공학박사 학위논문

Engineering Change Management using Bayesian Network

설계변경 관리를 위한 베이지안 네트워크 중심 접근법

2015 년 6 월

서울대학교 대학원

산업공학과

이 지 환

Abstract

Engineering Change Management using Bayesian Network

Jihwan Lee

Department of Industrial Engineering

The Graduate School

Seoul National University

An engineering change is defined as the changes in forms, fits, materials, dimensions or functions of a product or component. As a product evolves into a complex system, engineering changes become major driving force for determining schedules, costs, and quality of product development process. Change management is defined as company's effort of making changes to a product in a planned or systematic fashion. Several efforts have been tried from both academia and industry; these includes implementing computer aided systems to streamline change implementation process, establishing guidelines of product design, and developing tools and methodologies for analyzing engineering changes in advance

One major challenge in change management is change propagation. In a complex system where each part is associated with several other parts, a change presented into a part may influence other parts. As a result, a change in a component may propagate throughout entire system resulting in avalanches of changes. To cope with the risk of change

propagation, a number of strategies are proposed to avoid changes as much as possible. These strategies includes for example to make changes as early as possible, or to create design buffers. However, eliminating entire changes during the product development process is neither desirable nor possible.

In order to tackle these difficulties, this paper proposes a systematic method to predict and analyze engineering changes in advance. Bayesian network, which is a main modeling language of this paper, can effectively address both of the complex and uncertain aspects underlying engineering changes with the support of mathematical rigor. Based on a probability theory, one can anticipate risk of engineering changes in the form of conditional probability distribution. The resulting probability distribution can also be utilized for guiding optimal design decisions that reduce the risk of unnecessary changes.

This thesis consists of three parts. The first part focuses on change prediction. In order to avoid unnecessary changes, likelihood and consequence of design changes should be analyzed in advance. In this section, Bayesian network is utilized for encoding probabilistic relationship among components and generating conditional probability distribution of each component with respect to various change scenarios. Due to its ability to model and reasoning of uncertain domains, Bayesian-network-based approaches shows significant advantages over traditional approaches in terms of modeling, analysis, and data learning aspects.

The second part focuses on controlling change propagation. Especially, this part focuses on derivation of a design freeze sequence. De-

sign freeze is the end point of design phase at which a technical product description is handed over to production. One way to mitigate the risk of change propagation is to impose a design freeze on components at some point prior to completion of the design process. In this part, a Dynamic Bayesian Network was used to represent the change propagation process within a system. According to the model, when a freeze decision is made with respect to a component, a probabilistic inference algorithm within the BN updates the uncertain state of each component. Based on this mechanism, a set of algorithm was developed to derive optimal freeze sequence.

The third part focuses on learning change prediction model from engineering change log database. When a product becomes complex, identifying complex relationship among components complex based on expert elicitation would become almost impossible. One alternative is to automatically mine change propagation network from a collection of previous change records. As a modeling language, dependency network, a graphical model for representing probabilistic relationships among random variables, was utilized. As a result, a complete joint probability distribution that can predict the probability of change propagation was obtained. A case study on Azureus, an open-source software project, was conducted to validate the proposed approach.

Keywords: Engineering Change Management, Bayesian Network, Change Propagation, Change Prediction, Change Impact An alysis, Design Freeze Planning, Dependency network, Data Learning
Student Number: 2008-21237

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

The purpose of this study is to propose a methodology for predicting and managing engineering changes during product development process. This chapter presents backgrounds of research and specific research questions. In section 1.1, the definition and issues involved in engineering change management is introduced. Section 1.2 introduces the concept of change propagation which is one of challenges in engineering change management. Section 1.3 addresses current practice of managing change propagation and its limitation. And finally, Section 1.4 propose a Bayesian network framework for improving engineering change management.

1.1. Engineering Change Management

In order to survive from fierce competition, a company should continuously introduce their new products. It is noteworthy that most of new products are derived from their previous models. This makes the engineering change as one of the important managerial issue in product development management. Moreover, the engineering change becomes an important issue with the rise of new management concepts such as concurrent engineering, mass customization, or product platform development. For the successful execution of these disciplines, efficient and effective management of engineering change is required.

Contrary to the general concept of change in a business or organizational context, engineering change is directly concerned with modification of the products. In product development literature, there has been

numerous definitions of the engineering change. Wright (1997) defines an engineering change as a modification to a component of a product, after that product has entered production. Huang and Mak (1999) defines engineering changes as the changes in forms, fits, materials, dimensions or functions of a product or component. Terwiesch and Loch (1999) defines an engineering change as changes to parts, drawings or software that have already been released. Summarizing above definitions, Jarratt et al. defines an engineering change as an alteration made to parts, drawings or software that have already been released during the product design process. The change can be of any size or type; the change can involve any number of people and take any length of time.

Summarizing above definitions, the engineering change is routine task which occurs throughout the entire product development cycle. An engineering change may be as simple as documentary amendments, or as complicated as the entire redesign of products and manufacturing processes. Moreover, engineering changes can occur throughout entire product development lifecycle from early stage of product development process such as conceptual design or requirement specification to later stages when manufacturing process are established or even in stages when products are operated by customers. Thus, the engineering change is necessary and inevitable during product development process.

Engineering changes are motivated by several factors. According to Jarratt et al. (2011), there are two motivations of engineering changes: (1) emergent change and (2) initiated change. An emergent changes arises from the property of product itself. For example, emergent changes includes fixing design errors such as minor drawing errors or

problems occur during product integration; satisfying safety issues regulated by government; modifying specification to meet original functional requirement; or correcting design for resolving quality issues. On the other hand, an initiated change is requested by external stakeholders which are involved in product development process. For example, a customer may frequently request modifications of specification or insertion of functionality. Suppliers may suggest changes to comply with technical standards or alter material specification of the product. Production engineer may request modification of a product in order to improve the manufacturability. Maintenance staff may require engineering changes in order to improve the diagnosability of a product.

An engineering change also can occur at any point during the product development process. Jarratt et al. (2011) illustrates various time points where engineering change can occur. Once the concept of the design has been defined and corresponding design data and information is formally released to design team and suppliers, engineering change can be triggered at any point of times. It is noteworthy that an engineering change still occur after the design is handed over to manufacturing and production phase. Although these changes are referred to different terms such as product change (at manufacturing process), prototype change (at testing phase), they also can be included in the boundary of the engineering change.

Although the engineering change is ubiquitous concept in product development process, it is essential factors that determines the performance of the product development process. Actually, it is reported that one-third to one-half engineering capacity is consumed in handling en-

gineering changes during developing complex systems. Moreover, engineering change is also represents 20% to 50% of tooling cost. A survey of German engineering companies show that approximately 30% of all work effort has been consumed in handling engineering changes.

Due to the importance of the engineering change, many companies has adopted formal engineering change process. Jarratt et al. (2011) describes six-step procedure of change management process. At first, the person begin to raise change request (*change request*). He or she should record the change-related information such as the reason, the priority, types, components or subsystems that are likely to be affected in the formal document. This form is uploaded onto the database. Next, potential solution to the change request is selected once it seems feasible (*identification of solution*). After feasible solutions are proposed, the impact or risk of implementing this solution is evaluated (*Risk/impact assessment*). In evaluating the proposal, various factors such as overall schedule, supplier's over run contract, additional engineering cost, etc. are considered. Proposed solution is then approved by a change review organization (*approval of solution*). Engineering change board often consists of middle or senior staffs from various disciplines such as product design, manufacturing, marketing, supply-chain management, quality assurance, finance, supporting, etc. After approval of change request, it is actually implemented, and corresponding paper work also is updated (*Implementation*). Finally, whole change process is reviewed and any knowledge is accumulated in order to using them in future change process (*Review of particular change process*).

Although many companies has adopted formal engineering change process, many of them still suffer from properly controlling engineering changes. According to the survey of Acar et al. (1998) which examined engineering companies in UK who has capability of both design and manufacturing of the product, more than half of the companies agreed that engineering change is their major source of problems.

1.2. Change Propagation

One major pitfall of change management is change propagation. When developing a complex system where each part of system is highly inter connected with other components, a change made in one component often result in changes to other parts, which in turn propagate further. For example, consider the case of designing an illustrative product as shown in Figure 1-1. Suppose that a certain change request is made to an Engine. Since design parameters of Engine depends on parameters in other subsystem, a change made in Engine can spread to other parts such as Cabling and Piping, Engine Auxiliaries, and Engine Casing. In this way, a change made in only one component even can spread to entire subsystems. Furthermore, sometimes a change can even spread to other product when a changed component is commonly shared by other members of a product family. Therefore, if not managed properly, change propagation can have significant impact on entire product development performance.

Product architecture determines the behavior of change propagation within a system. Product architecture is defined as a scheme by which the function of a product is allocated to physical components. More

specifically, it consists of three elements: (1) the arrangement of functional element; (2) the mapping from functional element to physical components; and (3) the specification of interfaces among the interacting physical components. Product architecture can explain how an engineering change spread to entire system. Figure 1-2 illustrates an illustrative example of product architecture. With this viewpoint of product architecture, a change is initiated by changing requirement of functional element. This change of functional element then invokes the change of corresponding physical components. The change of the physical element then spread to other physical interacting components.

According to the arrangement between functional element and physical elements, the product architecture can be classified into two archetypes: (1) modular architecture, and (2) integral architecture. Modular architecture has the one-to-one connection between function and physical element and components are loosely coupled with each

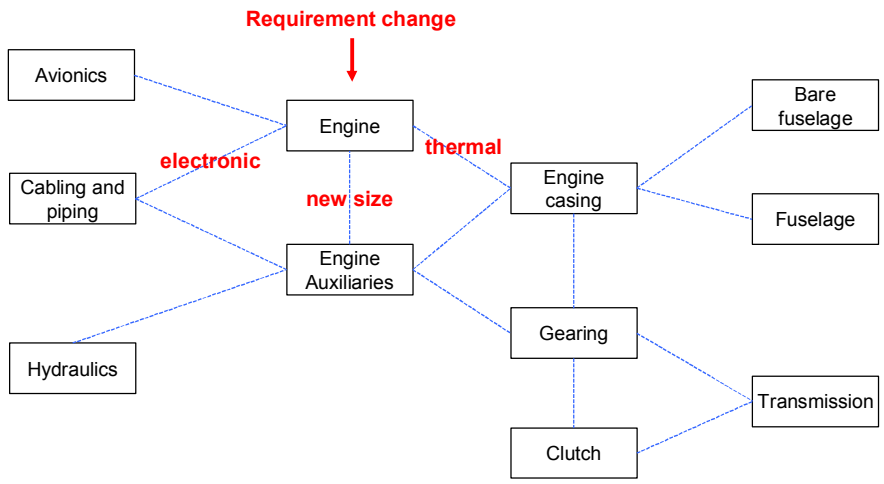


Figure 1-1. An illustrative example of change propagation

other. On the other hands, integral architecture has complex connection between function and components. In terms of engineering changes management, modular architecture is preferred over integral architecture because changes in modular architecture is less likely to propagate due to loose connection between elements. Although modular architecture is preferred over integral architecture in terms of the engineering change management, few products can achieve pure modular architecture. As the product become complex systems such as automobiles, aircraft, or complex software, number of components increase, it is impossible to implement fully modularized products by decoupling every interactions between components.In this sense, many authors have proposed a number of strategies to deal with the change propagation. Some strategies include following: (1) prevention of unnecessary change; (2) generate changes as early as possible (front loading); (3) prioritization of changes; and (4) improvement of product architecture, etc. Although these strategies are practically important, eliminating entire changes is

neither possible nor desirable for product development process.

There is a wide agreement that a scientific model for describing a behavior of change propagation is necessary. One of the main difficulty in modeling change propagation is that it shows an uncertain behavior. Eckert et al. (2004) identified different patterns of change propagation according to its uncertainty. These three patterns are illustrated in Figure 1-3. The most well-understood and predictable change propagation pattern is called as change ripples. Change ripple is controllable because it shows small-and-decreasing patterns of changes over time. Although change blossom shows significant number of changes in early stages, but it also reveals decreasing patterns. The most problematic scenario is change avalanche where change propagation effects are unpredictable and the number of changes continue to rise as time goes on. Unfortunately, interviewing many engineering companies, Eckert et al. (2004) found that many companies faces with change avalanche during

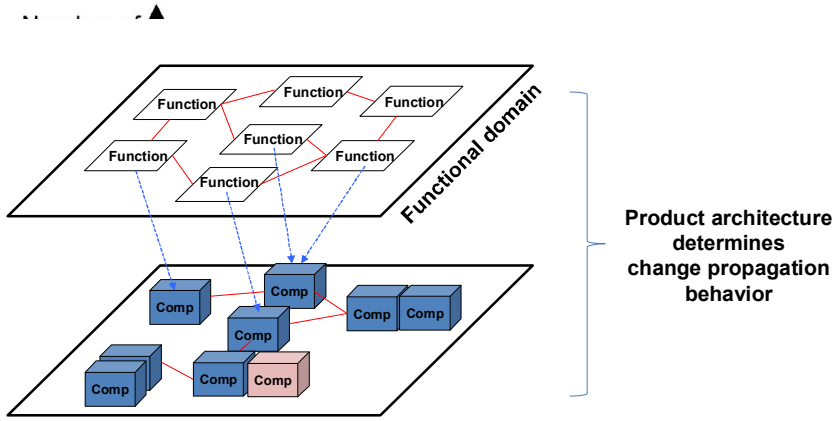


Figure 1-2. An illustrative example of the product architecture

the product development process.

1.3. Probabilistic Approach to Change Propagation Analysis

As can be seen in previous chapter, the change propagation can be problematic if they not properly managed or controlled. One way to improve engineering change management is thus to anticipate the consequence of changes in advance by which the design task can be guided (Clarkson et al., 2004). According to the survey of Jarratt et al. (2011), although most of commercial Product Lifecycle Management (PLM) systems provides the engineering change management tools, they only predict the impact of changes by considering only direct connection between components which are obtained by analyzing geometrical mismatches between changed components and its affected neighbor components. However, none of them address the consequence of change propagation when performing change impact analysis.

In the previous decade, some literature has proposed a model or tool for analyzing the impact of engineering changes considering not only in immediate changes but also in change propagation. Numerous terms such as change prediction (Clarkson et al., 2004), change propagation analysis (Giffin et al., 2009), or change impact analysis (Hamraz et al., 2013) has been interchangeably used to refer these sort of analyses. In this thesis, we would use ‘change propagation analysis’ to refer these analyses.

One approach to change propagation analysis is to use deterministic parameter linkage defined over a set of components. Under this ap-

proach, if a component is changed, then all components which are connected with this component are changed. That is, only the binary information is used to represent dependency between components. One of the first method based on deterministic assumption is Mokhtar et al. (1998) which proposes information model for linking design parameters and implement change propagation notification system using this parameter linkage information between components. Huang and Mak (1997) and Roubiah and Caskey (2003) proposed information exchange model which can calculating cascading effect of change propagation, but they didn't actually proposes an algorithm for change propagation analysis. Another famous methodology is Change Favorable Representation (C-FAR) which is proposed by Cohen et al. (2000). Overall methodology utilizes STEP (Standard for the Exchange of Product) data model which is the formal standard modeling language for representing a product. C-FAR represents design entities as vectors which consists of attributes of the design entities. After then, the interactions between attributes are recorded in matrix, named as C-FAR matrix. Only the binary value or qualitative measure (high, medium, low) can be used to specify the value of C-FAR matrix. By multiplying C-FAR matrix, a consequence of a change from a source entity to a target entity can be calculated.

Predicting change propagation based on deterministic assumption, however, is only valid when every possible interactions between components are identified and parameterized. When developing a new product, however, components interaction remains in ambiguous states. At the beginning of the product development phases, product architec-

ture is not stable, and new functions or technologies are continuously infused into a product. Therefore, one cannot sure whether a change can propagate to others or not. Moreover, as a product evolves into a complex system and as the number of components interactions explodes, binary interaction is not valid for representing an interaction between components. In actual product development situations, some components a likely to propagate changes than others. Moreover, even when dealing with a single change propagation event between two components, a change may propagate occasionally according to the reasons, motivation or types of change requests.

Considering the above nature of change propagation, a change propagation should be represented by a probability. An illustrative example of probabilistic viewpoint is illustrated in Figure 1-4. As can be seen, when component A is changed by a change request, this change does not necessary invokes changes to other components B or C. Instead, the likelihood of change propagation between components is represented by the conditional probability distribution defined over A and

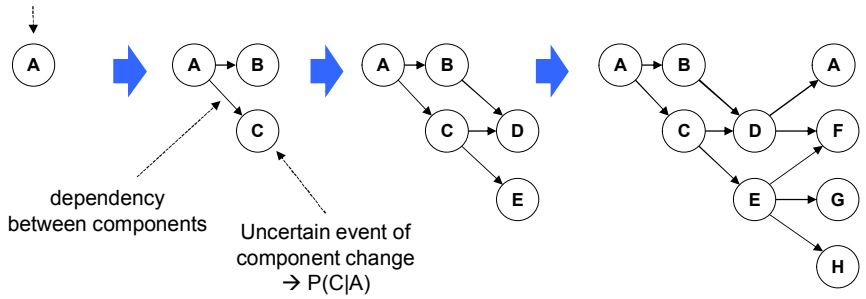


Figure 1-4. Stochastic viewpoint for modeling change propagation.

B or A and C. Some components, thus, is likely to propagate change than others. Therefore, the change propagation process can be regarded as a stochastic process of random variables.

One of the most advanced change propagation analysis methodology using probabilistic viewpoint is Change Prediction Method (CPM) proposed by Clarkson et al. (2004). Distinctive feature of the CPM is that it tries to model stochastic evolution of change propagation process using Design Structure Matrix (DSM) which encodes change propagation probability between every two components. Overall procedure of CPM is illustrated in Figure 1-5. At first, probabilistic dependency between components is estimated. A Design Structure Matrix (DSM) is utilized for quantifying likelihood and impact between adjacent components. It is noteworthy that every probability values are elicited from domain expert or component designers. After then, in order to address cascading effect of change propagation, an algorithm calculates the expected risk of each component by enumerating every possible change propagation path derived from DSM. Finally, combining both direct

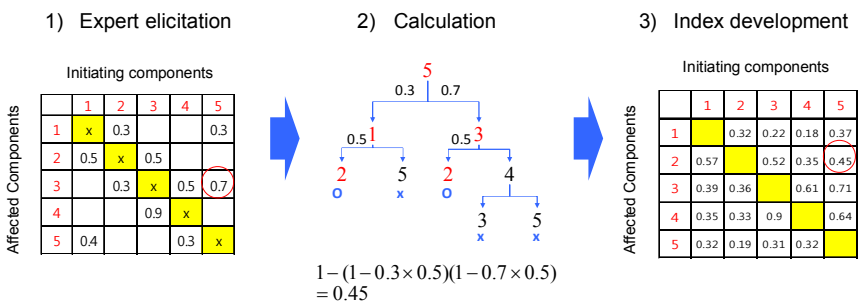


Figure 1-5. Overall procedure of CPM

and indirect effect of change propagation between components, a com-

bined risk measure is developed. This measure indicates that how likely a component can be affected by a change of one component.

1.4. Bayesian Network Approach to Improve Change Management.

Due to its ability to modeling stochastic behavior of change propagation, CPM has been extended in numerous literature. For example, extending DSM into MDM (Multi Domain Matrix), Koh et al. (2012) addresses change propagation among different domains such as organization, or manufacturing process. Keller et al. (2005) proposes a visualization tool for utilizing various information which can be obtained by CPM. Bayesian network to improve change management.

Although CPM is most recognized tool in academia, there are some limitation which makes it hard to be utilized in practice. One limitation of using DSM is due to its low assessment fidelity. CPM only can represent binary relationship between two components because component relationship is represented with DSM. In reality, however, a change propagation can occur among more than three components. That is, a component can be changed by the interaction more than two components, or vice versa. Moreover, a change can propagate to other component through different interfaces. For example, suppose that component A and B are connected by design parameters. Although there is a single change propagation path from component A to B, the degree of interaction between two components should be different according to design parameters linking these two components. Some parameter may be much stronger than others, or other may not be irrelevant in change

propagation. In order to handle complex relationship defined over multiple number of components, alternative representation scheme is required.

Second limitation is that DSM-based approach cannot consider architectural relationship between components. DSM basically assumes no hierarchical relationship between components. However, a complex system usually is a decomposed into a number of subsystems, each of which also can be decomposed into subsystems until it reaches. In developing such a complex system which has such hierarchical architecture, a change can propagate between different hierarchical levels. In order to represent interaction between different hierarchical level, another tool is required.

Third limitation of CPM is that it cannot analyze dynamic evolution of components status during product development process. Every index which the CPM generates is based on the assumption that every components can be changed during product development process. This assumption is only valid for early stage of the product development process, where every components status is fluid. However, as design progress, uncertainty involved in each component begin to reduce and after specific time point, its design is frozen and does not accept change from other components. Design freeze is defined as a time point when a design of specific component is completed and handed over to next (production phases). Actually, each component has different design freeze timining, and some components are frozen earlier than others. In this environment, a project manager should continuously monitor change propagation risk of each component given frozen components,

or should determine appropriate design freeze sequence which can mitigate the overall change propagation risk during product development process. In order to control and evaluate the control decision of engineering changes, therefore, another tool is required.

Final limitation of CPM is that it only utilizes expert's opinion as an input data for developing a model. That is, each probability within a DSM should be depicted by asking engineers. This expert elicitation is inefficient because it takes much time and cost for collecting entire information. Moreover, the resulting value may be inaccurate because probability value is based on subjective opinion of engineers. This make the CPM hard to be applied in large-scaled design projects which consists of large number of subsystems. From above discussion, if possible, using an objective component-interaction data is a better option than using expert elicitation. Fortunately, many engineering companies utilizes Product Data Management (PDM) system which supports documentation of engineering changes generated during product development process. However, CPM does not provide any structured methodology for automatic parameterization of change propagation probabilities from engineering change data. Considering above limitation, an alternative tool for modeling change propagation is required.

The objective of this study is to propose an alternative methodologies which can tackle the limitations of CPM. In this regard, Bayesian Network (BN), which is a kind of probabilistic graphical model is proposed. A BN is a directed acyclic graph (DAG) in which the nodes represent the system variables and the arcs symbolize the dependencies or the cause–effect relationships among the variables. Due to its ability to

compactly represent dependence and independence relationship among random variables, it has been used as a robust and efficient framework for modeling and reasoning uncertain knowledge which is often represented by large number of variables (Kjræulff and Madsen, 2012) including wide range of applications, such as fault detection (Bobbio et al., 2001), operational risk management (Cowell et al., 2007), or medical diagnosis (Heckerman et al., 1992). For more details about the real world applications of BN, refer to Heckerman et al. (1995a).

Figure 1-6 illustrates an example of a change propagation model represented by BN. As can be seen, a change of a component is represented by a random variable. And the arc represents change propagation relationship between components. Although overall mechanism for calculating change propagation may seem similar to CPM, BN contains many useful properties which can significantly improve the change propagation analysis.

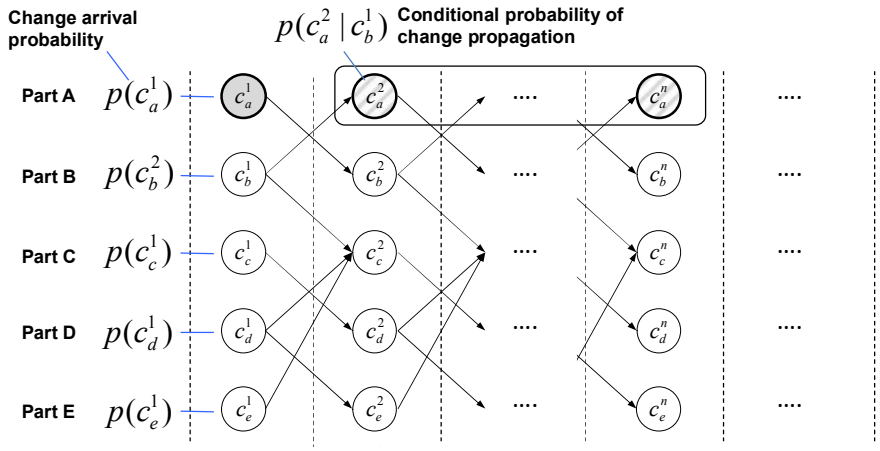


Figure 1-6. BN-based change propagation model

Some of advantages using BN are summarized as following. Firstly, BN-based model can enhance the assessment fidelity of change propagation relationship among components. For example, by adopting BN, one can represent interaction more than three components simultaneously since BN-formalism can encode complete probabilistic relationship among any number of components with conditional probability distribution. Seconds, BN can represent hierarchical relationship among components. Thirds, BN can perform more flexible analysis by using probabilistic inference algorithm defined over the BN. As a result, BN-based model can provide powerful scheme for calculating more complex change propagation patterns where CPM cannot adequately handle with. For example, BN-based model can calculate the change propagation impact generated by multiple change requests, or vice versa. Finally, data learning is possible. Since BN provides intuitive mechanism for learning change propagation model from real data. If such data learning is possible, it can reduce the effort and cost of expert elicitation as well

as retrieve more valuable information for managing engineering changes.

1.5. Structure of the Thesis

The remaining part of the thesis will present how the formalism of BN can be used to model change propagation process, and how it can be used to enhance current engineering change management practice. As illustrated in Figure 1-7, BN-based model can be utilized in entire product development lifecycle. In this regards, this thesis is organized as three parts, each of which corresponds to different product development stages (pre-during-post). Brief abstraction of each theme is presented as follows.

The first theme is development of change prediction model using Bayesian network framework**.** In pre-development stages, what is im-

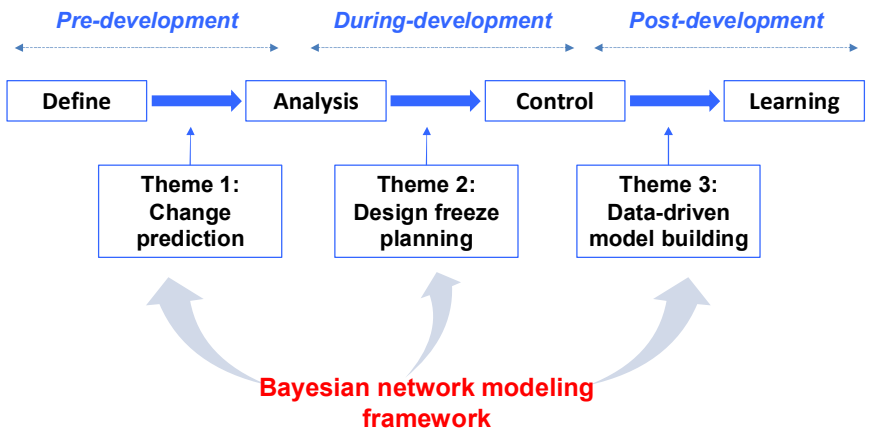


Figure 1-7. Outline of the thesis

portant is to assess change propagation in advance, and provides engineers with adequate design guidelines. In this regards, this chapter illustrates how the Bayesian network (BN), which is emerging tool for a wide range of risk management, can be used to predict change propagation. More specifically, BN-based model is compared with one of the well-recognized change prediction methods, namely CPM (change prediction model). Firstly, I show that CPM can be directly converted into an equivalent BN. In addition to this, I also show that BN has significant advantages over CPM at both modeling and analysis level. In the modeling level, various assumption over CPM can be relaxed and various kinds of modeling extension can be accommodated with BN. At the analysis level, BN's ability to performing probabilistic inference provides a user with another interesting measures, which cannot be obtained with CPM. Moreover, BN provides robust framework for learning change propagation probabilities from empirical data. My method can enhance the capability of engineering change management throughout entire product life-cycle.

The second theme is determination of design freeze sequence using Bayesian network framework. Controlling engineering change becomes important when actual product development process begins. One way to mitigate the risk of change propagation is to impose a design freeze on components at some point prior to completion of the process. This paper proposes a model-driven approach to optimal freeze sequence identification based on change propagation risk. A Dynamic Bayesian network (DBN) was used to represent the change propagation process within a system. According to the model, when a freeze decision is

made with respect to a component, a probabilistic inference algorithm within the BN updates the uncertain state of each component. Based on this mechanism, a set of algorithm was developed to derive optimal freeze sequence. Our methodology identifies the optimal sequence for resolution of entire-system uncertainty in the most effective manner. This mechanism, in progressively updating the state of each component, enables an analyzer to continuously evaluate the effectiveness of the freeze sequence.

Final theme proposes a structured method for managing change propagation by learning dependency network. The objective of this study was to propose a novel change prediction methods that can automatically mine a change propagation network from a collection of previous change records. As a modeling language, dependency network, a graphical model for representing probabilistic relationships among random variables, was utilized. As a result, a complete joint probability distribution that can predict the probability of change propagation was obtained. A case study on Azureus, an open-source software project, was conducted, which demonstrated that our approach effectively provides mathematically rigorous and efficient methods of change propagation learning and prediction.

Chapter 2. Literature Review

In this chapter, previous studies related to the thesis are reviewed in four perspectives. Literature introduced in first two chapters are related to Chapter 3. Section 2.1 presents a review of current change prediction methods. Section 2.2 introduces the definition of Bayesian network and its application in numerous fields. In Section 2.3, literature on design freeze planning that is handled in Chapter 4 is introduced. The last section 2.4 explains some data-drive models for learning change prediction model which are relevant to Chapter 5.

2.1. Change Prediction Methods

Prediction of engineering changes is a major research area in engineering change management. A number of tools have been developed in this domain to anticipate the impact of change propagation. One of the first models is Change Favorable Representation (C-FAR) proposed by Cohen et al. (2000). C-FAR quantitatively measures dependency between design parameters and proposes a mechanism for calculating the cascading effect of change propagation. Clarkson et al. (2004) proposed a more advanced tool named change prediction methods (CPM). The distinctive feature of CPM is that it predicts the impact of change propagation with the risk measure which is obtained by multiplying the likelihood and impact. CPM quantifies the likelihood and impact between adjacent components using a design structure matrix (DSM). Then, an algorithm, which enumerates every possible change propagation path derived with the DSM, calculates each component's expected risk. Af-

ter its successful initial implementation, CPM has been further extended by numerous studies. For example, Keller et al. (2005) proposed a data visualization tool for CPM. More recently, Koh et al. (2012), extending DSM to MDM (multi-domain matrix), addressed change propagation among different domains such as organizations or manufacturing processes.

In contrast, relatively little research has been done for the management of engineering changes during the product design process. Oh et al. (2007) uses change propagation information to design a system architecture which can effectively absorb change propagation. Wynn et al. (2010) and Maier et al. (2014) proposes methodologies for prioritizing design activities considering lead-time delays caused by change propagation between components. Yang and Duan (2012) develops a parameter linkage model which represents the propagation of change at the design parameter level, and proposes a methodology for searching an optimal change propagation path which can maximally mitigate the effect of propagation. Our design freeze sequencing method falls into this stream of research in that it provides a dynamic way to manage the propagation of change during the design phase. For more comprehensive literature review about change propagation analysis, please refer to Hamraz et al. (2013) and Jarratt et al. (2011).

As stated above the most well recognized change prediction method is CPM proposed by Clarkson et al. (2004). CPM begins with identifying dependence relationship among components using design structure matrix (DSM). An example of DSM is illustrated in Figure 2. Within the DSM, each column head represents instigating component which

propagates design changes to other component. On the other hands, each row head represents component whose design is changed by the instigating component. The 'X' mark in the (i, j) cell of the DSM, which is a binary indicator of the change relationship between two components means that the design change of component i is likely to induce the change of component j.

For each cell marked by x, the scale of change propagation is quantified by risk, which is defined as the product of the likelihood of the change occurring and the impact of subsequent changes. Likelihood $l_{i,\mathrm{j}}$ is defined as the probability that component i's design change might result in the change of component j. Because the likelihood value is defined in a probabilistic term, it is defined between 0 and 1. Likewise,

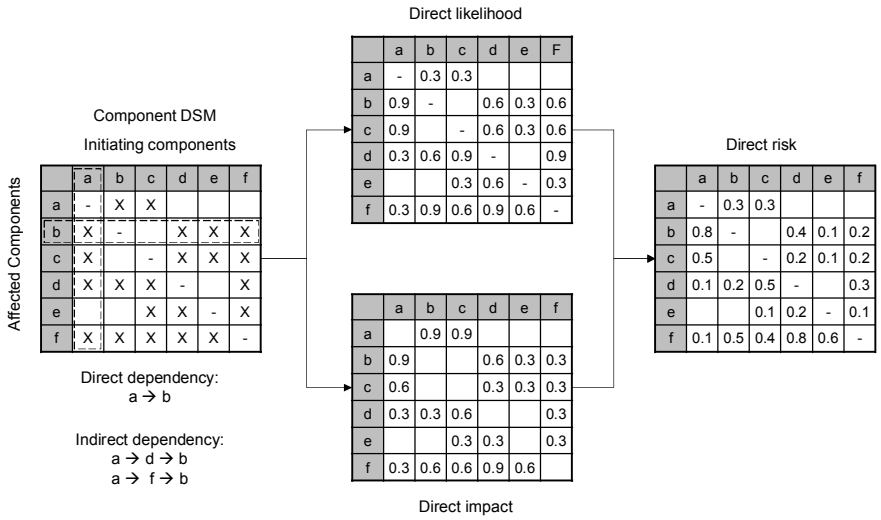


Figure 2-1. Likelihood, impact and risk DSM (adopted from Clarkson et al., 2004)

impact $i_{i,j}$ is defined as the average proportion of redesign work of component j if the change is propagated from component i. Now the risk matrix can be represented as a combination of both likelihood and impact as shown in Figure 2-1.

The change in one component can propagate throughout several intermediate components. As illustrated in Figure 2-1, component b is affected directly by component a, but also affected indirectly by component d and f. To enumerate every possible change propagation paths, a change propagation tree is constructed by referring the dependency structure from DSM. An example of change propagation tree is illustrated in in Figure 2-2. It shows the partial propagation paths from component a to component b. The change propagation tree assumes that each component is allowed to change only once during the change propagation process.

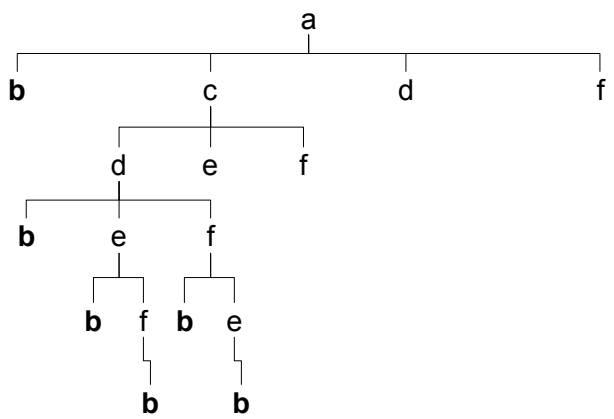


Figure 2-2. A partial change propagation tree from component *a* to *b* (adopted from Clarkson et al., 2004)

Now the combined effect of change propagation is obtained from the change propagation tree. Vertical lines in a tree reveals the events occurred in series, thus, their probabilities are combined with *AND* ($\cap$) gates. On the other hands, likelihood values on the horizontal line should be combined with *OR* ($\cup$) gates because they are mutually exclusive. Computation of each type of relationship is computed as follows, respectively:

$$l_{b,u} \cap l_{b,v} = l_{b,u} \times l_{b,v} \tag{2.1}$$

$$l_{b,\mathrm{u}} \cup l_{b,v} = 1-(1-l_{b,u}) \times (1-l_{b,v}) \tag{2.2}$$

where $l_{b,u}$ indicates the likelihood of change propagation from component b to component u. Following the above logic, we can obtain the combined likelihood by aggregating the probabilities from bottom to top.

Combined risk $R_{b,\mathrm{a}}$ of change propagation from component *a* to *b* is calculated as follows:

$$R_{b,a} = 1-\prod_{u \in P}(1-\rho_{b,u}) \tag{2.3}$$

where P is the set of penultimate components affecting ultimate component *u*, and $\rho_{b,\mathrm{u}}$ is the combined risk of change propagation from *b* to *u*. $\rho_{b,\mathrm{u}}$ is computed as follows:

$$\rho_{b,u} = \sigma_{u,a} l_{b,u} i_{b,u} \tag{2.4}$$

where $\sigma_{u,a}$ is combined likelihood of change propagation from component *a* to *u*; .. is the direct likelihood of change from component *u* to *b*, and $i_{b,u}$ is direct impact of component change *b* when penultimate component *b* is changed. Finally, the combined impact between two

components is obtained by dividing combined risk $R_{b,\mathrm{a}}$ by combined likelihood $L_{b,a}$.

2.2. Bayesian Network

A Bayesian Network (BN) is a graphical model for representing and reasoning about a set of random variables. A BN consists of both qualitative and quantitative part. The qualitative part is represented by the graph. The nodes in the graph represents a set of random variables, $X = \{X_1, X_2, ..., X_n\}$, from a domain. A set of directed edges (or arcs) connecting pairs of node $X_i \rightarrow X_j$, represents direct dependency between random variables, indicating X_i is the direct cause of X_j. The only constraint on the arcs is that they should not make any directed cycle in the graph. Therefore, whenever the directed acyclic graph (DAG) assumption is maintained, any kind of cause-and-effect relationship among random variable can be encoded with on the BN.

For a quantitative part, the strength of relationship between variables is quantified by Conditional Probability Tables (CPT). Assuming discrete state of each variable, the CPT contains a set of conditional probability distributions given every possible instantiation of its parents. An example CPT of a simple BN is illustrated in Figure 2-3.

Assuming every variable has binary state, CPT of C consists of four distinctive conditional probability distributions given the combination of A and B. On the other hands, for .. and B, which has no parents, the CPT's are specified with prior distributions with no conditional terms. These variables having no parents are called root variables.

BN utilizes conditional independent statements encoded on the variables so as to reduce the burden of calculating joint probability distribution. Conditional independence statements can be read off by d-separation (and the equivalent directed, global Markov property) which is originated from the topology of the graph. The d-separation property encoded in the graph enables the full joint probability distribution of entire random variables $X = \{X_1, X_2, ..., X_n\}$ to be factorized as in following equation (1), which is often called the chain rule:

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i \mid X_{pa(i)}) \qquad (2.5)$$

where $X_{pa(i)}$ denotes the random variables for all the parents of node i. For example, the full joint probability of four random variables in Figure 1 can be obtained by the product of only four different probabilities as in equation (2)

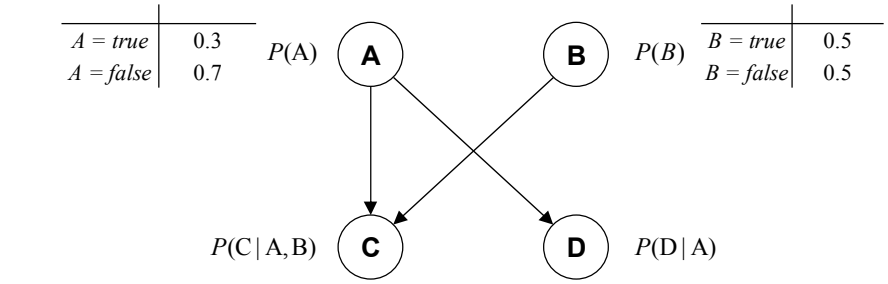
$$P(X_A, X_B, X_C, X_D) = P(A)P(B)\,\mathrm{P(C \mid A, B)}\,\mathrm{P(D \mid A)} \qquad (2.6)$$

In this way, BN can efficiently reduce the computational burden of calculating full joint probability distribution, which grows exponentially with the number of random variables. For the details about the concept of d-separation and conditional independence, see Pearl (1988).

After modeling a BN, it can be used to answer various types of probabilistic queries about each node. BN utilizes inference algorithm for exploiting conditional independence assumption encoded in the structure of the network in order to make this calculation efficiently. However, when the number of nodes increases, the exact inference on a BN becomes an NP-hard problem. Fortunately, efficient inference algorithms have been developed so that the inference task can be done with-

in a second even for a complex network consisting of more than hundreds of random variables. However, the detailed description of the inference algorithm is beyond the scope of this paper because there are many well-established commercially available tool which can support automatic calculation of posterior probability distribution (see Pearl (1988) for more detail.).

Over the last decades, a number of papers in engineering design domain has adopted BN in solving design problems. Since BN is a useful tool for modeling uncertainty, most of them use the formalism of BN in design decision making under uncertainty. For example, Matthieu et al. (2012) uses BN in formulating optimal disassembly strategy considering both product architecture and quality uncertainty. Moullec et al. (2012) and Shahan and Seepersad (2012) use the formalism of BN in representing probabilistic relationship between design parameters and predict the probability distribution of product performance. Mat-



A = true	0.3
A = false	0.7

B = true	0.5
B = false	0.5

	A = true		*A = false*	
	B = true	*B = false*	*B = true*	*B = false*
C = true	0.9	0.1	0.95	0.12
C = false	0.1	0.9	0.05	0.88

	A =true	*A = false*
D = true	0.8	0.3
D = false	0.2	0.7

Figure 2-3. An example of a BN with four random variables

thews (2011) develops a BN–based concept design support system which provides dynamic guides for selecting design elements within the morphological chart. However, the application of BN in engineering change management has not been reported in literature. Although Morkos et al. (2014) proposes a neural network in predicting engineering changes, their tool does not explicitly address the propagative property of engineering changes.

2.3. Design Freeze

One commonly perceived viewpoint about design process is an uncertainty reduction process, where the design description begins as a vague concept and gradually reduces the solution space until a precise final solution is reached (Herrmann, 2010). From this view point, a design for a system or a component cannot be made in a single step. Rather, it can be considered as a progressive process in which parameters are incrementally defined and frozen (Maier et al., 2014). Design freeze is defined as binding decision that defines the whole product, its parts or parameters and allows the continuation of the design based on that decision (Eger et al., 2005). In engineering management literature, design freeze is considered as a strategy for accelerating the product development time (Zirger and Hartly, 1996). A number of mathematical models has been proposed to determine timing for design freeze. Krishnan et al. (1997) evaluates a trade-off between early and late freeze timing between upstream and downstream design tasks. Bhattacharya et al. (1998) proposes a mathematical model for determining design freeze timing in the presence of competitors and market uncertainty.

Huchzermier and Loch (2001) proposes a real-option model which can evaluate the flexibility of design freeze timing under various types of product risk.

Although these models shed a light on the best timing of design freeze, most of them relied on a simple assumption that there is a single design freeze point during the design process. Based on the case studies of many engineering companies, however, Eger et al. (2005) found that individual parts are actually frozen at different times. At the part level, engineers sequentially freeze components in order to reduce the likelihood of further engineering changes. By freezing some components earlier than others, the designers can reduce the likelihood of change propagation and facilitate a design continuation of dependent components. This, in turn, calls for a structured method for deriving careful freeze sequence of components.

Motivated by Eger et al. (2005)'s work, Keller et al. (2008) proposes a structured method for determining design freeze sequence of components. They used Clarkson et al (2004)'s DSM-based method to calculate the combined risk of change propagation of each component. Then, a heuristic optimization method, simulation annealing, is used to reshuffle the rows and columns of matrix in order to freeze more influential components as early as possible. Although CPM is a good tool for assessing the risk of change propagation, it cannot directly take into account the freeze states of each part, which makes it impossible to address the dynamic evolution of component states during the freeze process. The present study adopts Bayesian network to overcome this limitation by modeling probabilistic relationship among components.

In the present study, we address a special type of decision problem, namely the sequential decision problem (Mookerjee and Mannino, 1997). In this problem, the decision maker has to choose among a set of alternative actions. To maximize his utility, the decision maker can sequentially gather new information through a series of tests. The objective of this problem, thus, is to choose the right test to perform next. BN is especially useful in evaluating the value of information of a sequence of testing because it can compute the probability distributions for a set of variables based on the observation of those variables (Mussi, 2002). A number of papers on the application of sequential decision problem using BN have been reported. Heckerman et al. (1995b) proposes an algorithm for deriving an optimal troubleshooting sequence from a BN model which models the relationship between the failure modes and their associated components. Similarly, Huang et al. (2008) proposes a method for deriving the sequence of diagnosis for automobile sound systems, Skaanning et al., (2000) a system for trouble shooting printers, Mirarab and Tahvildari (2007) an optimal test sequence for software systems, etc. Vomlel (2004) applies a similar approach in educational testing.

In the present study, the framework of sequential decision problem is adopted to obtain design freeze sequence. Probabilistic relationships between components are represented with BN. When a component is frozen, the information about this component is updated throughout the network since the associated uncertainty is resolved. Each time a component is frozen, thus, we can dynamically update the uncertainty levels of each component. In this fashion, we can quantitatively evaluate

the freeze sequences, which was not properly handled in previous literature.

2.4. Data-driven Change Prediction Methods

During the past decade, several change-propagation prediction methods have been proposed. They can be classified according to the approach by which the dependency structure is developed. Expert-based methods utilize the subjective opinions of experts to measure inter-component dependency. One of the first methods of this type to be introduced is C-FAR (change favorable representation), proposed by Cohen et al. (2000). C-FAR uses a previously established product data model, STEP (standard for the exchange of product), which decomposes a product into a set of elements and their corresponding attributes as input data. Expert opinion on attribute interaction is then measured on a qualitative scale (high, medium, and low), after which the result is translated to matrix form. This matrix is then used to calculate the consequence of a change of a source entity to a target entity. Clarkson et al. (2004) proposed an alternative, quantitative index for prediction of change propagation. Utilizing DSM (design structure matrix), a square matrix representing inter-component interaction, they quantify both the likelihoods and impacts of changes between adjacent components with numerical values between 0 and 1. They then combine these values to generate the change propagation risk of each component.

As already noted, expert-obtained numerical values can be biased as well as inaccurate. Parameter-based methods solve this problem by using a predefined mathematical function between components. For

example, Yang and Duan (2012) proposed the use of the mathematical constraint relationship between design parameters as a measure of inter-component dependency, which relationship is then used to find the optimal design path maximally mitigating the risk of change propagation. Hamraz et al. (2013) utilized a change propagation probability distribution obtained with reference to the tolerance ranges identified between subsystems. This probability distribution is compatible with DSM, which, in turn, makes it compatible with Clarkson et al. (2004)'s methods. Although parameter-based methods provide more accurate input data, such data are useful only when there is a well-defined mathematical function between subsystems. However, identifying this function is time-consuming and difficult to automate.

Data-based methods use previous design records to predict change propagation. Early papers in this line focused on the development of data models that can systematically capture those records. Casotto et al. (1991) proposed a data model that can capture a designer's activities in a CAD system and visualize them in graph model form. Similarly, Katz (1990) proposed the use of the version history of system components to extract the change propagation relationship. Unfortunately, neither work provides a systematic method for measuring the impact of change propagation. Recently, a few data-based methods incorporating the data-mining approach to predict change propagation have been introduced. Giffin et al. (2009) analyzed evolutionary patterns of engineering changes by mining the motif structure from the change propagation network. However, their approach focuses only on post-analysis, providing no systematic measures for change-propagation prediction.

Kocar and Akgunduz (2010) mined engineering-change association patterns from the database of previous change logs. But their method, in focusing on the extraction only of individual patterns, cannot address the issue of the combined effect of change propagation.

Our approach, utilizing dependency network, tends to improve data-based methods. Using it, probabilistic relationships between components can easily be visualized in graph form. This enables a user to readily identify, in a more holistic manner, the change propagation relationship between components. Moreover, it can calculate the combined effect of change propagation with complete probability distribution. In the following section, the concept of a probabilistic graphical model is introduced in more detail.

Chapter 3. Change Prediction using Bayesian Network

This chapter introduces a framework for predicting change propagation using Bayesian Network (BN) framework. Background of the research is briefly introduced in section 3.1. Detail procedure for developing BN-based change prediction model is illustrated in section 3.2, and the advantages of using BN formalism are analyzed through section 3.3 to 3.5. Finally, section 3.6 summarizes the framework and results.

3.1. Introduction

The difficulty of managing changes lies in the fact that a change does not occur alone. In complex system where each part is associated with different parts, a change made to a part may influence other parts. As a result, a change may propagate throughout the entire system. The change propagation process shows un-certain behavior (Eckert et al., 2004); it might be terminated within a few steps, but sometimes it generates an avalanche of changes as the redesign progresses. To cope with such uncertainty, a number of strategies are pro-posed in order to avoid changes as much as possible (Prasad, 1996). These strategies includes for example to make changes as early as possible, or to create design buffers. However, eliminating engineering changes is both undesirable and unrealistic in many cases.

Bayesian network (BN) is a complete representation about probabilistic relationship among (discrete or continuous) random variables (Jensen, 1996). Due to its ability to compactly represent dependence

and inde-pendence relationship among random variables, it has been used as robust and efficient framework for modeling and reasoning uncertain knowledge which is often represented by large number of variables (Kjræulff and Madsen, 2012). After its concept was first introduced in the field of artificial intelligence, it has been utilized in various real world application including medical diagnosis, document classification, bio-informatics, and fault detection of complex systems (Heckerman et al., 1995a).

The objective of this chapter is to explore the capabilities of the BN formalism in modeling and analysis of change propagation process. Firstly, we show that change propagation process described in CPM can be converted into an equivalent BN. This BN consists of a set of random variables, each of which indicates the uncertain state of the component. Because BN is a generic representation among random variables, BN has significant modeling advantages over the traditional CPM. For example, various types of modeling extension such as encoding probabilistic relationship among more than three components, representing hierarchical structure of a product, or accommodating of various types of linkage among components are possible via BN-based model. This enables a modeler to provide more flexible but mathematically sound framework for modeling change propagation process.

BN is able to perform various types of numerical analyses providing some useful measures which are not available in CPM. BN can be used to answer various types of probabilistic queries about them, of which task is often referred to probabilistic reasoning. An appealing point of BN is that it can perform probabilistic reasoning in any direc-

tion (Pearl, 1988). This enables an analyzer to freely simulate the change propagation process given various types of change scenarios. For example, BN can compute the combined effect of change propagation probabilities given the changes of multiple components. Moreover, diagnostic assessment of the change propagation is also possible. Thus, one can measure the severity of a change-initiating components given an observation about the component.

Moreover, BN provides intuitive ways for updating change propagation probabilities from lots of data records about engineering change orders. This enables an analyzer to adapt their change propagation process model continuously in the light of new data (Heckerman et al., 1995b). This ability of the BN, thus, can reduce the burden of building models which rely heavily on the knowledge elicited from a group of experts or designers.

3.2. Modeling Change Propagation with BN

Before addressing the conversion from CPM to BN, we need to introduce a special type of BN which is often called dynamic Bayesian network (DBN) (Murphy, 2002). DBN describes a system that is dynamically changing over time through modeling stochastic variables over time. Typical structure of a DBN is illustrated in Figure 3-1. It consists of a sequence of sub-models, each of which represents the state of the system at a certain point in time, or namely the time-slice. Temporal edges connecting nodes between consecutive sub-models reveals the temporal dependency between states. Although the DBN models the dynamic systems, its structure is time-invariant; the structure of the

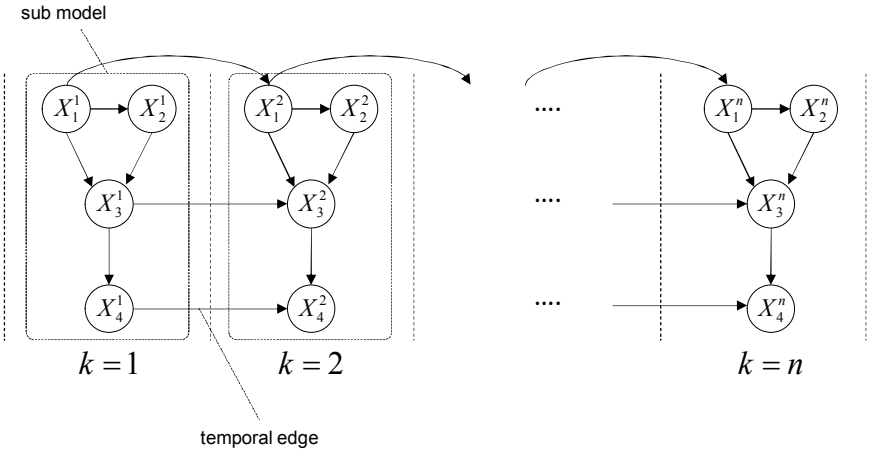


Figure 3-1. The structure of dynamic Bayesian network with n time-slice

network does not change over time with the exception of the root nodes (i.e. nodes at initial time-slice). Therefore, whenever the prior distribution of root node is specified, the DBN can recursively update the network, enabling a user to predict the further behavior of the system for the desired number of times.

Change propagation process can be naturally translated into an equivalent DBN. In Figure 3-2, a simple DSM is converted into its equivalent BN. In this network, each time slice consists of a set of nodes c_i^k which represent the states of component i at change propagation step k. This node is a discrete and binary random variable which takes values 'yes' if the corresponding component changes and 'no' otherwise. Now, the temporal edges connecting two nodes between consecutive change propagation steps $c_i^{k-1} \rightarrow c_j^k$, reveal the direct dependency from component i to j. Temporal edges can be easily identified by referring the (i, j) cell in the DSM. Now, a user can unroll the dynamic model for the desired number of time steps, duplicating a set of nodes and temporal edges.

Now, quantitative strength between components can be specified by the CPT. Because the state of component at step k only depends on the components at previous stages, parameters of each CPT are deter-

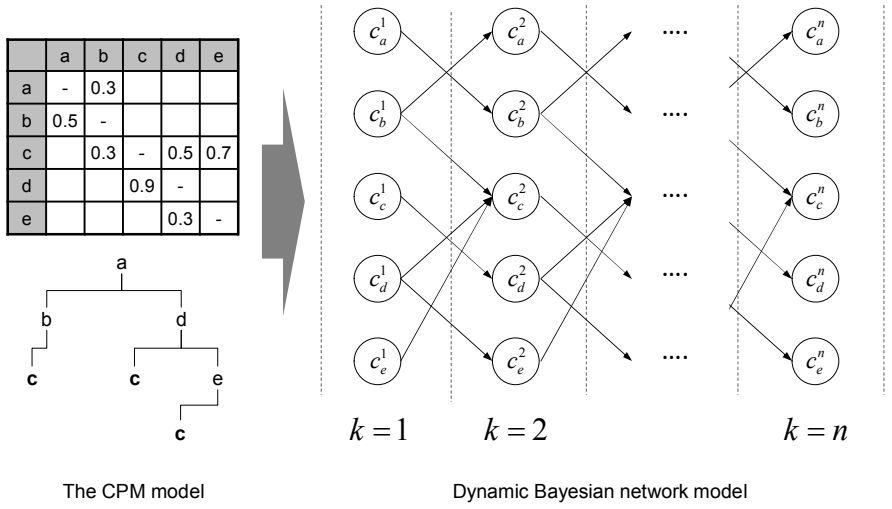


Figure 3-2. Conversion from the CPM model into BN

mined associated with every combination of the state of its affecting components at change propagation stage $k-1$. Suppose, for example, that component i is only affected by component j. If we let $l_{j,i}$ denote the likelihood that the change is propagated from component i to j, which is defined on the DSM, the CPT can be defined as follows:

$$\begin{aligned} &p(c_j^k = yes \mid c_i^{k-1} = yes) = l_{j,i} \\ &p(c_j^k = no \mid c_i^{k-1} = yes) = 1 - l_{j,i} \\ &p(c_j^k = yes \mid c_i^{k-1} = no) = 0 \\ &p(c_j^k = no \mid c_i^{k-1} = no) = 1 \end{aligned} \tag{3.1}$$

Although the component has multiple predecessors, its conditional probability distribution can be generalized with noisy-OR gate. Noisy-OR gates are applied when there are several causes $X_1, X_2, \ldots, X_n$ and a common effect variable Y, where (1) each of the cause X_i has probability p_i of being sufficient to produce effect, and (2) the ability of each cause being sufficient to produce effect is independent of the presence of other causes, i.e. causally independent with each other. In this case, the probability of y given an instantiation of its parent X_p is given by the following equation:

$$p(y \mid X_p) = 1 - \prod_{i:X_i \in X_p} (1 - p_i) \qquad (3.2)$$

For example, the CPT of component b in Figure 5 is constructed by following Table 3-1. As in above equation, the complexity of conditional probability distribution is reduced from exponential to linear in the number of parents. It is noteworthy, however, that the Noisy-OR gate is a special case of conditional probability distribution. Relaxing

Table 3-1. Conditional probability table of component *c* (based on Figure 3-2)

comp. *b*	comp. *d*	comp. *e*	comp. c	
			Yes	No
yes	yes	yes	1-(1-0.3)(1-0.5)(1-0.7)=0.895	(1-0.3)(1-0.5)(1-0.7)=0.105
yes	yes	no	1-(1-0.3)(1-0.5)=0.65	(1-0.3)(1-0.5)=0.35
yes	no	yes	1-(1-0.3)(1-0.7)=0.79	(1-0.3)(1-0.7)=0.21
yes	no	no	1-(1-0.3)=0.3	(1-0.3)=0.7
no	yes	yes	1-(1-0.5)(1-0.7) = 0.85	(1-0.5)(1-0.7) = 0.15
no	yes	no	1-(1-0.5) = 0.5	(1-0.5) = 0.5
no	no	yes	1-(1-0.7) = 0.7	(1-0.7) = 0.3
no	no	no	0	1

assumption of the Noisy-OR gate, flexible generalization of probability relationship is possible. We will address this issue in later sections.

3.3. Modelling Improvement

So far, we have shown that change propagation process can be naturally represented with BN. However, utilizing the modeling aspects underlying BN, more flexible generalization of the original model is possible. In later sub-sections, we will illustrate a number of modeling extensions and show how they are represented with BN.

3.3.1. Relaxing Distributional Assumption

In the CPM, only the interaction between two components are used as input for deriving combined likelihood. In the previous chapter, we have shown that an equivalent BN has the CPT constructed with the Noisy-OR gate. However, BN has no restriction on how parents interact with a child. Thus, any numbers can be used for constructing CPT whenever it is compatible with the graph structure. This enables a modeler to specify the interaction among more than three components at the same time.

For example, consider again CPT of component b in Table 1. The probability of component c's having changes given both d and e have changed is computed as $1-(1-0.3)(1-0.5)=0.65$. This calculation is based on the assumption that both d and e interact independently with c. However, BN can relax such independence assumption by manually specifying any numbers in the CPT. Therefore, the probabilistic impacts to a common child b can be amplified (by hav-

ing more than 0.65) or reduced (less than 0.65) in the presence of the change of both parents. In this way, user can represent more nuanced interaction among components, even when more than three components interact simultaneously.

However, it is noteworthy that relaxing distributional assumption lso indicate that expert elicitation burden increases exponentially as the number of parents increases. Therefore, the probability elicitation given every instantiation of its parents would become tedious when a component is associated with a large numbers of parents (Anthony et al, 2006). This would lead to the reduction of quality of the elicited numbers due to the limited time available for interaction with expert. A widely used technique for avoiding such full parameterization is to assume causal independence assumption as with noisy-OR gate. The elicitation exercise then reduces to eliciting the expert's belief about change probabilities between two components, which is the same task of constructing a DSM.

Noisy-OR gate used in previous chapters, is one of the mostly used models for constructing CPT. However, even when the causal independence is assumed, different types of probabilistic model can be assumed. For example, noisy-AND gate, which is considered as counterpart of the noisy-OR gate can be used. As the 'AND' indicates, the component has positive values if and only if every parents changes. Moreover, literature on BN provides variety of model which can exploit the causal independence assumption such as noise-threshold, noise-adder, noisy-MAX, noisy-MIN models etc. For details about the

generalization of causal independence models, please refer to Heckerman and Breese (1996).

3.3.2. Modelling Hierarchical Structure

As the product become more complex with increasing number of parts, hierarchical architecture is usually adopted. By hierarchical architecture, we mean that the product is consist of several layers of subsystems, each of which also being hierarchical in structure, until it reaches the elementary component at the lowest level.

When the hierarchical structure of a product is assumed, a change presented in a part or system may propagate across different hierarchical levels (Eckert et al., 2004). An example is illustrated in the left pane in Figure 3-3. At the system level a change of engine might require generic changes in bare fuselage. However, the changes in the engine system can also affect the casing, the subpart of bare fuselage. Moreover, within a system, changes in the part level can also lead to

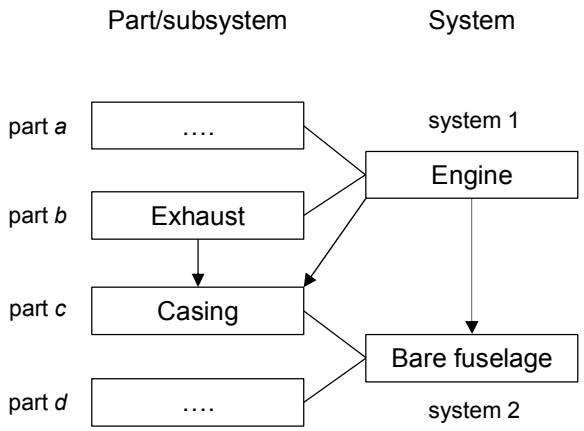


Figure 3-3. Change propagation between parts and subsystem (adopted from Eckert et al., 2004)

changes in the upper levels. In this situation, it might be helpful to analyze the change propagation process with varying degrees of hierarchical levels.

By introducing additional random variables, BN can easily accommodate hierarchical structure in the model. For example, the above example can be converted into an equivalent BN illustrated in Figure 3-4. This BN consists of two additional nodes s_1 and s_2 which represent the respective subsystem components. Connecting edges within each time-slice (solid line) represent interactions within a system. By specifying CPT of each system, we can represent the probabilistic relationship among the system and its subparts. Temporal edges connecting adjacent time-slice (dotted line) represents interaction between different systems. As illustrated in the figure, changes can propagate from system to component, component to system, or system to system.

The above framework can be applied in various context. For exam-

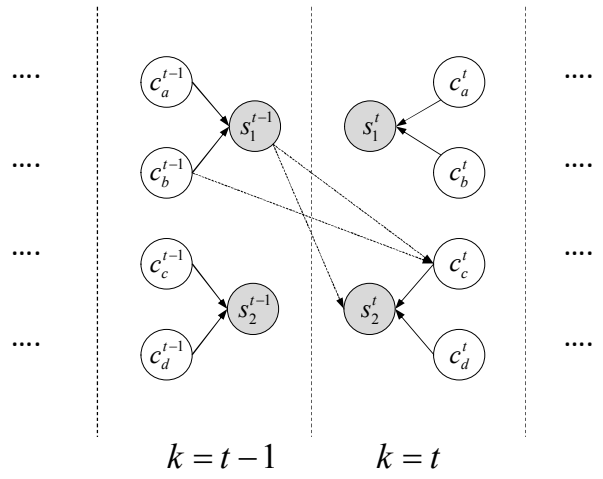


Figure 3-4. Representing hierarchical product architecture using BN

ple, when developing complex product such as airplanes, the company should communicate with several suppliers who design specific parts of the product. If we organize components with respect to their suppliers, then the above framework can be used to measure the communication burden between different suppliers. In addition to the physical chunks of the product, the concept of subsystems can be used to measure the interactions in various domains such as organizations, functions, processes or customer needs.

3.3.3. Modelling Multi-state Variables

In previous model, each component is allowed to have only the binary state, i.e. changes (yes) or does not change (no). However, with the introduction of multi-state variables in the model, one can represent more complicated change patterns. BN is easy to accommodate multi-state variables in the network.

One of the modeling extension using multi-state variables is to differentiate the state of the component according to the magnitude of its changes. For example, a three-state random variable consisting of high (changing considerable part), low (changing little part), and no change can reveal such difference. In this way, one can reveal the differences of degree of interaction among components in a more nuanced manner.

Another scenarios is to represent different types of interfaces through which the change can be propagated. Contrary to the CPM which assumes the unified interfaces for change propagation, there are various types of 'linking parameters' between parts and subsystems, such as geometry, force, heat, electronic or material, etc (Eckert et al.,

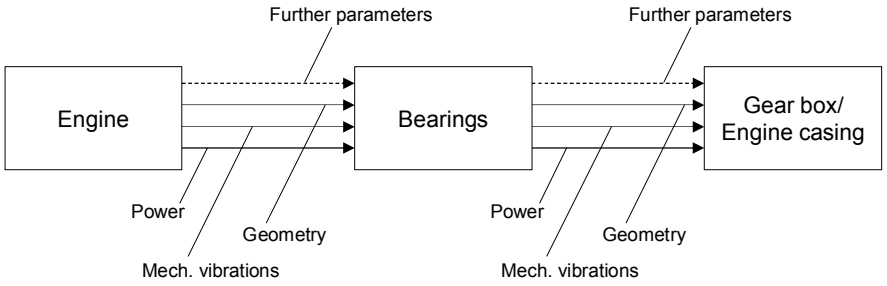


Figure 3-5. Multi-level parameter linkage (adopted from Eckert et al., 2004)

2004). An example of parameters that link elements is illustrated in Figure 8. This shows that each element is connected with four parameters – power, mechanical vibration, geometry, and further parameters. DSM is hard to encode multi-kind interaction because it only assumes the unified interface between two components. However, BN can easily accommodate the dependency between different parameters with the form of CPT. For example, the interaction between engines and bearings can be specified via CPT as illustrated in Table 3-2. If engines geometry parameter might affects the bearings mechanical vibrations, this relationship can be quantified via the conditional probability $p(bearing=mech.vibration|engine=geometry)$. In this way, one can reveal the interfaces among components in a more detailed manner.

Multi-state variables can also be used to represent more complicated behavior of components during the change process. For example, before executing design changes on each component, engineers usually evaluate the change request and determine whether or not to implement the request. Therefore, sometimes an engineer might reject the change request and pass it to another parts or subsystems. Giffin et al. (2009) shows that considerable amount of engineering change orders are rejected or reflected to other components. Therefore, another state 'change reflection' can lie between 'change' and 'no change'. In addition to change reflection, the component also can absorb changes. This implies that a component accepts the changes but decide not to propagate those changes to other components.

3.4. Analysis Improvement

Most common application using a BN is to answer probabilistic queries about it. A user can specify the values of any combination of nodes in the network they have observed. This evidence e propagates across the network, updating a new posterior probability distribution $p(X \mid e)$

Table 3-2. Example of CPT when multi-level linkage is assumed

Engine	Bearing			
	Power	Mech. vibrations	Geometry	Further parameters
Power	0.7	0.2	0.1	0
Mech. vibrations	0.1	0.8	0.05	0.05
Geometry	0.3	0.3	0.3	0.1
Further parameters	0	0	0.5	0.5

for each variable in the network. This task of computing posterior probability of the network given an evidence is often called probabilistic inference.

In our setting, probabilistic inference can be used to simulate the state of each component given the change of other component. Most common types of inference we wish to perform might be to find out updated posterior probability of components given the change of the component at initial stages. In the BN, this process of querying about the state of the system at future time given current evidence is referred to predictive inference. By predicting the future state of component given the change of current component, we can expect similar result with CPM.

However, BN has advantages over CPM because it can perform different types of inference supporting any direction of reasoning. For example, suppose a simple BN illustrated in Figure 3-6. An edge $A \to C$ implies that if we find a true state of A, then we can predict the state of C. However, BN also support reasoning with opposite direction – finding out the true value of C also makes A more credible. This types of reasoning from symptoms to causes is often called diagnostic inference. A further reasoning may involve reasoning about the mutual causes of a common effect, or vice versa. For example, suppose that we learn that common effect C is true. In this situation, learning that A is the true cause of the effect C lowers the possibility that B is the true cause of C. This kind of inference between mutual causes (or effects) is often called an intercausal reasoning. Moreover, in some situation, the reasoning does not fit neatly into one of the types described

above. This type of inference, performed by a combination of several types of reasoning, is often called mixed inference.

Utilizing such different variety of inferences, BN can provide a user with another interesting measures, which cannot be obtained with typical CPM. In the following sub chapters we will illustrate each types of inference, and how they can be utilized in change propagation analysis.

3.4.1. Predictive Inference

Predictive inference is the task of reasoning from new causes to effects. In our settings, it is the task of calculating the future state of components given the initial change of a component. This is similar to the typical analysis performed in CPM. An example of predictive inference is given in Figure 3-7. Suppose that we want to estimate the change prop-

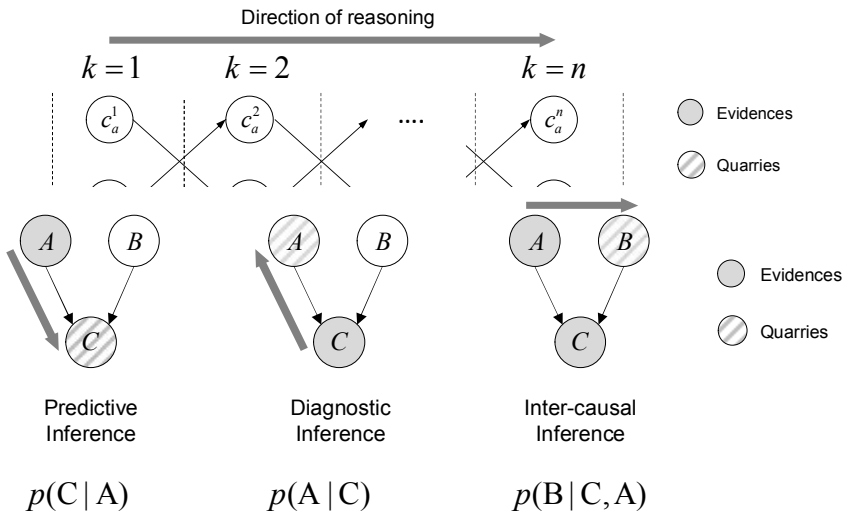


Figure 3-6. Common types of probabilistic inference

Figure 3-7. An example of query and evidence for predictive inference

agation probability from component c to e. We can encode this scenario into BN by setting the state of c_c^1 with 'yes'. This evidence, then propagates across the network, updating a posterior probability distribution of component e throughout the intermediate stages, i.e. stages from 2 to *n*. The probability of component e's change can be obtained by the following equation:

$$p(c_e^2 = yes \cup c_e^3 = yes \cup ... \cup c_e^n = yes \mid c_c^1 = yes) \qquad (3.3)$$

As illustrated in Figure 3-6, the query term can be simplified by introducing node c_e which is connected with query variables. By associating c_e with its parents through deterministic-OR gates, we can compactly represent the event of component e's change with a single node. Now, we can obtain the change propagation probability from c to e by the equation (3.3). Assuming the DSM of Figure 3-2 is directly converted into the BN, Figure 3-8 shows the result of change propagation probability among five components using predictive inference.

	a	b	c	d	e
a	-	0.3			
b	0.5	-			
c		0.3	-	0.5	0.7
d			0.9	-	
e				0.3	-

	a	b	c	d	e
a	-	0.3			
b	0.5	-			
c	0.17	0.33	-	0.61	0.7
d	0.15	0.3	0.9	-	0.63
e	0.04	0.12	0.34	0.42	-

$p_{ij} = p(c_j = yes \mid c_i^1 = yes)$

When component *i* is changed, what is the probability of component *j* being changed?

Figure 3-8. Result of predictive inference

However, the current method allows the initiating components to change more than once during the change propagation process. This might result in slightly higher probabilities than the CPM. To obtain the same result as CPM, the change initiating components should be controlled not to be changed during the propagation process. In BN, such action of controlling values is referred to as 'external intervention'. Distinction should be made between a passive observation (setting evidence) and an external intervention. Whereas the passive observation impacts the beliefs of the ancestors of a variable, the external intervention enforce a certain state on a variable does not under the assumption of causal ordering. External intervention is denoted by do-operator in BN. For example, $p(X \mid \mathrm{do}(Y \leftarrow y^*))$ means the posterior probability distribution of X given that we enforce the variable Y to the value y^*. Thus, the change propagation probability from component i to j can be obtained by the following equation:

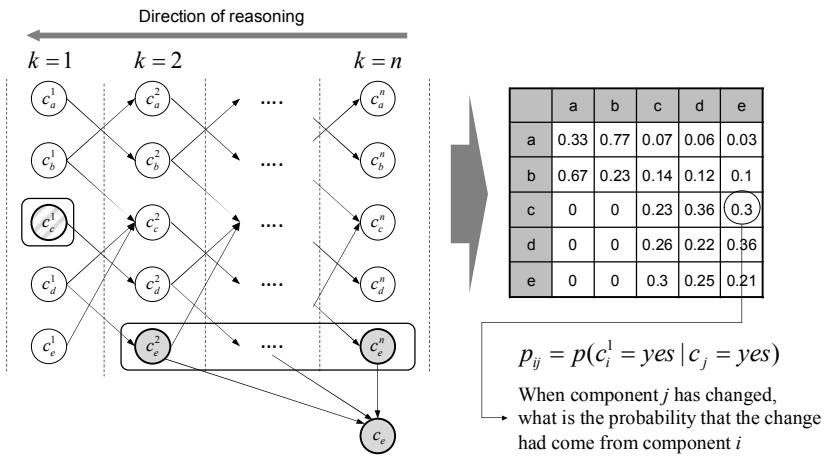
$$p(c_j \mid c_i^1 = yes, \mathrm{do}(c_i^{t\in[2,\mathrm{T}]} \leftarrow no^*)) \qquad (3.4)$$

where component x is enforced not to change during the change propagation process. Figure 3-9 shows the change propagation probabilities obtained by intervention. The obtained values are shown to be slightly less than those in Figure 3-8.

In some situation, an analyst may be interested in predicting the combined effect of change propagation caused by the initial changes of multiple components. Contrary to the CPM, this can be easily obtained in BN by posing a query having multiple c_i^1's changes as evidence. For example, suppose the situation where both components *a* and *b* are changed at initial stage. To compute the combined effect of change propagation, we can simply calculate the posterior probability distribution $p(c = yes \mid a = yes, b = yes)$.

3.4.2. Diagnostic Inference

Diagnostic inference is the reasoning of causes from symptoms. An example of diagnostic inference is given in Figure 3-10. Contrary to the predictive inference, by calculating the posterior probability distribution $p(c_c^1 \mid c_e = yes)$, we can measure how likely the component *e*'s change has come from component *c*. Thus, diagnostic inference pro-



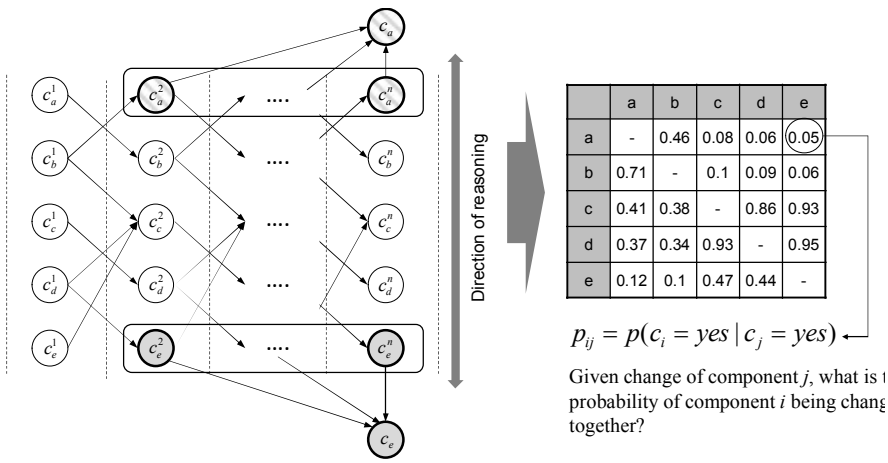
	a	b	c	d	e
a	0.33	0.77	0.07	0.06	0.03
b	0.67	0.23	0.14	0.12	0.1
c	0	0	0.23	0.36	0.3
d	0	0	0.26	0.22	0.36
e	0	0	0.3	0.25	0.21

Figure 3-10. Illustrative example of diagnostic inference

vides an analyst with information about the relative criticality of each component given a change of specific component. The complete result of diagnostic inference is illustrated in Figure 3-10. For example, by referring the third column in the matrix, we can identify that the component *c*'s change is influenced by component *e* most.

3.4.3. Inter-causal Inference

Inter-causal inference is utilized when we reason about the common-cause effects affecting the distinct symptoms. For example, as illustrated on the left-hand side of Figure 3-10, the reasoning between affected component $c_a \leftrightarrow c_e$ can be considered as an inter-causal inference. Utilizing this, we can identify how likely the components can change simultaneously due to the change of other components. Complete result of inter-causal inference between two components are illustrated in Figure 3-11. We need to pay attention to the components *c d*, *c e*, and *d e* which have high values in the matrix, because they are likely to



	a	b	c	d	e
a	-	0.46	0.08	0.06	0.05
b	0.71	-	0.1	0.09	0.06
c	0.41	0.38	-	0.86	0.93
d	0.37	0.34	0.93	-	0.95
e	0.12	0.1	0.47	0.44	-

Figure 3-11. Illustrative example of inter-causal inference

change at the same time given some external changes. This measure can be used as a guideline for determining modules or subsystems at the higher level. Moreover, intercausal inference can be performed in the opposite direction. For example, reasoning between initial components $c_j^1 \leftrightarrow c_k^1$, given an instatiation of component i, is also an intercausal inference. In this way, one can measure how likely both components j and k have affected to component i simultaneously.

3.5. Learning: Updating the Probabilities with Empirical Data

One of the advantages of using BN is that it has ability to learn from data. In most change management systems, there are a bunch of data which can be used for estimating probabilities between components. For example, most engineering change management systems contain the change logs showing the historical records of what was changed, why it was changed, or who changed it, etc. Utilizing these data, BN provides an intuitive way to learn the probabilities between components, combining the experts' opinion and empirical data.

The basic idea is similar to the standard statistical modelling approach. Under the circumstances that the distribution of real data is unknown, we assume that it belongs to a specific family of possible distributions. We can assume that each probability within CPT belongs to a distinct member of this families of distributions, which is represented by parameter θ. This contrast to the previous model where each parameter is specified in an explicit manner. When new data is available,

our task is to find the optimal parameter θ which provides most plausible explanation about the current set of data.

Among the various ways of learning parameters, Bayesian update approach is utilized. The overall procedure is illustrated in Figure 3-12. Firstly, an analyst expresses the uncertainty about the parameters in the form of prior distribution. This implies that an expert is not only asked the most likely values (means) but also their confidence about these values represented in error terms (variances). When the design project begins, we can obtain the dependency data from a set of engineering change logs. These statistical data is combined with prior distribution to calculate posterior distribution according to the Bayes' theorem. Now the posterior parameters become the priors for the next set of data. In this way, the model continuously adapts their optimal parameters in

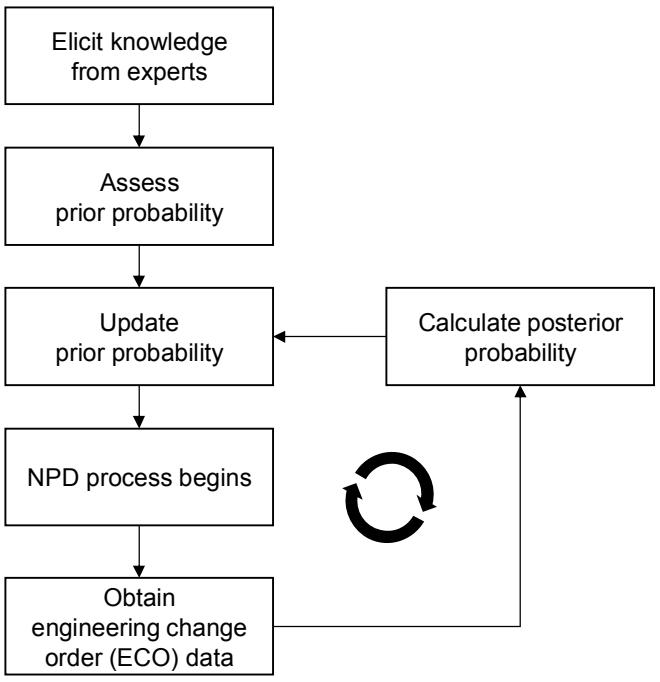


Figure 3-12. Bayesian parameter update procedure

light of the new data. As more data is utilized in parameter learning, more accurate parameters can be obtained.

In Bayesian updating of parameters, conjugate prior is used as a convenient approximation to facilitate the analysis. If a prior distribution has the same functional form as the posterior distribution for any data, it is often called a conjugate prior. For our case, we obtain binomial samples from the database, where each data is represented via binary states, i.e. changed or not changed, given some specific conditions. In this case, if we define our conjugate priors as Beta distributions, then the prior-to-posterior calculation can be performed easily (Cowell, 2006). Furthermore, the Dirichlet distribution can be used as conjugate priors for multinomial sampling, which is the generalization of binomial samples.

To construct a Beta prior for CPT, an expert is asked to estimate not only mean values about change probabilities but also the standard deviations around the means indicating the uncertainty about his estimation. The specified mean and standard deviation then determines the shape parameters α_y and α_n of the Beta distribution. These parameters can be obtained by solving following equations:

$$\begin{aligned} c_y : E(\theta_y) &= \frac{\alpha_y}{\alpha_y + \alpha_n} \quad \text{and} \quad Var(\theta_y) = \frac{\alpha_y \alpha_n}{(\alpha_y + \alpha_n)^2 (\alpha_y + \alpha_n + 1)} \\ c_n : E(\theta_n) &= \frac{\alpha_n}{\alpha_y + \alpha_n} \quad \text{and} \quad Var(\theta_n) = \frac{\alpha_y \alpha_n}{(\alpha_y + \alpha_n)^2 (\alpha_y + \alpha_n + 1)} \end{aligned} \tag{3-5}$$

We can regard α_y and α_n as the imaginary numbers of changes and no changes, respectively. Sum of shape parameters $\alpha_y + \alpha_n$ is often called the equivalent sample size, which can be thought of as imaginary

counts from our prior experience. Therefore, the larger the equivalent sample size, the more confident we are about our prior.

Table 3 shows an example of a Beta prior of CPT based on the assessment made in Table 1. Given four combinations of its parents *d* and *e*, an expert is asked for specifying four independent Beta distribution. In the rightmost columns in Table 3-3, the shape parameters for each distribution are identified.

After observing the data, shape parameters are updated following the Bayes' theorem. Updating procedure is simple; if *y* and *n* denote the numbers of cases corresponding to α_y and α_n, respectively, then prior distribution $\mathrm{Be}(\alpha_y,\ \alpha_n)$ is updated to posterior distribution $\mathrm{Be}(\alpha_y + y,\ \alpha_n + n)$. Table 3-4 illustrates updated posterior probability distributions shown in Table 3-3 in the light of some hypothetical data. Suppose we accept the assessment made in Table 3 and then we observe that the change in component *c* in two out of three cases when both *d* and *e* have changed simultaneously. Now, the prior distribution $\mathrm{Be}(1.8954,\ 0.3281)$ is updated to the posterior distribution $\mathrm{Be}(1.8954+2,\ 0.3281+1)$ which has the mean value of changes

Table 3-3. Beta prior CPT with expressed mean and standard deviation

comp d	comp e	std.	yes		no		parameter	
			mean	range	mean	range	α_y	α_n
yes	yes	0.2	0.85	0.75-0.95	0.15	0.05-0.25	1.8954	0.3281
yes	no	0.1	0.5	0.25-0.35	0.5	0.65-0.75	12	12
no	yes	0.1	0.3	0.55-0.65	0.7	0.35-0.45	6	14
no	no	-	0	0	1	1	-	-

Table 3-4. Beta posterior CPT after change log accumulated

compd	compe	prior parameter		hypothetical data		posterior parameter		mean change	std. change
		α_y	α_n	yes	no	α_y	α_n		
yes	yes	1.8954	0.3281	2	1	3.8954	1.3281	0.7457	0.1745
yes	no	12	12	5	7	17	19	0.4722	0.0821
no	yes	6	14	8	15	14	29	0.3256	0.0706
no	no	-	-	-	-	-	-	-	

0.7457 and the standard deviation 0.1754. The obtained results are fairly intuitive: by observing two out of three cases of changes, prior probability of component change is reduced from 0.85 to 0.7457. Moreover, by observing the additional cases, the standard deviation about the mean value is also reduced from 0.2 to 0.1754. Similar updates can be performed for all nodes in the network without much cost to perform the analysis.

3.6. Summary

This chapter illustrates the applicability of BN to the change propagation modeling and analysis. The main advantage of using BN is the ability to incorporate expert opinion in a more flexible manner. Because BN has no restriction in defining the structure of network or specifying the CPT of each node, it is applicable to a more generic situation, which makes it a useful tool in practice. At the analysis level, BN can be used to simulate various change scenarios to help the engineering change managers. In this paper, we have shown that various types of probabilistic inference, such as predictive, diagnostic, and intercausal inference, can generate a variety of measures which cannot be

obtained with CPM. Finally, another positive aspect using BN is that it provides a mathematically rigorous framework for updating change propagation probabilities. Based on the Bayes' theorem, we have shown that the BN can continuously adapt the model in light of observed data.

However, there are also challenges to overcome in applying the BN to the change propagation model. One of such challenges is that the model becomes complex with many nodes to specify, which requires a significant effort of domain expert. Especially for a complex product which consists of a large number of subsystems, it might be impossible to even specify the structure of the graph. This challenge addresses the interesting future research toward structural learning from data. Actually, developing an algorithm for inferring both the network structure and its CPT from empirical data is one of the most important issues in applying BN to change propagation problems.

Chapter 4. Design Freeze Sequencing using Bayesian Network

This chapter introduces a method for determining optimal design freeze sequence. Section 4.1 introduces the concept of design freeze sequence and explains why design freeze sequencing problem is important in managing change propagation. Section 4.2 introduces a formal definition of the freeze sequencing problem. Section 4.3 introduces the BN-based model for determining freeze sequence. Section 4.4 proposes algorithms for determining freeze sequences. Section 4.5 proposes a case study of EH101 Helicopter development project to validate our model. Finally, the results and future works are discussed in Section 4.6.

4.1. Introduction

Many product design processes are large and interdisciplinary in nature. A product usually consists of a set of components each of which is designed by a separate group of engineers (Eppinger et al, 1994). Complex systems such as automobiles or aircraft can involve even thousands of engineers making millions of design decisions over the course of years. As a product becomes complex and comes to involve many more decision makers, coordinating the design of components grows very complicated. Management of complexity within the design and development process, not surprisingly, is a frequent focus of engineering management research.

One way of mitigating the risk of change propagation is successive freezing of components (Thomke, 1997). Design freeze is defined as the end point of the design phase at which a technical product description is handed over to production. By freezing some components earlier, company can expect two main advantages; first, overall process can be quickly stabilized by limiting the engineering changes on that component; second, since an early-frozen component can be handed over to the manufacturing phase in advance, it can reduce overall product development lead-time. However, at the same time, an early-frozen component might still be vulnerable to changes propagated from other components. It is not uncommon that, due to component interdependency, an early-frozen component has to be redesigned, which can lead to significant rework costs. In this light, careful planning of design freeze sequence is required for efficient management of change propagation during design process.

This chapter proposes a model for change-propagation-risk-based determination of optimal freeze order. In order to identify the optimal freeze sequence, the Bayesian network (BN), which is an emerging tool for a wide range of risk management tasks, is used as a modeling framework for representing a sequential freeze process. When a freeze decision is made with respect to one component, the change propagation risk associated with it is removed from the system. In this setting, the optimal freeze sequence is that which reduces risk to the system in most effective manner.

4.2. Problem Definition

The problem of finding the optimal freeze sequence can be formulated as follows:

- *Given:* C, *a set of components;* σ_C, *the set of permutations of* C, *and* f, *a function that evaluates* σ_C,
- *Problem: find* $S' \in \sigma_C$ *such that* $f(S') \geq f(S'')$ *for* $\forall S''\ (S'' \in \sigma_C, S'' \neq S)$.

Here, σ_C represents the set of all possible freeze sequences, and f is the function that evaluates the effectiveness of freeze sequence. Thus, the problem is to choose the best sequence of components maximizing the user-defined evaluation function f. To further define our problem as a sequential decision problem, the following assumptions are required.

- Sequential freeze process: decision maker freezes one component at a time.
- Initial change probability is identified for each component: represents the probability that a change first arrives at that component. Change can occur due to safety issues, new technical solutions, or a change of customer request. Note, however, that this probability only describes the initiation of changes. The component might have a probability higher than the initial one, due to change propagation from other components.
- Change propagation probability is identified among components: represents the probability that a change that appears in one component results in changes to the other components.

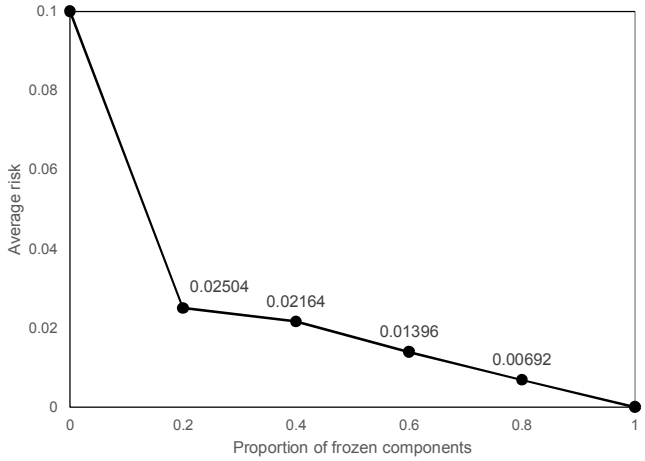


Figure 4-1. Example of change propagation risk trajectory

When a component is frozen, it does not initiate change propagation to other components, therefore, decrease the change propagation risk of entire system. Figure 4-1 illustrates a trajectory of change propagation risk of a design freeze sequence of five components. As can be seen, each point represent the design freeze point and the values assigned over each point is average residual risk of unfrozen components. BN is utilized in calculating dynamic evolution of change propagation risk given a current freeze sequence. As more components are frozen, the design begins to stabilize, and when the final component is frozen, the change probability of each component falls to zero. In this setting, our objective is to find optimal design freeze sequence which can maximally mitigate the change propagation during design process. For, the rest of the paper, we would discuss more detail about this process.

4.3. Modeling Freeze Process using Bayesian Network

4.3.1. Modeling of Cause-effect Relationship among Components

In order to sequentially calculate the uncertainty of each component given freeze states of components, a BN is proposed. In this BN, the node corresponds to the component of which design can be frozen by a separate group of engineers. This node is a discrete and binary random variable the state of which takes 'yes' if it accepts a design change, and 'no' otherwise. Edges connecting two nodes indicate direct dependency between components.

To represent cascading effect of changes which occurs through several intermediates steps, each node should be extended with temporal dimension. This types of BN is often called dynamic Bayesian network

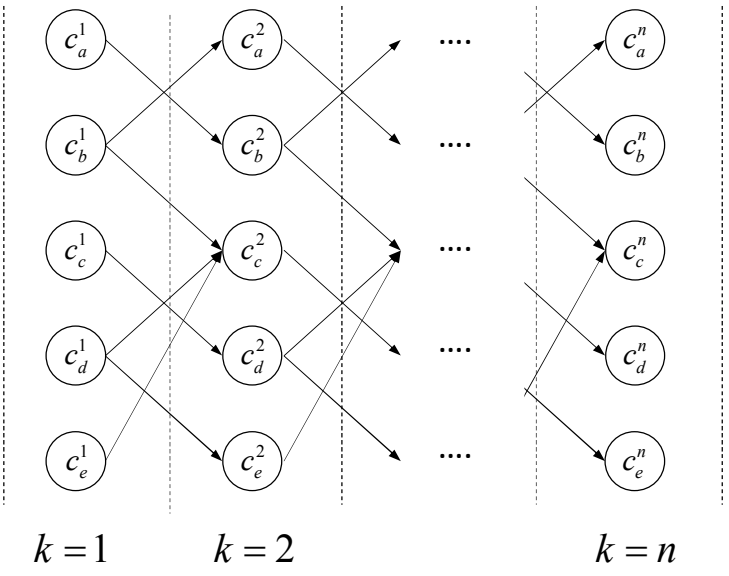


Figure 4-2. Change propagation network of five components

(DBN). It consists of a sequence of sub-models, each of which represents the state of the system at a certain point in time, which representation is called a time point. Temporal edges connecting nodes between consecutive sub-models reveal such temporal dependency between states. Although a DBN models a dynamic system, its structure is time-invariant; that is, the structure of the network does not change over time, with the exception of the root node (i.e. the node at time period 1). Therefore, whenever the prior distribution of the root node is specified, the DBN can recursively update the network, enabling a user to predict the further behavior of the system for the desired number of iterations.

As an illustrative example, the DBN of change propagation process among five components is illustrated in Figure 4-2. The temporal node c_i^k represents the uncertain state of component i at change propagation stages k. Each time period indicates an intermediate change propagation step during the change propagation process. If a change arrives in the root nodes, it can cascade through the change propagation path revealed by temporal edges $c_i^{k-1} \rightarrow c_j^k$ connecting nodes between consecutive stages. For example, in Figure 4-2, if a change arrives in c_c^1, this change can propagate c_d^2. Again, the change of c_d^2 can result in changes to c_e^3 and c_c^3, respectively. In this way, a user can unroll the change propagation process for the desired number of time stages, duplicating a set of nodes and temporal edges. However, it is noteworthy that temporal edges connecting nodes between consecutive time-slices do not change during the change propagation process, since the structure of the DBN is time-invariant.

4.3.2. Assessment of Change Propagation Probability

In order to quantify the degree of change propagation between components, each node should be identified with the CPT. For the root node, only the prior probability is required. This probability represents the arrivals of engineering changes to the component. Meanwhile, the nodes in the intermediate time periods (from 2 to n) describes the behaviors of the change propagation process. Since these nodes are influenced by their immediate predecessor, the conditional probability distribution given every instantiation of their parents should be specified. Consider, again, the CPT of c_b^2 in Figure 4-2. As can be seen, the state of c_b^2 is affected by c_a^1. Since c_b^2 does not change without the change of c_a^1, the conditional probability $p(c_b^2 = yes \mid c_a^1 = no)$ becomes zero. In contrast, c_b^2 might change with some probability if the state of c_a^1 is set to *yes*. Suppose that such chance probability is 30%. Then, the CPT of c_b^2 can be defined as follows:

$$\begin{aligned} &p(c_j^k = yes \mid c_i^{k-1} = no) = 0 \\ &p(c_j^k = no \mid c_i^{k-1} = no) = 1 \\ &p(c_j^k = yes \mid c_i^{k-1} = yes) = 0.3 \\ &p(c_j^k = no \mid c_i^{k-1} = yes) = 1 - 0.3 \end{aligned} \quad . \qquad (4\text{-}1)$$

Since the DBN has a time-invariant structure, each c_b^t during intermediates states has the same CPT as that of c_b^2.

When a component is connected by many components, the number of parameters to be estimated increases exponentially; so, if n parents is connected to a common child, then the number of probability distributions to be estimated is 2^n. Therefore, as n increases, identifying

the conditional probability distribution given every parent combination becomes computationally burdensome.

The knowledge elicitation burden of constructing the CPT can be reduced by assuming a conditional independence relationship among parents. Suppose that there are several causes $X_1, X_2, \ldots, X_n$ and a common effect variable Y, where each of the causes X_i has the probability p_i of being sufficient to produce the effect. The conditional independence among $X_1, X_2, \ldots, X_n$ with respect to Y holds when each parent's ability to produce the effect Y is not influenced by the presence of other parents. When conditional independence is assumed, the CPT of Y can be obtained by the equation

$$p(y \mid X_p) = 1 - \prod_{i:X_i \in X_p} (1 - p_i) \qquad (4\text{-}2)$$

where X_p is the set of every instantiation of its parent. This special structure of CPT is referred to as the Noisy-OR model. For example, consider the example of c_b^2 in Figure 4-2, which is affected by both c_d^1 and c_e^1. The CPT of c_b^2 under Noisy-OR model is depicted in Table 4-1. As can be seen, if we know the change propagation probability of $p(c_b^2 = yes \mid c_d^1 = yes)$ and $p(c_b^2 = yes \mid c_e^1 = yes)$ respectively, then, $p(c_b^2 = yes \mid c_d^1 = yes, c_e^1 = yes)$ can be calculated without further knowledge elicitation using equation (4).

Table 4-1. Conditional probability table of component c (based on Figure 4-2)

comp. d	comp. e	Yes	No
yes	yes	1-(1-0.3)(1-0.5) = 0.85	(1-0.3)(1-0.5) = 0.15
yes	no	1-(1-0.3) = 0.5	(1-0.3) = 0.5
no	yes	1-(1-0.3) = 0.3	(1-0.3) = 0.7
no	no	0	1

4.3.3. Calculation of Change Propagation Risk given Design Freeze Decision

After BN is constructed, it can be used to answer various probabilistic queries. When a certain state of a node is observed by a decision maker, this node is called 'evidence'. This evidence e, then, propagates across the network, updating a new posterior probability distribution $p(X \mid e)$ for each variable. The BN provides a mechanism for calculating the posterior probability distribution of a certain hypothetical variable x given the availability of a set of evidence e. This $P(x \mid e)$-calculation task is often called probabilistic inference.

The concept of probabilistic inference is used to update the change propagation risk as design freeze sequence progress. Suppose that we want to calculate the remaining risk of each component after component *a* is frozen in Figure 4-2. Since the frozen component does not allow changes, the true state of the corresponding root node c_a^1 is set to 'no' regardless of the external event. Now the posterior probability distribution $p(c_x = yes \mid c_a^1 = yes)$ indicates the remaining risk to all five components. We can then proceed to the next component by making additional observations on the root nodes. The change propagation risk

of freezing component *b* after freezing component *a*, then, can be calculated by computing $p(c_x \mid c_a^1 = no, c_b^1 = no)$. In this way, we can update the change propagation risk of every component until the final component is frozen. By means of this scheme, we can compare the effectiveness of all freeze sequence alternatives and identify the optimal one among them.

4.4. Derivation of Optimal Freeze Sequence

Finding optimal freeze sequence can be viewed as finding the optimal sequence of making evidence on root node. Such a sequential decision problem is difficult to solve, however, because there is no simple closed-form solution that has been found. Therefore, one must either check all possible sequences or use a heuristic to check, based on a greedy approach. In the following section, we will introduce several algorithms for identifying the component freeze order.

4.4.1. All Enumeration Algorithm

An algorithm that can always guarantee the optimal freeze sequence needs to consider all possible freeze sequence orderings. The all enumeration algorithm, correspondingly, identifies the optimal freeze order by searching all possible sequences. When a product consists of *n* components, each of the $n!$ orderings is checked in order to determine the optimal order. It is clear that the freeze-sequencing problem is a subset of the traveling salesman problem, a famous NP-hard problem, even if its computation procedure is deterministic. The all enumeration algorithm can be described as follows.

All enumeration algorithm

Step 1: Extract initial sequence from the permutation set

$S' \leftarrow$ the initial sequence from the permutation set σ_C

$\sigma_C \leftarrow \sigma_C \setminus \{S'\}$

Step 2: Proceed to next sequence and extract it from the permutation set

$S'' \leftarrow$ next sequence to consider

$\sigma_C \leftarrow \sigma_C \setminus \{S''\}$

If $\sigma_C = \{\varnothing\}$ then terminate.

Step 3: Compare sequence

If $f(S') \le f(S'')$ then $S' \leftarrow S''$

Otherwise remains same.

Step 4: Go back to step 2.

As an illustrative example, the all enumeration algorithm was applied to the five-component example from Figure 1. However, since the inherent complexity of probabilistic inference is NP-hard, when the size of the network increases, searching the optimal sequence from among all possible combinations becomes intractable.

4.4.2. Greedy Algorithm

When solving a sequential decision problem, greedy algorithm can be utilized as a good approximation (Li et al., 2007). The underlying strategy of the myopic search algorithm is to always choose the best option given the current situation. In our problem, the myopic approach was applied such that the component that can maximally mitigate the overall risk is frozen first, followed by the second best component, and so

on until to the final component. Although the solution optioned by the myopic search might be suboptimal, it can reduce the complexity of searching sequence alternatives to n without any significant loss of accuracy. The myopic algorithm can be described as follows.

Myopic Search Algorithm

Step 1: Initialize remaining component K and ordered set of optimal sequences J

Step 2: Identify components that can maximally mitigate change propagation risk R

Compute $\arg\min_x R$

$J \leftarrow J \bigcup \{x\}$

$K \leftarrow K - \{x\}$

If $J = \varnothing$ then Stop.

Step 3: Go to step 2.

If every component is assigned a uniform cost, the myopic search algorithm can guarantee an optimal solution, because the trajectory of change propagation risk always shows non-increasing patterns. However, if the cost of every component is different, the myopic search algorithm cannot guarantee an optimal solution.

4.4.3. Non-greedy Algorithm

The myopic search algorithm looks only at the effect of freezing one component at a time. As such, this approach cannot guarantee the optimality of the solution. In order to improve the solution without sacrificing too much complexity, a non-myopic algorithm can be utilized.

One promising approach is to use the K-optimal algorithm. The principle of this algorithm is simple: it enumerates K-pairs of decision elements together, and selects the best pairs for each algorithm cycle. In our problem, the K-optimal algorithm, rather than search one component at a time, freezes K-pairs of components. After freezing the best component pairs, the same procedure is applied to the remaining components. The K-optimal algorithm can be described as follows.

K-optimal algorithm

Step 1: Initialize remaining component K and ordered set of optimal sequences J

Step 2: Enumerate all possible k-pairs of components from K. Denote this set as S_k.

Step 3: Identify best component pairs that can maximally mitigate change propagation risk R

Compute $s = \arg\min_{S_k} R$

Step 4: Re-order s such that it can maximally mitigate change propagation risk r.

Step 5: Update freeze sequence J and remaining component K.

$J \leftarrow J \bigcup \{s\}$

$K \leftarrow K - \{\mathrm{s}\}$

If $K = \varnothing$ then Stop.

Step 6: Go to step 2.

Although solution quality improves with increasing value of K, computation time also increases, due to the increased number of component pairs. Johnson and McGeoch (1997) showed that for k > 3, the computation time increases considerably faster than the solution quality, indicating that the 2-optimal approach is both fast and effective.

4.5. Case study

4.5.1. Product Descriptions

For illustration, our model is applied to the product data of Westland Helicopter EH101, which was originally obtained by Clarkson et al. (2004). They obtain this data through workshops and interviews of component designers. They used DSM to decompose the helicopter into subsystems, such as engines, weapons, or avionics etc. Figure 5 shows the DSM of EH101. The (i, j) element in the matrix indicates the change propagation probability from component i to component j. As illustrated, this helicopter consists of 19 subsystems, the components of which are interrelated with complex interdependency.

Firstly, the DSM information is converted into our BN-based change propagation model. This process is straightforward. Each component in the DSM is converted to a node in the graph. Edges can be easily identified by referring the dependency structure in the DSM. After the graph structure is identified, the CPT of each node should be identified. In case

4.5.2. Dynamic Bayesian Network Model Representation

When the number of parents is too many, eliciting change probabilities given every possible combination might be intractable. The Noisy-OR model can then be used to reduce the knowledge elicitation burden because it requires only the pairwise relationship between each combination of parent and component. In our case example, only the Noisy-OR model is applicable because DSM only identifies the interactions between two components.

One problem of DSM-based information is that the data is intrinsi-

cally static. The data cannot be adjusted during the design process once

it is obtained. BN, on the other hand, provides intuitive mechanism for

updating or learning the network using the engineering change log da-

tabase. For adjustment of parameters, analysts express the uncertainty

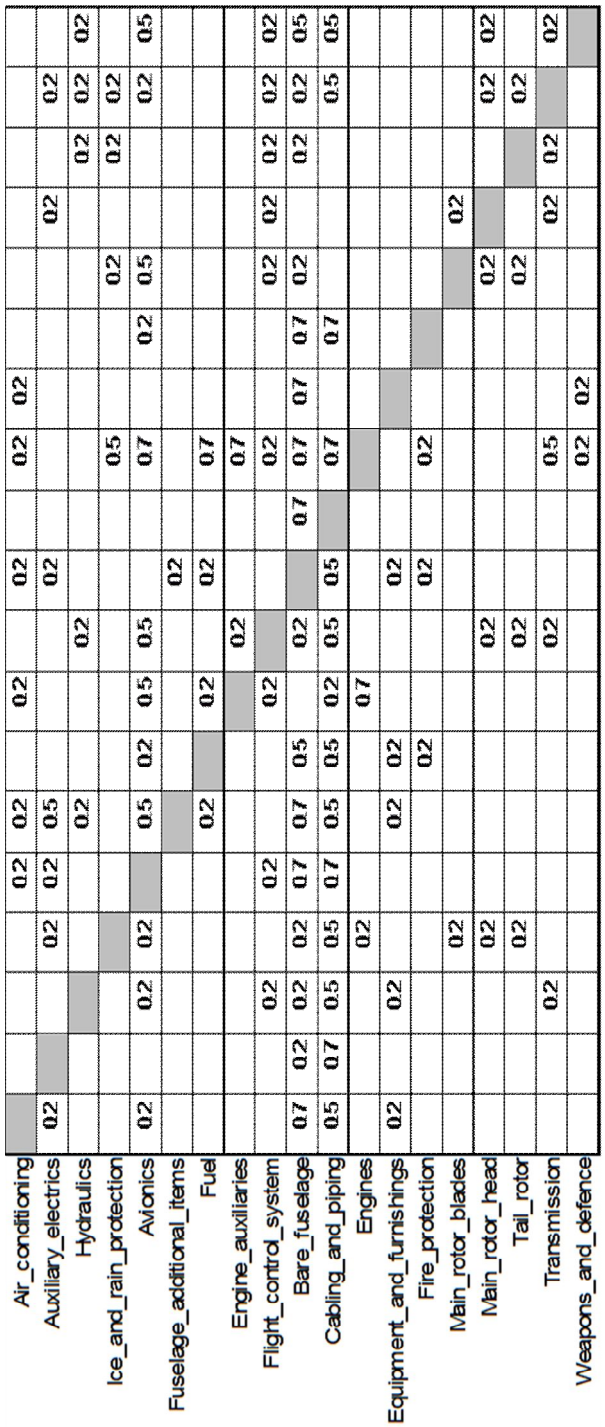


Figure 4-3 DSM of EH101 (adopted from Clarkson et al., 2004). The values in the matrix are obtained from Cambridge Advanced Modeler (Wynn, 2010), which is an implemented toolbox of CPM.

about the parameters in the form of their prior distributions. The under

tain parameters are adjusted as the engineering change data are combined with prior distributions to calculate the posterior distributions according to the Bayes' theorem. Even when the DSM data is not available, BN can also automatically construct the network. For details about the parameter learning and updating procedure, please refer to Lee and Hong (2015).

4.5.3. Scenario I: Minimizing Overall Change Propagation Risk

After BN-based model is constructed. It can be used to derive optimal freeze sequence. Depending on the objective of the decision maker, a different freeze order can be obtained. The first scenario concerns a

Table 4-2. Design freeze sequence of Scenario I

Seq.	Subsystems	Avg. risk	Subsystems	Avg. risk
1	Engines	0.449844	Air conditioning	0.494207
2	Engine auxiliaries	0.410602	Auxiliary electrics	0.493049
3	Flight control system	0.389146	Hydraulics	0.485151
4	Ice and rain protection	0.366664	Ice and rain protection	0.468637
5	Transmission	0.346614	Avionics	0.460993
6	Weapons and defense	0.324962	Fuselage additional items	0.447768
7	Fuselage additional items	0.30282	Fuel	0.440808
8	Main rotor blades	0.278877	Engine auxiliaries	0.39881
9	Tail rotor	0.258822	Flight control system	0.375276
10	Hydraulics	0.239103	Bare fuselage	0.362122
11	Avionics	0.218099	Cabling and piping	0.356957
12	Bare fuselage	0.195187	Engines	0.264193
13	Air conditioning	0.171562	Equipment and furnishings	0.245958
14	Fuel	0.144983	Fire protection	0.225532
15	Fire protection	0.114941	Main rotor blades	0.190475
16	Equipment and furnishings	0.080211	Main rotor head	0.169274
17	Main rotor head	0.050342	Tail rotor	0.137691
18	Cabling and piping	0.024354	Transmission	0.083015
19	Auxiliary electrics	0	Weapons and defense	0

case in which the decision maker wants to obtain a design freeze order that minimizes system-level risk. This scenario can be applied to a case in which the main objective of the design freeze is to stabilize the design as early as possible, thereby facilitating convergence to the final design. System-level risk can be obtained by aggregating the change propagation probabilities of the respective components.

In this example, the user-defined evaluation function f calculates the average change propagation probability of each of the 19 components as the result of the design freeze decision. Let F_k be the set of frozen components at k th freeze decision. Then, average change propagation probability among 19 components at k th freeze decision can be calculated as following

$$\sum_{i=1}^{19} \frac{p(c_i = yes \mid c_{j \in F_k} = \text{no})}{19}. \qquad (4\text{-}2)$$

However, as stated earlier, different measure can be used to measure the change propagation risk during freeze decision. For example, instead of average probability of components, a manager may be interested in reducing the variability of components probabilities. In this situation, variance related measure may be a useful measure. On the other hands, he or she may be interested in reducing the maximum change propagation probability. In this way, different measure can be utilized for freeze sequencing.

The average change propagation risk would show a monotonically decreasing pattern as the number of frozen components increases. The freeze sequence obtained by greedy algorithm is provided on the left side of Table 4-2. The first and second columns represent the freeze

order and corresponding components, respectively. The third column represents the average change propagation probability of each of the 19 components given the currently frozen components. To illustrate the effectiveness of our methodology, we compared the performance of our solution with that of an arbitrary sequence. This arbitrary sequence is indicated on the right side of Table 4-2. The risk trajectories of the two freeze sequences are illustrated in Figure 4-4. Unlike the arbitrary sequence, the myopic algorithm rapidly remove the change propagation risk from the system. The area under risk trajectory in Figure 4-4 was 0.3082 with the arbitrary sequence and 0.2180 with the optimal sequence, indicating that our sequencing method effects a significant risk

mitigation improvement. As a result, our myopic sequencing method mitigates more than 30% of change propagation risk.

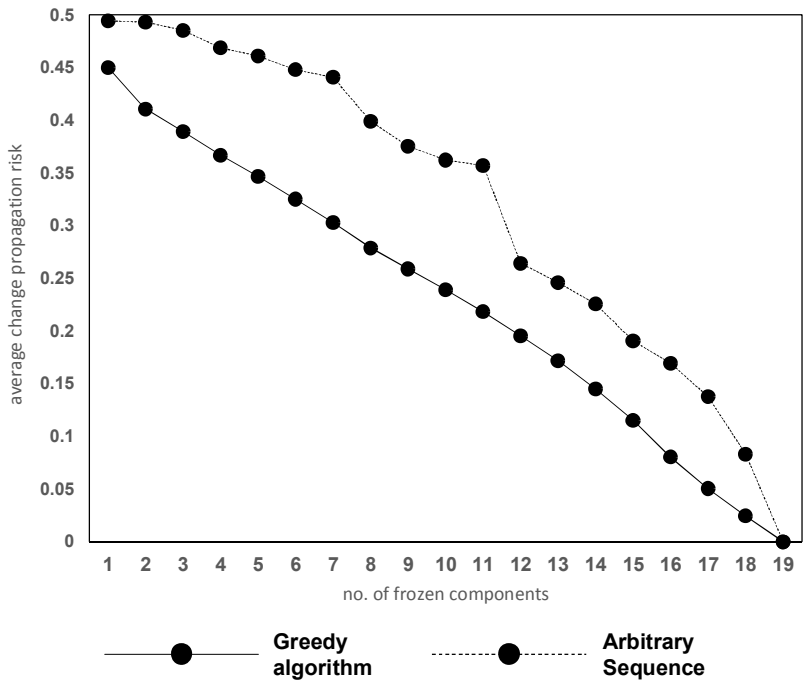


Figure 4-4. Freeze sequence of scenario I (myopic algorithm vs arbitrary sequencing)

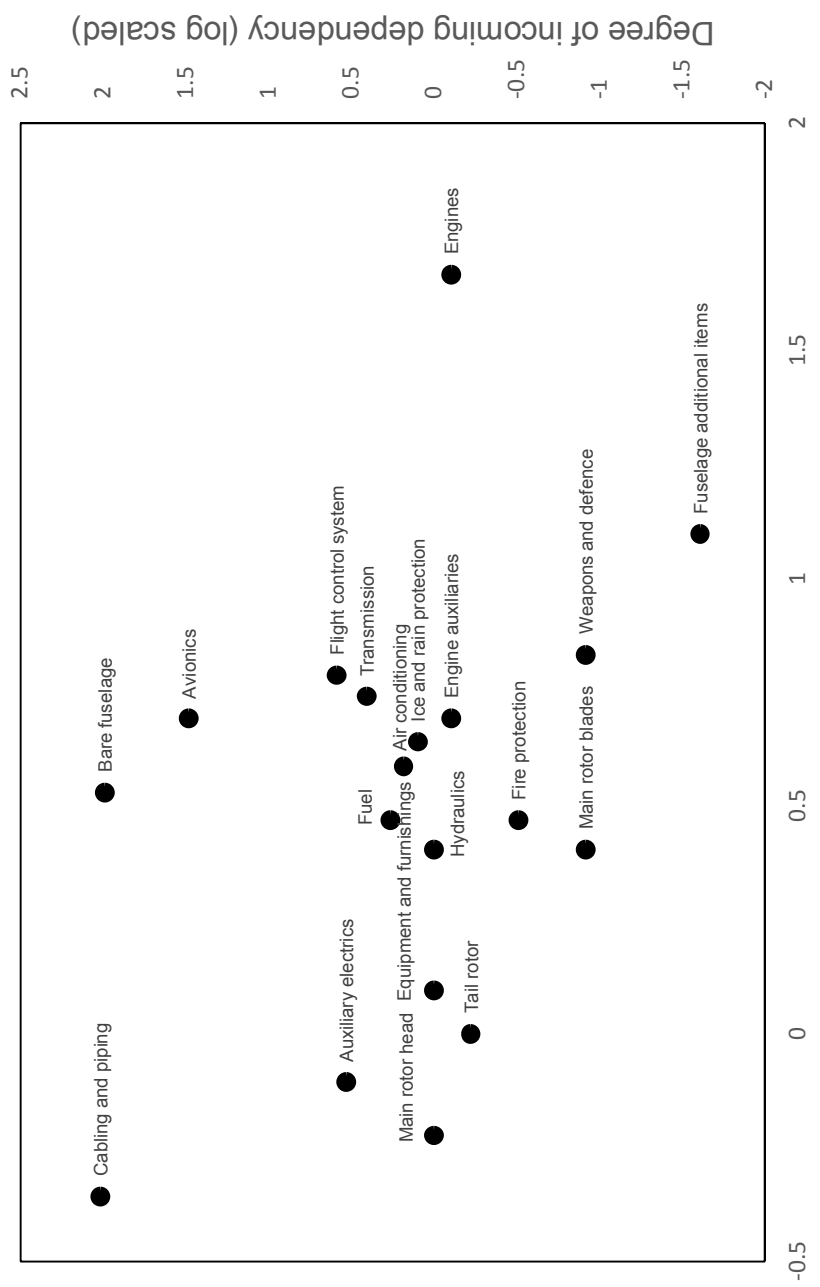


Figure 4-5 Relative position of components The numbers associated with components indicate optimal freeze sequence obtained from Table 4-2.

For interpretation of our result, we plot each component with respect to the degree of incoming/outgoing risk. The incoming risk, which is the indicator of change absorber, is obtained by summing the DSM rows. On the other hand, the outgoing risk, the indicator of change multiplier, is obtained by summing the DSM columns. Figure 4-5 illustrates the relationship between the freeze sequence and the degree of incoming/outgoing risk of each component. As can be seen, the strong change multipliers are likely to be frozen earlier, while the change absorbers are frozen later. However, some exceptional cases can also be found in the figure. For example, Fuselage additional items, which is the second strong multiplier, is frozen in the seventh decision. This may be due to the fact that the relative position on the risk plot continuously changes after each freeze decision is made.

4.5.4. Scenario II: Minimizing Change Propagation Risk of Already-frozen Components

So far, we have derived a design freeze order that minimizes every component's average change propagation risk that is incurred during the execution of component freezing. However, a decision maker might be interested only in minimizing the risk of already-frozen components. This scenario can be applied to the case in which the redesign cost of an already-frozen component is much more expensive than an unfrozen component. Since our objective is to obtain a freeze sequence that can minimize the change propagation risk of an already-frozen component, now, the evaluation function f computes the average change propagation risk only of frozen components. Let F_i also be the set of frozen

Table 4-3. Design freeze sequence of Scenario II

Seq.	Myopic Algorithm	Avg. risk	Arbitrary sequence	Avg. risk
1	Main rotor blades	0.134516	Air conditioning	0.641448
2	Weapons and defense	0.149489	Auxiliary electrics	0.675127
3	Engines	0.148517	Hydraulics	0.554937
4	Engine auxiliaries	0.108071	Ice and rain protection	0.480047
5	Ice and rain protection	0.092182	Avionics	0.533423
6	Tail rotor	0.087999	Fuselage additional items	0.479137
7	Transmission	0.082403	Fuel	0.475601
8	Main rotor head	0.069468	Engine auxiliaries	0.401292
9	Flight control system	0.051856	Flight control system	0.375420
10	Hydraulics	0.041406	Bare fuselage	0.418887
11	Fuselage additional items	0.049941	Cabling and piping	0.460733
12	Avionics	0.057744	Engines	0.327048
13	Fuel	0.061620	Equipment and furnishings	0.300808
14	Fire protection	0.061236	Fire protection	0.265031
15	Air conditioning	0.057300	Main rotor blades	0.214300
16	Equipment and furnishings	0.045304	Main rotor head	0.186738
17	Bare fuselage	0.036918	Tail rotor	0.148091
18	Auxiliary electrics	0.023609	Transmission	0.087152
19	Cabling and piping	0	Weapons and defense	0

components at ith freeze decision. Then, our evaluation function then can be defined as following,

$$\sum_{i \in F_k} \frac{p(c_k = yes \mid c_{j \in F_k} = \text{no})}{k}. \qquad (4\text{-}2)$$

At first, the greedy algorithm is applied to obtain the freeze sequence. The average change propagation risk of frozen components with respect to freeze decisions is illustrated in Table 4-3. Additionally, the performance of the freeze order obtained by arbitrary sequencing is illustrated on the right side of the table. The two different sequencing methods are illustrated in Figure 4-6. Compared with the first scenarios, the change propagation risk does not show a monotonically decreasing pattern; instead it repeats up and down until it reaches the final components. This is due to the fact that the average risk can increase as the

number of frozen components increases. The area under risk trajectory in Figure 4-6 was 0.3529 with the arbitrary sequence and 0.068 with the myopic algorithm sequence, indicating that our sequencing method provides an almost five times better performance than the arbitrary sequencing method.

Table 4-4. Comparison between myopic algorithm and non-myopic algorithm

Seq.	Myopic Algorithm	Avg. risk	Arbitrary sequence	Avg. risk
1	Main rotor blades	0.134516	Engines	0.218953
2	Weapons and defense	0.149489	Engine auxiliaries	0.132653
3	Engines	0.148517	Main rotor blades	0.114560
4	Engine auxiliaries	0.108071	Ice and rain protection	0.098246
5	Ice and rain protection	0.092182	Weapons and defense	0.092182
6	Tail rotor	0.087999	Tail rotor	0.087999
7	Transmission	0.082403	Transmission	0.082403
8	Main rotor head	0.069468	Main rotor head	0.069468
9	Flight control system	0.051856	Flight control system	0.051856
10	Hydraulics	0.041406	Hydraulics	0.041406
11	Fuselage additional items	0.049941	Fuselage additional items	0.049941
12	Avionics	0.057744	Avionics	0.057744
13	Fuel	0.061620	Fuel	0.061620
14	Fire protection	0.061236	Fire protection	0.061620
15	Air conditioning	0.057300	Air conditioning	0.061236
16	Equipment and furnishings	0.045304	Equipment and furnishings	0.057300
17	Bare fuselage	0.036918	Bare fuselage	0.045304
18	Auxiliary electrics	0.023609	Auxiliary electrics	0.036918
19	Cabling and piping	0	Cabling and piping	0.023609

Contrary to the first scenario, which shows monotonically decreasing patterns of change propagation risk, in this second scenario, the myopic algorithm cannot guarantee the optimality of the solution. To improve the solution, a non-greedy algorithm can be applied. In this example, two-step-look-ahead approach is applied. That is, in current decision point, the algorithm searches further two set of components. This algorithm first searches component pairs and then rearranges each of them. The results of the two-optimal search algorithm are illustrated in Table 4-4. As seen in Figure 4-7, the two algorithms show different freeze orders for the first six components. The area under risk trajectory of the new freeze order was 0.0670, which represents a slight improvement in the performance.

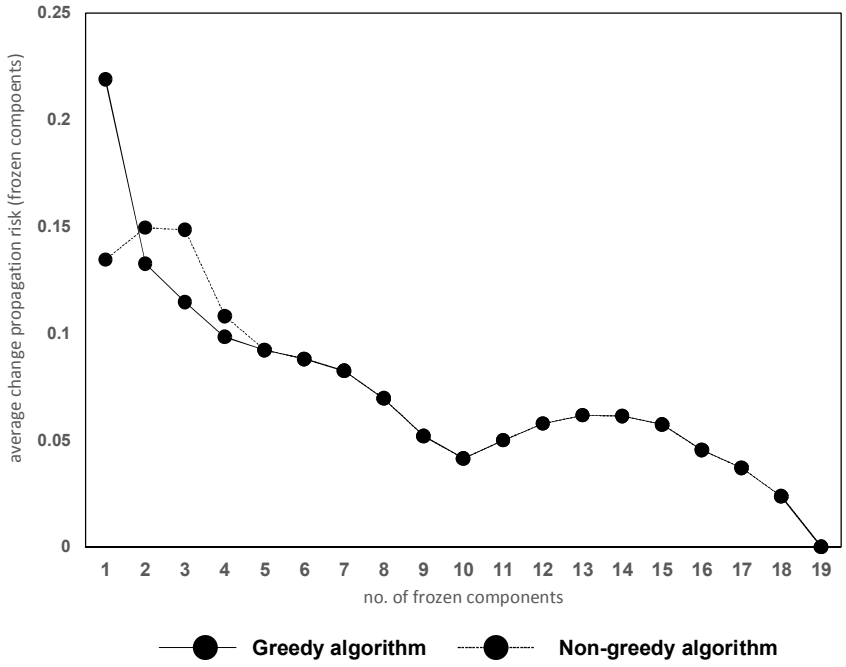


Figure 4-7. Comparison between greedy and non-greedy algorithm

4.6. Summary

Change propagation is the major source of schedule delays and cost overruns in design projects. One way to mitigate the risk of change propagation is to impose a design freeze on components at some point prior to completion of the process. In this situation, identifying the most appropriate freeze sequence is crucial, as it can effectively mitigate change propagation risk and, thereby, improve design project outcomes.

This paper proposes a Bayesian network (BN)-based model for deriving the optimal freeze sequence. In this study, a dynamic Bayesian network (DBN) was used to represent the change propagation process within a system. According to the model, when a freeze decision is made with respect to a component, a probabilistic inference algorithm

within the BN updates the uncertain state of each component. Utilizing this mechanism, we can identify the trajectory of risk according to the freeze sequence. And since identification of the optimal freeze sequence is similar to the sequential decision problem, we propose efficient algorithms for identifying near-optimal solutions. In a case study, we derived the optimal freeze sequence of a helicopter design project from real product development process. The experimental result showed that our proposed method can significantly improve the effectiveness of freeze sequencing compared with arbitrary freeze sequencing.

We believe that our model provides a useful guidance for freezing decisions in complex engineering design projects. Practically, our model can be utilized in following applications. First, our model can be used to planning freeze sequence: Using the model, one can derive an optimal freeze sequence plan in advance before the actual design process begins. This might be useful for better planning and structuring the design process. Seconds, proposed model can be utilized as a tool for assessing change propagation risk during product development process. Our model can monitor the current level of change propagation risk of each component given the component freeze status. This measure helps a project manager or component designers to check the current design progression and dynamically adjust their change implementation plan. Thirds, it can be used to justify competing freezing proposals: Our model can also be used to evaluate the effect of additional freeze decisions given the current freeze status. The project manager can quantitatively evaluate the competing freeze proposals when a multiple of them

are proposed. Different types of objective functions can also be used to evaluate each scenario. For example, one can evaluate the freeze proposals by comparing how much change propagation risk can be reduced, or how vulnerable the already frozen components become as a consequence of changes of unfrozen components.

Chapter 5. Structured Management of Change Propagation by Learning Dependency Network

This chapter introduces a methods for learning change prediction model using engineering change logs database. Section 5.1 explains the reason and backgrounds why data learning approach is required. Section 5.2 introduce the formal problem definition of proposed model. In section 5.3, Dependency network which is an approximated version of Bayesian network is introduced. Section 5.4 introduces a mechanism for learning change propagation model from data. Section 5.5 explains detail procedure for data learning. In section 5.6 proposed methodology is validated through real software project change log data. Finally, the contribution of this part and results are summarized in Section 5.7.

5.1. Introduction

The major difficulty of managing engineering change is the fact that it does not occur independently. Since each component is intricately interconnected with others, a change made in one components often necessitates changes in another. Moreover, multiple changes interact with each other (Eckert et al., 2004). This “knock-on” effect is often referred to as change propagation. When change propagation occurs, it can severely degrade the performance of a product development process by

consuming engineering resources. The impact of change propagation becomes more problematic when developing complex systems, because the more parts that are interconnected with each other, the greater is the chance that changes will propagate within the system.

However, a common limitation of these approaches is their sole dependence on expert opinion for extraction of dependency information (Jarratt et al., 2011). Expert elicitation of this kind has several disadvantages. First, identification of the entire connectivity structure entails too much effort, as the number of interactions to be identified increases exponentially with the number of components. Second, the numerical value, the product of the expert's unavoidably subjective opinion, is likely to be biased. Third, the significant expense of the process of expert elicitation makes adjusting the change propagation model during product development process difficult.

The objective of this chapter is to propose novel change prediction methods that can automatically identify connectivity structures by data mining of previous change records. As a modeling language, the dependency network (Heckerman et al., 2001), a graphical model for representation of the joint probability distribution among a set of random variables, was utilized. The dependency network captures the conditional statistical dependence and independence relationships of system components in a way that can be applied to the estimation of change propagation probability. Since the dependency network can be automatically learned from the previous change logs, data-gathering cost can be saved and change prediction accuracy can be improved.

The remainder of this paper is organized as follows. Section 5.2 briefly introduces the previous change prediction methods and the relative position of our current research. Section 5.3 introduces the probabilistic graphical model and shows how it can be used to model change propagation. Section 5.4 explains the overall procedure by which a product architecture is learned from data and then used to predict change propagation. Section 5.5 provides a case study that illustrates the use of our method with real data. Finally, section 6 draws conclusions and anticipates future work.

5.2. Problem Definition

Suppose that raw data in the form of historical records of engineering changes occurring in the course of product development are available. The structure of such data is shown in Figure 5-1. Each change record contains information on the components that were modified, the reasons for the changes, and the change initiator. This information is easily obtainable from the ECM (engineering change management) system, which computerizes engineering change (EC) work-flows to ensure that each EC is requested, notified, and executed by appropriate engineers. Our objective, then, was to develop a model that can predict the propagation of change from a source component to a target component.

Before developing this model, we assumed that simultaneous change of components when a single change propagation event occurs. According to this assumption, individual change records can be divided into irrelevant groups, which is called a transaction. Figure 5-1 provides an example of transaction data. From such data, the pattern of change propagation can be predicted by identifying which components frequently are changed together. For example, in the example shown in the figure, we can determine that components *a* and *b* frequently have been changed together. Although this information does not provide actual change propagation sequence, it remain valid for change-

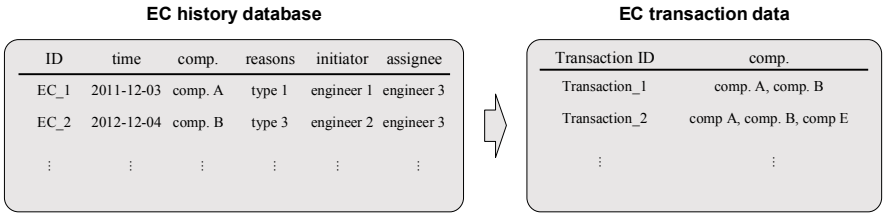


Figure 5-1. Input data for predicting change propagation

propagation prediction, because component *a*'s change will enhance the chance of component *b*'s change, or vice versa.

From the probabilistic viewpoint, the task of predicting change propagation can be viewed as the calculation of the conditional probability distribution. Suppose that we wish to calculate the probability of incurring changes to component *i* as the result of a change of component *j*. If we denote the random variables representing the states of the respective components as X_i and X_j this probability is equivalent to conditional probability distribution $P(X_i \mid X_j)$. In order to calculate such conditional probability distribution freely, what is required is to develop a complete joint probability distribution defined for a system components.

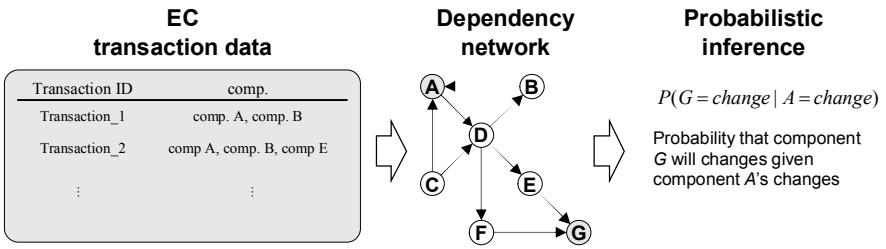


Figure 5-2. Overall framework

Dependency network provides a formal language for learning a probabilistic model from transactional data. As shown in Figure 5-2, the dependency structure represents the joint probability distribution, wherein each random variable corresponds to one of the system components. Once the dependency network is identified, it can also be used to calculate the conditional probability distribution of variables of interest. On this basis, the probability of change propagation generated by the change of any of the system components can be predicted. Throughout the remainder of this paper, we will describe and explain how the dependency network is used to learn a change propagation model from an EC database and how the resulting model can be used to predict change propagation.

5.3. Dependency Network

In this section, we show how the change propagation model can be represented by the dependency network (DN). In order to fully understand the DN, it is necessary to first introduce the Bayesian network (BN), as the DN is proposed as an approximated model of the BN.

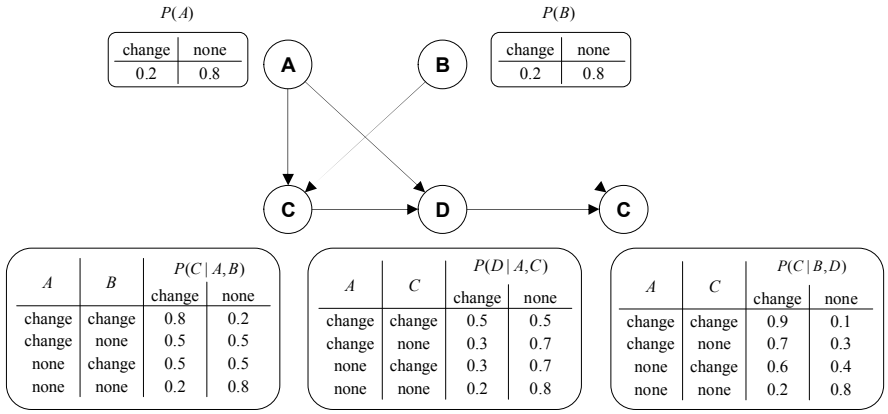


Figure 5-3. An example of BN-based change propagation model

5.3.1. BN−based Change Propagation Model

Suppose that a product consists of *n* parts. For this scenario, our basic change propagation model is equivalent to the Bayesian network just described. Consider the following analogy: A node corresponds to the random variable representing the uncertain state of each component. This is a binary random variable that takes 'change' if the corresponding component changes and 'none' otherwise. The edges connecting two nodes $X_i \rightarrow X_j$ represent the direct change propagation path from component *i* to *j*. The CPTs specify the conditional probability of change propagation given every combination of parents.

An example of this BN is illustrated in Figure 5-3. For the components (*A* and *B*) that are not affected by other components, only the prior probability is assigned. This probability can be considered as the likelihood that this component is the first to receive a change request. Conditional probability of change propagation, on the other hand, is assigned to components (*C, D* and *E*) that are affected by other compo-

nents. For example, the CPTs of component C consist of conditional probabilities given the state of its parents: A and B. As can be seen, C changes with probability 0.5 when either one of its parents changes. And note that this probability would increase to 0.8 when both A and B change.

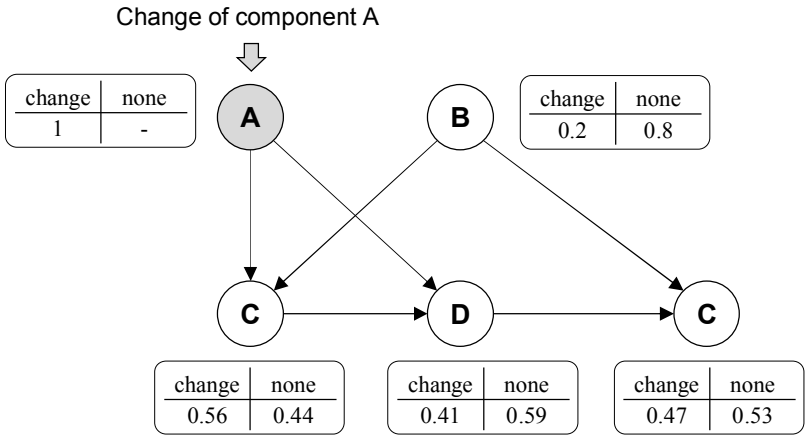


Figure 5-4. Posterior probabilities of components given change of component A

Once the BN is defined, it can be used to predict change propagation. Suppose that we wish to predict the state of a component given a change of component *A* in Figure 5-4. Since we know that component *A* has already changed, the state of X_A in the BN is set to 'change' regardless of its prior probability distribution. This change of state in the node then propagates to the rest of the network, updating the CPTs of the other nodes in the network as it goes. Figure 5-4 shows the updated network after component *A* has changed. As can be seen, the posterior probability distribution of each component represents the change propagation probability as affected by component *A*. This method of calculating the posterior probability of random variables given known values is called probabilistic inference. With the BN's efficient algorithm for probabilistic inference, the change propagation probability between each pair of components can be calculated. For full details on probabilistic inference, please refer to Darwiche (2009).

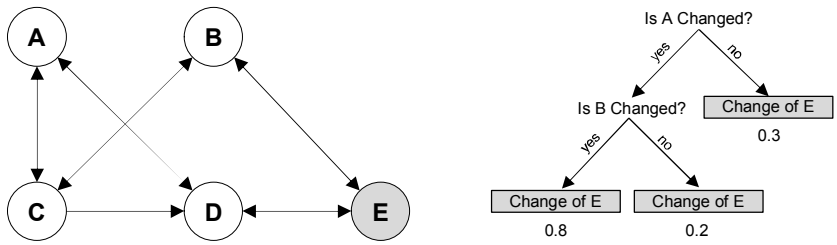
Although the BN can naturally represent change propagation, it has several disadvantages. One limitation is the complexity of the process of learning a BN from data. Learning a BN involves two steps: 1) graph searching and 2) CPT parameter estimation. Both tasks are NP-hard problems; that is, they become increasingly challenging as the number of components increases. Since a product usually consists of several hundreds of components, identifying both the graph and CPTs from such data can be computationally intractable. Another limitation is the BN's incapacity for representing a bi-directional change propagation path between components, due to its having been based on the imposition of the directed acyclic assumption on a graph. Finally, the exponential size of the CPT can lead to poor prediction performance. For each CPT, the number of parameters to be estimated increases exponentially with the number of parents. Thus, a binary random variable of n parents requires a total 2^n distinct parameters from the data. Thus, as the number of components increases, the resulting model is likely to require too many parameters, which can diminish the reliability of model. In light of all of the above limitations, an alternative model is required.

5.3.2. Learning of Change Propagation Model using Dependency Network

The Dependency Network (DN) is an alternative form of graphical model that approximates the full joint probability distribution over a set of random variables (Heckerman et al., 2001). The DN is distinguished from BN in its capacity to learn from data. The DN is constructed by learning the conditional probability distribution of each random varia-

ble $P(X_i \mid X_{pa(i)})$, which can be estimated independently from the others. This probability distribution is called a local distribution. To represent this distribution, any conditional learners, such as probabilistic decision tree, a generalized linear model, a neural network, or support vector machine, can be used. During the learning process, a feature selection method is employed to identify only statistically meaningful predictors of this variable. The graph structure of the DN, then, can be automatically inferred from the dependencies that appear within each local distribution.

Figure 5-5 shows an example of a DN of five components. As can be seen, each local distribution is independently learned from data using probabilistic decision tree. Figure 5-5b) shows an example of a decision tree of component *E* in the network. It shows that both components *B* and *D* are parents of *E*. Each leaf in the decision tree indicates a probability for *E*, along with the specified conditions of its parents. From this, we can find that *E* would change with probability 0.8 when both parents change. On the other hand, *E* changes with probability 0.2 when only *A* changes. Finally, *E* would change with probability 0.3



(a) Dependency network for A,B,C,D,E *(b)* Probabilistic decision trees for E

Figure 5-5. An example of dependency network-based change propagation model

when neither parent changes.

The DN has several advantages over the BN. First, the DN can represent the bi-directional change propagation path between components. As can be seen in Figure 5-5a), since each local distribution can be estimated independently without regard for the acyclic constraint, mutual interaction among system components can naturally be represented in the graph. Second, the process of learning the DN is straightforward and computationally efficient, because simple statistical models such as decision tree can be utilized. Therefore, it is possible to learn a complex network consisting of several hundreds of system components.

After constructing the DN, it can also be used to perform probabilistic inference. Instead of directly manipulating CPTs in the network, the DN adopts a simulation approach called Gibbs sampling, which can approximate the conditional probability distribution. In performing Gibbs sampling, a random population of each variable is repeatedly generated, from which a conditional distribution is re-sampled. Although this approach only approximates the task probability distribution, Heckerman et al. (2001) showed that Gibbs sampling produces nearly consistent values. For full details on the DN and Gibbs sampling approach, please refer to Heckerman et al. (2001). In the following sections, we describe and explain the detailed procedures by which a DN-based change propagation model is learned from data.

5.4. Proposed Approach

In this section, we illustrate overall framework for learning the DN from data. Figure 5-6 shows the framework, which consists of three

steps: data preparation, model learning, and change prediction. In the data preparation step, the raw data on the previous change records are extracted from the database, and are converted into a suitable form so as to make them DN-compatible. In the model learning step, an algorithm is utilized to learn the graph and CPTs of the DN from the data. Finally, in the change prediction step, the DN is utilized for visualization and analysis of the change propagation. In following subsections, each step will be described in more detail.

5.4.1. Data Preparation

First, the raw data is converted to transaction data. One important issue in this step is to determine an adequate level of data abstraction. If the

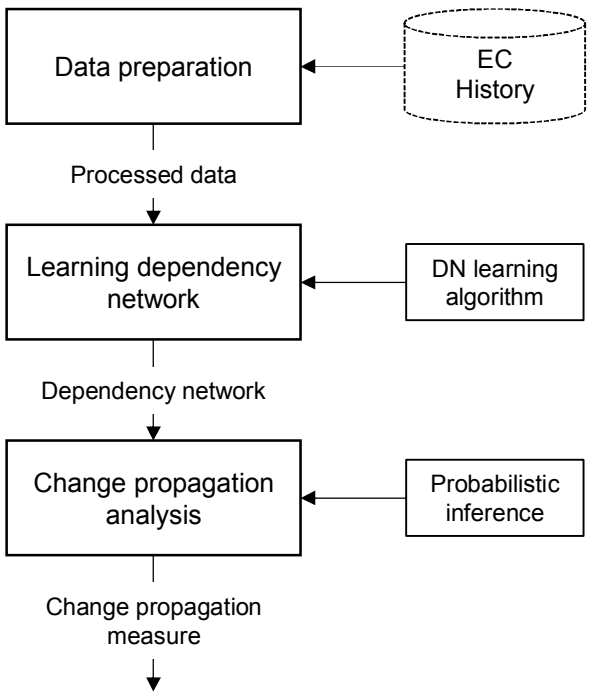


Figure 5-6. Overall framework for predicting change propagation using dependency network

data is too specific, the resulting model will become too complex to provide reliable values. Conversely, if the data is too abstract, the resulting model will not be able to provide the user with meaningful values.

The abstraction level can be affected by several factors. For example, it can be affected by the hierarchical levels of components. A component can be represented by various system hierarchy levels ranging from small parts to large subsystems. The abstraction level also can be affected by the criteria for identifying an individual transaction. These criteria can vary in different product domains. For example, in developing software, a collection of files modified by a user within one day consists of a transaction (Zimmermann et al., 2005). On the other hand, when using ECM, a series of changes triggered by a single change request also can be considered as a transaction (Katz, 1990). Considering both aspects, the analyzer should carefully determine the basic unit of change propagation.

5.4.2. Learning of Dependency Network from Data

Once the transaction data is obtained, the DN is learned from it. As noted above, in the DN, the local distribution is independently estimated for each component using numerous statistical models, for example, probabilistic decision tree, neural networks, or a logistic regression model. Among these, probabilistic decision tree is utilized for learning a local distribution. Learning a decision tree for X_i begins with a singleton root node having no children. Then, each leaf node is added to the tree with a binary split until no such replacement increases the

Bayesian score (Chickering et al., 1997). Throughout these procedures, only a subset of components that are statistically significant predictors of X_i remains in the tree. This remaining variable becomes its parents. Once the decision tree of every component is learned, the graph structure is constructed by connecting the edges from parents to children.

Using the graph structure encoded in the DN, the direct change propagation relationship between system components can be identified. Note that the edges connecting two nodes represent the statistical relationship between components. Therefore, even if two components are not physically connected, they can be connected by an edge whenever they are frequently changed together. If $X_i \rightarrow X_j$ is in the graph, X_i should be interpreted as a significant predictor of X_j. However, $X_i \leftrightarrow X_j$ should be interpreted such that either component can become a significant predictor of the other.

5.4.3. Change Propagation Analysis using DN

Once the DN is learned from the data, it can be used in a variety of change propagation analyses including visualization, and change prediction. This section explains how the probability distribution can be used in such analyses.

5.4.3.1. Prediction of Change Propagation

Although the graph can visualize the direct change propagation path between components, it cannot address the domino effect of ECs that propagate through multiple intermediate components. The DN, however, can predict which components are likely to be affected by changes.

Two measures can be used to support the identification of such components: conditional probability, and odds ratio. Conditional probability is defined as the probability that a target component changes given a change of the source component. The Gibbs sampling approach can be used to obtain this kind of probability distribution. For example, the degree of change propagation from component *j* to *i* can be represented by conditional probability $P(X_j = change \mid X_i = change)$.

Although the conditional probability distribution $P(X_j = change \mid X_i = change)$ indicates the absolute probability of target component *j*, it can be affected by components other than source component *i*. Another measure that can support change prediction analysis is odds ratio. Odds ratio is a comparative measure indicating how strongly the available evidence supports a hypothesis. To determine the unique contribution of this evidence component to a target component, the odds ratio from component *i* to *j*, odd_{ij} can be defined as

$$odds_{ij} = \frac{P(X_j = 1 \mid X_i = 1)P(X_j = 0 \mid X_i = 0)}{P(X_j = 0 \mid X_i = 1)P(X_j = 1 \mid X_i = 0)}, \qquad (5\text{-}1)$$

which indicates the ratio of the conditional probability of component *j* given the change of component *i* to the conditional probability distribution without changing component *i*. From this equation, the impacts of components other than X_i are normalized, and only the sensitivity of component *i* on component *j* remains. In this way, a user can identify the relative strengths of the change propagation relationship among components.

This approach can be distinguished from previous data-based methods, which adopt sequential pattern mining for prediction of change propagation. By these methods, which lack a supportive quantitative measure, only a small sets of sequential patterns are extracted from the database. By contrast, our mathematically rigorous probability-measuring methods can calculate the impact of the change propagation of every system component.

5.4.3.2. Calculation of Change Propagation Caused by Multiple Engineering Changes

Thus far, we have addressed the issue of change propagation caused by an individual change request, in which case, one change is considered at a given time. In practice, however, often multiple ECs arrive at the same time, due to either the bundling practice of EC boards or customer requirements for multiple simultaneous changes (Jarratt et al., 2011). Multiple changes of these sorts can amplify or cancel out the impact of change propagation to other components.

The DN can effectively address such multi-component issues, because in its performance of probabilistic inference, there is no restriction on either the number of queries or evidence terms. For example, suppose that a user wishes to predict the change propagation of component *i* given the change request of multiple components *i* and *j*. This can be easily obtained by calculating the conditional probability distribution $P(X_i \mid X_j, X_k)$. Reverse-wise, the DN also can calculate the change propagation from a single source to multiple targets.

For example, the conditional probability distribution $P(X_i, X_j \mid X_k)$ indicates the joint probability distribution of i and j given the change request of k. Additionally, combining both scenarios, the change propagation from multiple sources to multiple targets can be calculated. This method of calculation enables a user to perform more flexible analysis of change propagation.

5.5. Case Study

To demonstrate our approach, a case study of Azureus, an open-source software development project, was conducted. Azureus is a BitTorrent client that allows users to transfer, view, publish and share torrent files. After its first version was introduced in 2003, it became one of the most popular BitTorrent clients. We chose this project due to its moderate size and architectural complexity, which characteristics make it comparable with, and the present results applicable to, many other product types and studies, respectively.

EC_ID	Developer	Date	Affected source file
r40461	pgardner	2014-06-11 08:09:11	M/azureus2/src/com/aelitis/azureus/core/peermanager/unchoker/SeedingUnchoker.java
r40461	pgardner	2014-06-11 08:09:11	M/azureus2/src/com/aelitis/azureus/core/peermanager/unchoker/UnchokerUtil.java
r40461	pgardner	2014-06-11 08:09:11	M/azureus2/src/org/gudy/azureus2/core3/peer/impl/control/PEPeerControlImpl.java
r40460	pgardner	2014-06-11 05:04:20	M/azureus2/src/org/gudy/azureus2/ui/swt/win32/Win32UIEnhancer.java
r40459	pgardner	2014-06-11 05:01:33	M/azureus2/src/org/gudy/azureus2/ui/swt/win32/Win32UIEnhancer.java
r40458	amogge	2014-06-11 04:44:34	M/azureus2/src/org/gudy/azureus2/ui/swt/win32/Win32UIEnhancer.java
r40457	amogge	2014-06-11 04:41:40	M/azureus2/src/org/gudy/azureus2/ui/swt/win32/Win32UIEnhancer.java
r40456	amogge	2014-06-11 03:04:04	M/azureus3/src/com/aelitis/azureus/ui/swt/shells/main/MainWindowImpl.java
r40456	amogge	2014-06-11 03:04:04	M/azureus2/src/org/gudy/azureus2/ui/swt/Utils.java
r40453	pgardner	2014-06-07 08:58:10	M/azureus2/src/Azureus2.jardesc
⋮	⋮	⋮	⋮
r44	gudy	2003-07-10 12:41:28	M/azureus2/src/org/gudy/azureus2/ui/swt/ConfigView.java
r44	gudy	2003-07-10 12:41:28	M/azureus2/src/org/gudy/azureus2/core/GlobalManager.java
r44	gudy	2003-07-10 12:41:28	M/azureus2/src/org/gudy/azureus2/ui/swt/MainWindow.java

Figure 5-7. Change records in Azureus

5.5.1. Data Preparation

As raw data, the historical change records during the project's development period were extracted from the concurrent versions system (CVS) repository. In the open-source development environment, the participants, who are mostly volunteers, are distributed among different geographic regions. CVS aids collaboration between participants by helping them to save and retrieve different development versions of source code concurrently. When the developer finished a set of changes, he/she should check (or commit) them back into the CVS repository. Therefore, this repository contains every record of software module change. Figure 5-7 provides a snapshot of a change record. As can be seen, each change record is specified by date, developer, and affected files. We extracted change records dating from July 2003 to August 2014. In that period, there were 58,882 change records, 41 developers, and 10,940 modified files.

Table 5-1. Experiment scenarios

Scenarios	*Software hierarchy*	*Transaction criteria*	*Number of variables*
1	Source _file (*.java)	Single developer // same day	10,940
2	Package (directory)	Single developer // same day	915
3	Package (directory)	Single developer // same day // Freq. >10 times	385

This raw data was then transformed into transactional data. We grouped a set of files modified at the same time and by the same developer as a single transaction. According to the ways in which transactions are defined, we generated three experiment scenarios, which are listed in Table 5-1. In scenario 1, a change is considered to occur at the source file level. In this setting, a set of files that has been modified by a single user within one day is considered as a single transaction. As a result, int-

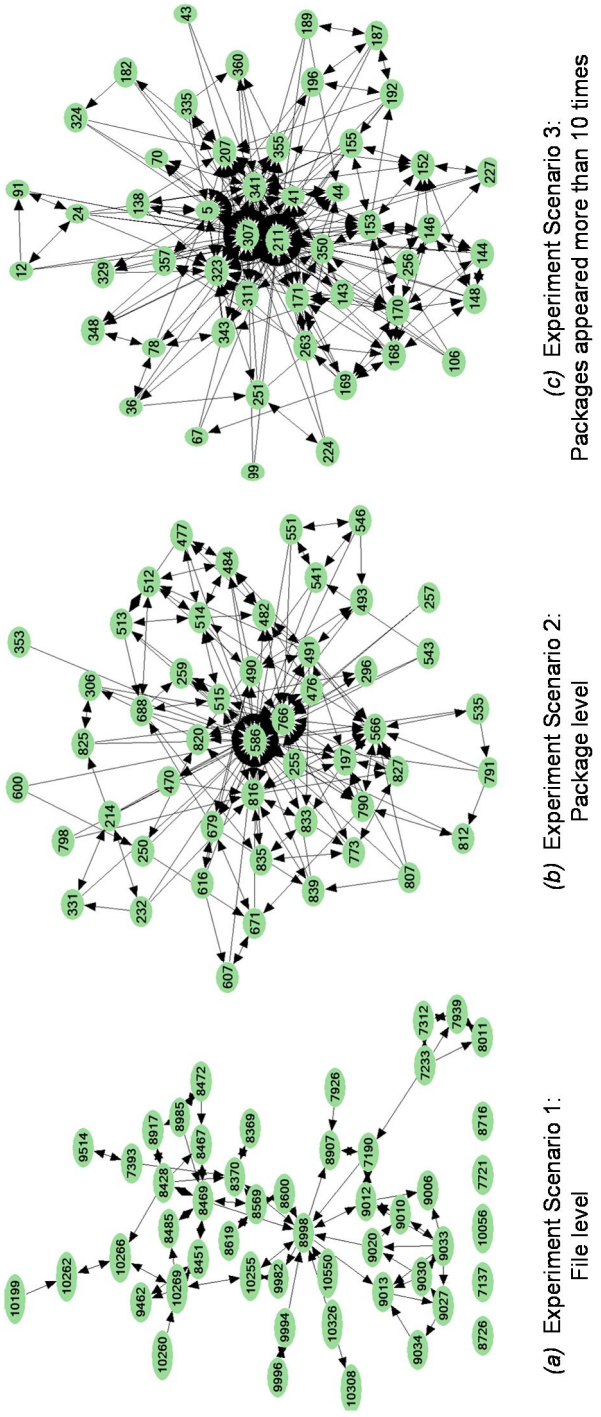


Figure 5-8. Dependency networks with respect to three scenarios in Table 5-1

eractions among a total of 10,940 files were encoded in the DN. In scenario 2, a change is considered to occur at the package level. As a collection of source files included in the same directory, a package is a higher-level software element. Therefore, the number of variables to be estimated was reduced to 915. Scenario 3, while also involving package-level data, increases the data density by eliminating packages that appear less than 10 times.

5.5.2. Learning Dependency Network

To learn the DN from data, WinMine, a tool developed by Microsoft Research, is used (Chickering, 2002). With WinMine, three different networks are derived from each scenario. The resulting network is visualized in Figure 5-8. By inspecting the graph structure, the direct predictor of each component can be identified. As can be seen, each graph shows

a different density. In scenario 1, in which the DN is learned from file-level change propagation data, only a small subset of file elements are connected by edges. On the other hand, in scenarios 2 and 3, using package-level data, denser network is obtained. This means that when performing an analysis with package-level data, more meaningful patterns can be found. The graph also provides the user with an overview of the change network. This way, especially, influential components in the network, package 586 or 766 in Figure 9b) for example, each of which is connected with large numbers of other packages and located, accordingly, at the center of the network, can easily be found. The user needs to exercise care when changing such packages, as they are likely to amplify change propagation throughout entire system. With proba-

bilistic decision tree, the quantitative relationships between software elements can be examined. Figure 5-9 shows a probabilistic decision tree example for package 616 (name) in scenario 2. As can be seen, the histograms at the leaves correspond to the probabilities of package 616 being change and not change, respectively. The structure of the branches shows the packages influential with regard to that package's change. In this case, the most influential package determining the change of 616 is 720. Note that according to the state of 720, the decision tree branches up or down. The upper branch of the tree shows the case where package 720 changes. From this, we find that package 616 changes with probability 0.982 when both package 720 and 679 have changed. Correspondingly, from the downward branch of the tree, we find that packages 669, 679, and 658 can individually affect package 616 with moderate probability. In this way, the statistical relationships between software elements can be identified without using the CPT structure.

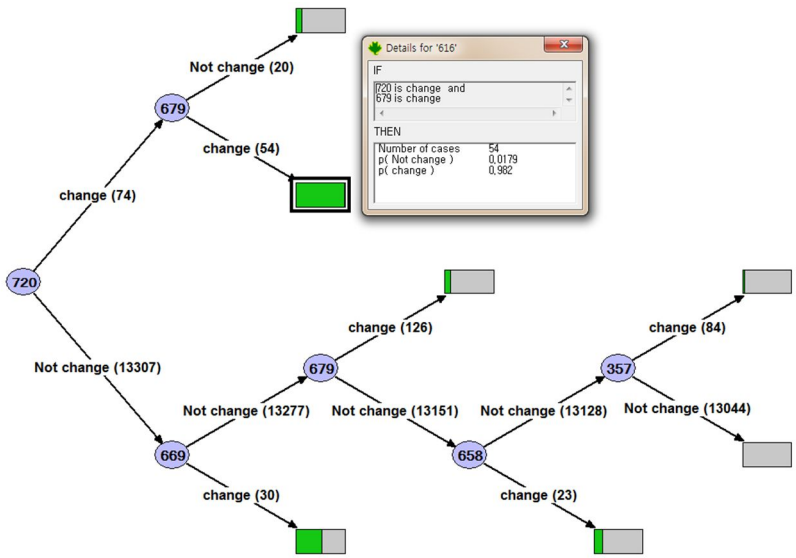


Figure 5-9. Probabilistic decision tree of package 616 in scenario 2.

The probability assigned by each branch indicates the strength of change propagation between components. Using this information,

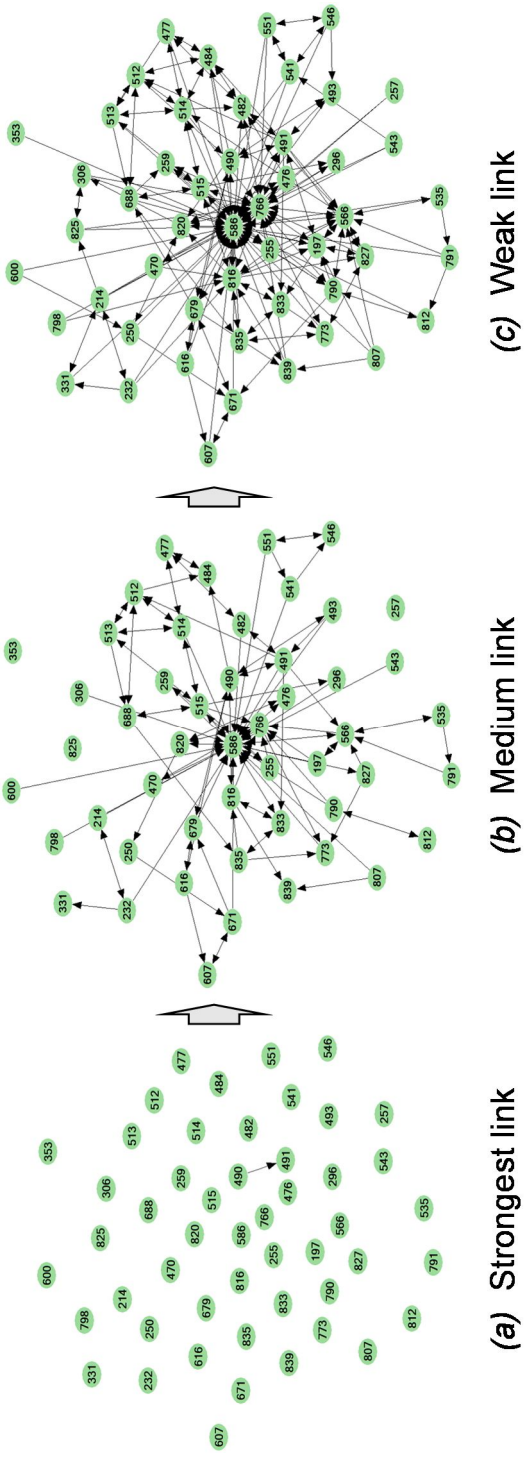


Figure 5-10. Variation of edges in graph according to the strength of probabilistic relationship be-tween components

WinMine enables a user to navigate the strength of dependency by means of the slider on the left. Figure 5-10 shows how a connection emerges according to the varying degree of strength. As can be seen, the strongest change propagation relationship in scenario 2 is that from package 490 to packages 491.

5.5.3. Prediction of change propagation

After learning the DN from the data, the DN can be used to predict the probability of change propagation in the form of a conditional probability distribution. In this study, in order to perform Gibbs sampling from the DN, the Libra toolkit (Lowd and Rooshenas, 2014) was used. Table 5-2 shows the conditional probabilities of the software packages in scenario 3, which were obtained by conditioning some of the software packages as evidence. In the leftmost part the table, the software packages

likely to be affected by the change of package 211 (‘org.gudy.azureus2.internat’) are listed in the order of their conditional probability. As can be seen, when package 211 was changed, the most affected package was 343 (‘org.gudy.azureus2.ui.swt.views.configsections’), the probability of which was approximately 11 percent. Table 5-2 also shows how these conditional probabilities change according to additional changes to packages. When package 171 (‘org.gudy.azureus2.core3.peer.impl.transport’) is added to the evidence, each package shows a significant probability increase. As can be seen, several packages show change propagation probabilities of around 30%. This indicates that when both components change togeth-

Table 5-2. Predicted changes of packages (represented by conditional probability distribution)

Given change of 211		Given change of 171, 211		Given change of 171, 211, 350	
no.	*prob.*	*no.*	*prob.*	*no.*	*prob.*
211	0	171	0	171	0
343	0.1092	211	0	350	0
341	0.1041	341	0.3745	211	0
307	0.0708	170	0.3522	341	0.3748
143	0.0678	343	0.3244	170	0.3598
323	0.0527	169	0.3212	169	0.312
207	0.0399	143	0.2982	343	0.2724
5	0.0325	168	0.2688	112	0.2597
153	0.0295	263	0.2296	113	0.2583
355	0.027	357	0.1629	168	0.2542
155	0.0216	153	0.1305	349	0.2376
251	0.0206	207	0.1019	263	0.2152
348	0.0175	355	0.1014	143	0.214
152	0.0167	235	0.0992	357	0.1581
360	0.0164	187	0.098	244	0.1304
256	0.0152	152	0.0812	355	0.1298
227	0.0139	146	0.077	153	0.1219
36	0.0126	148	0.0766	311	0.113
170	0.0125	307	0.0761	207	0.0904
146	0.0112	68	0.0718	235	0.0853

er, significant change propagation within the system is likely to be generated. When package 350 is further added to the evidence, many more components show increased probabilities.

As an alternative change propagation measure, odds ratio can be used to measure the unique contribution of each component. Specifically, the odds ratio indicates how strongly the presence of changes in a package can increase the probabilities of other packages.

Table 1 lists packages in the order of their odds ratios. The left side of the table shows the odds ratio when only package 211 changes. In this case, package 321, with an odds ratio of 9.3764, is identified as the most affected. This indicates that in the presence of change 211, the probability of package 321 increases by a factor of about 9. As can be

seen, this result also shows that the package having the highest conditional proba-

Table 5-3. Predicted change of packages (represented by odds ratio)

Given change of 211				Given change of 171,211				Given change of 171,211,350			
no.	prob (none)	prob. (change)	odd ratio	no.	prob (none)	prob. (change)	odd ratio	no.	prob (none)	prob. (change)	odd ratio
321	0.9948	0.0052	9.3764	171	0.5	0.5	43.0402	112	0.7403	0.2597	92.1205
319	0.9966	0.0034	6.5639	263	0.7704	0.2296	34.6783	113	0.7417	0.2583	84.3653
71	0.9972	0.0028	5.3777	169	0.6788	0.3212	31.945	349	0.7624	0.2376	56.694
191	0.9952	0.0048	3.8817	235	0.9008	0.0992	29.3698	171	0.5	0.5	43.0402
211	0.5	0.5	3.7866	357	0.8371	0.1629	27.3084	263	0.7848	0.2152	31.8942
215	0.9977	0.0023	3.115	168	0.7312	0.2688	26.199	169	0.688	0.312	30.6163
238	0.996	0.004	3.0992	68	0.9282	0.0718	24.7207	244	0.8696	0.1304	26.7457
343	0.8908	0.1092	3.0619	203	0.9853	0.0147	23.3548	357	0.8419	0.1581	26.3504
107	0.9915	0.0085	2.9755	170	0.6478	0.3522	16.5968	235	0.9147	0.0853	24.8896
287	0.9972	0.0028	2.8598	191	0.9806	0.0194	16.0079	168	0.7458	0.2542	24.29
183	0.9974	0.0026	2.6372	358	0.9371	0.0629	15.6834	350	0.5	0.5	23.8984
42	0.9944	0.0056	2.5646	80	0.981	0.019	15.1834	68	0.9418	0.0582	19.7465
117	0.9971	0.0029	2.397	62	0.9625	0.0375	14.649	358	0.9304	0.0696	17.461
302	0.9956	0.0044	2.3495	262	0.9793	0.0207	14.5978	170	0.6402	0.3598	17.154
143	0.9322	0.0678	2.3389	143	0.7018	0.2982	13.6731	321	0.991	0.009	16.358
330	0.9957	0.0043	2.3099	59	0.93	0.07	13.072	352	0.9168	0.0832	15.3606
300	0.9945	0.0055	2.258	51	0.9631	0.0369	12.3038	262	0.9787	0.0213	15.0246
356	0.9964	0.0036	2.2266	65	0.9713	0.0287	12.2899	62	0.9634	0.0366	14.2956
333	0.9982	0.0018	2.2008	319	0.9937	0.0063	12.0254	59	0.9256	0.0744	13.9619
267	0.9943	0.0057	2.1844	321	0.9934	0.0066	11.9955	342	0.9697	0.0303	13.5559

bability does not necessary have a high odds ratio. When package 171 is added as evidence, the odds ratio increases up to around 30, which indicates a factor of 30. When additional package 350 is added to the evidence, the odds ratio increases even up to 90. In this way, the components that are most likely to be affected by changes can be identified.

Table 5-3 lists the components that have both a high conditional probability and a high odds ratio. When modifying components, such packages must be carefully considered, because they are highly likely to be changed as the result of changes made to the source component.

5.6. Summary

Change propagation is the major source of schedule delay and cost overruns in the product development process. To mitigate the risk of change propagation is to anticipate its impact in advance. Change prediction is the scientific determination of the likelihood of component changes resulting from previous changes made to other components. Although several change prediction methods have been addressed in the literature, most of them suffer from the use of a manual data gathering process to establish a change propagation model.

As a remedy to this drawback, this paper proposes a novel data-driven approach that can automatically learn a change propagation model from a previous EC database. As a modelling language for encoding probabilistic relationships between components, the DN, a graphical model for representing joint probability distributions, is adopted in this approach. Each component is considered as a random variable, and its holistic relationship with the other components is rep-

resented by a joint probability distribution. When the data is available, the parameters within this joint probability distribution are learned from it. Once this joint probability distribution is obtained, it can be used to predict the probability of change propagation from the source component to a target component. And utilizing the probabilistic inference algorithm, the DN can calculate the probability of change propagation in the form of a conditional probability distribution. Unlike the previous methods, which can address only changes that are caused by an individual change, probabilistic inference can address changes caused simultaneously by multiple components. This approach, thereby, provides a user with more flexible means of analyzing change propagation. In order to validate our approach, a case study of Azureus, an open-source software project, was conducted. In this study, the DN was obtained from change records accumulated over the course of the previous 10 years.

Chapter 6. Conclusion and Future works

The thesis proposed methodologies for managing change propagation using Bayesian network. This chapter summarizes conclusions of this study, reveals contribution of methodologies, and discuss its future direction.

6.1. Summary of Contributions

This research develops a methodologies for managing engineering changes using Bayesian network. Although engineering change is ubiquitous concept in product development process, it is essential factors that determines the performance of the product development process. In this regards, managing engineering change has been important research issues in product development management literature. The difficulty of managing changes lies in the fact that a change does not occur alone. In complex system where each part is associated with different parts, a change made to a part may influence other parts. As a result, a change may propagate throughout the entire system. In this regards, anticipating change propagation and controlling them during product development process becomes very important.

The first contribution of the thesis is to propose a step-by-step procedure for modeling change propagation using Bayesian network. In this framework, change propagation is represented by the complex network of random variables, each of which represents the uncertain state of each component. The formalism of BN can represents complex

probabilistic relationship among components. To validate our methodology, this thesis shows that the proposed methodology has significant advantages over CPM, which is most well recognized method in this change propagation analysis.

The main advantage of using BN is the ability to incorporate expert opinion in a more flexible manner. Because BN has no restriction in defining the structure of network or specifying the CPT of each node, it is applicable to a more generic situation, which makes it a useful tool in practice. At the analysis level, BN can be used to simulate various change scenarios to help the engineering change managers. Finally, BN provides a mathematically rigorous framework for updating change propagation probabilities. Based on the Bayes' theorem, we have shown that the BN can continuously adapt the model in light of observed data.

The second contribution of this research is to propose a methodology for controlling change propagation during product development process. Design freeze is defined as the end point of the design phase at which a technical product description is handed over to production. One way of mitigating the risk of change propagation is successive freezing of components. By freezing some components earlier, company can reduce overall product development lead-time and reduce entire change propagation risk. However, early-frozen component which is transferred to production phases, also has the risk of redesign due to change propagation. Considering above trade-off involved in the timing of design freeze, careful sequencing of components are required.

In order to sequentially calculate the uncertainty of each component given freeze states of components, a BN-based model is proposed. According to the model, when a freeze decision is made with respect to a component, a probabilistic inference algorithm within the BN updates the uncertain state of each component. Finding optimal freeze sequence can be viewed as finding the optimal sequence of making evidence on root node. In this regards, a series of algorithm has been proposed for identifying optimal or near-optimal solution which can maximally mitigate the change propagation risk during product development process..

To valid the freeze sequencing methodology, our model is applied to the product data of Westland Helicopter EH101, which was originally obtained by Clarkson et al. (2004). The result show that the freeze sequence obtained from our methodology shows significant improvement over arbitrary sequence, thus proves that our methodology can be effectively and efficiently used in controlling change propagation during product development process.

The final contribution of this research is to propose a structured method for learning change prediction model from real database. One of the main limitation of current change prediction method is that it relies heavily on expert elicitation in identifying change propagation relationship between components. When the size and complexity of the product increases, identifying entire interaction among components become almost impossible. One possible alternative to this problem is to learn change propagation model from database where change related information is automatically accumulated. This information is easily obtainable from the ECM (engineering change management) system,

which computerizes engineering change (EC) work-flows to ensure that each EC is requested, notified, and executed by appropriate engineers.

Although Bayesian network provides an intuitive mechanism for learning probabilistic model from real data, it has two following limitation to be applicable in practice. First, Bayesian network cannot represent bi-directional relationship between components, thus cannot represent interaction between two components. Seconds, it is computationally too expensive to learn Bayesian network from large-scaled data, because it is NP-hard problem.

In this regard, this thesis proposes Dependency network, which is an approximated version of the Bayesian network is utilized. First, the DN can represent the bi-directional change propagation path between components. Because each local distribution in Dependency network can be estimated independently without regarding for the acyclic constraint, mutual interaction among system components can naturally be represented in the graph. Second, the process of learning the DN is straightforward and computationally efficient, because simple statistical models such as decision tree can be utilized. Therefore, it is possible to learn a complex network consisting of several hundreds of system components.

To validate our methodology, a case study of Azureus, an open-source software project, was conducted. In this study, the DN was obtained from change records accumulated over the course of the previous 10 years. This case study show that even a complex software project, where more than ten thousands of software elements are intercon-

nected with each other, Dependency network-based model can predict significant change propagation relationship among software elements.

Summarizing above results, contributions of this study are stated as follows. First, Bayesian network provides more flexible and efficient formalism for modelling change propagation, contrary to current change prediction methodologies. Second, BN-based framework can be used in entire product development lifecycle including early (change prediction), during (change control), and post (learning from data). Third, Bayesian network provides an intuitive mechanism to learn from database, thus, it is applicable to manage engineering changes in complex product.

6.2. Limitations and Future Research Directions

Until now, this thesis shows that Bayesian network can be used to improve overall change management lifecycle. However, there are some limitations and issues deserve further discussion and require future research.

Firstly, proposed methodologies should be further validated. The current BN-based model adopts stochastic viewpoints for representing a change propagation between components. Indeed, by representing change propagation relationship with probability measure which are represented by values between 0 to 1, this viewpoint allows a user to represent more complex change propagation behavior among components. However, most of engineering change management system still depends on simple binary data in representation of relationship between

components. Therefore, one of the viable future work is to implement BN-based change prediction system and evaluate its performance. By comparing numerous performance measure such as precision/recall ratio or time and cost for constructing components relationship model, the proposed model can be validated.

Another issue which is specific to freeze sequence planning is that sequential freeze process assumed in this thesis might be unrealistic in some situations. We assumed that a freeze decision is made one-by-one until every parts within the system is frozen. However, in real engineering design projects, some components, such as outsourced modules, might be uncontrollable; therefore, only a subset of components may be frozen in advance of completion. Moreover, freeze decisions might not be as frequently made as in our model. According to Prasad (1996), a freeze decision is made, at most, twice or three times during engineering design phases. Therefore, in real situations, a chunk of subsystems might be frozen concurrently, and the freeze period might be longer than in current models. Fortunately nonetheless, the BN provides a flexible model for addressing all of the aforementioned issues. To that end, more realistic case studies of real engineering projects remain as future work.

Another issue is that there are be several factors that can affect the design freeze decision. The current model considers only the risk of incurring redesign costs for the components. However, one of the important motivations of design freeze is the reduction of the overall development schedule. For example, Eastman (1980) states that a design project schedule can be reduced by by early freeze of long-lead-time

items. Combining our methods with existing project management methods might be an interesting avenue of future research.

Some possible extensions deserve further research and discussion in final contribution of the thesis. One possible extension is to include multi-state random variable in the network. In the current model, each component is allowed to have only a binary state, that is, changes (yes) or not changes (no). A more realistic model would require multi-state variables in the network. Multi-state variables, for example, can represent the different types of changes an individual component can have, such as modification (color, shape, concept), strength (high, medium, low), or rationale (regulation, requirement, previous error, etc.). By learning the probabilistic relationships among such multi-state random variables, much more complex interaction among components can be represented.

Another possible extension is to use the information from individual engineers. Individual change records contain information on who has made a change to a component. Utilizing this information, a DN among engineers can be constructed. This network could provide an analyzer with an overview of engineers' social network structure. Finally, as our DN-based model is currently applied only to the software domains, integrating it into commercial CAD systems looms as another goal of future research.

Bibliography

Acar, B. S., Benedetto-Neto, H., & Wright, I. C. (1998). Design Change: Problem or Opportunity. *In Proceedings of Engineering Design Conference*, Brunel University, UK

Bhattacharya, S., Krishnan, V., and Mahajan, V. (1998). Managing New Product Definition in Highly Dynamic Environments. *Management Science*, 44(11-2), pp. 50-64.

Bobbio, A., Portinale, L., Minichino, M., and Ciancamerla, E. (2001). Improving the Analysis of Dependable Systems by Mapping Fault Trees into Bayesian Networks. *Reliability Engineering & System Safety*, 71(3), pp. 249-260.

Casotto, A., Newton, A. R., and Sangiovanni-Vincentelli, A. (1991). Design Management based on Design Traces. *In Proceedings of The 27th ACM/IEEE Design Automation Conference* (pp. 136–141).

Cheng, H., and Chu, X. (2012). A Network-based Assessment Approach for Change Impacts on Complex Product. *Journal of Intelligent Manufacturing*, 23(4), pp. 1419-1431.

Chickering, D. M., Heckerman, D., and Meek, C. (1997). A Bayesian Approach to Learning Bayesian Networks with Local Structure. *In Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence* (pp. 80–89).

Chickering, D.M. (2002). *The WinMine Toolkit* (Tech. Rep.). Technical Report MSR-TR-2002-103, Microsoft, Redmond, WA.

Clarkson, P. J., Simons, C., and Eckert, C. (2004). Predicting Change Propagation in Complex Design. *Journal of Mechanical Design*, 126(5), pp. 788-797.

Cohen, T., Navathe, S., and Futon, R. E. (2000). C-FAR, Change Fa-

vorable Representation. *Computer-Aided Design*, 32(5), pp. 321-338.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. (2006). *Probabilistic Network and Expert Systems.* Springer.

Cowell, R. G., Verrall, R. J., and Yoon, Y. K. (2007). Modeling Operational Risk with Bayesian Networks. *Journal of Risk and Insurance*, 74(4), pp. 795-827.

Darwiche, A. (2009). *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Eastman, R. M. (1980). Engineering Information Release Prior to Final Design Freeze. *IEEE Transactions on Engineering Management*, 27(2), pp. 37-42.

Eckert, C., Clarkson, P. J., and Winfried, Z. (2004). Change And Customisastion in Complex Engineering Domains. *Research in Engineering Design*, 15(1), pp. 1-21.

Eger, T., Eckert, C. M., and Clarkson, P. J. (2005). The Role of Design Freeze in Product Development. *In 15th International Conference on Engineering Design (ICED'05)* (pp. 1-11). Melbourne, Australia, 15-18 August.

Eppinger, S.D., Whitney, D.E., Smith, R.P., and Gebala, D.A. (1994). A Model-based Methods for Organizing Tasks in Product Development. *Research in Engineering Design*, 6(1), pp. 1-13.

Giffin, M., Keller, R., Eckert, C., de Weck, O. Bounova, G., and Clarkson, P. J. (2009). Change Propagation Analysis in Complex Technical Systems. *Journal of Mechanical Design*, 131(8), pp. 1-14.

Hamraz, B., Caldwell, N. H., and Clarkson, P. J. (2013). A Holistic Categorization Framework for Literature on Engineering Change Management. *Systems Engineering*, 16(4), pp. 473-505.

Hamraz, B., Hisarciklilar, O., Rahmani, K., Wynn, D. C., Thomson, V.,

and Clarkson, P. J. (2013). Change Prediction using Interface Data. *Concurrent Engineering*, 21(2), pp. 141–154.

Heckerman, D. E., and Nathwani, B. N. (1992). An Evaluation of the Diagnostic Accuracy of Pathfinder. *Computers and Biomedical Research*, 25(1), pp. 56-74.

Heckerman, D., and Breese, J. S. (1996). Causal Independence for Probability Assessment and Inference using Bayesian Networks. Systems, *Man and Cybernetics, Part A: Systems and Humans, IEEE transactions on*, 26(6), pp. 826-831.

Heckerman, D., Breese, J. S., and Rommelse, K. (1995b). Decision-theoretic Troubleshooting. *Communications of the ACM*, 38(3), pp. 49-57.

Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R., and Kadie, C. (2001). Dependency Networks for Inference, Collaborative Filtering, and Data Visualization. *The Journal of Machine Learning Research*, 1, pp. 49–75.

Heckerman, D., Mamdani, A., and Wellman, M. P. (1995a). Real-world Applications of Bayesian networks. *Communications of the ACM,* 38(3), pp. 24-26.

Herrmann, J. W. (2010). Progressive Design Processes and Bounded Rational Designers. *Journal of Mechanical Design*, 132(8), pp. 081005-1-081105-8.

Huang, G. Q., and Mak, K. L. (1999). Current Practice of Engineering Change Management in UK Manufacturing Industries. *International Journal of Operations and Production Management*, 19(1). pp. 21-37.

Huang, Y., McMurran, R., Dhadyalla, G., and Jones, R. P. (2008). Probability based Vehicle Fault Diagnosis: Bayesian Network Method. *Journal of Intelligent Manufacturing*, 19(3), pp. 301-311.

Huchzermeier, A., and Loch, C. H. (2001). Project Management under Risk: Using the Real Options Approach to Evaluate Flexibility in R&D. *Management Science*, 47(1), pp. 85-101.

Jarratt, T. A. W., Eckert, C. M., Caldwell, N. H. M., and Clarkson, P. J. (2011). Engineering Change: An Overview and Perspective on the Literature. *Research in Engineering Design*, 22(2), pp. 103-124.

Jensen, F. V. (1996). *An Introduction to Bayesian Networks*. Springer.

Johnson, D. S., and McGeoch, L. A. (1997). The Traveling Salesman Problem: A Case Study in Local Optimization. *Local Search in Combinatorial Optimization*, 1, pp. 215-310.

Kang, C. M., and Hong, Y. S. (2009). Evaluation of Acceleration Effect of Dynamic Sequencing of Design Process in a Multi Project Environment. *Journal of Mechanical Design*, 131(2). pp. 021008.1-021008.11.

Katz, R. H. (1990). Toward a Unified Framework for Version Modeling in Engineering Databases. *ACM Computing Surveys (CSUR)*, 22(4), pp. 375–409.

Keller, R., Clarkson, P. J., and Eckert, C. M. (2008). Determining Component Freeze Order: A Redesign Cost Perspective using Simulated Annealing. *In ASME 2008 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2008)* (pp. 1-10). Brooklyn, New York, 3-6 August.

Keller, R., Eger, T., Eckert, C. M., and Clarkson, P. J. (2005). Visualising Change Propagation. *In 15th International Conference on Engineering Design (ICED'05)* (pp. 280-291). Melbourne, Australia, 15-18 August.

Kjaerulff, U. B., and Madsen, A. L. (2012). *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis.*

Springer.

Kocar, V., and Akgunduz, A. (2010). ADVICE: A Virtual Environment for Engineering Change Management. *Computers in Industry*, 61(1), pp. 15–28.

Koh, E. C. Y., Caldwell, N. H. M., and Clarkson, P. J. (2012). A Method to Assess the Effects of Engineering Change Propagation. *Research in Engineering Design*, 23(4), pp. 329-351.

Krishnan, V., Eppinger, S. D. and Whitney, D. E. (1997). A Model-based Framework to Overlap Product Development Activities. *Management Science*, 43(4), pp. 437-451.

Lee, J., H., and Hong, Y., S. (2015). A Bayesian Network Approach to Improve Change Propagation Analysis, *In Proceedings of 20th International Conference on Engineering Design (ICED15)*, Forthcoming.

Li, Z., Harman, M., and Hierons, R. M. (2007). Search Algorithm for Regression Test Case Prioritization. *IEEE Transactions on Software Engineering*, 33(4), pp. 225-237.

Loch, C., and Terwiesch, C. (1998). Communication and Uncertainty in Concurrent Engineering. *Management Science*, 44(8), pp. 1032–1048

Lowd, D., and Rooshenas, A. (2014). *User manual for libra 1.0* (Tech. Rep.).

Maier, J. F., Wynn, D. C., Biedermann, W., Lindemann, U., and Clarkson, P. J. (2014). Simulating Progressive Iteration, Rework and Change Propagation to Prioritise Design Tasks. *Research in Engineering Design*, 25(4), pp. 283-307

Matthews, P. C. (2011). Challenges to Bayesian Decision Support Using Morphological Matrices for Design: Empirical Evidence. *Research in Engineering Design*, 22(1), pp. 29-42.

Matthieu, G., François, P., and Tchangani, A. (2012). Optimising End-

Of-Life System Dismantling Strategy. *International Journal of Production Research*, 50(14), pp. 3738-3754.

Mirarab, S., and Tahvildari, L. (2007). A Prioritization Approach for Software Test Cases based on Bayesian Networks. *Fundamental Approaches to Software Engineering (pp. 276-290)*. Springer.

Mookerjee, V. S.,and Mannino, M. V. (1997). Sequential Decision Models for Expert System Optimization. *IEEE Transactions on Knowledge and Data Engineering*, 9(5), pp. 675-687.

Morkos, B., Mathieson, J., and Summers, J. D. (2014). Comparative Analysis of Requirements Change Prediction Models: Manual, Linguistic, and Neural Network. *Research in Engineering Design*, 25(2), pp. 139-156.

Moullec, M. L., Bouissou, M., Jankovic, M., Bocquet, J. C., Réquillard, F., Maas, O., and Forgeot, O. (2013). Toward System Architecture Generation and Performances Assessment Under Uncertainty Using Bayesian Networks. *Journal of Mechanical Design*, 135(4), pp. 041002.

Murphy, K. P. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. Doctoral Dissertation, University of California.

Mussi, S. (2002). Sequential Decision-Theoretic Models and Expert Systems. *Expert Systems*, 19(2), pp. 99-108.

O'Hagan, A., Buck, C. E., Daneshkhah, A., Garthwaite, P. H., Jenkinson, D. J., Oakley, J. E., and Rakow, T. (2006). *Uncertain Judgments: Eliciting Experts' Probabilities*. John Wiley & Sons.

Oh, S., Park, B., Park, S., and Hong, Y. S. (2007). Design of change-absorbing system architecture for the design of robust products and services. *Human-Computer Interaction. HCI Applications and Services* (pp. 1110-1119). Springer.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Net-*

work of Plausible Inference. Morgan Kaufmann Publishers.

Prasad, B. (1996). *Concurrent Engineering Fundamentals: Integrated Product and Process Organization, Volume I*. Prentice Hall.

Rouibah, K., & Caskey, K. R. (2003). Change Management in Concurrent Engineering from a Parameter Perspective. *Computers in Industry*, 50(1), pp. 15-34.

Shahan, D. W., and Seepersad, C. C. (2012). Bayesian Network Classifiers for Set-Based Collaborative Design. *Journal of Mechanical Design*, 134(7), 071001.

Skaanning, C., Jensen, F. V., and Kjærulff, U. (2000). Printer Troubleshooting using Bayesian Networks. *Intelligent Problem Solving. Methodologies and Approaches* (pp. 367-380). Springer.

Terwiesch, C., and Loch, C. H. (1999). Managing the process of engineering change orders: the case of the climate control system development. *Journal of Product Innovation Management*, 16(2): pp. 160-172.

Thomke, S.H. (1997). The Role of Flexibility in the Development of New Products: An Empirical Study. *Research policy*, 26(1), pp. 105-119

Vomlel, J. (2004). Bayesian Networks in Educational Testing. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12(1), pp. 83-100.

Wright, I. C. (1997). A Review of Research into Engineering Change Management: Implications for Product Design. *Design Studies*, 18(1), pp. 33-42.

Wynn, D. C., Caldwell, N. H., and Clarkson, P. J. (2010). Can Change Prediction Help Prioritise Redesign Work in Future Engineering Systems?. *In Proceedings of DESIGN 2010, the 11th International Design Conference*, Dubrovnik, Croatia.

Yang, F., and Duan, G. J. (2012). Developing a Parameter Linkage-

Based Method for Searching Change Propagation Paths. *Research in Engineering Design*, 23(4), pp. 353-372.

Zimmermann, T., Zeller, A., Weissgerber, P., and Diehl, S. (2005). Mining Version Histories to Guide Software Changes. *IEEE Transactions on Software Engineering*, 31(6), pp. 429–445.

Zirger, B. J., and Hartley, J. L. (1996). The effect of acceleration techniques on product development time. *IEEE Transactions on Engineering Management*, 43(2), pp. 143-152.

국문 초록

설계변경이란 제품개발 과정에서 발생하는 부품의 형태, 재료, 기능의 변화를 의미한다. 설계변경은 제품개발 프로젝트에서 일상적으로 일어나는 과정이자, 제품개발 비용, 일정, 품질을 결정하는 중요한 요인이다. 설계변경을 관리함에 있어 가장 어려운 점 중 하나는, 한 부품의 설계변경이 다른 부품을 연쇄적으로 변경시키며, 제품 전반으로 퍼져나가는 파급성을 지닌다는 데 있다. 설계변경 파급을 예측하고 이를 관리하지 못할 경우, 제품개발 비용을 폭발적으로 증가시킬 수 있다. 하지만, 제품개발 과정에서 설계변경을 완전히 제거 할 수 없으므로, 설계변경파급을 관리하기 위한 유일한 대안은 이를 사전에 예측하고 관리하는 것이다.

본 논문에서는 설계변경 및 파급효과를 효과적으로 관리하기 위한 방법론으로써 베이지안 네트워크의 활용을 제안한다. 베이지안 네트워크는 확률변수간의 관계를 분석하고 추론하기 위한 표현체계로써, 설계변경파급 과정을 불확실성에 기반한 확률적 사건으로 바라본다. 이러한 확률적 관점이 필요한 이유는, 제품이 소프트웨어를 포함한 복잡계 시스템으로 변화하면서 부품간의 관계가 불확실하고 모호해지기 때문이다. 베이지안 네트워크는 부품간의 확률적 설계변경 계를 그래프 모형으로 표현하고, 이들간의 관계를 쉽게 추론할 수 있도록 도와줌으로써, 불확실하고 복잡한 설계파급과정을 예측하고 관리할 수 있도록 도와준다.

본 논문은 크게 세 부분으로 구성되어 있다. 첫 번째 부분은 설계변경의 파급효과를 사전에 예측하는 방법론을 개발하는 것이다. 기존 설계변경 예측 방법론은 부품간의 관계 표현, 데이터 수집, 그리고 분석측면에서 단순하고 평면적인 제품만을 다룰 수 있다는 단점이 있었다. 본 연구에서는 설계변경 파급과정을 예측하기 위한 베이지안 네트워크로 모형을 소개하고, 해당 모형이 설계변경 파급 현상의 표현 및 분석 측면에서 어떠한 장점이 있는지 다룬다. 표현 측면에서 베이지안 네트워크는 부품 사이의 계층적이고 복잡한 확률적 관계를 쉽게 표현할 수 있으며, 분석 측면에서는 확률 추론알고리즘을 이용해 기존 방법이 분석하지 제공하지 못하는 다양한 설계변경파급 시나리오의 분석을 수행할 수 있게 된다.

두 번째 부분은 부품의 동결순서의 결정을 통해 프로젝트 수행 중 발생하는 설계변경파급의 위험(risk)를 조절하는 방법론을 제안한다. 제품개발 과정이 어느 정도 진행되고 나면, 설계변경을 더 이상 수용하지 않는 동결시점이 존재한다, 보통, 이러한 동결시점은 부품마다 다르며, 대부분 순차적인

동결과정을 취하게 된다. 만약 부품의 동결상태에 따라 전체 부품에 남아있는 설계변경 파급확률의 위험을 측정할 수 있다면, 이 정보를 바탕으로 효율적인 설계변경 동결과정의 실행계획을 세울 수 있게 된다. 해당 부분에서는 설계변경 파급과정을 베이지안 네트워크 모형으로 표현하고, 이를 활용해 최적의 설계동결 순서를 찾을 수 있는 알고리즘을 제안하고 이를 실제 제품개발 데이터에 적용하였다.

세 번째 부분은 설계변경 모형의 학습 및 데이터 수집을 위한 관리체계를 제안한다. 기존의 설계변경 관리 방법론의 가장 큰 한계 중 하나는, 부품간의 관계를 측정함에 있어 전문가의 주관적 의견에 의존한다는 것이다. 이러한 전문가 의존방식은 부품의 수가 많아지고 연관관계가 복잡해짐에 따라 효율적인 예측모형의 구축이 어렵다는 한계가 있다. 본 장에서는, 대다수 기업이 보유하고 있는 PLM(Product Lifecycle Management) 시스템에 남아있는 설계변경 이력 데이터를 자동으로 학습하여 부품간의 설계변경 파급 확률을 자동적으로 학습하고 관리할 수 있게 된다. 모형의 학습 및 예측을 위한 방법론으로써, 베이지안 네트워크의 근사모형인 Dependency Network 가 활용되었다. 제안한 방법론을 소프트웨어 설계변경 이력 데이터에 적용해 봄으로써 제안한 방법론의 효율성과 효과성을 검증하였다.

핵심어: 설계변경관리, 베이지안 네트워크, 설계변경파급, 설계변경예측, 디자인동결순서, 디펜던시 네트워크, 설계이력 학습
학 번: **2008-21237**