12TH INTERNATIONAL DEPENDENCY AND STRUCTURE MODELLING CONFERENCE, DSM'10
22 – 23 JULY 2010, CAMBRIDGE, UK

PATH-BASED AND PATTERN-BASED APPROACHES FOR CHANGE MANAGEMENT

Simon Li and Elmira Rajinia
Concordia Institute for Information Systems Engineering, Concordia University, Canada

Keywords: change management, decision making

1 INTRODUCTION

When changes are required for an existing complex system, there can be several feasible change options that can be applied to achieve the change requirements. Due to the complexity of the original system, it is not trivial to choose an appropriate change option. This paper is intended to address this issue in two steps. Firstly, via the matrix-based modelling techniques, we characterize the possible options to address a change scenario. Secondly, we apply different quantitative approaches to help select the appropriate option for change management.

In general, product design and development can be envisioned as a system of interconnected product, process and organization entities (e.g., product components, process activities and project teams) (Eppinger & Salminen, 2001). One essence of matrix-based modelling in this context stems from the explicit description of product (or process) entities and characterization of their dependency relationships. As the matrix models (e.g., design structure matrix and domain mapping matrix) clearly shows the relationships between two entities, they help us to understand how one entity would influence another entity due to changes. Then, this dependency information becomes the major means to evaluate different change options due to some initial changes in a system.

In literature, dependency-based (especially using matrix) models have been used to manage and control the propagation of changes. Ollinger and Stahovich (2004) have used a causal model to capture the causal dependency among design parameters for managing design changes. Clarkson et al. (2004) used likelihood, impact and risk DSMs to perform risk management for design changes. Chen et al. (2007) and Li & Chen (2007) developed a model-based rapid redesign methodology that is able to control complex design change propagation using their matrix-based decomposition techniques.

Among these research efforts, this paper presents the path-based and pattern-based approaches to formulate and evaluate the change options in change management. The path-based approach uses the dependency information available in the matrix models to trace the propagation paths for different change options. This path-tracing effort can be utilized to evaluate the quality of different change options. Alternatively, the pattern-based approach intends to identify the matrix-based structures of a system to estimate the scope of change propagation for different change options. We will present two cases in the following to illustrate the path-based and pattern-based approaches respectively for change management.

2 ILLUSTRATION OF THE PATH-BASED APPROACH: SOFTWARE PROGRAM DEBUGGING

The software program in this illustration is a guess-the-number game, which is obtained from Sierra and Bates (2005). The game starts from generating three random integers between 1 and 9. Then, the player is asked to guess the values of these three numbers. At the end, the program will return the number of guesses that the player has made. This is an object-oriented program (OOP), and 25 OOP entities are identified, including different classes, methods and interfaces. Then, a 25×25 DSM is constructed to capture the dependency of these OOP entities. The DSM is shown in Figure 1. Each matrix entry, denoted as m_{ij}, is referred to the potential change on the ith entity due to the change of the jth entity.

A program's bug is found in the original program. Due to this bug, the program will increase the number of hits every time the user guesses one of the generated numbers, even if that number had already been guessed. To remove this bug, we need to distinguish the case if the user has repeatedly guessed the same number generated by the program. If this is the case, the program will not count it as a new hit. To correct this bug, four change options are proposed (Sierra and Bates, 2005), and they are summarized in Table 1.

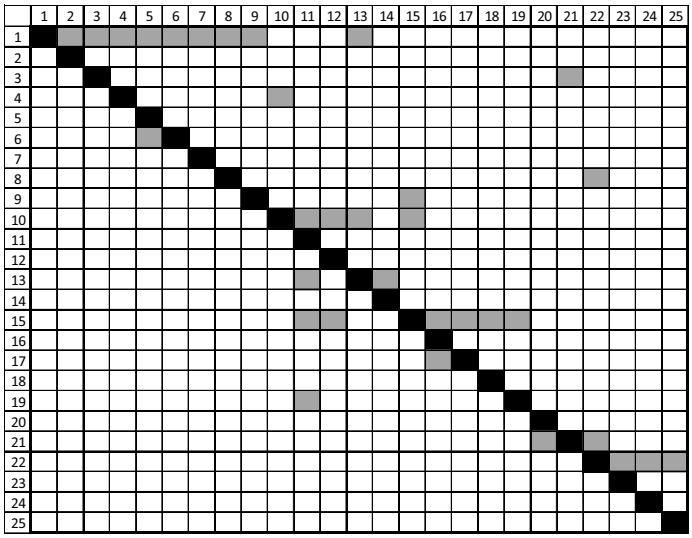


Figure 1. DSM of 25 OOP entities

Table 1. Four options to modify and debug the program

Option 1	A second array is created. At each time when the user makes a guess, the modified program stores the guessed number in the second array. When the user makes another guess, the program will check the second array for any repeated guesses.
Option 2	The original array would be kept but the values of any correctly guessed numbers in the array would be changed to -1. In this option, there is only one array to check and manipulate.
Option 3	The location of each number that is guessed correctly should be removed from the original array, and the array should be modified to a smaller one. Since the size of an array cannot be changed, we have to make a new array and copy the remaining cells from the old array to the new and smaller array.
Option 4	Use a specialized function from the Java library to perform the modification in the Option 3.

The options shown in Table 1 indicate that different initial OOP entities need to be changed for debugging. The decision question is which option we should select to debug the program. The selection strategy in this case is to select the option that has the minimum changes on the original program. In the path-based approach, we first identify the entities that correspond to the initial changes. For instance, the Option 1 needs to modify the entities #10 and #15 (based on the DSM labels) to create the second array and update the check conditions. By checking the 10^{th} column of the DSM in Figure 1, we found that the entity #10 will potentially affect the entity #4. In the same way, we found that the entity #15 will potentially affect the entities #9 and #10. The propagation can continue by inspecting the next affected entities due to the changes of entities #4, #9 and #10. Figure 2 illustrates the propagation paths pertaining to Option 1.

After identifying the propagation paths, it is assumed that the effect of the initial changes will decrease along the propagation paths. That is, the entities located farther on a propagation path (or at a higher propagation level) will have less chance to be actually affected after the implementation of all the changes. In this case, we denote the propagation probability as the probability that the change propagation takes place due to direct dependency. In this example, we set the propagation probability

equal to 0.5 for simplicity. That is, we have a half chance that the change of one entity will directly modify another immediate entity on the propagation path. Another simplification in this example is to limit the length of the change propagation paths up to three levels. Based on this probabilistic approach, we can estimate the expected numbers of other OOP entities that are affected by the initial changes.

Table 2 summarizes the results of this case. The column of *estimated propagation scope* lists the normalized rating (from 0 to 1) of the scope of change propagation for each option. The higher rating value indicates the larger scope of changes. All four options are eventually implemented for comparison in this case study. The last column of Table 2 records the number of lines that are actually modified according to the corresponding change option. As seen, the order of the estimated propagation scope follows the order of the actual number of lines modified in the program. This result is quite satisfactory.

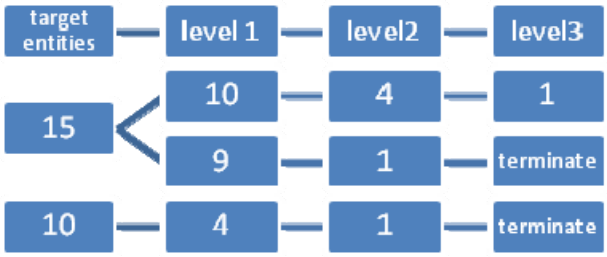


Figure 2. Change propagation paths of Option 1

Table 2. Estimation and verification of change results

Change option	Estimated propagation scope	Number of modified lines of codes
#1	0.383	8
#2	0.274	3
#3	0.673	14
#4	1.000	20

3 ILLUSTRATION OF THE PATTERN-BASED APPROACH: RELIEF VALVE REDESIGN

The relief valve example is obtained from Kannapan and Marshek (1993), and we have modelled this example using 29 design functions and 49 design parameters. Then, a 29×49 DMM, shown in Figure 3, is constructed to capture the dependency between functions and parameters. In particular, the rows of the DMM correspond to the design functions, and the columns correspond to the design parameters. Then, each matrix entry, m_{ij}, indicates the presence of dependency. That is, if the jth parameter is related to the ith function, the entry m_{ij} is equal to 1 (or shaded as shown in Figure 3). Otherwise, the entry m_{ij} is equal to 0 (or un-shaded).

Given the matrix-based design model, new requirements are invoked to improve some design parameters. The verbal statements of the change requirements are summarized in Table 3, which shows that the parameters of Columns 9, 19 and 26 (symbolized C_9, C_{19} and C_{26}) need to be modified and thus labelled as target parameters. By checking the DMM, all the functions that depend on them are considered the target functions. Then, all the parameters that these functions depend on are also labelled as the target parameters. The indices of the target function rows and the target parameter columns are shaded in the DMM in Figure 3.

Based on the DMM, if changes occur to the parameter C_{19}, the function R_4 (i.e., the function labelled with '4' in the row) will be affected and this R_4 will potentially affect other parameters (e.g., C_{16} and C_{41}). In change management of this case, we will have the options of which parameters (C_{16} and/or C_{41}) should be relaxed and modified to achieve the new requirements. Different options can lead to different paths and scopes of change propagation.

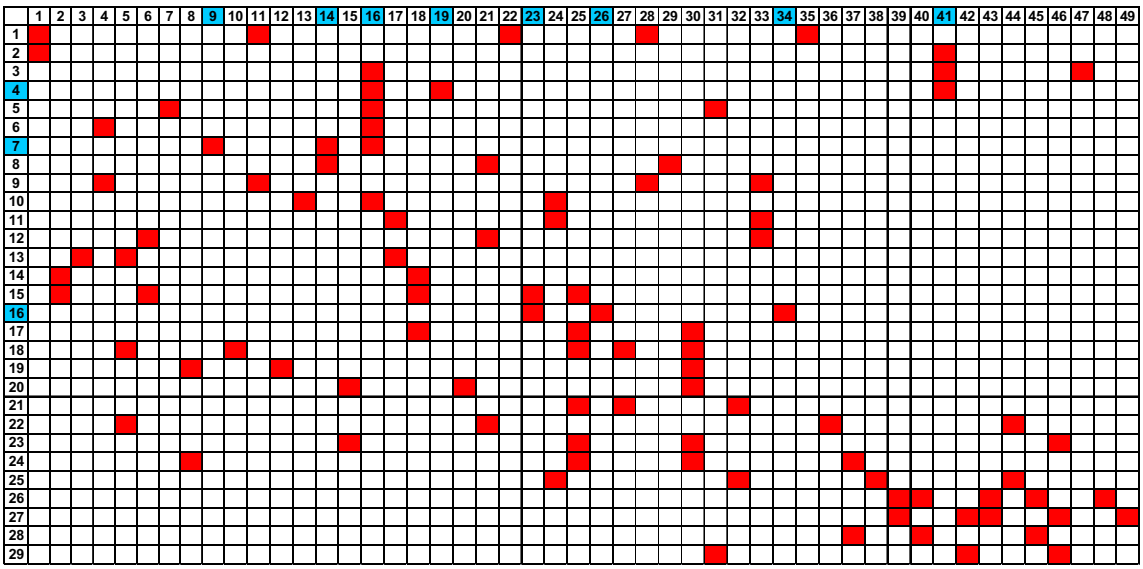


Figure 3. DMM of 29 functions (rows) and 49 parameters (columns)

Table 3. Summary of initial change requirements

C_9	Increase seal thickness for more secure fluid flow seal-off
C_{19}	Decrease valve head loss to below a new maximum allowable value
C_{26}	Decrease allowable helical spring material stress for greater valve reliability and overall safety

In the above context, the purpose of the pattern-based approach is to estimate the scopes of change propagation using matrix patterns. Using the decomposition method reported in Chen et al. (2007), five different patterns are generated and shown in Figure 4. In brief, these patterns are obtained by first clustering the rows and columns of the original matrix, thus yielding several block-angular matrix solutions. Then, the blocks that contain the target rows and columns are highlighted to form a pattern. Each pattern essentially conveys two pieces of information: matrix-based structure and distribution of target entities. The matrix-based structure shows the formation of blocks (along the diagonal) and the blocks' interactions (i.e., interaction columns on the left and/or interaction rows on the top). In these structures, target entities are distributed in different ways. For better visualization, the blocks that contain target entities are shaded in darker colour in Figure 4.

While these target entities are the initial points to propagate changes, the matrix-based structures help to investigate how the changes will be confined within a block or propagated to other blocks via the interactions parts. For instance, Pattern 1 in Figure 4 shows that two blocks contain target entities (as highlighted). These blocks essentially contain the subsets of design functions and parameters that will likely be affected in redesign. Yet, if the interaction parameters in Pattern 1 are fixed in their values, changes can be confined in these two target blocks. In this case, we intend to estimate the scope of change propagation entailed in matrix patterns via two factors: intensity and interdependency.

The first factor, intensity, focuses on the size of the blocks and the interaction parts that contain target entities. If the size of these "target" blocks and interactions is larger, the value of the intensity index will be higher. In contrast, the second factor, interdependency, focuses on the coupling relationships between blocks. If the interactions between blocks are heavy, the value of the interdependency index will be higher. The details of these two indices (e.g., formulations) can be found in Chen et al. (2007).

Table 4 summarizes the values of normalized intensity and interdependency indices for each pattern. From Table 4, we observe that Pattern 1 and Pattern 5 pertain to a trade-off relationship. First of all, Pattern 1 (see Figure 4) has relatively large target blocks, leading to a high intensity value (i.e., 0.92). In contrast, Pattern 5 has many interaction rows and columns, thus leading to a high interdependency value (i.e., 1.00). In the case of applying even weights on intensity and interdependency values, Pattern 5 is selected. The justification and validation of this selection can be found in Li and Chen (2007).

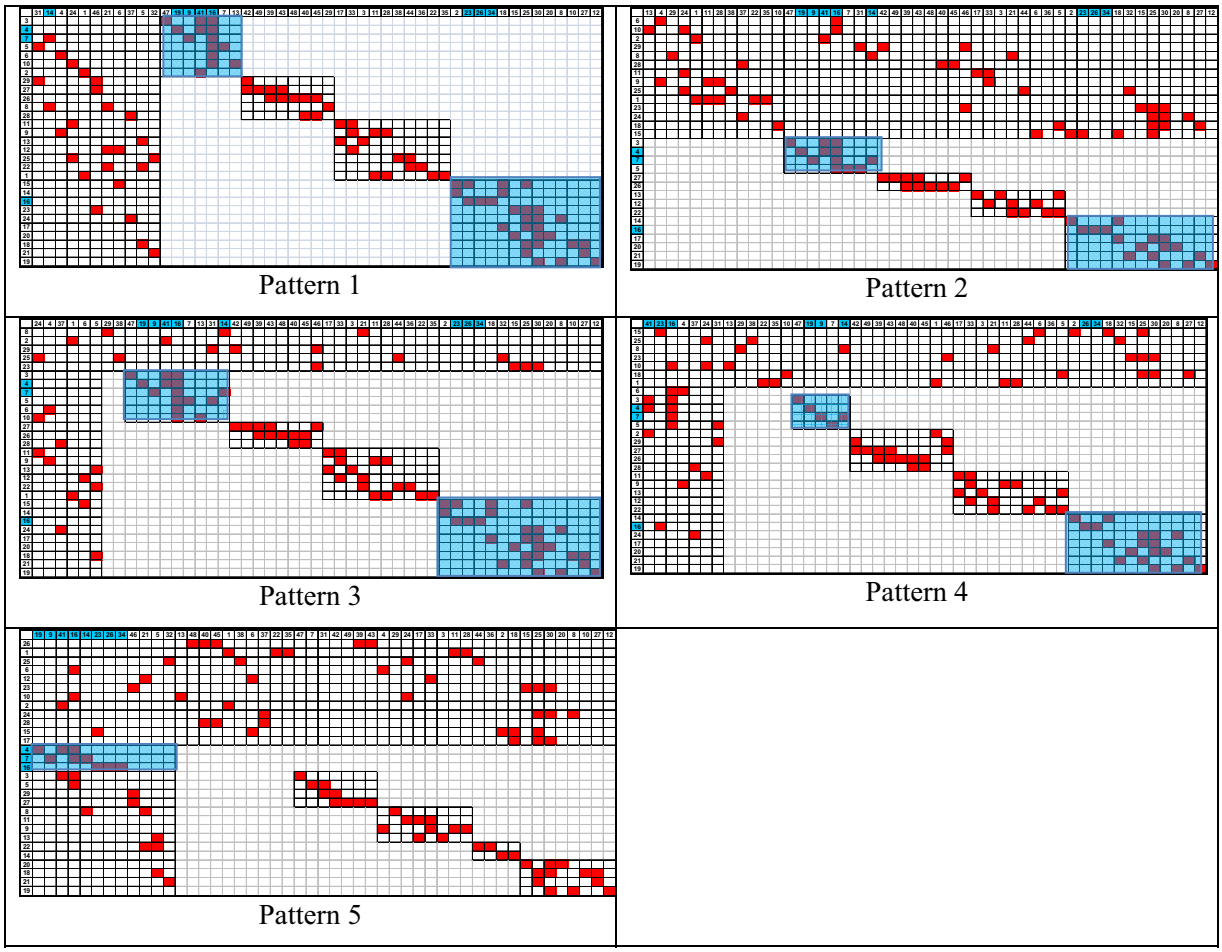


Figure 4. Five patterns for change management

Table 4. Intensity and interdependency values of five patterns

	Intensity Index	Interdependency Index
Pattern 1	0.92	0.53
Pattern 2	0.81	0.62
Pattern 3	1.00	0.61
Pattern 4	0.79	0.78
Pattern 5	0.28	1.00

4 CONCLUSION

This paper focuses on the decision aspect in change management. When changes are required towards a complex system, it can be the case that several change options are available to execute the changes. This paper discusses the path-based and pattern-based approaches to estimate the scopes of change propagation. At this preliminary stage of research, the future work includes the following:

- More case studies to examine the nature of path-based and pattern-based approaches
- Effective algorithms to generate paths and patterns in change management
- Comprehensive software tools to support path-based and pattern-based approaches in change management

At the current stage, we are studying the proposed change management approaches in the context of software systems, project management, and manufacturing systems.

REFERENCES

Chen, L., Macwan, A., and Li, S. (2007). Model-Based Rapid Redesign Using Decomposition Patterns. *Journal of Mechanical Design*, 129, 283-294.

Clarkson, P.J., Simons, C., and Eckert, C. (2004). Predicting Change Propagation in Complex Design. *Journal of Mechanical Design*, 126, 788-797.

Eppinger, S.D. and Salminen, V. (2001). Patterns of Product Development Interactions. *Proceedings of International Conference on Engineering Design (ICED)*, Glasgow, Aug. 21-23.

Kannapan, S.M. and Marshek, K.M. (1993). An Approach to Parametric Machine Design and Negotiation in Concurrent Engineering. In A. Kusiak (Ed.), *Concurrent Engineering: Automation, Tools and Techniques*, John-Wiley, New York, pp. 509-533.

Li, S. and Chen, L. (2007). Towards Rapid Redesign – Pattern-Based Redesign Planning for Large-Scale and Complex Redesign Problems. *Journal of Mechanical Design*, 129, 227-233.

Ollinger, G.A. and Stahovich, T.F. (2004). RedesignIT – A Model-Based Tool for Managing Design Changes. *Journal of Mechanical Design*, 126, 208-216.

Sierra, K., and Bates, B. (2005). *Head First Java*, O'Reilly Media, Inc., Sebastopol, CA, USA.

Contact: Simon Li
Concordia University
Concordia Institute for Information Systems Engineering
1455 de Maisonneuve Blvd. West, S-EV 007.648
Montreal, Quebec
Canada, H3G 1M8
Phone 1-514-848-2424 ext. 5621
Fax 1-514-848-3171
e-mail lisimon@ciise.concordia.ca

Path-Based and Pattern-Based Approaches for Change Management

Simon Li
Elmira Rajinia

Concordia Institute for Information Systems Engineering, Concordia University, Canada

Technische Universität München

UNIVERSITY OF CAMBRIDGE

MANAGING COMPLEXITY
BY MODELLING DEPENDENCIES

Index

- Introduction
- Path-based Approach
- Pattern-based Approach
- Summary and Future Works

Introduction

- Changes are inevitable in systems.
- Several feasible change options can be available to address one change scenario.
- Due to the complexity of the original system, it is not trivial to choose an appropriate change option.
- Complex system → a system of interconnected product, process and organization entities
  - Matrix-based modelling: explicit description of these entities and their dependency relationships
  - To understand how one entity would influence another entity due to changes
- Paper's purpose:
  - Present the path-based and pattern-based approaches to formulate and evaluate the change options in change management

Path-Based and Pattern-Based Approaches

- Question: how to estimate the scope of change propagation given a specific change on a system
- Path-based approach
  - Use the dependency information available in the matrix models to trace the propagation paths.
- Pattern-based approach
  - Identify the matrix-based structures of a system to estimate the scope of change propagation.

Path-Based Approach – The Example

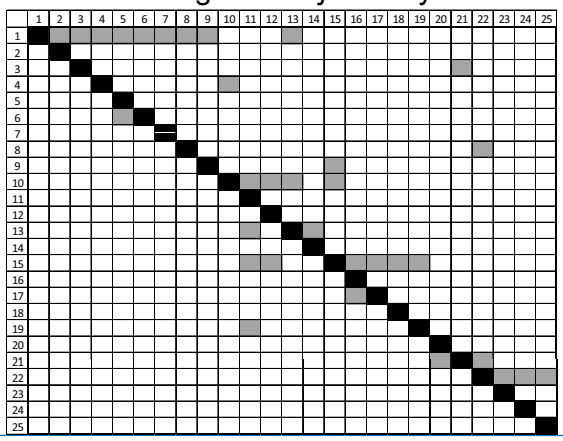
- Software: a guess-the-number game
  - The game starts from generating three random integers between 1 and 9.
  - Then, the player is asked to guess the values of these three numbers.
  - The program will return the number of guesses that the player has made.
- The bug
  - The program will increase the number of hits every time the user guesses one of the generated numbers, even if that number had already been guessed.
- Four options are available to fix this bug.
- Change Management Question: which option we should select to minimize the modifications of the original software program?

Path-Based Approach – Options to Remove the Bug

Option 1	- Create a second array. - Store the guessed number in the second array every time when the user makes a guess. - When the user makes another guess, the program will check the second array for any repeated guesses.
Option 2	- Keep the original array but the values of any correctly guessed numbers in the array would be changed to -1. - There is only one array to check and manipulate.
Option 3	- The location of each number that is guessed correctly should be removed from the original array, and the array should be modified to a smaller one. - Since the size of an array cannot be changed, we have to make a new array and copy the remaining cells from the old array to the new and smaller array.
Option 4	- Use a specialized function from the Java library to perform the modification in the Option 3.

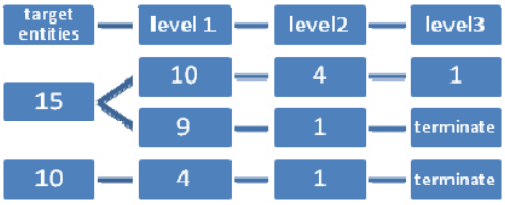
Path-Based Approach – The DSM

- This is an object-oriented program (OOP)
- 25 OOP entities are identified, including different classes, methods and interfaces.
- A 25×25 DSM is constructed to capture the dependency of these OOP entities.
- Each matrix entry, denoted as m_{ij}, is referred to the potential change on the *i*th entity due to the change of the *j*th entity.



Path-Based Approach – Change Scope Estimation

- Each change option initially triggers changes of different OOP entities.
- Via the DSM, trace the change propagation path from initial changes to other OOP entities.
- For example, we have the following tracing paths for Option 1:



- Estimation of the scope of change propagation:
  - The effect of the initial changes will decrease along the propagation paths.
  - Propagation probability → chance about changing one entity will directly modify another entity
  - Estimate the expected numbers of OOP entities that are affected by the initial changes

Path-Based Approach – The Results

- Four change options have been implemented to verify the results.
- The order of the estimated propagation scope follows the order of the actual number of lines modified in the program. This result is quite satisfactory.

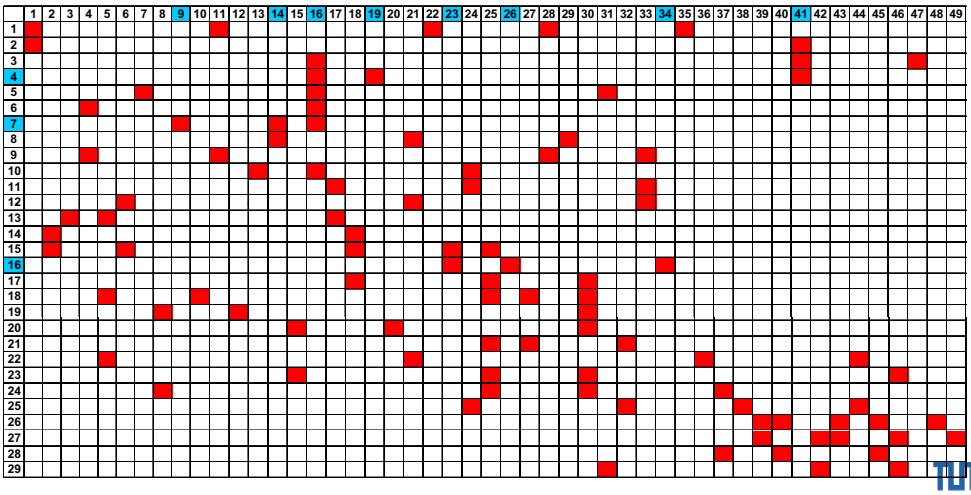
Change Option	Estimated Propagation Scope	Number Of Modified Lines Of Codes
#1	**0.383**	**8**
#2	**0.274**	**3**
#3	**0.673**	**14**
#4	**1.000**	**20**

Pattern-Based Approach – The Example

- The example is about the redesign of a relief valve system.
- The example consists of 29 design functions and 49 design parameters.
- Redesign requirements:
  - Increase seal thickness for more secure fluid flow seal-off (related to the parameter C_9).
  - Decrease valve head loss to below a new maximum allowable value (related to the parameter C_{19}).
  - Decrease allowable helical spring material stress for greater valve reliability and overall safety (related to the parameter C_{26}).
- Change Management Question: which parameters we should modify in order to achieve the redesign requirements.

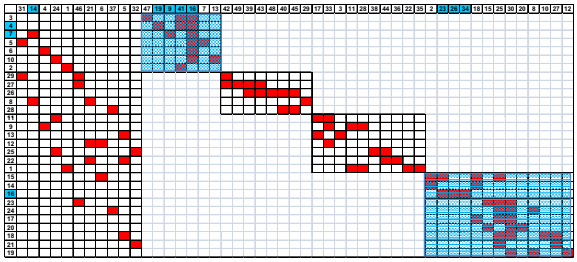
Pattern-Based Approach – The DMM

- A 29×49 DMM is constructed to capture the dependency between 25 functions and 49 parameters.
- Each matrix entry, m_{ij}, indicates the presence of dependency. That is, if the *j*th parameter is related to the *i*th function, the entry m_{ij} is shaded.
- The shaded row / column labels indicate the target entities.



Pattern-Based Approach – Matrix Patterns

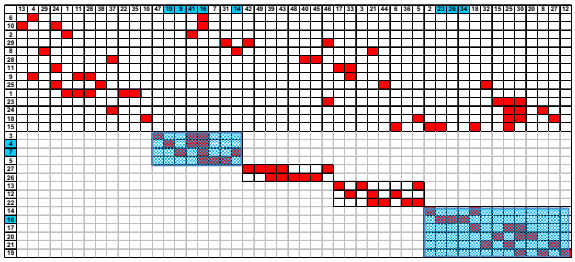
- How does a matrix pattern help for change management?
  - Target entities indicate the initial points to propagate changes.
  - The target blocks define the subsets of “likely affected” entities.
  - The interactions help to control the propagation.



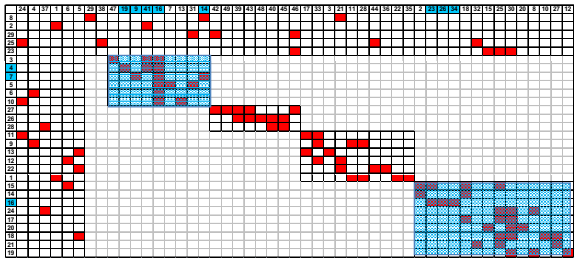
Pattern 1

- How to get matrix patterns?
  - Step 1: cluster the matrix’s rows and columns to form block-angular matrices (i.e., blocks on the diagonal with interaction columns on the left and/or interaction rows on the top)
  - Step 2: identify the locations of target entities in the block(s) and the interactions

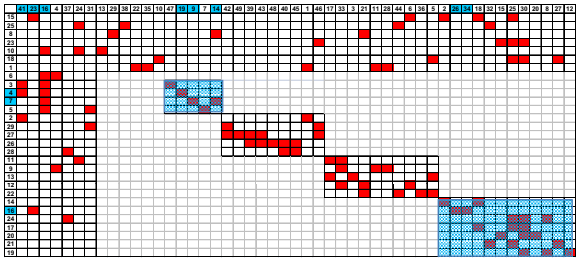
Pattern-Based Approach – Some Other Patterns



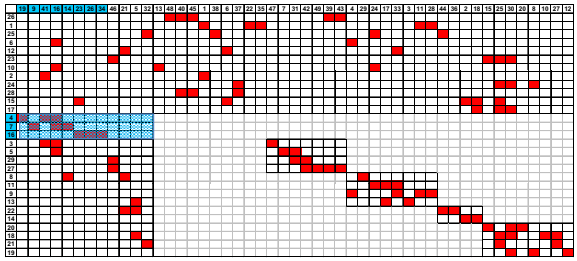
Pattern 2



Pattern 3



Pattern 4



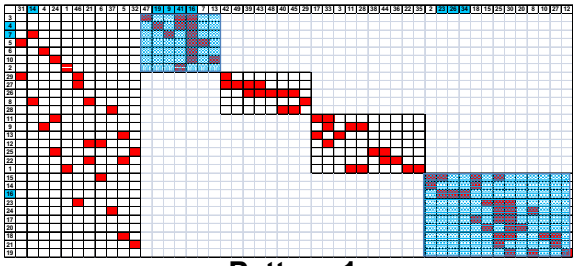
Pattern 5

Pattern-Based Approach – The Selection Problem

- Each pattern indicates a unique way to control the change propagation.
- The selection problem → which pattern should be selected for redesign?
- Estimation of the scope of change propagation is done via two factors: intensity and interdependency.
- Intensity
  - Focus on the size of the blocks and the interaction parts that contain target entities.
  - If the size of these “target” blocks and interactions is larger, the value of the intensity index will be higher.
- Interdependency
  - Focus on the coupling relationships between blocks.
  - If the interactions between blocks are heavy, the value of the interdependency index will be higher.

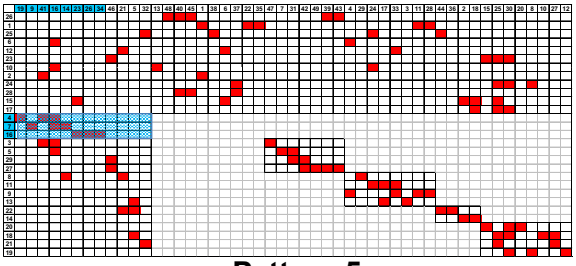
Pattern-Based Approach – The Example Results

	Intensity Index	Interdependency Index
Pattern 1	**0.92**	***0.53***
Pattern 2	**0.81**	**0.62**
Pattern 3	**1.00**	**0.61**
Pattern 4	**0.79**	**0.78**
Pattern 5	***0.28***	**1.00**



Pattern 1

- Large target blocks → high intensity
- Small interactions → low interdependency (0.53)



Pattern 5

- Target entities grouped in interaction only → low intensity (0.28)
- Large interactions with all target entities → high interdependency

Summary and Future Works

- There can be more than one option to fulfill the initial change requirements (e.g., debug a software program and redesign a mechanical device).
- Based on some matrix models, the scope of change propagation can be estimated for different change options.
- The path-based and pattern-based approaches have been used to perform the estimation of change propagation.
- Future works
  - More case studies to examine the nature of path-based and pattern-based approaches
  - Effective algorithms to generate paths and patterns in change management
  - Comprehensive software tools to support path-based and pattern-based approaches in change management