* END INITIALIZE CLUSTERING *************************



% ***************************************************************************
%                         Clustering Routine                              *
% ***************************************************************************

% Initialize parameters
stable = 0;        %  toggle to indicate if the algorithm has met the stability
criteria
change = 0;        %  toggle to indicate if a change should be made
accept1= 0;        %  toggle to indicate if the solution should be acccepted
first_run = 1;     %  toggle to indicate if it is the first run through
pass = 1;       %  index to count the number of passes through the algorithm

% store best cost solution and data found up to this point
old_data(pass).Cluster_matrix = Cluster_matrix;
old_data(pass).cluster_size = cluster_size;
old_data(pass).total_coord_cost = total_coord_cost;






% **** start the clustering routine ****
% continue until the results have met the stability criteria
% AND the final solution is the same or better than any intermediate solution
that may
% have been found.  Due to simulated annealing, a final solution may be worse
than an
% intermediate solution that had been found before making the random change
%
% If the final solution is not equal to or less than any best solution, then
% return to the best solution and continue to search for a better solution.
% Do this until a better solution is found or we have looped back max_repeat
times.
% If we reach max_repeat, then report the best solution thathad been found.

while ( ((total_coord_cost > best_coord_cost) & (pass <= max_repeat))|
(first_run == 1))
   if first_run == 0    % only if not first pass through clustering routine
```

```
      pass = pass+1; % increment # passes
      % store the current data
      old_data(pass).Cluster_matrix = Cluster_matrix;
      old_data(pass).cluster_size = cluster_size;
      old_data(pass).total_coord_cost = total_coord_cost;
      total_coord_cost  = best_curr_cost;
      Cluster_matrix = best_curr_Clust_mat;
      cluster_size      = best_cluster_size;
      history_index     = history_index+1;
      cost_history(history_index,1) = total_coord_cost;
   end
   % reset the toggles back to zero
   first_run  =  0;
   stable     =  0;
   accept1     =  0;
   change     =  0;


   while (stable ~= stable_limit)
      for k=1:DSM_size*times;
         % (1)  pick an element in a random location in the DSM
         elmt = cei_  SM_size*rand(1,1));% random number between 1 and DSM_size


         % (2) Accept bids for the elemement from all clusters
         %            cluster_bid(i) holds the bid for cluster i in
cluster_matatrix
         [cluster_bid]=bid(elmt, DSM_matrix, Cluster_matrix, max_cluster_size,
pow_dep, pow_bid, cluster_size);


         % (3) Determine the best bid & second best bid
         best_cluster_bid = max(cluster_bid);
         secondbest_cluster_bid =
max((best_cluster_bid~=cluster_bid).*cluster_bid);

         % simulated annealing
         if (floor(rand_bid*rand(1,1)) == 0)       % pick second best bid 1 out
of rand_bid times
            best_cluster_bid = secondbest_cluster_bid;
         end


         if (best_cluster_bid>0)

            % (3a)  Determine if the BID is acceptable
            % Initialize
            cluster_list(:,1) = 0;

            % Determine the list of clusters affected
            cluster_list(1:n_clusters,1) = ((cluster_bid(:)==best_cluster_bid)
& (Cluster_matrix(:,elmt)==0));


            % copy the cluster matrix into new matrices
            new_cluster_matrix   = Cluster_matrix;
            new_cluster_size     = cluster_size;

            % proceed with cluster changes in the new cluster
            new_cluster_matrix(1:n_clusters,elmt) =
new_cluster_matrix(1:n_clusters,elmt) | cluster_list;
            new_cluster_size(1:n_clusters) = new_cluster_size(1:n_clusters) +
(cluster_list==1)*1;
```

```
            % delete any duplicate or empty clusters
            [new_cluster_matrix, new_cluster_size]=
Delete_Clusters(new_cluster_size, new_cluster_matrix);

            % determine the change in the coordination costs
            [new_total_coord_cost] = Coord_Cost(DSM_matrix, new_cluster_matrix,
new_cluster_size, pow_cc);
            if (new_total_coord_cost <= total_coord_cost)
               accept1 = 1;
            else
               if (floor(rand_accept*rand(1,1)) == 0)% still accept 1 out of
approx random_accept times
                  accept1 = 1;
                  % if we are going to accept a total cost that is not less
than our current cost
                  %  then
                  % save the current cost as the best current cost found so far
(only if the current cost
                  % is lower than any best current cost previously saved)
because we may not find
                  % a cost that is better than the current cost
                  %
                  % When we think we are finished we will check the final cost
against any best cost
                  % if the final cost is not better than the lowest cost found,
then we will move back to that best cost
                  if total_coord_cost < best_curr_cost
                     best_curr_cost = total_coord_cost;
                     best_curr_Clust_mat = Cluster_matrix;
                     best_cluster_size = cluster_size;
                  end
               else
                  accept1 = 0;
               end
            end
         end

         if (accept1)
            accept1=0;

            % (4) UPDATE the clusters
            total_coord_cost = new_total_coord_cost;
            Cluster_matrix = new_cluster_matrix;
            cluster_size = new_cluster_size;
            history_index = history_index+1;
            cost_history(history_index,1) = total_coord_cost;

            if (best_coord_cost > total_coord_cost)
               best_coord_cost = total_coord_cost;
               change = change + 1;  % improvement in coord.cost
            end
         end
      end

      % (5) Test the system for stability
      if (change >0)
         stable = 0;
         change = 0;
      else
         stable = stable + 1;
      end
```

```
    end
end
```

Coord_cost

```
function [total_coord_cost] = Coord_Cost(DSM_matrix, Cluster_matrix,
cluster_size, pow_cc);
%[total_coord_cost] = Coord_Cost(DSM_matrix, Cluster_matrix, cluster_size,
pow_cc);
%
%
%  Function to calculate the coordination cost of the cluster matrix
%
%    Inputs:
%            DSM_matrix      The DSM matrix
%            Cluster_matrix     The cluster matrix (Cluster,Element)
%            cluster_size(n) array specifying the number of elements in cluster
n
%            pow_cc          value for the exponential weighting of the cost
function
%
%
%    Outputs:
%            total_coord_cost   total coordination cost
%
% This routine checks all DSM Interactions.  If a DSM interaction is contained
in
% in one or more clusters, we add the cost of all intra-cluster interactions.
% If the interaction is not contained within any clusters, then a higher cost
is
% assigned to the out-of-cluster interaction.
%
% Looking at all DSM interactions i and j,
%  Are DSM(i) and DSM(j) both contained in any clusters
%    if yes: coordination cost is the sum of (DSM(i) +
DSM(j))*cluster_size(cluster n)^pow_cc
%            across all clusters
%    if no: coordination cost is (DSM(i) + DSM(j))*DSM_size^pow_cc
%  Total coordination cost is equal to the sum of all coordination costs


% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
%                                                                           *
%  File:   Coord_Cost.m                                                     *
%                                                                           *
%  Created by: Ronnie E. Thebeau                                            *
%              System Design and Management Program                         *
%              Massacusetts Institute of Technology                         *
%                                                                           *
%  Date: December 2000                                                      *
%                                                                           *
%  Function to cluster the elements of a matrix                             *
%  Algorithm based on work developed by Carlos Fernandez                    *
%                                                                           *
%  This function calculates the "coordination cost" of the clustering       *
%  solution.  Higher values are assigned to interactions outside of         *
%  than those occuring inside of clusters.                                  *
%                                                                           *
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************


% get the number of clusters and the size of the DSM
[n_clusters, DSM_size] = size(Cluster_matrix);

% intialize coordination costs
total_coord_cost = 0;
coordination_cost= zeros(1,DSM_size);

DSM_labels = cell(DSM_size,1);      % dummy variable for reorder function



% reorder the DSM acording to the cluster matrix
% NOTE: this re-ordering will duplicate entries for elements that show up
% in more than one cluster.  If an element is in three clusters, the new
% DSM matrix will have three seperate entries for the element.
% Therefore this new DSM matrix may be much larger than the original DSM
% This essentially assigns a higher cost if the element is a memeber of
% more than one cluster
[New_DSM_matrix, New_DSM_labels] = reorder_DSM_byCluster(DSM_matrix,
Cluster_matrix, DSM_labels);
New_DSM_size = size(New_DSM_matrix,1);



% get the number of elements in each cluster
Num_cluster_elements = sum(Cluster_matrix,2);

n=1;

New_Cluster_matrix = zeros(New_DSM_size, New_DSM_size);

% Create a new cluster matrix that matches the new reordered DSM matrix
% Because the DSM was reordered, columns of the cluster matrix must be re-
ordered
% to match the order of the new DSM Matrix.  This is done for cost calculation
% purposes only.
for i =1:n_clusters
   New_Cluster_matrix(i,n:n+Num_cluster_elements(i)-1) =
ones(1,Num_cluster_elements(i));
   n= n+Num_cluster_elements(i);
end

% get new cluster size array that matches the new cluster matrix
New_Cluster_size = sum(New_Cluster_matrix,2);


% replace the old data with the new data for the cost calculation
DSM_size = New_DSM_size;
DSM_matrix = New_DSM_matrix;
Cluster_matrix = New_Cluster_matrix;
cluster_size = New_Cluster_size;

[n_clusters, DSM_size] = size(Cluster_matrix);
total_coord_cost = 0;
coordination_cost= zeros(1,DSM_size);

% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ****************************************************************************
%                    CALCULATE THE COST OF THE SOLUTION                       *
% ****************************************************************************
% ****************************************************************************
% ****************************************************************************
% ****************************************************************************
% ****************************************************************************


for i=1:DSM_size
   for j=i+1:DSM_size   % j=i+1 to skip the diagonals
      if (DSM_matrix(i,j)>0 | DSM_matrix(j,i)>0)% if a dependancy exists
between i & j
         cost_total = 0;

         % check if any clusters contain the both elements
         for (cluster_index=1:n_clusters)
            if
(Cluster_matrix(cluster_index,i)+Cluster_matrix(cluster_index,j)==2) % ie both
i & j belong to the same cluster
               cost_total = cost_total + (DSM_matrix(i,j) + DSM_matrix(j,i)) *
cluster_size(cluster_index)^pow_cc;
            end
         end

         if (cost_total>0)
            cost_c = cost_total;
         else
            cost_c = (DSM_matrix(i,j) + DSM_matrix(j,i))*DSM_size^pow_cc;
         end
         coordination_cost(i) = coordination_cost(i) + cost_c;
      end
   end
end

total_coord_cost = sum(coordination_cost);
% ****************************************************************************
% ******************** END COST CALCULATION **********************************
% ****************************************************************************
```

reorder_dsm_bycluster

```
function [New_DSM_matrix, New_DSM_labels] = reorder_DSM_byCluster(DSM_matrix,
Cluster_matrix, DSM_label);
%[New_DSM_matrix, New_DSM_labels] = reorder_DSM_byCluster(DSM_matrix,
Cluster_matrix, DSM_label);
%
%
%    Inputs:
%            DSM_matrix      DSM Matrix to be re-ordered
%            Cluster_matrix    Cluster Matrix to control the re-ordering
%            DSM_label       labels of the DSM elements (lebels need to be re-
ordered also)
%
%
%
%    Output:
%            New_DSM_matrix    reordered DSM matrix
%            New_DSM_labels    reordered label array
%
%

% **************************************************************************
% **************************************************************************
% **************************************************************************
% **************************************************************************
% **************************************************************************
%                                                                          *
%  File:   reorder_DSM_byCluster.m                                         *
%                                                                          *
%  Created by: Ronnie E. Thebeau                                           *
%              System Design and Management Program                        *
%              Massacusetts Institute of Technology                        *
%                                                                          *
%  Date: December 2000                                                     *
%                                                                          *
%  Function to reorder the DSM Matrix accroding the the Cluster matrix.    *
%  Place all elements in the same cluster next to each other.  If an      *
%  an element is a member of more than one cluster, duplicate that element *
%  in the DSM for each time it appears in a cluster                        *
%                                                                          *
%  The new DSM will have all elments in a cluster next to each other       *
%                                                                          *
% **************************************************************************
% **************************************************************************
% **************************************************************************
% **************************************************************************


% place zeros along the diagonals of the DSM matrix
DSM_matrix = tril(DSM_matrix,-1) + triu(DSM_matrix,1) +
diag(zeros(length(DSM_matrix),1));

% get the size of the DSM matrix
DSM_size = size(DSM_matrix,2);

% find all element-to-cluster assignments
[Cluster_number,Element] = find(Cluster_matrix);

%sort the element-to-cluster list in ascending order of clusters
[ordered_cluster_number, ClusterList_index] = sort(Cluster_number);
```

```
new_number_elmts = length(ClusterList_index);

% Get the new rows of the DSM matrix
for i = 1:new_number_elmts
   temp_DSM_matrix(i,:) = DSM_matrix(Element(ClusterList_index(i)),:);
   New_DSM_labels{i,1} = DSM_label{Element(ClusterList_index(i)),1};
end

% Now add the new columns of the DSM matrix
for i= 1:new_number_elmts
   New_DSM_matrix(:,i) = temp_DSM_matrix(:,Element(ClusterList_index(i)));
end
```

reorder_cluster

```
function [New_Cluster_matrix] = reorder_cluster(Cluster_matrix);
%[New_Cluster_matrix] = reorder_cluster(Cluster_matrix);
%
%
% Function sort and reorder the cluster matrix by size
% Small cluster will be at the bottom of the list and large cluster
% will be moved to the top of the list
%
% Inputs:
%          Cluster_matrix     The Cluster matrix
%
% Outputs:
%          New_cluster_matrix The new sorted Cluster Matrix

% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
%                                                                           *
% File:   reorder_cluster.m                                                 *
%                                                                           *
% Created by: Ronnie E. Thebeau                                             *
%             System Design and Management Program                          *
%             Massacusetts Institute of Technology                          *
%                                                                           *
% Date: December 2000                                                       *
%                                                                           *
% Function sort and reorder the cluster matrix by size                      *
% Small cluster will be at the bottom of the list and large cluster         *
% will be moved to the top of the list                                      *
%                                                                           *
%                                                                           *
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************



Num_clstelm = sum(Cluster_matrix,2);

[Y,I] = sort(Num_clstelm,1);
flipped_I = flipud(I);
New_Cluster_matrix(:,:) = Cluster_matrix(flipped_I,:);
```

delete_clusters

```
function [new_cluster_matrix, new_cluster_size]= Delete_Clusters(cluster_size,
cluster_matrix);
%[new_cluster_matrix, new_cluster_size]= Delete_Clusters(cluster_size,
cluster_matrix);
%
%
% Function to delete duplicate clusters or cluster that are within clusters
%
%
%
%  Inputs:
%          cluster_size       Array of containing size of each cluster
%          cluster_matrix     The cluster matrix
%
%  Outputs:
%          new_cluster_matrix Cluster Matrix after duplicates have been removed
%          new_cluster_size      New array of cluster sizes
%
%
%


% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
%                                                                           *
%  File:   Delete_Clusters.m                                                *
%                                                                           *
%  Created by: Ronnie E. Thebeau                                            *
%              System Design and Management Program                         *
%              Massacusetts Institute of Technology                         *
%                                                                           *
%  Date: December 2000                                                      *
%                                                                           *
%  Function to calculate the bids from clusters for the selected element.   *
%  Each cluster makes a bid for the selected element based on the           *
%  bidding parameters.                                                      *
%                                                                           *
%  This algorithm is based on work by Carlos Fernandez.                     *
%                                                                           *
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************



[n_clusters, n_elements] = size(cluster_matrix);
new_cluster_matrix = zeros(size(cluster_matrix));
new_cluster_size = zeros(size(cluster_size));

% if the clusters are equal or cluster j is completely contained in cluster i
% delete cluster j
for i=1:n_clusters
   for j=i+1:n_clusters
      if (cluster_size(i) >= cluster_size(j) & cluster_size(j)>0)
```

```
         if all(((cluster_matrix(i,:) & cluster_matrix(j,:)) ==
cluster_matrix(j,:)))
            cluster_matrix(j,:)=0;
            cluster_size(j) = 0;
         end
      end
   end
end

% if cluster i is completely contained in cluster j, delete cluster i
for i=1:n_clusters
   for j= i+1:n_clusters
      if (cluster_size(i) < cluster_size(j) & cluster_size(i)>0)
         if all(((cluster_matrix(i,:) & cluster_matrix(j,:)) ==
cluster_matrix(i,:)))% cluster i is contained in j
            cluster_matrix(i,:)=0;
            cluster_size(i)=0;
         end
      end
   end
end


% delete clusters with no tasks

non_empty_cluster_indx = find(cluster_size);

new_cluster_matrix(1:length(non_empty_cluster_indx),:) =
cluster_matrix(non_empty_cluster_indx,:);
new_cluster_size(1:length(non_empty_cluster_indx)) =
cluster_size(non_empty_cluster_indx);
```

dsm_autolabel

```
function [numeric_DSM_labels] = DSM_autolabel(DSM_matrix);
%[numeric_DSM_labels] = DSM_autolabel(DSM_matrix);
%
%
% Function to create numeric text labels that correspond the the index
% in the DSM matrix.  Element 1 gets a text lebel of "1", etc.
%
%
%
%  Inputs:
%          DSM_matrix         The DSM matrix
%
%  Outputs:
%          mumeric_DSM_labels Cell array of numeric text labels

% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
%                                                                           *
%  File:   DSM_autolabel.m                                                  *
%                                                                           *
%  Created by: Ronnie E. Thebeau                                            *
%              System Design and Management Program                         *
%              Massacusetts Institute of Technology                         *
%                                                                           *
%  Date: December 2000                                                      *
%                                                                           *
%  Function to create a cell array of labels that correspond to each        *
%  entry in the DSM                                                         *
%                                                                           *
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************


% get labels from the rows (will work for cluster matrix also)
num_labels = size(DSM_matrix,1);      % get the number of labels needed

numeric_DSM_labels = cell(num_labels,1);   % create empty cell array

for i = 1:num_labels
   numeric_DSM_labels{i,1} = num2str(i);
end
```

place_diag

```
function [new_matrix] = place_diag(old_matrix, diagonal_element);
%[new_matrix] = place_diag(old_matrix, diagonal_element);
%
%
% Function function to place the desired number along the diagonal of a matrix
%
%
%
%  Inputs:
%          old_matrix         matrix to be manipulated
%          diaganol_element      element to be place along the diagonal
%
%  Outputs:
%          new_matrix         matrix witrh the desired element along the diag.

% ****************************************************************************
% ****************************************************************************
% ****************************************************************************
% ****************************************************************************
% ****************************************************************************
%                                                                            *
%  File:   place_diag.m                                                      *
%                                                                            *
%  Created by: Ronnie E. Thebeau                                             *
%              System Design and Management Program                          *
%              Massacusetts Institute of Technology                          *
%                                                                            *
%  Date: December 2000                                                       *
%                                                                            *
%  Function to place a desired number along the diagonal of a matrix.        *
%                                                                            *
%                                                                            *
% ****************************************************************************
% ****************************************************************************
% ****************************************************************************
% ****************************************************************************


diag_matrix = diag(diagonal_element*ones(length(old_matrix),1));

new_matrix = tril(old_matrix,-1) + triu(old_matrix,1) + diag_matrix;
```

graph_matrix

```
function []=graph_matrix(matrix, x_title, y_title, graph_title, x_tcklabel,
y_tcklabel, print_flag, Cluster_matrix);
%  []=graph_matrix(matrix, x_title, y_title, graph_title);
%
%  Function to graph the DSM or cluster matrix.
%  The data is plotted as a scatter graph and the size and color
%  of the marks are conrtrolled according to the value of the input
%
%       Inputs:
%               matrix          matrix to be plotted on scatter graph
%               x_title      string containing label for x-axis
%               y_title      string containing label for y-axis
%               graph_title     title for the graph
%               x_tcklabel      labels to be placed along the x-axis
%               y_tcklabel      labels to be placed along the y-axis
%               print_flag      flag to toggle printing 1=print, else no print
%               Cluster_matrix  Optional Cluster_matrix of plotting the
clusterd DSM
%
%       Outputs:
%
%
%
%

% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
%                                                                          *
%  File:   graph_matrix.m                                                  *
%                                                                          *
%  Created by: Ronnie E. Thebeau                                           *
%              System Design and Management Program                        *
%              Massacusetts Institute of Technology                        *
%                                                                          *
%  Date: December 2000                                                     *
%                                                                          *
%  Function to graph the DSM matrix or cluster matrix.  If the DSM matrix  *
%  is to plotted in clustered form, the cluster matrix must also be provided.
%                                                                          *
%                                                                          *
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************


% Are we plotting cluster boxes?
if nargin < 8
   Cluster_plot = 0;
else
   Cluster_plot = 1;
end


[row_input, column_out, m_value] = find(matrix);   % find non_zero entries
```

```
max_row = max(row_input);
max_col = max(column_out);
ax = [0 max_col+1 0 max_row+1];
max_value = max(m_value);
data_scale = ceil(500/max_value)/20;

x_tcklabel_g{1,1} = '0';
y_tcklabel_g{1,1} = '0';

for i = 1:size(x_tcklabel,1)
   x_tcklabel_g{i+1,1} = x_tcklabel{i,1};
end

for i = 1:size(y_tcklabel,1)
   y_tcklabel_g{i+1,1} = y_tcklabel{i,1};
end

figure;
clf;
scatter(column_out, row_input, m_value*data_scale, m_value,'filled','d');
axis(ax);
cur_ax = gca;
axes_scale = get(cur_ax,'Position');
set(cur_ax,'Position',[axes_scale(1) axes_scale(2) axes_scale(3)
axes_scale(4)*0.9]);
set(cur_ax,'XTick',(0:1:max_col+1));
set(cur_ax,'YTick',(0:1:max_row+1));
set(cur_ax,'XAxisLocation','top','YDir','Reverse');
set(cur_ax,'XTickLabelMode','manual');
set(cur_ax,'YTickLabelMode','manual');
set(cur_ax,'XTickLabel',x_tcklabel_g);
set(cur_ax,'YTickLabel',y_tcklabel_g);
set(cur_ax,'FontSize',4);
set(cur_ax,'Box','On');


xlabel(x_title);
ylabel(y_title);
title(graph_title);
orient('landscape');

% draw squares around the clustered elements in the DSM
if Cluster_plot ==1
   Number_clusters = size(Cluster_matrix,1);

   sq_s = 0.5;  %Start of the square
   for cluster_indx = 1:Number_clusters
      n_el = sum(Cluster_matrix(cluster_indx,:),2);     % number of elements in
cluster
      line([sq_s (sq_s + n_el) (sq_s + n_el) sq_s sq_s], [sq_s sq_s (sq_s +
n_el) (sq_s + n_el) sq_s]);
      sq_s = sq_s + n_el;
   end
end

if print_flag ==1
   print
end
```

line_mult_cluster

```
function [] = line_mult_cluster(Cluster_matrix, fig_handle, ax_handle);
%[] = line_mult_cluster(Cluster_matrix, fig_handle, ax_handle);
%
%
% Function add vertical lines to the cluster matrix graph to indicate
% which elements are members of more than one cluster
%
%
%
%  Inputs:
%             Cluster_matrix     The Cluster matrix
%             fig_handle         Figure Handle to add the lines
%             ax_handle          Figure Axes handles
%
%  Outputs:

% **********************************************************************
% **********************************************************************
% **********************************************************************
% **********************************************************************
%                                                                 *
%  File:   line_mult_cluster.m                                                       *
%                                                                 *
%  Created by: Ronnie E. Thebeau                                     *
%             System Design and Management Program                *
%             Massacusetts Institute of Technology                *
%                                                                 *
%  Date: December 2000                                               *
%                                                                 *
%  Function to add lines to the Cluster Matrix plot.  The vertical lines   *
%  will be placed along the elements that are members of more than one       *
%  cluster.                                                     *
%                                                                 *
%                                                                 *
% **********************************************************************
% **********************************************************************
% **********************************************************************
% **********************************************************************


[num_clusters, num_elmts] = size(Cluster_matrix);

X_lims = get(ax_handle, 'XLim');
Y_lims = get(ax_handle, 'YLim');

num_el_clusters = sum(Cluster_matrix,1);


for i = 1:num_elmts
   if num_el_clusters(1,i) > 1
      line([i i],[0 Y_lims(2)],'LineStyle',':','MarkerSize',0.5,'Color',[0.5
0.5 0.5]);
   end
end
```

plot_cluster_list

```
function [] = plot_cluster_list(Cluster_matrix, DSM_matrix, E_name_list,
current_date, plot_flag);
%[] = plot_cluster_list(Cluster_matrix, DSM_matrix, E_name_list, current_date,
plot_flag);
%
%
% Function to calculate the bids from an element to each cluster of elements
%
%
%
% Inputs:
%             Cluster_matrix  The Cluster Mtrix
%             DSM_matrix      The DSM Matrix
%             E_name_list     Array of strings that list the elements in the
cluster
%             current_date    Date string
%             plot_flag       toggle for printing; 1=print, else don't print
%
% Outputs:
%

% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
%                                                                         *
% File:   plot_cluster_list.m                                                           *
%                                                                         *
% Created by: Ronnie E. Thebeau                                              *
%             System Design and Management Program                        *
%             Massacusetts Institute of Technology                        *
%                                                                         *
% Date: December 2000                                                        *
%                                                                         *
% Function to plot a text list of the clusters and the elements contained     *
% in the clusters.                                                        *
%                                                                         *
%                                                                         *
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************


   % ******************************************************************
   %                        Simulation  Data
   % ******************************************************************
   % scaling factors for the plotting area
   top_of_list = 0.92;
   stepsize1 = 0.025;
   ttl_offset = 0.05;
   n=0;

   [non_zero_clusters, nz_el] = find(Cluster_matrix);
   num_clusters = max(non_zero_clusters);
   num_elements = size(Cluster_matrix,2);
   el_num_clusters = sum(Cluster_matrix,1);

   fig_row = 1;
   fig_base = set_plot_props(current_date);




   for i=1:num_clusters
      num_entries = sum(Cluster_matrix(i,:) > 0);
      if num_entries+n> 25
         fig_row = fig_row+1;
         n = 0;
      end

      if fig_row>2
         if plot_flag==1
            print;
         end
         fig_base = set_plot_props(current_date);
         fig_row = 1;
      end

      if fig_row ==1
         xpos1 = 0;
      else
         xpos1 = 0.6;
      end

      [cluster_yes,Element_yes] = find(Cluster_matrix(i,:));
      n=n+2;
      text(xpos1+ttl_offset,top_of_list-stepsize1*n,['\bf Cluster #'
num2str(i)],...
         'HorizontalAlignment','Center');
      n = n+1;
      text(xpos1-0.05,top_of_list-
stepsize1*n+0.02,'________________________',...
         'HorizontalAlignment','Left',...
         'interpreter','none');
      n=n+1-1;

      for z=1:num_entries

         n=n+1;
         el_DSM_num = Element_yes(z);
         listentry = [E_name_list{el_DSM_num,1} ' (' num2str(el_DSM_num) ')'];

         if el_num_clusters(Element_yes(z)) > 1
            text(xpos1,top_of_list-stepsize1*n,listentry,...
               'HorizontalAlignment','Left',...
               'interpreter','none',...
               'FontAngle','normal');
            text(xpos1-0.03,top_of_list-stepsize1*n,'*');
         else
            text(xpos1,top_of_list-stepsize1*n,listentry,...
               'HorizontalAlignment','Left',...
               'interpreter','none');
         end
      end
   end

   if plot_flag == 1
      print;
```

```
end



function [fignumber] = set_plot_props(current_date);

fignumber = figure;
orient tall;
Result_Data_window = figure(fignumber);
clf reset;
set(Result_Data_window,'Name','Cluster List');
ax_handle = axes;
set(ax_handle,'Visible','Off');

wd_units =  get(Result_Data_window,'Units');
set(Result_Data_window,'Units','normalized');
wd_position = get(Result_Data_window,'Position');
wd_left = wd_position(1);
wd_bottom = wd_position(2);
wd_width = wd_position(3);
wd_height = wd_position(4);
wd_left_new = 0.1;
wd_height_new = 0.75;
wd_bottom_new = 0.1;
wd_width_new = 0.5;
wd_new_position = [wd_left_new wd_bottom_new wd_width_new wd_height_new];
set(Result_Data_window,'Position',wd_new_position);
set(Result_Data_window,'Units',wd_units);
% ****************************************************************
%                    Title & Header  Data
% ****************************************************************
text(0.5,1,'\bf\fontsize{14} Cluster Member List',...
   'HorizontalAlignment','Center');

text(0.5,0.97,['\it ' current_date],...
   'HorizontalAlignment','Center');
```

likeness_calc

```
% likeness_calc

% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
%                                                                           *
%  File:   likeness_calc.m                                                  *
%                                                                           *
%  Created by: Ronnie E. Thebeau                                            *
%              System Design and Management Program                         *
%              Massacusetts Institute of Technology                         *
%                                                                           *
%  Date: December 2000                                                      *
%                                                                           *
%  Script file to get the average match between clusters of several runs    *
%  of the clustering algorithm.  Must change the master file below          *
%  to run the proper DSm calculations                                       *
%                                                                           *
%                                                                           *
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************


print_flag = 0;    % to toggle printing; 1=print, else no print

plot_flag  =  1;      % toggle plotting of the likeness averages for all of the
runs
printplot_flag  =  0;    % toggle to print the plots of the likeness averages
num_runs   =  10;

for get_data = 1:num_runs
   get_data
   run_cluster_A  % ******** name of master cluster DSM file to complete the 10
runs ********
   close all;
   % Extract some of the clustering results for analysis
   Results(get_data).Cluster_matrix = Cluster_matrix;
   Results(get_data).total_coord_cost = total_coord_cost;
   Results(get_data).cost_history = cost_history(1:max_run);
   Results(get_data).New_DSM_matrix = New_DSM_matrix;
   Results(get_data).New_DSM_labels = New_DSM_labels;
   Results(get_data).params = Cluster_param;
   Results(get_data).DSM_matrix = DSM_matrix;
end

get_date = now;
current_date = datestr(get_date,0);

% now plot the average matches
[Union_match] = find_cluster_matches(Results);
[Best_match, Average_match, Max_match, BclI,totalmean] =
get_match_avg(Union_match,Results, plot_flag, printplot_flag);
```

find_cluster_matches

```
function [Union_match] = find_cluster_matches(Results);
%[Union_match] = find_cluster_matches(Results);
%
%
% Function to find matching clusters from different runs of the clustering
algorithm
%
%
%
%  Inputs:
%          Results          Structure containing the results of multiple
%                           runs of the clustering algorithm
%
%  Outputs:
%          Union_match        Cell array of the match between every cluster
%                           of a run with every cluster of all other runs
%                           {Matrix_A,Matrix_B,Matrix_A_Cluster_x,
Matrix_B_Cluster_y} = match measurement

% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
%                                                                           *
%  File:   find_cluster_matches.m                                           *
%                                                                           *
%  Created by: Ronnie E. Thebeau                                            *
%              System Design and Management Program                         *
%              Massacusetts Institute of Technology                         *
%                                                                           *
%  Date: December 2000                                                      *
%                                                                           *
%  Function to find matching clusters from different runs of the            *
%  clustering algorithm                                                     *
%                                                                           *
%  This algorithm is based on work by Carlos Fernandez.                     *
%                                                                           *
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************


% match or likeness is calculated as follows

% [2*sum(#elements in both clusters)]/[Sum(# of elements in Cluster A) + Sum(#
of elements in Cluster B)]

num_mat = length(Results);


test_data = Results(1:num_mat);

for test_a = 1:num_mat
   [row_a, col_a] = find(test_data(test_a).Cluster_matrix);
   test_data(test_a).n_clusters = max(row_a);
end
```

```
for aa = 1:num_mat
   for bb = aa+1:num_mat
      for cc = 1:test_data(aa).n_clusters
         for dd = 1:test_data(bb).n_clusters
            Cluster_union{aa,bb,cc,dd} = ...
               (test_data(aa).Cluster_matrix(cc,:) .* ...
               test_data(bb).Cluster_matrix(dd,:));
            Union_sum{aa,bb,cc,dd} = sum(Cluster_union{aa,bb,cc,dd});
            Union_total_elmts{aa,bb,cc,dd} = ...
               (sum(test_data(aa).Cluster_matrix(cc,:) + ...
               test_data(bb).Cluster_matrix(dd,:)));
            if Union_total_elmts{aa,bb,cc,dd} > 0
               Union_match(aa,bb,cc,dd) =
2*Union_sum{aa,bb,cc,dd}/Union_total_elmts{aa,bb,cc,dd};
            else
               Union_match(aa,bb,cc,dd) = 0;
            end
         end
      end
   end
end
```

get_match_avg

```
function [Best_match, Average_match, Max_match, BclI, totalmean] =
get_match_avg(Union_match,Results, plot_flag, printplot_flag);
%[Best_match, Average_match, Max_match, BclI] =
get_match_avg(Union_match,Results);%
%
% Function averagve of matches between two runs of cluster calculations
%
%
%
%  Inputs:
%          Union_match           Cell array of likeness calculations between the
clusters
%          Results            Structure of Results from running the clustering
%                             algorithm multiple times
%          plot_flag          toggle the plotting of figures summarizing the
results (1=plot, 0=no plot)
%          printplot_flag        toggle printing of the plots
%
%  Outputs:
%          Best_match         the best match average for a cluster of run x to
any cluster
%                             in run y
%          Average_match      the average match for a cluster of run x to all
other best matched
%                             clusters in other runs
%                             in run y
%          Max_match          the best match for a cluster of run x to all other
best matched
%                             clusters in other runs
%                             in run y
%          BclI               the index of best matches between clusters
%          totalmean          mean of the likeness averages for all of the
clusters of a run


% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
%                                                                           *
%  File:   get_match_avg.m                                                  *
%                                                                           *
%  Created by: Ronnie E. Thebeau                                            *
%              System Design and Management Program                         *
%              Massacusetts Institute of Technology                         *
%                                                                           *
%  Date: December 2000                                                      *
%                                                                           *
%  Function to calculate the bids from clusters for the selected element.  *
%  Each cluster makes a bid for the selected element based on the           *
%  bidding parameters.  An plot the results.                                *
%                                                                           *
%                                                                           *
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
% ***************************************************************************
```

```
test_data = Results;              % get the clustering results
[num_runs] = size(Union_match,2);

for test_a = 1:num_runs
   [row_a, col_a] = find(test_data(test_a).Cluster_matrix);
   test_data(test_a).n_clusters = max(row_a);
end

for Run1 = 1:num_runs
   for Run2 = 1:num_runs
      if Run1 ~= Run2
         for Cluster_R1 = 1:test_data(Run1).n_clusters
            if Run1< Run2
               [Best_match(Run1,Run2,Cluster_R1),BclI(Run1,Run2,Cluster_R1)] =
max((Union_match(Run1,Run2,Cluster_R1, :)),[],4);
            else
               [Best_match(Run1,Run2,Cluster_R1),BclI(Run1,Run2,Cluster_R1)] =
max((Union_match(Run2,Run1,:, Cluster_R1)),[],3);
            end
         end
      end
   end
end

c_run = [];
for Run1 = 1:num_runs
   for j=1:num_runs
      if j ~= Run1
         c_run = [c_run, j];
      end
   end

   for Cluster_r1 = 1:test_data(Run1).n_clusters
      Average_match(Run1,Cluster_r1) = mean(Best_match(Run1,c_run,Cluster_r1));
      Max_match(Run1,Cluster_r1) = max(Best_match(Run1,c_run,Cluster_r1));
   end
end


if plot_flag == 1

   plots_per_page = 5;

   [num_runs, num_clusters] = size(Average_match);


   plot_num = 0;
   figure;
   for i=1:num_runs
      n_cluster = nnz(Average_match(i,:));
      totalmean(i) = mean(Average_match(i,1:n_cluster));
   end

   Total_run_mean = num2str(mean(totalmean));
   Total_run_median = num2str(median(totalmean));

   for i=1:num_runs
      plot_num = plot_num+1;
      if plot_num > plots_per_page
         if printplot_flag ==1
            print
         end
         figure;
         plot_num = 1;
      end
      subplot(plots_per_page,1,plot_num);
      n_cluster = nnz(Average_match(i,:));
      bar(Average_match(i,:),'k');
      ylabel(['Run ' num2str(i) ' Avg']);
      Tcost = Results(i).total_coord_cost;
      xlabel(['Clusters  (total Avg: ' num2str(totalmean(i)) ' ); Coord Cost: '
num2str(Tcost)]);
      if (i == 1) | (mod(i,(plots_per_page+1))==0)
         title(['Mean: ' Total_run_mean '      Median: ' Total_run_median]);
      end
      grid;
      orient tall;
      hold on;
      plot(Max_match(i,:),'rx');
   end


   if printplot_flag ==1
      print
   end
end
```

Bibliography

Bartkowski, Glenn D. “Accounting for System Level Interactions in Knowledge Management Initiatives” SM Thesis. Massachusetts Institute of technology, Feb 2000.

Dong, Qi. "Representing Information Flow and Knowledge Management in Product Design Using the Design Structure Matrix" SM Thesis. Massachusetts Institute of Technology, Jan 1999.

Fernandez, Carlos Iñaki Gutierrez. “Integration Analysis of Product Architecture to Support Effective Team Co-location” SM Thesis. Massachusetts Institute of Technology, 1998.

The MIT Design Structure Matrix – DSM – Home Page. Massachusetts Institute of Technology. 19 Dec 2000. < http://web.mit.edu/dsm/>

Otis Impact Resource CD. Vers. 3.1. CD-ROM. Otis Elevator Company 2000

Otis.com. Otis Elevator Company. 19 Dec 2000. <http://www.otis.com/>

Pimmler, Thomas U. and Stephen D. Eppinger, “Integration Analysis of Product Decomposition” Working Paper. Alfred P. Sloan School of Management, Massachusetts Institute of Technology, May 1994: WP #3690-94-MS

Rechtin, Eberhardt. Systems Architecting. Englewood Cliffs: Prentice Hall, 1991

Rechtin, Eberhardt and Mark W. Maier. The Art of Systems Architecting. NewYork: CRC Press, 1997

Ulrich, Karl T. and Stephen D. Eppinger. Product Design and Development. New York: McGraw-Hill, 1995

THESIS PROCESSING SLIP

FIXED FIELD: ill. ______ name ______

index ______ biblio ______

► COPIES: Archives Aero Dewey [✓ Barker] Hum

Lindgren Music Rotch Science Sche-Plough

TITLE VARIES: ► ☐ ______

NAME VARIES: ► ☐ [illegible]

IMPRINT: (COPYRIGHT) ______

► COLLATION: ______

► ADD: DEGREE: ______ ► DEPT.: ______

► ADD: DEGREE: ______ ► DEPT.: ______

SUPERVISORS: ______

NOTES:

cat'r: date:

page:

► DEPT: [illegible]

► YEAR: ______ ► DEGREE: ______

► NAME: ______