PROCESS-ORIENTED KNOWLEDGE DISCOVERY TO SUPPORT PRODUCT DESIGN USING TEXT MINING

LAN LIJUN

(*B.Sc., M.Sc., CQU*)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2017

Supervisor:
Associate Professor Lu Wen Feng

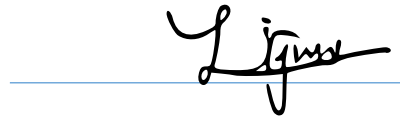
Examiners:
Associate Professor Tay Eng Hock
Associate Professor Hong Geok Soon
Professor Ashutosh Tiwari, Canfield University

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety.

I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.



LAN LIJUN

13 January 2017

ACKNOWLEDGEMENTS

I would first like to express my great appreciation and gratitude to my supervisor Prof. Lu Wen Feng for his continuous support of my PhD study and related research, for his patient, motivation, and encouragement. I would also like to thank my former supervisor Prof. Liu Ying. The research would not have been at this level of quality and completeness without his valuable guidance in the first year of my PhD study.

Besides my advisors, I also appreciate the valuable feedback from all the committee members, for their insightful comments, critical suggestions, and constructive discussions, which inspired me to widen my research from various perspectives.

Many thanks to my seniors, particularly, Dr. Wang Sibao, Dr. Liu Ning, Dr. Wang Pengfei, and Dr. Hu Huicong, for the stimulating informal discussions, for the continuous help and support, and for all the fun we have had in the four years.

My sincere thanks also goes to my friends who always took the time to listen to my intimate feelings even complaints. Special thanks to Ms. Zhao Xiaohong, Ms. Chen Huling and Mr. Du Dongyang because they have been like a second family to me in Singapore.

Last but not the least, I would like to thank my family. You are always there for me.

TABLE OF CONTENTS

SUMMARY

Design documents and design project footprints accumulated by corporation IT systems have increasingly become valuable sources of evidence for design information and knowledge management. Identification and extraction of the embedded information and knowledge into a clear and usable format will greatly accelerate continuous learning from past design efforts for competitive product innovation and efficient design process management in future design projects. To efficiently reuse the embedded design information, most of the existing design information extraction systems focus on either organizing design documents for efficient document retrieval or extracting relevant product information for product optimization. This study seeks to further extract design information deep into the document content, with a focus on process-oriented design knowledge. For this purpose, a process-oriented knowledge discovery system for extracting process relevant knowledge from archival design documents is developed, and three subjects are investigated.

Firstly, considering the design documents generated during design processes are mostly unstructured, unlabeled, and textual data, two information extraction approaches have been investigated to extract process-oriented information from the design documents with minimal prior knowledge. Unlike most existing methods which heavily rely on a large quantity of training data, the two presented approaches extract coarse-grained information at the document level in an unsupervised manner and fine-grained information at the sentence level in a semi-supervised manner respectively. Experimental results indicate that

the extracted information could help decision makers to get a fast and brief understanding of the underlying design process.

Secondly, as design processes are usually flexible and iterative, two process mining approaches, i.e., bottom-up process mining and top-down process mining, are proposed to discover the underlying design process model in a hierarchical and modular representation. Different from conventional process mining techniques that aim to capture all the process behaviors in a flat model, the outcome of the two proposed process mining approaches is a hierarchical process model that provides different degrees of details at different abstraction levels. To discover such a hierarchical process model, the two process mining approaches work in two opposite directions, with one from specification to generation and the other from generation to specification. Experimental results indicate that the hierarchical process models have a good reflection of the reality, and outperform the flat models in capturing the flexibility of the underlying design process.

Thirdly, as the process model discovered only reflects the workflow aspect of the underlying design process, the process model is further refined to distill multi-faceted knowledge patterns by applying a number of statistical analysis methods. The outcomes range from cooperation patterns from social net analysis, functional and organizational roles from role mining, and irregular task executions from temporal behavior analysis. Results show that the extracted knowledge patterns include not only knowledge the interviewed expert had already known but also hidden knowledge he was unaware.

LIST OF TABLES

LIST OF FIGURES

LIST OF ABERRATIONS

BIO	Beginning, Inside, and Outside
BMDTK	Best Match Based Dependency Tree Kernel
DBN	Deep Belief Network
DP	Dependency Tree
ERE	Entity Relation Detection
HCA	Hierarchical Clustering Analysis
HMM	Hidden Markov Model
IE	Information Extraction
KD	Knowledge Discovery
KDD	Knowledge Discovery in Database
KNN	k-Nearest Neighbors
LDA	Latent Dirichlet Allocation
ML	Machine Learning
NE	Named Entity
NLP	Nature Language Processing

NN	Neural Networks
NER	Named Entity Recognition
PDP	Product Design/Development Process
PKDD	Process Knowledge Discovery in Texts/Documents
PM	Process Mining
POS	Part-Of-Speech
RBM	Restricted Boltzmann Machines
SVM	Support Vector Machine
TP	Topic Modeling
XML	Extensible Markup Language

CHAPTER 1 INTRODUCTION

1.1 Background

In today's modern manufacturing environment, globalization, unpredictable markets, increasing customer requirements, and the pursuit for competitive advantages are some of the main challenges that are pushing enterprises to move products quickly from concept to market [1]. These challenges can trigger more intractable problems such as causing frequent changes in product design, increasing the complexity of both products and product development processes, and raising the product development cost. The ability of efficiently solving the above problems has become a pre-requisite for designing a successful product.

One crucial key to solve the above problems is a good product design process that is well supported with immense amounts of valuable and available design information. In general, a product design process is a sequence of activities, involving from concept through realization, to turn ideas into products [2]. Its nature is often viewed as an information and knowledge intensive process. This nature is presented from two aspects. On one hand, product design processes heavily rely on the personal knowledge of the designers. It is estimated that throughout a design process, designers spend 20-30% of their time searching for information and another 20-30% for handling information [3]. On the other hand, with the wide spread of computer-supported systems, a large volume of digital design data has been accumulated at various stages of design processes. Examples of such design data include CAD models, regular progress reports, claims, configuration files, email, and chat

transcripts. Invaluable design information, patterns, and knowledge are embedded in such documents.

To efficiently reuse the embedded design information for supporting product design, most of the existing design information extraction systems mainly focus on either organizing design documents for efficient document retrieval or extracting relevant product information for product optimization. Although these systems indeed can reduce the time spent in seeking for useful design information, human efforts are still required to locate and understand the retrieved design information. In this context, identifying and extracting the embedded information and knowledge into a clear and usable format would greatly accelerate continuous learning from past design efforts for competitive product innovation and efficient design process management in future design projects.

This study seeks to further extract design information deep into the document content, with a focus on process-oriented design knowledge. For this purpose, a process-oriented knowledge discovery system is developed. The developed system aims to automatically discover multi-faceted design process knowledge such as design process model, social network of the involved people, and resource utilization, from design documents archived in existing or old product design processes. The discovered design process knowledge is well organized in an understandable and interpretable manner, so as to help decision makers to quickly get right information at the right time.

Before describing the motivations of this research in detail, it is worth explaining two significant concepts involved in this thesis: knowledge discovery and process mining.

1.1.1 Knowledge Discovery

Knowledge discovery is methodically similar to information extraction and data mining, but goes beyond information extraction and data mining. It requires not only extracting useful patterns from structured or unstructured source data but also understanding and reusing the extracted patterns.

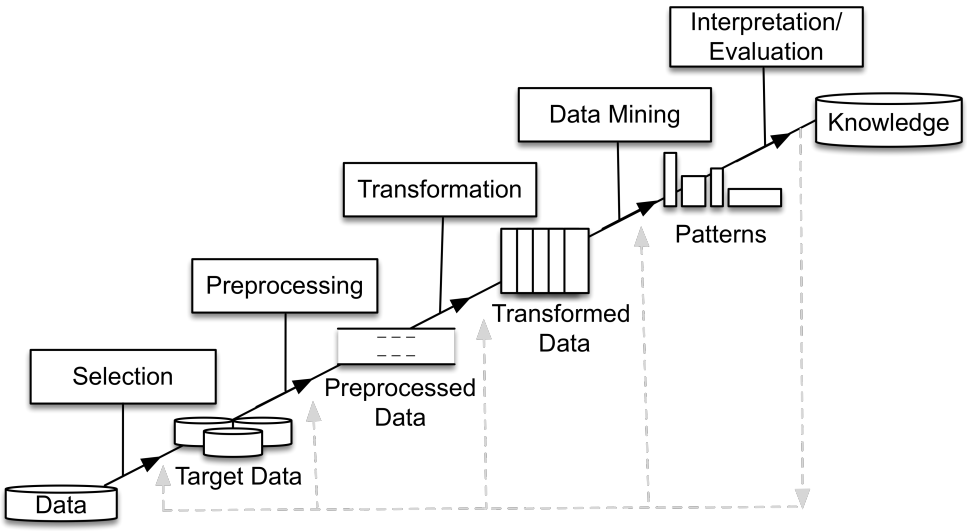


Figure 1.1 Knowledge discovery in databases [4]

Figure 1.1 illustrates the most classical framework of knowledge discovery in databases (KDD) [4]. There are five main steps: selecting the target data from the given dataset, preprocessing the target data, transforming data into operable formats, extracting hidden patterns, and interpreting the extracted patterns. Referring to Fig. 1.1, the starting point of KDD is the raw data, which are usually unrefined and unfiltered information. After some forms of processing, the raw data evolve into information, e.g., relations and patterns

detected from the source data. At last, knowledge is a high-value form of information that is understandable and ready for use in decision making [5, 6]. For example, if the traffic light is the considered data, the information should be “the light is red”, while the knowledge should be “to stop”.

1.1.2 Process Mining

Process mining is the task of using specialized data mining algorithms to automatically extract business workflows from event logs recorded by information systems [7]. Figure 1.2 gives an example of process mining. In Fig. 1.2, event logs are execution records generated by information systems. The resulting model can be seen as a special type of knowledge that identifies workflow sequences among the activities recorded in the event logs. Extraction of such a model can provide decision makers with great help in obtaining deep insight into control flows, organizational structures, and resource utilization, especially when no formal description of the process can be obtained by other approaches or when the quality of an existing model is questionable.

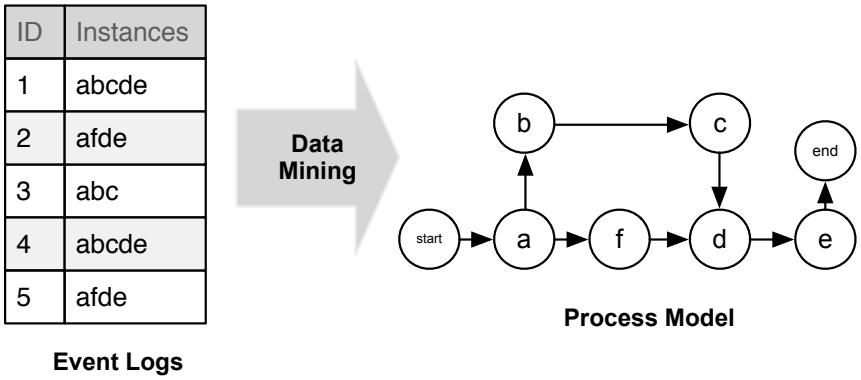


Figure 1.2 Process mining

1.2 Motivations

It has been widely realized that extracting useful information and knowledge from the archival design documents could continuously support decision making in a new design project. For this purpose, most of the existing applications of information extraction in product design mainly focused on discovering technology trend from patents [8-10], retrieving and reutilizing product design on the basis of CAD files [11-13], and mining customer opinions from online customer reviews [14, 15]. One common point of the three applications is that they are trying to improve product quality by putting focus on product itself. For example, by finding and reusing the best matching old product from the existing CAD files, a new product can be modified to better fit the customer requirements. Although such design knowledge reuse systems can be greatly helpful in supporting detailed design, they are not compatible to the entire product design process.

To the best of the author's knowledge, this is no previous study on extracting design process information and knowledge from design documents accumulated during design processes. Different from product-relevant design information like CAD models, design process information such as design process models, social network of the designers, and organizational structures, are of great value to decision makers in improving design process management throughout the entire lifecycle of the product design process. Typical benefits of process information include tracking root causes of delayed design activities, optimizing resource utilization and resource allocation based on the the historical performance of the

designers, and optimizing process planning based on the historical task dependencies. However, such types of design information have been much ignored in the existing studies.

In the domain of intelligent business management, process mining has become a popular tool for business process analysis and improvement, such as conformance checking [16-19], bottleneck detection [20], and decision support [21, 22]. However, existing process mining techniques can not be directly transplanted to mine design process models from design documents. There are two significant reasons caused by the nature of both design data and product design process.

The first reason is the textual nature of product design data. In contrary to the numeric datasets of business processes, 80% of design process data is semi-structured or unstructured texts [23], e.g., regular progress reports, claims, configuration files, emails and chat transcripts. However, the input data for most process mining approaches is restricted to event logs and relation database data, which are totally structured. That is to say, there is an irreconcilable conflict between product design data and the data required by traditional process mining approaches. To overcome this conflict calls for intelligent techniques that can not only extract design process information such as people, time, and tools, from design documents, but also encode the extracted information into a well-defined structure like event logs.

The second reason is the difference between business processes and product design processes: the former is formal and repetitive, while the latter is unpredictable and iterative

[24]. In practice, designers usually produce in advance some sequences of basic design activities that are supposed to reach some goals. However, as the product design process progresses, changes caused by new information and new ideas require designers to take new actions to solve some new problems. This nature of product design processes causes the design tasks have nonlinear and complex correlations. This makes traditional process mining approaches not suitable to model the behavior of product design processes because most of them try to model the process behavior in a flat and linear model. To solve this problem calls for specialized process mining approaches that is able to capture the flexible and iterative nature of product design processes, for example, modeling product design processes in a hierarchical manner by providing appropriate abstractions.

Motivated by the growing need of and the lack of techniques for reutilizing the stored design data for product design process improvement, this thesis aims to develop a design process knowledge discovery system. The proposed system could automatically discover multi-faceted design process knowledge such as design process model, social network of the designers, and resource utilization, from design documents archived in existing or old product design processes.

1.3 Research Objectives and Scope

As mentioned above, the aim of this research is to develop a knowledge discovery system to extract much-ignored process knowledge from design documents. Due to the

characteristics of design documents and the nature of product design processes, the proposed system must achieve the following three sub-goals.

I. Process information extraction – to extract process relevant information from the design documents. The extracted information include topics discussed in the documents and physical objects such as people, organizations, locations, and tools, which have been mentioned in the design documents. To address this problem, two information extraction approaches, topic modeling and Named Entity Recognition (NER), would be investigated. The former aims to extract topics at the document level in an unsupervised manner, and the latter aims to identify fine-grained information at the sentence level in a semi-supervised manner.

II. Process mining – to discover product design process models that are able to capture the flexibility of the product design process. To solve this problem, Entity Relation Extraction (ERE) approaches would be studied to extract design events recorded in design documents by finding the higher-order relations among process-relevant entities. In addition, in order to capture the flexible structure of product design processes, hierarchical process mining approaches would be investigated to model design processes with different levels of abstraction.

III. Process knowledge interpretation – to enhance and interpret multi-faceted design knowledge on the basis of the discovered process model and to reuse the discovered knowledge in decision making. Two significant aspects of product design process

will be analyzed to illustrate what valuable design knowledge could be distilled from the design process model discovered. They are organizational mining and temporal behavior analysis. The former focuses on the people involved in the underlying design process by discovering their social network, functional roles, and organizational structures. The latter focuses on the temporal behaviors of both the detected design tasks and the involved people. In addition, process knowledge visualization is also stressed at this stage, with the goal of organizing the discovered information in an accessible, understandable, and interpretable manner.

The discovered process knowledge is expected to help designers, even novices, to 1) recognize unknown problems of existing product design processes (know-what), 2) understand the cause of problems (know-why), and 3) get the capacity to act quickly and correctly in the design process of a new product (know-how). Such abilities would support and facilitate product design in many situations. For example, before a manager making decision to assign a person to a new design task, he may want to know 1) what tasks this person have been in charge of in the existing projects, and 2) how this person have performed in his previous tasks. In another situation, if a person is assigned a new task, he may want to know what experience he can learn from previous tasks so as to improve the new task. With the help of the discovered process knowledge, such concerns could be easily resolved, thus reducing the time-to-market.

1.4 Organization

The dissertation is presented in nine Chapters. Chapter 2 reviews the state of the art related to each research topic. Chapter 3 gives an overview of the methods integrated in the developed system of discovering process knowledge from design texts (PKDT). Figure 1.3 shows the organization structure of the core steps of the proposed PKDT system. At the upstream stage of the PKDT system, Chapters 4 and 5 address the design information extraction problems. Chapters 6 and 7 address the process mining problems, which is the second major stage in the PKDT system. Chapter 8 addresses the knowledge interpretation and visualization problems via multi-faceted process analysis, which forms the downstream stage. Finally, Chapter 9 gives the conclusions as well as the future works.

Chapter:	Chapter 4	Chapter 5	Chapter 6	Chapter 7	Chapter 8
KDD Procedure:	Transformation		Data Mining		Interpretation
PKDT Procedure:	Coarse-grained Information Extraction	Fine-grained Information Extraction	Event Detection	Process Modeling	Multi-faceted Process Knowledge Discovery
Approach:	DBN-based Topic Modeling	A Hybrid NER	Graph Partition Based Higher-order ERE	Bottom-Up Mining, Top-Down Mining	Organizational Mining, Temporal Behaviour Analysis
Keyword:	Topic Modeling	NER	ERE, Graph Partition	ERE, Process Modeling	Social Network, Statistic

Figure 1.3 Thesis organization

CHAPTER 2 LITERATURE REVIEW

2.1 Text Mining in Product Design

Text mining has its roots in linguistics and data mining, but extends the data mining workbenches by extracting high-quality information in form of patterns and terms from texts [25, 26]. The typical techniques include Naïve Bayes [27], support vector machine (SVM) [28], Conditional Random Fields [29], and decision tree [30]. In product design, although extracting and utilizing design information from design documents via text mining techniques has long been recognized, high-quality design information extraction is still a big challenge. One major reason is that design processes are often ill-structured and ad hoc, and vary greatly due to uncertainty [3].

Based on the type of design information that produced, typical applications of text mining techniques in design document analysis can be classified into three categories: document retrieval, document classification, and information extraction. Document/information retrieval approaches [8, 11, 31, 32] allow end-users to retrieve their interesting design documents/information from large indexed document collections, using keyword matching techniques. Document classification approaches [33-38] focus on providing more efficient design document organization by clustering documents into predefined categories, using unsupervised (e.g., K-means and hierarchical clustering algorithm) or supervised (e.g., SVM, neural networks and decision trees) approaches based on document content or labeled training samples. Unlike previous two approaches that

processing design documents at the document level, information extraction aims to extract domain-specific information, e.g., design rationales [39], task topics [40] and structural models [41, 42], from text fragments at the semantic level. In most cases, this task requires processing natural language texts by a combination of techniques, such as natural language processing, data mining, machine learning, and probability statistics.

Because of the ability of locating highly relevant information, information extraction is gaining increasing popularity in design document processing. Among present works, the largest proportion focus on patent processing for technology trend analysis and design innovation inspiration. For example, in order to collect design rationales from patent documents for a engineering design purpose, a 3-layers ontology model, ISAL (issue, solution and artifact), is automatically built using a series of text mining algorithms [39]. With a similar purpose of market-driven technology innovation, potential product concepts of solar-lighting devices are identified from a collection of domain-specific patents [43]. Focusing on the patent claim parsing, a data-driven parser is proposed to identify the scope of intellectual property protection [10]. Besides patent processing, discovering customer opinions from website data, e.g., online customer requirements and reviews, is another leading stream for design information extraction. For instance, based on the concurrent information between keywords, customer reviews are automatically translated into engineering characteristics for Quality Function Deployment [15].

The literature review indicates that compared to patent documents and website data, other types of design text, e.g., repair verbatim [44], production configuration data [45] and social media data [40], have obtained much fewer attention for design information extraction. Furthermore, most of the design documents accumulated during design processes are less-structured or even unstructured textual data. This characteristic of design documents also narrowly restricts the application of information extraction in engineering design, especially for the process-oriented information discovery.

2.2 Topic Modeling

In machine learning and natural language processing (NLP), topic modeling is a task of automatically developing statistical models that learn low-dimensional latent representation of a collection of documents [46]. Usually, the topics produced by topic modeling techniques are clusters of similar words. The statistics of the words in each document offer insight for us to quickly organize and understand a large collection of textual bodies.

In a typical topic model, per-document word assignments are observed variables, while topics and per-document's topic distributions are hidden variables. In this context, the central problem of topic modeling is using observed variables to infer hidden variables. In the literature, Latent Dirichlet Allocation (LDA) [47] and its extensions [48-50] have been fund as the most popular topic models, in which documents are regarded as mixtures of topics and joint distribution is utilized to compute the posterior distribution of hidden

variables. However, exact inference in these models is difficult, so that the posterior distributions are only computed approximately.

More recently, neural network based undirected graphical models are witnessed to outperform LDA models in fast inferring a document's semantic structure. One typical approach is using a two-layer Restricted Boltzmann Machine (RBM) to model word-count vectors as a Poisson distribution [51]. In order to deal with documents with different length, [52] proposed the Replicated Softmax model on the basis of RBM. Even though the inference is efficient, the representation ability of these undirected graph models is usually constrained by their simple network structure. To fix this problem, a good alternative is to use Deep Belief Nets (DBN) [53, 54]. A typical DBN model consists of one input layer of observation, one output layer of reconstruction, and several hidden layers. The deep architecture of DBN allows it to learn more complex topic features. Due to the great learning capability brought by the deep architecture, DBN is attracting more concerns from different application domains, such as document topic modeling [54], image processing [55] and speech recognition [56]. However, none is found in design document processing.

In addition, all the above topic models are constructed on word-frequency or word-count representations, ignoring the inherent appearance sequence of words. One direct result is the difficulty of understanding the learned topics straightforwardly in practical and realistic scenarios. Furthermore, most existing topic models assume that the whole document collection shares the same set of flat topics. However, in the particular application

domain of product design, one mostly common case is that documents relevant to a specific task are highly overlapped and shares a limited set of local topics. The lack of mechanism to identify local topics within groups of highly relevant documents further limits the applications of topic modeling in extracting easily understandable information.

2.3 Named Entity Recognition

As a subtask of information extraction, Named Entity Recognition (NER) seeks to identify and classify information elements, called as Named Entities (NE), in texts into pre-defined categories, such as Person, Organization, Location, etc. [57]. These annotated NEs serve as a crucial basis for many other areas of information extraction and management. For example, in the domain of automatic construction of ontologies [58-61], NEs recognized from texts are automatically filled into predefined ontologies without human intervention. By this means, NER could add structure to unstructured texts, which plays a significant role in information management and knowledge reutilization.

Early NER systems were essentially rule-based approaches, which use grammar-based rules created by human experts to match satisfied writing expressions over texts [62]. Although rule-based NER systems typically obtain high precision, the constrains for specific domains or languages limit the type of entities and texts to which they could apply, thus result in low recall in practice. To tackle this problem, modern NER systems resort to statistical machine learning (ML) algorithms. The main motivation behind is to explore more expressive features that can be utilized to learn sets of dis-ambiguous rules, which

capture the discriminative characteristics of positive and negative examples in the training corpus that are manually annotated prior. Most widely and successfully used ML models include conditional random field (CRF) [63] and support vector machine (SVM) [64]. However, one critical weakness of these statistic NER systems is their heavy dependence on large amounts of manually annotated NEs, which is extremely time-consuming for construction.

In order to reduce the annotation labor, recent research studies are directed to explore hybrid approaches, which use a mix of previous two by taking advantage of a small degree of rule-based supervision. The main technique of such semi-supervised NER systems is called “mutual bootstrapping”, which grows NEs and learns their contexts in turn on the basis of several seed NEs [65]. Precisely, they start with a small number of example NEs that are manually selected, then extract context clues from seed NEs, such as textual [65], syntactic and semantic context patterns [66, 67], followed by using context clues to learn new NE examples. This learning process is repeated. Recent experiments report that such semi-supervised NER approaches have obtained a high-quality aggregation of the supervised and unsupervised ones.

The success of information extraction has motivated the broadening of its application in different domains, where the adaption for recognizing new categories of NEs is required according to the specific applications. For example, the most concerned entities in bioinformatics are names of genes and genetic products [68]. In the clinical domain, various

types of medication related entities, such as medical problem, treatments and lab tests, are extracted from clinical notes [69, 70]. In addition, drug NER systems have added more chemical and drug related entities for complex biomedical NLP tasks [71, 72]. There also has an increasing interest in the recognition of NEs in web social media like tweet [57], e.g., Product, Band, Movie, TV-shown, etc. However, little attempt has been made to discover deep process-oriented design information for design document processing.

2.4 Entity Relation Extraction

Entity relation extraction (ERE) aims to identify relationships between pairs of named entities (NEs) in text. In information extraction, ERE is treated as a further step beyond NER, towards a more structured semantic analysis of texts [73]. For example, persons might be related to organizations, an organization may locate at some physical place, and persons can also be related to others via marriage, friendship or colleague relationships. Detection of such semantic relations, e.g., person-affiliation, organization-location and social-with, can add structure to unstructured texts and allows more powerful nature language understanding. However, high-quality relation detection especially from a large quantity of texts is not a trivial task and involves diversity machine learning techniques.

Existing approaches for ERE can be generally divided into three categories, namely, supervised, semi-supervised and unsupervised methods. The supervised methods solve the ERE task as a classification task, mostly binary classification between two named entities. A lot of machine learning algorithms have been applied in ERE, such as Conditional

Random Fields (CRF) [74], Neural Networks (NN) [75], Maximum Entropy models (MEM) [76], Hidden Markov Models (HMM) [77], and Supported Vector Machines (SVM) [78]. According to the type of the training data, these supervised ERE approaches can further be divided into feature-based and kernel-based methods. Feature-based methods require that the classifier must be trained on feature vectors. Usually, it is difficult to select a suitable feature set. In order to tackle this problem, kernel-based methods are developed to explore entity relations in a higher dimensional space. Lodhi [79] uses string-kernels to classify entity relations by computing the number of common subsequences in two strings. Bunescu and Mooney [80] extend this string-kernel from character level to word level. Zelenko [81] further replaces the strings in the string-kernels by shallow parse trees of sentences, and obtains higher reliability. Furthermore, in the form of trees, rich and diverse features (i.e., lexical, syntactic and semantic features) could be introduced to increase the learning performance [82-86]. These features cover words, entity types, POS, dependency trees, shortest paths and third-party semantic resources.

Recently, semi-supervised/bootstrapping methods have become hot for entity relation extraction. They construct weak learners on a small set of relation examples then use the output of weak learners as training data for next iteration. Typical bootstrapping algorithms and tools include DIPRE [87], Snowball [88], KnowItAll [89], and TextRunner [90]. One biggest disadvantage of these approaches is that the extracted patterns degrade iteratively because of the semantic drift [91]. In contrast, another weakly-supported approach, called

as distant supervision [92-94], has been proposed to train classifiers on a large quantity of facts. The rationale is gathering rich features by finding sentences that contain the same pair of entities. For example, a linear-regression classifier is trained by learning from only positive and unlabeled relation examples derived from Freebase, a large Wikipedia corpus [93].

Parallel to the above approaches, Open Relation Extraction (ORE) has emerged to extract open relations without previously tagged corpora using completely unsupervised methods. For example, in [95], NE pairs with similar context are grouped, and NE pairs in the same group may have the same relation. The state-of-the-art ORE systems include REVERB [96] and WOE [97]. Both systems target large corpora, e.g., World Wide Web and Wikipedia, by extracting relations that are mediated by verbs.

Besides binary relation extraction mentioned above, higher-order relation extraction has recently gained increasing popularity due to its wide applicability for various purpose. A typical application of higher-order relation is event detection [98-100], which detect complex relations among multiple entities such as people, locations, and time. A detailed review of data-driven, knowledge-driven, and hybrid event detection approaches is reported in [101]. It is stated that data-driven approaches require a lot of training data, knowledge-driven approaches work on small datasets on the basis of more expert knowledge, while hybrid approaches take advantage of both previous two approaches.

2.5 Process Mining

Process mining, also known as event mining or workflow mining, is a general methodology used to diagnose business processes by discovering models (e.g., Petri net, BPMN or event graph models) that describe reality from historical event data [102]. The resulting business process models can be used for conformance checking by comparing them to priori models [18, 103], decision support by treating them as simulation models [104-106], process monitoring by predicting how ongoing traces will unfold up to their completion [107], and process optimization by setting up optimization parameters of business process optimization models [108].

Traditionally, process mining has been focusing on control-flow discovery-that is, automatically discovering the causal dependencies or execution patterns between activities from enactment logs. Agrawal et al. [109] proposed the first concrete algorithm for event mining based on workflow graph. After that, variety process mining algorithms [110-117] have been proposed to address problems such as parallelism, noise, concurrence, loop, invisible tasks and duplicate tasks. Each algorithm has its own advantages and disadvantages. For example, as the most classic process mining algorithm, alpha algorithm [110, 111] is simple and the computation time is short, but it is not suitable for complex process with loop, duplicate tasks or noises. In contrast, genetic miner based algorithms [113, 114] are good at handling noises and tackling complex structures, but are sometimes very time consuming.

Unfortunately, above process mining algorithms have problems dealing with processes that constitute a kind of very flexible workflows [118], e.g. industrial systems and product design processes. The weakness is that they look at the discovered model from the viewpoint of paths in a flat graph model, usually WorkflowNets (WFN). As a consequence, the discovered models are often intricate networks and are typically not understandable. For this reason, Gunther [119] proposed a fuzzy mining to simplify the discovered model with the concept of roadmap abstraction. Maggi ect. [120-122] used a semi-structured process scheme called as declarative workflow to present unstructured processes with a set of constrains that state the rules among activities. Diamantini [123] applied hierarchical graph clustering to the set of instance graphs generated by a process so as to identify meaningful collaboration work practices.

Recently, an increasing amount of process mining techniques [124-126] are focusing on other perspectives, e.g. organizational perspective, performance perspective and data perspective. For example, resource perspective was addressed by grouping performers into roles based on the metrics of joint activities[124], while the temporal perspective is emphasized by using normal distribution to approximate the waiting time and execution time for activities [127].

Now, process mining is applicable to a wide range of systems and has been applied successfully in real cases, e.g., business managements [128-130], transaction fraud detection [102], shipbuilding industry [131], risk management [132], financial service [130,

133], manufacturing [131], and healthcare process [134-136]. However, there is a crucial prerequisite that the actual behavior must be recorded in well-defined structure, usually event logs in which HTML tags are used to identify the type of data elements. This presents no difficulty to formal business companies which have well-developed information systems that records sequential events. However, process mining is disappointing for highly flexible processes like product design processes, where the footprint of a process is hid in a huge amount of textual data. In this context, special text mining techniques should be carefully designed and adequately introduced to mine process model from textual data.

2.6 Summary

This chapter reviews all the aspects involved in the related research topics, starting with the applications of text mining in product design, discussing information extraction algorithms including topic modeling, NER and ERE, and finally culminating with process mining techniques.

Based on the literature review, it is found that although the existing information extraction techniques have made significant advances in product design, they have not been widely used to extract design process information. The most significant reason is that design data accumulated in product design processes are mostly unstructured texts in free natural language format. This is quite different from the two most popularly studied product design data, CAD models and patents, which are structured and semi-structured respectively. This difference means that process-relevant design documents have no prior knowledge to

supervise and support the information extraction procedure as CAD and patent files do. Consequently, traditional information extraction approaches are not suitable for extracting process relevant information from unstructured design documents, as most of them are supervised on a large amount of manually labeled training data. To solve this problem, an alternative is to use open information extraction approaches which require no prior knowledge or training data. However, open information approaches heavily rely on public knowledge bases like Wikipedia. There are no such knowledge bases related to product design processes. In this context, a semi-supervised information extraction approach with minimum human intervention is desired for product design process information extraction.

The literature review also indicates that the flat models resulting from most of the existing process mining approaches are not suitable in representing product design processes. One fundamental reason is that product design processes are highly unpredictable and iterative [24]. This characteristic of product design processes results in nonlinear and complex correlations among design tasks. However, most of the traditional process mining approaches try to capture all the process behavior in one flat model. Directly applying flat models in modeling product design processes may generate incomprehensive and unmanageable models. To solve this problem, hierarchical process models are more suitable in capturing the flexibility of product design processes. However, the existing hierarchical process mining approaches [137, 138] often construct process hierarchy by computing the

similarity of execution traces. This strategy is suitable for formal and repetitive business processes, but not for product design processes as there are no repetitive execution traces.

In addition, existing design knowledge reutilization systems are generally based on product. Such systems are suitable in the early stages of product design, e.g., conceptual design and specification design, but might not be compatible with the entire product design process. As product design is an integrated process of product, logical workflows, and resources, a good alternative approach is to integrate product design rationale and organization structure with the design process. The challenges of developing such an integrated knowledge reutilization system include making design knowledge from multiple aspects reusable, structuring heterogeneous design knowledge in a compatible knowledge base, and presenting design knowledge lucidly.

To solve the aforementioned issues and challenges, a knowledge discovery system has been developed for discovering product design process models and process relevant knowledge from design documents. An overview of the proposed framework is presented in Chapter 3.

CHAPTER 3 FRAMEWORK OVERVIEW

This chapter gives an overview of the proposed knowledge discovery system as well as the methods developed in this system. Firstly, the framework overview gives the key components of the developed system and the data flow for each component. Next, the methods developed to achieve the goal of each component are introduced correspondingly.

3.1 Framework Overview

According to the inherent characteristic of product design processes, together with the traditional KDD process [4], a framework for process-oriented knowledge discovery in product design Texts/Documents (PKDT) is proposed. Figure 3.1 shows the overview of the proposed framework. Referring to Fig. 3.1, the PKDT framework consists of five steps: data selection, data preprocessing, process information extraction, process mining, and process knowledge interpretation. The last three steps compose the core components of the PKDT framework.

The starting point of the PKDT framework is the design documents collected from various stages of a product design process. There are many types of design documents, which may be related to products, e.g., CAD models, or related to product design processes, e.g., emails and progress reports. As this research mainly focuses on process-oriented knowledge extraction, only documents relevant to process executions are selected as the target data.

Next, the target data is preprocessed by a series of natural language processing (NLP) techniques, including tokenization, lemmatization, Part of Speech (POS) tagging, and stop-words removing. The aim of data preprocessing is to enrich the target data with more linguistic features.

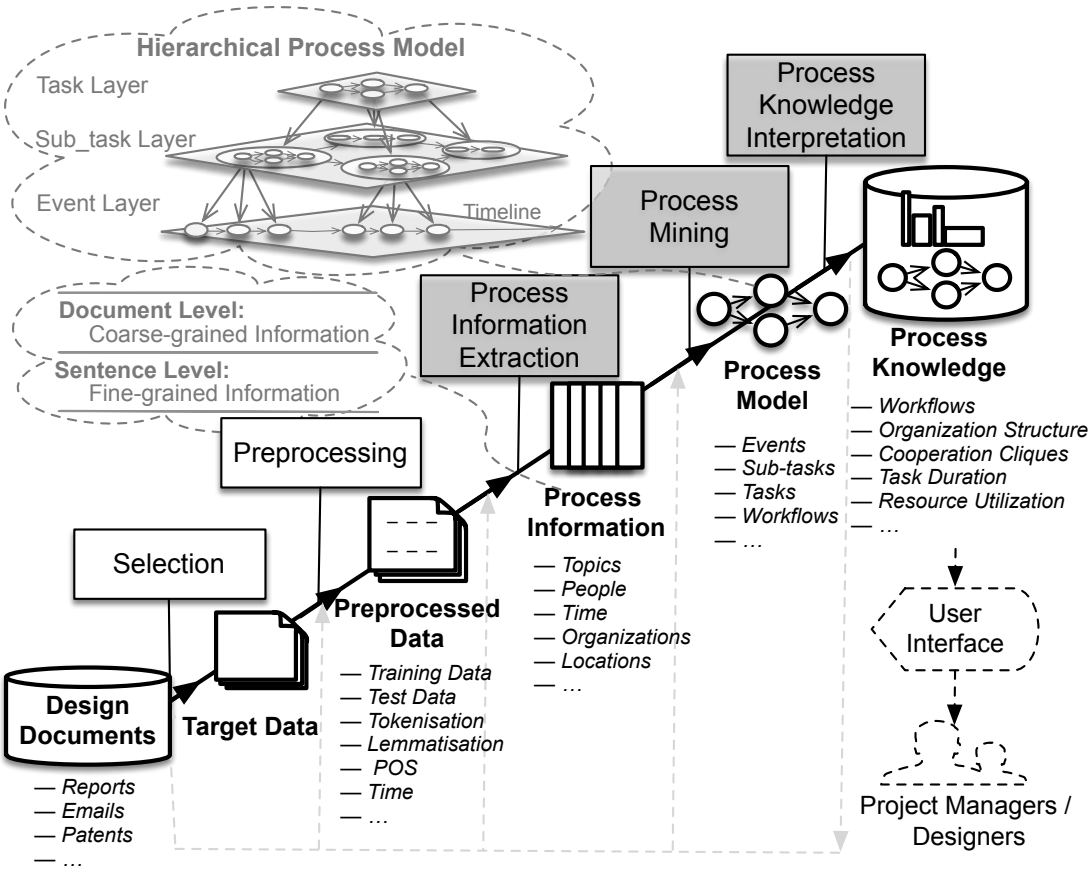


Figure 3.1 Process-oriented knowledge discovery in design texts/documents (PKDT)

The core of the PKDT process starts from the third step, process information extraction. As the target data are texts, the information related to task executions are scattered in the target data in natural language format. To identify process relevant information from the target texts, text mining and information extraction techniques are applied to recognize

special writing expressions that may point to physical objects, e.g., product components, people, organizations, and time. Such writing expressions can be seen as low-level process information as they are insufficient to directly describe the underlying design process recorded in the target data. In addition, this information extraction step also leads to a data purification and reduction as only the extracted process information would be used as the input data of the next step. This part of research will be presented in Chapters 4 and 5.

The second core component of the PKDT framework is the forth step, process mining. Given the process information extracted in the third step, process mining advances to construct a workflow model that describes the underlying product design process. The discovered model itself could be seen as a type of process knowledge, which describes the relations among the low-level process information from the workflow point of view. To achieve this goal requires selecting the appropriate model to capture the characteristics of product design processes and developing the efficient data mining algorithms to mine the relationship among the model elements. This part of research will be presented in Chapters 6 and 7.

In the last step, the extracted workflow model serves as the backbone, based on which multi-faceted process knowledge such as organizational structure and social network patterns could be enhanced and integrated. Using statistical analysis methods, more specific aspects of a product design process could be further interpreted, for example, the most active designers and the longest design tasks. Such upgraded information composes the ready-to-

use knowledge, which could help designers to make more efficient decisions. To achieve the above goals requires understanding the application domain where the analyzed results will be applied, selecting the accurate data from the discovered process model, and using appropriate visualization techniques to interpret the analyzed results in a user-readable manner. This part of research will be presented in Chapter 8.

3.2 Methods

New methods have been developed to achieve the goal of the three core components. As shown in Fig. 3.1, all the developed methods are systematically integrated in the PKDT system.

3.2.1 Methods for Process Information Extraction

Two information extraction methods which identify process information in texts with different granularity are proposed. They are a DBN based topic modeling approach for coarse-grained information extraction and a hybrid NER approach for fine-grained information extraction.

For the purpose of extracting process information without any training data or prior knowledge, a DBN based topic modeling approach is proposed to automatically find a finite set of topics from the input documents. The extracted topics are groups of meaningful words that condense the document contents and may reflect some issues of the target product design process. Based on the extracted topics, one could get a fast understanding of some process behaviors, for example, design tasks by connecting topics to design activities, and

temporal dynamics of a design process by identifying changes in topics over time. However, such topics cannot precisely reflect how a design task was executed. This is why the extracted topics are called as coarse-grained process information in the PKDT framework. The proposed topic modeling method will be presented in Chapter 4.

For the purpose of extracting fine-grained process information as well as controlling the human intervention to the minimum, a hybrid NER approach is proposed to recognize special writing expressions that point to physical objects that were involved in the target product design process, e.g., designers, tools, and organizations. The main idea of the proposed NER approach is to combine the advantages of both the rule based and the machine learning based NER approaches. In addition, to increase the recognition accuracy, a local dependency tree is proposed to utilize more linguistic features of NEs. The experiment data for evaluating the proposed approach are a set of emails collected from a real-life product design project. The comparison to two baseline approaches shows an increase in the detection accuracy by using the proposed approach. The proposed NER approach will be presented in Chapter 5.

3.2.2 Methods for Process Mining

To capture the flexibility of product design processes, a high-order ERE approach is proposed to detect small design events at the most concrete level, and two process mining approaches are developed to obtain a hierarchical process model with different degrees of abstraction.

An event, which is the smallest component of a process, describes the most detailed task executions. An event usually can be presented as a higher-order relation among several objects, e.g., someone did something at some time. To capture this higher-order relation of design events in an unsupervised manner, a graph partition based ERE approach is proposed. The main idea is to decompose the higher-order relation of an event into several binary relations, then reconstruct the higher-order relation by finding the maximum NE cliques centered at each task NE. The proposed event detection approach will be given in Chapter 6.

To capture the flexibility of product design processes and to reduce the complexity of the discovered process model, two process mining approaches are proposed, i.e., top-down and bottom-up process mining. Both approaches aim to decompose a product design process into modules in a hierarchical manner, then refine the detailed transitions within each module. Their biggest difference is the strategy used to construct the hierarchy structure. The top-down approach mines the process model from generation to specification by iteratively decomposing the underlying process based on the similarity of document contents. In contrast, the bottom-up approach proceeds from specification to generation by iteratively merging design events into bigger ones based on the context similarity of design events. Both process mining approaches will be presented and compared in Chapter 7.

3.2.3 Methods for Process Knowledge Interpretation

To overcome the problem that most of the existing design knowledge reutilization systems are not compatible with the whole design process, an integrated design knowledge reutilization framework is proposed. The proposed framework treats the discovered design process model as the central element of design knowledge, and links other types of design knowledge such organizational structure, cooperation patterns, and temporal process behaviors to the process model.

To enrich the discovered process model with multi-faceted knowledge, statistic analysis methods are applied to distill more understandable design knowledge from the process model. Based on the proposed knowledge reutilization framework, two significant aspects of product design processes are studied: organization mining and temporal behavior analysis. Firstly, to investigate the performance of the project participants, their cooperation patterns, and their functional roles, the developed organization mining methods analyze the performance of the project participants via the design events/tasks they have participated in. Secondly, to study the dynamic changes of the target design process, the temporal behavior of both the design tasks and the project participants are analyzed, including the duration, waiting time, and idle time of the design tasks, as well as the temporal and overall contribution of the project participants.

All the analyzed results are well presented in a user-readable manner using Gantt chars, social network graphs, dot charts, and bar charts. A real case study is conducted in Chapter 8 to illustrate the discovered design process knowledge.

3.3 Case Study Description

The developed process-oriented knowledge discovery system is tested and illustrated on an email dataset collected from a university-hosted design project, named as traffic wave project (TWP). It aimed to design a traffic control system to ease the traffic congestion on expressway and published the study results in a conference paper. It was a sub-project of a university-hosted project, which had several sub-projects focusing on solving different related problems. The main participants include students and professors from three different disciplines. In addition, participants from other sub-projects were also involved in this traffic wave project more or less. Throughout the design process, the participants used emails as their major communication tool to exchange opinions. The design process was originally planed with seven phases: concept design, thesis proposal, specific design phase I, specific design phase II, experimental simulation, hardware level validation, and thesis submission. However, this plan had been interrupted by several unpredicted problems during the actual design process. The whole design process lasted about two years, from March 2011 to February 2013.

The experimental dataset is a set of emails collected from this traffic wave project. Throughout the design process, all the participants always sent a copy to a specific common

account when they used emails to exchange and discuss their opinions. This culminated in a total of 569 emails saved in a MS Outlook file. Each email contains information about the design tasks discussed in the email body, the involved people are mentioned as either the email sender/receiver or in the email body, and the time is indicated by the creation time of the email.

3.4 Summary

This chapter gives an overview of the proposed knowledge discovery system, which consists of five steps: data selection, data preprocessing, process information extraction, process mining, and process knowledge interpretation. The rationale behind each step and the methods integrated are briefly introduced correspondingly. This chapter culminates in a detailed description of the used case study.

CHAPTER 4 COARSE-GRAINED PROCESS INFORMATION EXTRACTION BY TOPIC MODELLING

4.1 Introduction

The first problem addressed by the PKDT system is extracting coarse-grained process information at the document level. The coarse-grained process information is presented as topics that condense the content of the input documents and relate to process executions. In this chapter, a Deep Belief Network (DBN) based topic modeling approach is proposed to discover such topics from design documents in an unsupervised manner. From a technical perspective, the topics produced by topic modeling are clusters of similar or highly correlated words, and the order of these words is not taken into account [139] [140, 141]. This is why the word "coarse-grained" is used to describe the process information extracted in this chapter.

As discussed in Chapter 1 and Chapter 2, there are two significant motivations for using DBN-based topic modeling to extract process information. Firstly, the DBN-based topic modeling approaches are unsupervised. That is to say, there is no requirement of training data or prior knowledge. Secondly, the extracted topics could serve as the base for developing new ways to search, browse, summarize, and understand the original design documents. More specifically, some of the topics might correspond to the design tasks, e.g., concept design, specific design, and validation. Automatically uncovering such topics can benefit decision makers to develop a fast understanding of the underlying process recorded

by the design documents. For example, decision makers can find a topic that they are interested in, and then zoom in on the documents related to this topic for getting more detailed information. At a broader level, designers can track the complete history of the product design process via investigating the topic changes over time.

The framework of the proposed approach is presented in Section 4.2. Section 4.3 introduces the details of the DBN-based topic modeling approach. In addition, to deal with documents with different length, real-valued units that represent documents in word-frequency vectors are used at the input layer of the proposed DBN topic model. Furthermore, to make the learned topics relate to process executions, "label" information, e.g, document title, keywords, and abstract, which summarize the central theme of a document, is used as the output layer to fine-tune the topic model. In Section 4.4, the extracted topics are used to develop a fast understanding of the recorded design process from three aspects: design tasks discussed, their dynamic changes and interactions. In Section 4.5, the performance of the coarse-grained process information extraction is tested on the email dataset from the TWP project introduced in Section 3.3.

4.2 Framework Overview

Figure 4.1 illustrates the framework of the proposed topic modeling approach for extracting coarse-grained process information. The proposed framework consists of two steps, training a DBN topic model and applying the DBN model in understanding design

task structure. In Fig. 4.1, the thin arrows indicate the workflows within each step, and the blank arrows indicate the input and output flows running through each step.

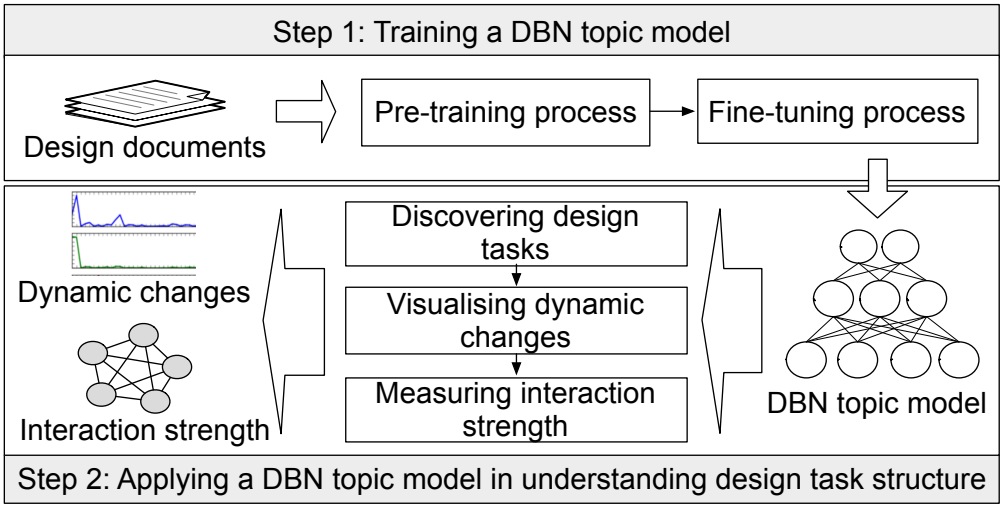


Figure 4.1 The framework of coarse-grained process information extraction using DBN-based topic modeling

In Fig. 4.1, the starting point of the first step is a set of time-stamped design documents from a product design project. The result is a DBN topic model which learns a set of topics recorded in the whole document archive and the topic distribution of each document. The training process of the topic modeling consists of two stages, pre-training and fine-tuning. At the pre-training stage, the body of the input documents are used to train a primary topic model. Next, the title or keywords of the input documents are used to fine tune the primary model at the fine-tuning stage.

In the second step shown in Fig. 4.1, the discovered topic model is used to understand the underlying design process from three aspects. They are 1) what design tasks can be told

by the topics, 2) how did the tasks change over time, and 3) How did the tasks interact with each other.

4.3 Training a DBN Topic Model

4.3.1 DBN Topic Model

Figure 4.2 shows the structure of a typical DBN model which consists of one input layer of observations, one output layer of reconstructions of the input data, and several hidden layers [53]. The units in each hidden layer aim to learn the topic representation of the input data (observation) at different abstraction levels. Generally, topics in an upper hidden layer tend to become more complex.

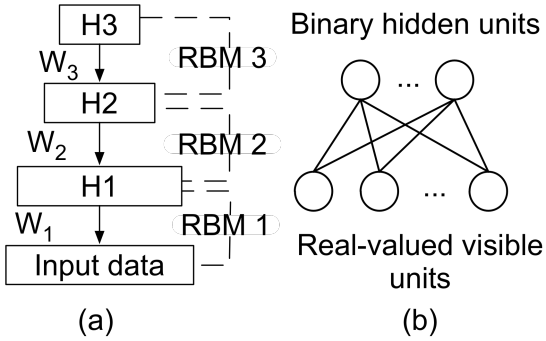


Figure 4.2 The architecture of deep belief network (DBN): (a) an example of DBN, (b) The restricted Boltzmann machine (RBM)

As shown in Fig. 4.2 (a), the layers of a DBN model can be split pairwise. Each pair forms a separated Restricted Boltzmann Machine (RBM), which is shown in Fig. 4.2 (b). Each RBM aims to learn the statistical relationship between the visible units and the hidden units. In this context, the DBN can be greedily trained in a layer-by-layer manner, where the output of the lower-layer RBM is the input data for training a higher-layer RBM.

In order to deal with documents with difference length and to distinguish words with different degrees of contribution, each RBM is configured with a real-valued visible layer and a binary hidden layer. In detail, given the hidden topic features *H*, a normal distribution $P(v_i|H)$ is used to model the possibility of a word v_i appearing in a document. In contrast, given the observed per-document word distribution *V*, a sigmoid function $P(h_j = 1|V)$ is used to model the hidden topic features *H*. Equations (4.1) and (4.2) formulate the two functions.

$$p(v_i|H) = Normal(\frac{\exp\left(\sum_{j=1}^{j=k} w_{ji}h_j + a_i\right)}{\sum_{l=1}^{l=M} \exp\left(\sum_{j=1}^{j=K} w_{jl}h_j + a_l\right)}, \delta) \tag{4.1}$$

$$p\left(h_j = 1\middle|V\right) = sigm(b_j + \sum_{i=1}^{i=M} w_{ji}v_i) \tag{4.2}$$

where w_{ji} is the symmetric interaction weight between a visible unit (word) v_i and a hidden topic h_j, σ is the standard deviation, a_i is the bias of a visible unit v_i, and b_j is the bias of a hidden unit v_i. The value of the visible units stands for the relative frequency of the corresponding words in a document, valued in a range of 0 to 1. Given a set of topic features, the occurrence frequency over all the words sum up to be one, which is import to deal with documents with different lengths.

4.3.2 Training Process

Before training the DBN model, the metadata in the input document are divided into two parts, i.e., "body" data $\{D_1, \dots, D_i, \dots, D_N\}$ and “label” data $\{Y_1, \dots, Y_i, \dots, Y_N\}$. The "body" data of each document is represented by a word-frequency vector $D_i =$

$(v_1, \ldots, v_j, \ldots, v_M)$, where v_j is the occurrence frequency of the j^{th} word in the body text of document D_i, and M is the vocabulary size of all the documents. The "label" data is the words in document title, keywords, and abstract, which summarize a document's central theme. The information contained in such data provide significant supplements to text analysis. Taking advantage of such "label" data to supervise and fine-tune the training process is quite helpful in guaranteeing that the learned latent topics are related to the central theme of a document. Therefore, the "label" data of a document is defined as a word-occurrence vector $Y_i = (y_1, \ldots, y_j, \ldots, y_M)$, where $y_j \in \{0,1\}$ and "1" indicates the occurrence of the j^{th} word in the "label" parts of D_i. Due to the low frequency of words in the "label" parts, all the words in Y are treated with the same significance. In other words, y_j indicates the occurrence of a word in the "label" data rather than the occurrence frequency in v_j.

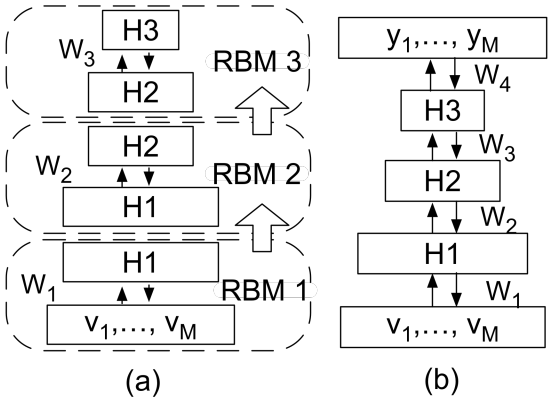


Figure 4.3 Training process of DBN topic model: (a) Pre-training process, in which a stack of RBMs are learned layer by layer, (b) Fine-tuning process, where an extra layer of "label" data is added to fine tune the entire network

Figure 4.3 illustrates the training process of the DBN topic modeling, which consists of two steps: pre-training and fine-tuning. In Fig. 4.3 (a), the the visible units $(v_1, \dots, v_j, \dots, v_M)$ in the bottom layer stand for the word-frequency vectors obtained form the "body" data. The output units $(y_1, \dots, y_j, \dots, y_M)$ in top layer of Fig. 4.3 (b) correspond to the word-occurrence vectors obtained from the "label" data. The aim of the pre-training process is to learn a topic model that can reconstruct the input data to the largest extent. Given the "label" data, the fine-tuning process aims to enhance the topic model by making the learned topics reflect the central theme of a document.

4.3.2.1 Pre-training Process

The pre-training step aims to greedily approximate parameters in Eq. (4.1) and Eq. (4.2). Each RBM in Fig. 4.3 (a) is trained separately. Given the word-frequency vectors, the bottom RBM is expected to learn a set of low-level topic features from the documents. The renormalized topic features over the learned posterior distribution $P(H|V)$ is then used as the input data for training a higher-level RBM. The topics learned at a higher level try to capture the complex combination of the low-level topics. This layer-by-layer training process is repeated several times to learn a deep belief network in Fig. 4.3 (a).

In each iteration of the training process, the 1-step Contrastive Divergence [142] is adopted to updating the parameters by

$$w_{ij} = \varepsilon(E_{Pdata}[v_i h_j] - E_{Precon}[\hat{v}_i \hat{h}_j]) \tag{4.3}$$

$$a_i = \varepsilon(E_{Pdata}[v_i] - E_{Precon}[\hat{v}_i]) \tag{4.4}$$

$$b_i = \varepsilon(E_{Pdata}[h_i] - E_{Precon}[\hat{h}_i]) \tag{4.5}$$

where ε is the learning rate, and $E_{Pdata}[v_i h_j]$ equations $p(h_j|v_i)p(v_i)$, indicating the expectation of the co-occurrence frequency of word v_i and hidden feature h_j given the observed input data. Similarly, $E_{Precon}[\hat{v}_i \hat{h}_j]$ corresponds to the expectation of the co-occurrence frequency given the reconstructed data after one-step Gibbs sampling.

4.3.2.2 Fine-tuning Process

After pre-training, an extra layer of binary units is added to the top of the DBN, as shown in Fig. 4.3 (b). The "label" data $Y_i = (y_1, \dots, y_j, \dots, y_M)$ is used to back-propagate the whole network to enhance the weights of topics that are mostly related to document "labels". For those documents without "label" data, the words with high frequency in a document are used for substitution because words with low frequency are usually less significant to reflect the main idea of a document.

4.4 Applying DBN Model in Understanding Design Task Structure

4.4.1 Discovering Design Tasks

This step advances to use the learned topics to interpret design tasks that are recorded in the input documents. After training, each hidden unit in the topic model is connected to a set of words in the visible layer by the weights in W. In turn, the words that are strongly

connected reveal the semantic meaning of the corresponding topics, which might refer to a design task in the real word.

In detail, each topic learned by the lowest hidden layer (e.g., H_1 in Fig. 4.3) is directly represented by words with strongest positive weights to the corresponding hidden unit. Take Fig. 4.4 as an example, where the thick lines indicate strong connections between words and topics. According to Fig. 4.4, three words $i \in \{1, 2, 3\}$ with largest w_{ji} in $w_{j,:}$ are selected to compose $Topic_j^1$. Similarly, using the low-level topics in place of words in the visible layer, the topics learned in a higher layer can be represented by groups of strongly connected topics learned in a lower layer.

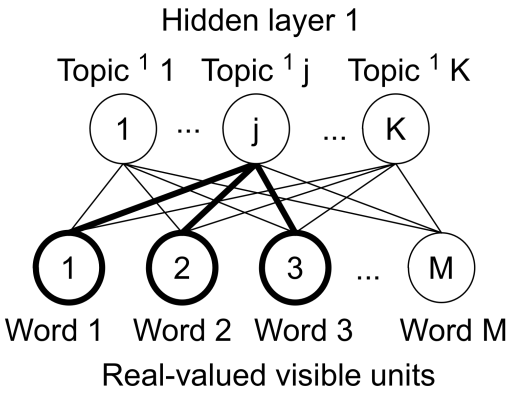


Figure 4.4 Illustration of mapping design tasks from hidden topic features. The thick lines indicate words with strongest connections to the j^{th} topic.

After presenting the topics using strongly connected words, a small amount of human intervention is required to select the topics that are relevant to task executions. The selected topics can then be analyzed to obtain deep insight into process behaviors.

4.4.2 Visualizing Dynamic Changes of Design Tasks

Based on the task-relevant topics and their distribution throughout the input documents, the temporal feature of the design tasks is characterized by the change of their temporal frequency over time. In other words, the time regions, within which a task-relevant topic is frequently discussed, can reflect the lifecycle of the corresponding task. Furthermore, the dynamic changes of all the design tasks reflect the temporal behavior of the entire product design process.

Let h_t be a topic relevant to a design task, its temporal frequency within a short time window is computed as:

$$tf(h_t, Win) = \frac{\sum_{V_d \epsilon D} p(h_t|V_d)\varphi(D_d, Win)}{|Win|} \tag{4.6}$$

where, $p(h_t|V_d)$ is the possible frequency of h_t appearing in a document, the value of $p(h_t|V_d)$ is determined by the learned topic model, Win stands for the time window, $|Win|$ indicates the size of the time window, and the function $\varphi(D_d, Win)$ returns a binary value to judge whether the creation time of a document is within the time window or not.

4.4.3 Measuring The Interaction Strength of Design Tasks

Since design tasks that have strong correlation are usually mentioned together, the interaction strength between pairs of design tasks are estimated by the co-occurrence frequency of the corresponding topics in the input documents.

Let h_a and h_b be two task-relevant topics. The learned topic model estimates the possible frequency that h_a and h_b are mentioned in a document V_d as $p(h_a|V_d)$ and $p(h_b|V_d)$ respectively. Next, the interaction strength of h_a and h_b can be estimated by computing their overall co-occurrence frequency in the whole document set:

$$IS(h_a, h_b) = \left(\sum_{d=1}^{d=N} P(h_a|V_d)P(h_b|V_d) \right)/N \tag{4.7}$$

where N is the size of the document set, and $P(h_i|V_d)P(h_j|V_d)$ computes the co-occurrence frequency of h_a and h_b in a single document V_d.

4.5 Experimental Results and Discussions

4.5.1 Dataset and Data Preprocessing

The performance of the coarse-grained process information extraction approach was tested and illustrated using the email dataset from the TWP project described in Section 3.3.

The original dataset was preprocessed by removing meaningless stop-words, stemming, and eliminating words that occurred less than two times throughout the entire email collection. All these preprocessing operations were performed with the help of NLTK[1], which is a open-source toolkit for natural language processing with python. The vocabulary size of the preprocessed dataset is 1630. The words in both the email subjects and the email bodies composed the input data $\{D_1, \ldots, D_i, \ldots, D_N\}$ for pre-training the topic model. In

[1] http://www.nltk.org

contrast, only the words in email subjects composed the "label" data $\{Y_1, \dots, Y_i, \dots, Y_N\}$ for fine-tuning the topic model.

4.5.2 Performance Measures

The performance of the proposed approach was evaluated from two aspects: the effectiveness of full-text document retrieval and the ability for discovering some valuable characteristics of the actual design process.

The full-text document retrieval experiment aims to evaluate the influence of the topic model structure on the document retrieval effectiveness. Each email in the training set was used as a query to search those ones with biggest similarity to it. The content similarity of any two emails was calculated by using the Euclidean distance between their latent topic distributions $P(H|V)$. Using email subject as the evaluation criterion, the document retrieval precision was computed as follows:

$$Precision(i) = N_{correct}^{total}(i)/N_{total}(i) \tag{4.8}$$

where N_{total} is the number of emails that have the same subject with the i^{th} email, and $N_{correct}^{total}$ is the number of correctly retrieved emails, which are among the top N_{total} rank in the retrieved emails and have the same subject with the target email.

To test the ability of the proposed approach for discovering the reality of the underlying design process, the original process model planned at the beginning of this project was used as the baseline model for comparison. If the discovered design tasks contain the planned

tasks, the proposed method was able to discover actual process executions from design documents. In addition, one participant, who played a admin role in this project, was interviewed to check the correctness of the design tasks that were not originally planned. Furthermore, to test the understandability of the discovered results, the results, including task-relevant topics, their timeline, and interactions, were also checked by two novices who had no idea of this project.

4.5.3 Results and Discussions

4.5.3.1 Document Retrieval Evaluation

Figure 4.5 compares the performances of different DBN topic models in full-text document retrieval. Each DBN topic model in Fig. 4.5 is different from the others by having a different number of hidden units or a different number of hidden layers. Referring to Fig. 4.5, the structure of the DBN topic models are indicated in the format of XX-XX, where XX means the number of neurons in a layer of the DBN model. For example, 1630-50 means a DBN model having two layers, one visible layer of size 1630 and one hidden layer of size 50. All topic models were trained under the same parameter settings: 2000 iterations for pre-training process, 1000 iterations for fine-tuning process, 0.2 for weight learning rate, and 0.05 for biases learning rate.

Figure 4.5 (a) shows the average retrieval precision of six DBN topic models, which have one hidden layer but different numbers of hidden topic units. As seen from the symbol curve in Fig. 4.5 (a), the average retrieval precision increases dramatically when the

numbers of hidden units are relatively small, but it becomes stable after the number is greater than 50. Based on the well-known experience that more hidden units tend to need more training data and more training time, a moderate number of hidden units is suggested to remain effective in training topic models. Therefore, the number of hidden units was set to be 50 in the next experiments.

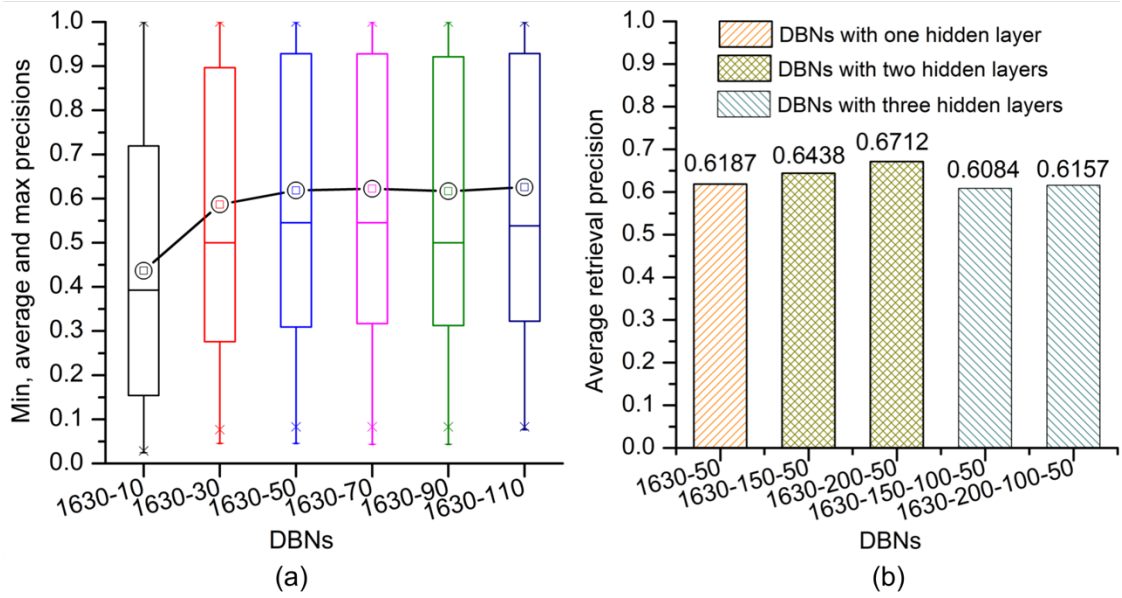


Figure 4.5 Document retrieval effectiveness of DBNs: (a) Comparison of DBNs of one hidden layer but different hidden units, (b) Comparison of DBNs of the same hidden units in the top layer but different numbers of hidden layers

In Fig. 4.5 (b), five DBN models with different numbers of hidden layers are compared. As observed from Fig.4.5 (b), compared to the one-hidden-layer model (1630-50), DBNs with two hidden layers (1630-150-50 and 1630-200-50) improve the precision score from 0.6187 to 0.6438 and 0.6712 respectively. However, when a larger number of hidden layers is specified in the two three-hidden-layers DBN models, the accuracies drop to 0.6084 and 0.6147. This conflicting result indicates that the effectiveness of full-text document retrieval

is not proportional to the number of hidden topic layers. One possible reason might be insufficient training. As DBN models of more hidden layers contain more parameters, to correctly learn parameters in a large DBN model requires sufficient training data. Therefore, the number of hidden layers in a DBN model should be selected according to the size of the training data. Based on the findings from Fig. 4.5, a moderate number of hidden layers is suggested.

Figure 4.6 compares the DBN topic models with the Latent Direchilet Allocation (LDA) [20], which is one of the most popular topic models. For fairness, the LDA models in Fig. 4.6 were trained with the similar parameters as the DBN models did, i.e., 2000 iterations for training and the same numbers of hidden topics. The comparison result in Fig. 4.6 confirms the previous conclusion that DBN outperforms LDA in learning the latent topic representation of documents.

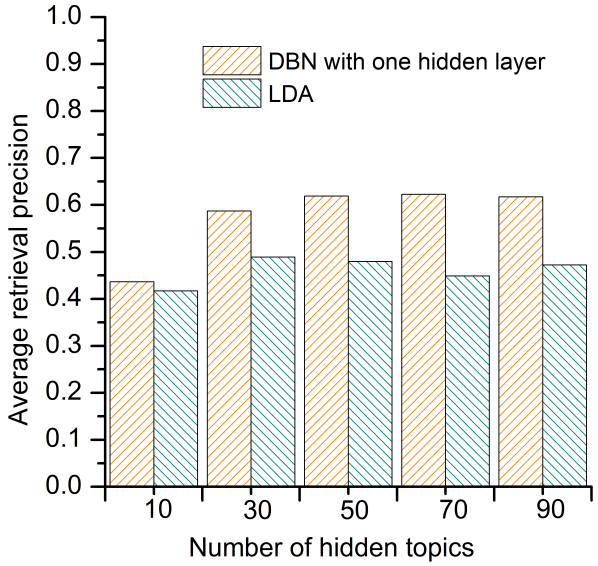


Figure 4.6 Document retrieval effectiveness of one-hidden-layer DBNs and LDAs

4.5.3.2 Learned Design Tasks

The second experiment aims to inspect that, given a set of design documents, whether the DBN topic model is able to identify meaningful latent topics that uncover design tasks recorded in these documents. Based on the findings from Fig. 4.5, the DBN model of structure 1630-200-50 was selected to learn topics from the email dataset. For each learned latent topic, the top five words with the strongest connections to it are used to name the corresponding design tasks.

Table 4.1 Illustration of design tasks learned by DBN topic model

Words	**Probability**	**Words**	**Probability**	**Words**	**Probability**
Task 1 (XXX project proposal)		Task 2 (Concept paper submission)		Task 3 (ASME conference paper)	
XXX	0.591	Concept	0.598	Revise	0.706
Meeting	0.291	Submission	0.556	ASME	0.315
Proposal	0.245	Revise	0.276	Dates	0.268
Project	0.230	Paper	0.267	Congress	0.259
Importance	0.022	Conference	0.223	Ants	0.190
The Words	**Probability**	**Words**	**Probability**	**Words**	**Probability**
Task 4 (IRB application)		Task 5 (Traffic data collection)		Task 6 (Simulation software)	
Application	2.496	Traffic	0.514	Paramics	0.527
IRB	2.696	AYE	0.499	Simulation	0.293
Review	2.235	Data	0.492	Key	0.256
XXX	0.597	Project	0.482	Software	0.193
Form	0.566	Program	0.433	Wei	0.137

50 latent topics were learned from the email dataset. The feedback from the interviewed project participant revealed that some of the 50 latent topics truly related to the actual design tasks while some not. Due to the space limitation, Table 4.1 only lists 6 topics that are most

relevant to design tasks that had been carried out during this TWP project. In the following parts, these topics will be referred to as design tasks. For each design task, only words of top-5 strongest connections are listed in Table 4.1, and the probability column displays the weights connecting words and topics. For privacy reasons, XXX is used in place of the names of organizations and persons.

According to Table 4.1, most words associated with each design task are quite intuitive in the sense of conveying a semantic meaning that reflect what were actually done during the design process. Take the six design tasks as an example, namely XXX project proposal, concept paper submission, ASME conference paper, IRB application, traffic data collection, and simulation software. According to the feedback from the core participant, the TWP project is only a sub-project of the "XXX project", which consists of several sub-projects. At the beginning, each sub-project was required to submit a "project proposal", as reflected by Task 1 in Table 4.1. Next, a detailed "concept paper" about their ideas and plans were completed after several "modification" iterations. This part is reflected by Task 2 in Table 4.1. In the middle stage, an unexpected task was conduced to obtain some supporting "documents" from a significantly relevant department. The words of Task 4 imply the information of the "IRB application". After developing the core techniques, which are not shown in Table 4.1, real life "traffic data" was fetched from the traffic department. Next, the project members utilized the data to evaluate the developed traffic control system on several simulation platforms. One of the simulation tools was "Paramics", which is correctly

listed in Task 6. Finally, as shown in Task 3, this project was ended with writing and publishing an "ASME paper".

Figure 4.7 illustrates the conformance of the tasks shown in Table 4.1 with the tasks scheduled at the beginning of the TWP project. The arrows in Fig. 4.7 connect the automated tasks to the planned tasks if they are related. It can be observed that four tasks in Table 4.1 are related to the planned tasks, while two tasks were not scheduled at the beginning of this project. According to the expert feedback, the two unplanned tasks also reflect the reality of this TWP project.

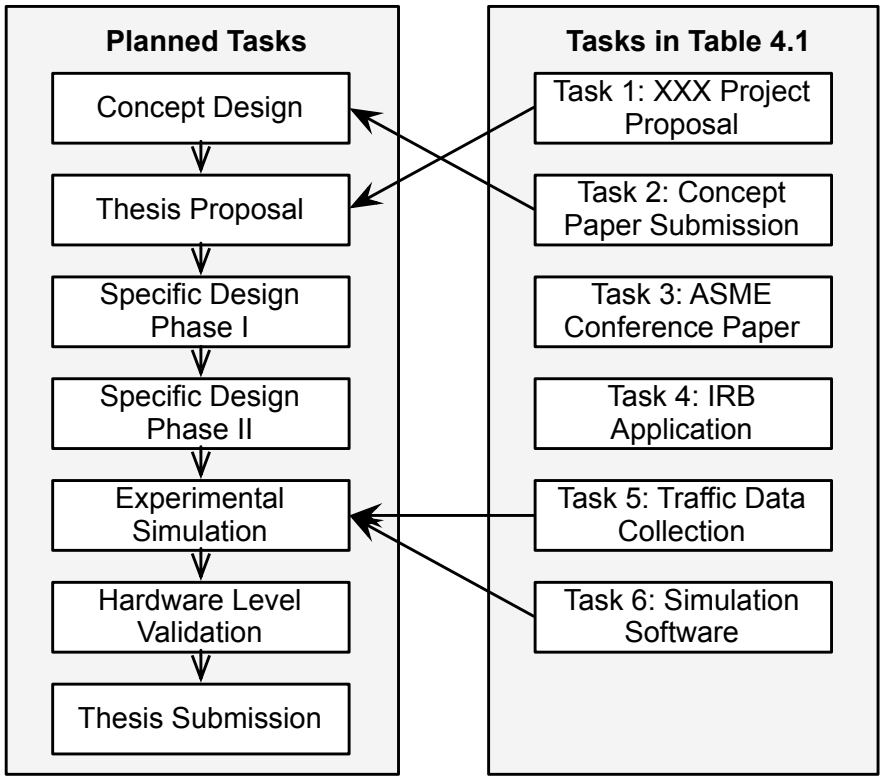


Figure 4.7 Conformance checking

4.5.3.3 Timeline of Design Tasks

In order to track the regions of the timeline when the project participants were truly working on the different tasks, Figure 4.8 plots the temporal frequency of the six task-relevant topics in Table 4.1with a window size of 15 days.

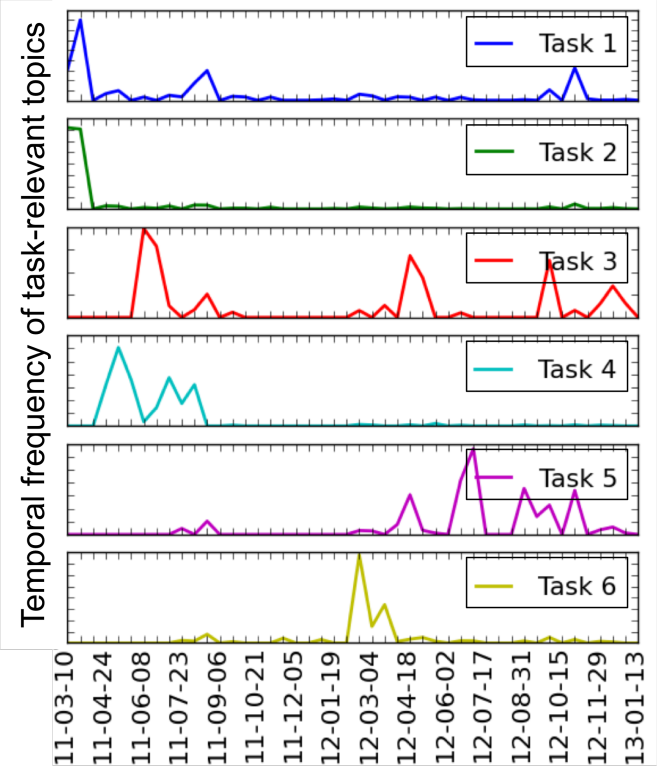


Figure 4.8 Temporal frequency of task-relevant topics in Table 4.1 with a window size of 15 days

According to Fig. 4.8, the timeline of each task in Table 4.1 aligns well with the feedback discussed in Section 4.5.3.2. It can be seen that the participants conducted on the project proposal issue (Task 1) at the beginning. Next, they achieved a concept paper (Task 2) during the first month after the project started out. By the second month, the participants proceeded to obtain the IRB support (Task 4) before they could advance to the technical

part, which took them about 4 months. According to the timelines of Task 5 and Task 6, traffic data collection (Task 5) and simulation software purchase (Task 6) were started out almost simultaneously after about 10 months. However, Figure 4.8 shows that the participants spent much longer time in getting and processing the traffic data.

4.5.3.4 Learned Design Task Interactions

The last experiment tries to investigate how design tasks had interacted with each other in practice. Figure 4.9 illustrates the interaction strength between the six tasks in Table 4.1. In Fig. 4.9, nodes indicate tasks, the size of nodes reflect the overall interaction between one task and all others, and the thickness of edges present the strength connecting tasks.

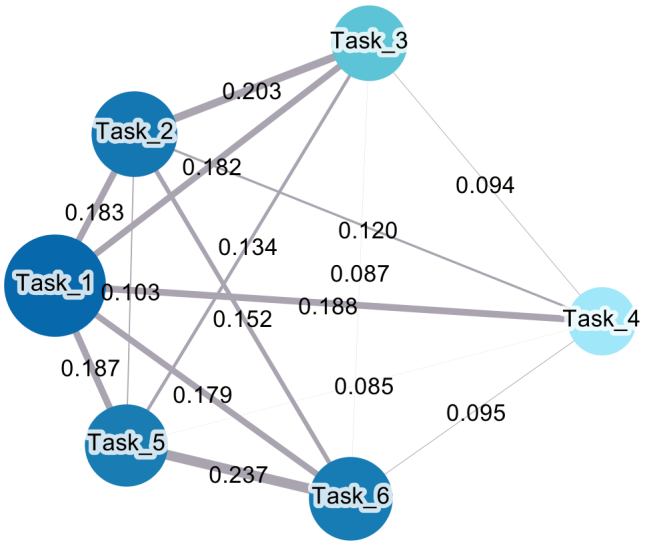


Figure 4.9 Illustration of interaction strengths between selected design tasks

From Fig. 4.9, one notable observation is that Task 1 (XXX project proposal in Table 4.1) might have strongly interacted with all others. This finding is not difficult to explain. Because all the initial design ideas were generated in this task, it is nature that all the

remaining tasks had connections with it more or less. The edges connecting Task 4 (IRB application) show an exactly inverse interaction pattern. Figure 4.9 shows that Task 4 only has strong connections to two tasks, Task 1 and Task 2, with a strength of 0.188 and 0.120 respectively. This observation is consistent with the feedback that Task 4 is not a part of the design project itself, but required to get support from a relevant department based on the results of Task 1 and Task 2. According to Fig. 4.9, the strongest interaction, valued at 0.237, is found between Task 5 and Task 6. This is validated by the relevant emails that the two tasks were carried out concurrently, and both were about validating the developed traffic control system.

4.6 Summary

To extract design information for product design process understanding, a DBN-based topic modeling approach was proposed to automatically learn process-relevant topics in the design documents. Using the data collected from a real-life design project, three significant aspects have been considered: design tasks, their timelines, and their interaction strengths. The findings were evaluated by the project participant. The feedback revealed strong positive comments to the results.

Some limitations were also observed. Firstly, the feedback from the interviewed project participant indicated that the learned topics could reveal some design tasks in practice, but the feedback from the two interviewed novices revealed that the words composing a topic were difficult for interpretation, especially for novices. Take Task 1 in Table 4.1 as an

example. Based on words, i.e., “Meeting”, “Proposal”, “Project”, and “Importance”, participants of this project can easily recollect the corresponding tasks, but these words might be difficult for novices to connect them to a real-world task. This is caused by the learning mechanism of topic models, which discover abstract “topics” only based on the statistics of words, overlooking their occurrence order. Secondly, although the co-occurrence frequency of topics can uncover the task interaction to some extent, it is not sufficient to explain how design tasks interacted. One most significant reason is that the complex interaction among design tasks are jointly determined by multiple process variables, e.g., resources, tools, and deadline. Consequently, identifying the process-related variables from design documents is critical for estimating the interaction strength among the design tasks more comprehensively and correctly. Both limitations require extracting and analyzing design information with a more fine-grained granularity.

CHAPTER 5 FINE-GRAINED PROCESS INFORMATION EXTRACTION BY NAMED ENTITY RECOGNITION

5.1 Introduction

In previous chapter, the experiment results show that the proposed topic modeling approach is successful in providing a rough and quick overview on "what happened", but fails to convey detailed information about task executions. This means that individual documents still have to be fetched and skimmed through manually to locate concrete information such as who did what, when, where, and how. Such a reworking operation charges extra time on knowledge reutilization. To overcome this problem, a good alternative is exploring computational approaches for pinpointing highly concrete design information.

Based on the above analysis, this chapter continues to extract process information from design documents, but down to a fine-trained granularity. Specifically, the goal of this chapter is to extract special writing expressions or terms which may point to some physical objects such as design tasks, people, organizations, tools, and locations, from the design documents.

Named entity recognition (NER) is a popular information extraction technique that seeks to classify named entities (NEs) in text into pre-defined categories. However, as discussed in Chapter 2, traditional NER approaches are not suitable for extracting process information in design documents. Traditionally, the supervised NER approaches are based on a large amount of training data that are manually labeled or automatically labeled with

the help of public knowledge bases like Wikipedia. However, considering the high flexibility of product design processes, creating a training dataset for design process information extraction is neither economical (the time cost is high) nor effective (there are significant differences between documents from different product design processes). In this context, a semi-supervised NER approach can be helpful, as human intervention could be controlled in an acceptable range. However, most of the semi-supervised NER approaches are narrowly restricted to some specific domains. Therefore, one information extraction system developed for one domain usually does not perform well in other domains.

To close the above gaps, a hybrid NER approach is proposed in this chapter to identify process relevant entities from design documents in a stepwise manner. The main steps of the proposed NER approach include sentence classification via a prior-trained Bayes classifier, seed entity generation via speech act rules, entity expansion via kernelized machine learning approaches, and co-reference resolution via clustering. The speech act rules are used to reduce the human intervention in creating training data to a minimum. In addition, a kernel function of local dependency tree is proposed to capture the complex linguistic features of NEs for the training purpose.

The symbolic representation of the problem of this chapter is refined in Section 5.2. The proposed NER approach and the relevant data structure of the kernel function are detailed in Section 5.3. The experiment results and discussions are given in Section 5.4. Lastly, the conclusion is drawn in Section 5.5.

5.2 Problem Statement

As defined in [2], the product design process is a set of interrelated activities that engineers/designers use basic sciences, mathematics and engineering sciences to create functional products. The fundamental elements of the design process include designers/engineers (who), design activities/tasks (what), time (when), techniques/tools (how), and locations (where). From this viewpoint, this chapter describes the fundamental ingredient of the product design process (PDP) as a seven-tuple, $PDP = (TE, PE, SE, OE, LE, IE, ME)$, where,

- TE → Design tasks/activities that were carried out to achieve some objectives
- PE → Persons who were involved in at least one design activities
- SE → Time, including the starting/ending time or duration of the activities
- OE → Organizations
- LE → Locations
- IE → Inputs or outputs of the design activities
- ME → Methods, techniques or tools used in the design activities

Based on the above representation, the problem statement of this research is to identify special writing expressions called as named entities (NEs) from a collection of design documents $D = \{D_1, \dots, D_N\}$ and to classify the extracted NEs into pre-defined information categories, namely, TE, PE, SE, OE, LE, IE and ME. Furthermore, considering that the same object might be mentioned by different groups of people using

different vocabularies on particular occasions, each task entity $te_i \in TE$ is further decomposed into a list of mentions $te_i = [tm_1, \ldots, tm_{N_i}]$ that appear in different morphological forms but refer to the same entity object.

5.3 A Hybrid Named Entity Recognition Approach

Figure 5.1 shows an overview of the proposed NER approach. As shown in Fig. 5.1, the proposed NER approach consists of four steps: sentence classification, seed entity generation, entity expansion, and entity clustering. To reduce the human intervention without influencing the accuracy of the recognized NEs, several supervised or unsupervised machine learning techniques are integrated in the four steps.

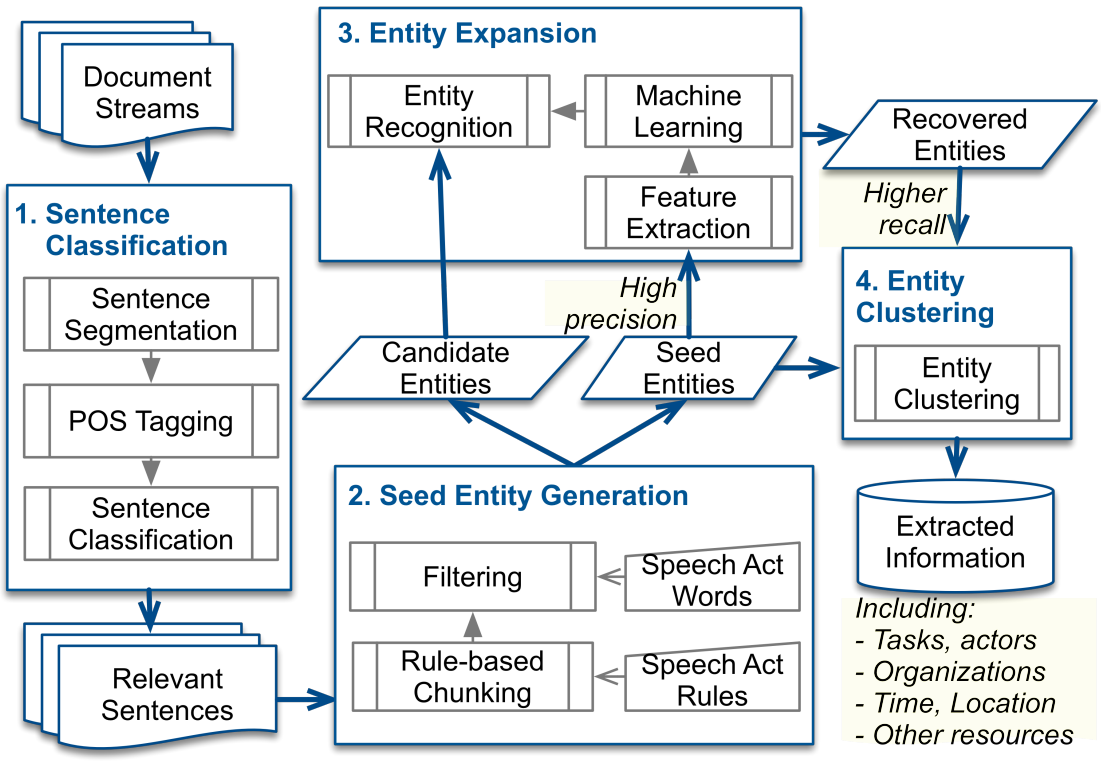


Figure 5.1 A hybrid NER approach for fine-grained process information extraction

Referring to Fig. 5.1, given the input documents, a sentence classifier is trained on a small set of sample sentences. The major function of the sentence classifier is to purify the input data by eliminating sentences that are not relevant to the underlying design process. In the second step, a small set of seed NEs are generated by a set of speech act rules with a little amount of expert knowledge. Next, these seed entities are utilized to grow more general instances of NEs from the input texts. To do so, a kernelised SVM classifier is trained on the seed NEs to automatically learn their discriminative linguistic features. Lastly, all the detected NEs are fed into an unsupervised classifier to automatically find different mentions that refer to the same object.

It is noteworthy that to make sure the whole system in Fig. 5.1 work well, the NEs identified in the step of seed entity generation must be with high precision (might low recall). Because the SVM classifier in next step is trained on the seed entities, the accuracy of the seed entities directly determines the quality of the SVM classifier. The rationale and algorithm design for each step will be detailed in the following subsections.

5.3.1 Sentence Classification

In consideration of the high precision required by the step of seed entity generation, the first step aims to identify sentences that carry semantic meanings relating to the very recorded design process. Only sentences that are predicted as relevant are allowed for further processing.

Because nouns and verbs often bear more semantic meaning for understanding a sentence than other types of words, all the sentences in the documents are simplified by removing words that are not nouns or verbs. After this operation, all the sentences are represented as binary vectors, within which "1" denotes the occurrence of a noun or verb word in a sentence.

Based on the binary representation of the sentences, a Bayes prediction model is trained on a small set of sample sentences. The sample sentences are selected and labeled with a little amount of human intervention. In detail, let D be the whole document set reordered by time, D is automatically sampled with a time interval g. This sampling operation results in a smaller set of documents D'. Similarly, for each document $D'_i \in D'$, n sentences are randomly picked out for manual annotation. Let S'^i be the sentences selected from D'_i, the final size of annotated training dataset is $\sum_{D'_i \in D'} |S'^i| \leq n * |D'|$. Compared to completely artificial processing, the human intervention is largely reduced from $\sum_{D_i \in D} |S^i|$ to $\sum_{D'_i \in D'} |S'^i|$, where $|D'| < |D|$ and $|S'^i| < |S^i|$.

Next, the Bayes classifier is used to predict the relevance of the remaining sentences. Only sentences that are predicted as relevant can be used to generate seed entities in the next step.

5.3.2 Seed Entity Generation by Speech Act Rules

Inspired by the "speech acts" concept, a rule-based NER approach is developed to generate a set of seed entities SE_E for each entity class $E \in [TE, PE, SE, OE, LE, IE, ME]$.

Originally, "Speech acts" is defined as "illocutionary" verbal utterances that have a performative function to present a speaker's intentions, such as promising, ordering, requesting and inviting [143]. In the context of process information extraction, "speech acts" are considered as writing statements bearing semantic meanings, such as execution of design tasks, personnel assignment, requesting for special tools or data, and so on.

On the basis of "speech acts" theory, this work considers verb phrases associated with special verbs (e.g., "submit", "complete", and "use") and noun phrases containing special words (e.g., "input", "output", and "technique") as significant clues to trace the statements about task executions. The selected verbs and nouns of hint function are called as speech act words. Each entity type E in PDP reserves a speech act dictionary W_E. To collect W_E, the same set of sentence samples $\bigcup_{D_i' \in D'} S'^i$ used in the previous step are provided to domain experts or users for annotating a preliminary set of speech act verbs and nouns from their domain. Furthermore, in order to get a more general speech act dictionary, the preliminary W_E is expanded by including their hyponyms using WordNet[2], which is a large lexical database of English.

Given W_E of each entity type, the seed entities are directly matched from the design documents via pattern search. With the help of Stanford CoreNLP[3], a open source tool for natural language analysis, all the verb phrases and noun phrases are identified as candidate

[2] https://wordnet.princeton.edu

[3] http://stanfordnlp.github.io/CoreNLP/

seeds. Next, only noun phrases that contain speech act nouns or follow speech act verbs in W_E are selected to form the seed entities SE_E. If in the following example sentence, "Group 1" implies some involved people, "modify concept paper" is found as a task entity, "technical requirements" refers to some kinds of input data, and "Jan. 01, 2012" is a time entity, if "group", "modify", "requirements", and "Jan" are labeled as speech act words.

Example 1—Group 1 need to modify your concept paper according to the attached technical requirements by Jan. 01, 2012.

5.3.3 Entity Expansion by SVM

The most likely case associated with handcrafted rules is that the recognized NEs are often of high precision but low recall. To improve recall, this step aims to explore more general entities via machine learning approaches. In detail, the seed entities obtained in previous step are used as the training data. In contrast, all the noun phrases (NP) that can not match the speech act rules are candidate entities. The kernerlized SVM is adopted to learn the linguistic context of the seed entities and to apply the learned discriminative features to retrieve more general instances from the candidate entities.

Firstly, the linguistic context of a seed or candidate entity is characterized by a dependency tree, which is constructed by words in a local context. The local context of a noun phrase (NP) is defined as the sequence of words from the end of its preceding NP to its own last word. For example, the sentence in Fig. 5.2 (a) contains two NPs, and their local contexts are highlighted by underlines.

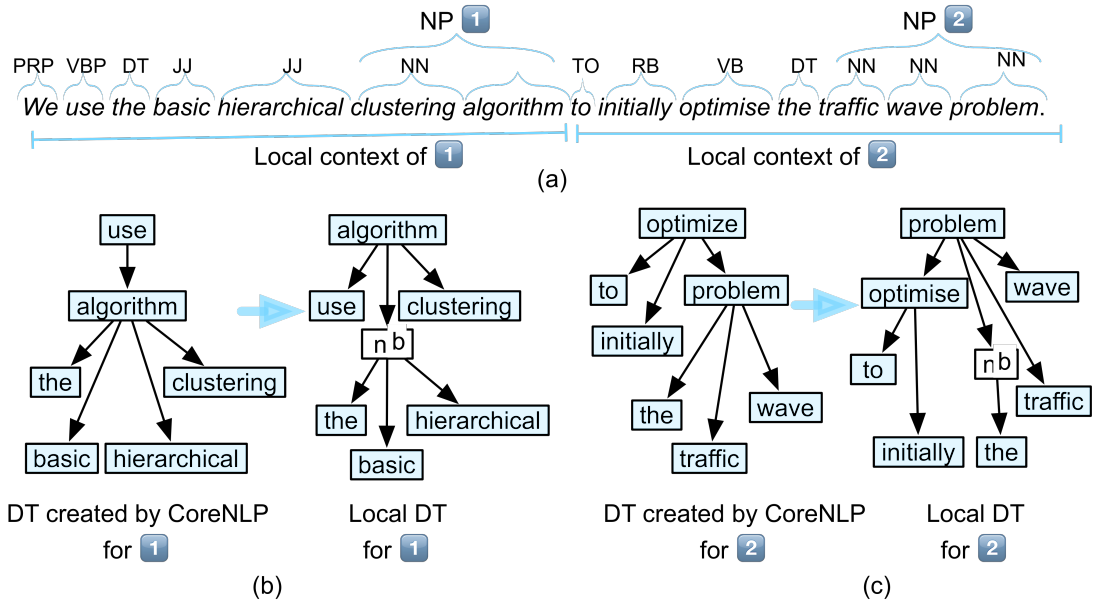


Figure 5.2 Illustration of local dependency tree construction: (a) example of local context, (b) local dependency tree of the first NP in (a), (c) local dependency tree of the second NP in (a)

Based on the local context, the local dependency tree is constructed for each NP to capture its lexical, syntactical and semantic features in a tree structure. Figure 5.2 (b) and (c) illustrate the local dependency tree of the two candidate NPs in Fig 5.2 (a). Each NP including its local context is represented as a tree $T = (N, E)$ with nodes $N = \{n_1, \ldots, n_r\}$ and edges $E \subset N \times N$.

As shown in Fig. 5.2 (a) and (b), each node n_i in T corresponds to a word in the local context. The linguistic feature of each node is further characterized by a set of expressive features, $n_i = (f_1, \ldots, f_F)$, which are compiled in Table 5.1. Each edge e_i in T indicates the dependency relation between the two connected words. The dependency relation is

obtained with the help of CoreNLP[4]. However, different from the traditional dependency tree rooted at a verb, the local dependency tree assumes that the headword (usually, the last word) of a NP is of the most essential significance to reveal its semantic meanings. Therefore, the headword is treated as the root node in the local dependency tree. Other words such as remaining component nouns (if, multi-word NPs), verbs, and adjectives, dependent on the root headword directly or connect to the root headword through a path of dependencies. Besides, an extra blank node n^b is created to degrade the significance level of adjectives from remaining component nouns. By this means, the significance of the words in the local context can be scaled in a hierarchical manner based on their distance to the root.

Given two candidate NPs and their local dependency trees $T_1(root_1)$ and $T_2(root_2)$, a best match based dependency tree kernel $k_{BMDTK}(T_1, T_2)$ is proposed to compute their semantic similarity, on the basis of the structure similarity between T_1 and T_2. Equation (5.1) formulates this tree kernel as:

$$k_{BMDTK}(T_1, T_2) = \Delta(root_1, root_2) \tag{5.1}$$

where, $root_1$ and $root_2$ are the root nodes of T_1 and T_2, and the node kernel function $\Delta(n_1, n_2)$ computes the structure similarity between two sub-trees which are rooted at n_1 and n_2 respectively. Therefore, the tree kernel k_{BMDTK} can be expressed as the node kernel $\Delta(root_1, root_2)$ of the two corresponding root nodes.

[4] http://stanfordnlp.github.io/CoreNLP/

Table 5.1 List of features used for entity recognition

Feature category	Feature	Description	Example
Lexical features	Token	The token in original text	"tokens"
	Lemma	The base form of tokens after lemmatization	"token" for "tokens"
	Prefix	The prefixes of length from 2 to 3	"to" for PRE_2_"tokens"
	Suffix	The suffixes of length from 2 to 3	"ens" for SUF_3_ "tokens"
	Orthography	The binary indicators of tokens containing special symbols besides of letters	ALL_LETTER_UPPER, FIRST_LETTER_UPPER, HAVE_DIGIT, HAVE_DOT
	Word shape	The orthographic pattern of tokens	"AA00a" for "HB25c"
Syntactical features	POS	The part-of-speech tag of tokens	"NNS" for "tokens"
	Distance	The distance from a token to the headword	0, 1, 2, etc.
Semantic features	Clusters	The cluster id of tokens based on WordNet	1, 2, 3, etc.

The computation of the node kernel is given in Eq. (5.2) to Eq. (5.4).

If n_1 and n_2 are leaf nodes:

$$\Delta(n_1, n_2) = \delta(n_1, n_2) \tag{5.2}$$

If n_1 and n_2 are parent nodes:

$$\Delta(n_1, n_2) = \delta(n_1, n_2) + \lambda \sum_{(n_1^c, n_2^c) \in BM(Cn_1, Cn_2)} \Delta\left(n_1^c, n_2^c\right) \tag{5.3}$$

$$\delta(n_1, n_2) = |\cap (n_1, n_2)|/|n_1| \tag{5.4}$$

where, $\delta(n_1, n_2)$ is the feature similarity function, which calculates the proportion of common features two nodes shares in Table 5.1, C_{n_1} and C_{n_2} are the child nodes of n_1 and n_2 accordingly, $BM(C_{n_1}, C_{n_2})$ retrieves the set of best matches between C_{n_1} and C_{n_2} on the basis of nodes' feature similarity, and $\lambda \in [0,1]$ is the significance subtracter.

Based on k_{BMDTK}, the kernel based SVM is adopted to predict the entity type of the candidate NPs. Given all the seed entities in $SE = \bigcup_E SE_E$, $E \in [TE, SE, PE, OE, LE, IE, ME]$, a kernelized SVM classifier is trained on SE firstly. The classifier is then used to predict the entity type $\hat{y}$ of a candidate entity cnp by computing a weighted sum of similarities over the seed entities:

$$\hat{y} = sgn \sum_{se \in SE} \partial_{se} y_{se} k_{BMDPK}(T_{se}, T_{cnp}) + b \tag{5.5}$$

where, $\partial_{se} \in R$ are the weight of the seed entities, as determined by the SVM learning algorithm.

5.3.4 Entity Clustering

The last step aims to find the different expressions that refer to the same entity. It is a common case that different mentions of the same entity tend to have different but synonymic vocabularies because of the varying linguistic environment. For example, John Smith, John, Dr. Smith, and Mr. Smith might be used to refer to the same person, namely to John Smith. To recognize such expressions, the hierarchical clustering algorithm (HCA) is adopted to cluster the mentions of the same entity into groups. Each group can be treated as a single entity in a whole.

The clustering operation is carried out based on the lexical similarity between pairs of entity mentions. It is assumed that the meaning of an entity mention is not only determined by its component words but also influenced by the words around it. Under this assumption, Equations (4.6-4.7) compute the lexical similarity between two mentions, m_1 and m_2.

$$sim(m_1, m_2) = w_{IF} sim_{cos}(IF^{m_1}, IF^{m_2}) + w_{EF} sim_{cos}(EF^{m_1}, EF^{m_2}) \tag{5.6}$$

$$sim_{cos}(F^{m_1}, F^{m_2}) = \frac{\sum_{i=1}^{i=V} v_i^{m_1} v_i^{m_2}}{\sqrt{\sum_{i=1}^{i=V} \left(v_i^{m_1}\right)^2} \sqrt{\sum_{i=1}^{i=V} \left(v_i^{m_2}\right)^2}}, F \in \{IF, EF\} \tag{5.7}$$

where, IF denotes the internal feature of an entity mention, the representation of IF is a binary vector of words that compose a mention; on the contrary, the external feature EF is a binary vector of words appearing before and after a mention; accordingly, w_{IF} and w_{EF} denote the weights of IF and EF; V is the vocabulary size. To reduce the ambiguousness brought by the different morphological forms of the same word for grammatical reasons, all the words in IF and EF are transformed back into their base forms using the lemmatization package provided by NLTK[5]. Within each feature space (IF and EF), the lexical similarity is calculated by the cosine metric in Eq. (5.7).

5.4 Experimental Results and Discussions

5.4.1 Dataset and Data Preprocessing

The proposed algorithms were tested on the same email dataset from the TWP project introduced in Section 3.3. Before the experiment investigation, the original dataset was

[5] http://www.nltk.org

cleaned by deleting cite text from earlier messages within the email thread. The cleaned dataset was then manually tagged by annotating process-relevant entities using the BIO (Beginning, Inside and Outside) labeling scheme. All the tagged emails served as the baseline in assessing the performance of the proposed hybrid NER approach.

5.4.2 Performance Measures

The experiments were carried out in four steps to assess the performance of each algorithm in the proposed NER approach shown in Fig. 5.1. There are three importance assumptions needing for verification:

- The Bayes classifier can eliminate the process-irrelevant sentences with high accuracy,
- The speech act rules can generate the seed entities with high precision,
- The tree kernel based SVM can improve the recall while keeping the precision.

Precision, recall and their harmonic combination (F_1-value) were calculated by comparing the recognized entities with the artificial annotations. The MUC evaluation metrics [144], which scores a NER system by assessing its ability to find both the correct type (TYPE) and the exact text (TEXT), were adopted as the guideline for performance measurement. However, as the proposed NER approach treats all the noun phrases as candidate entities, the TEXT aspect of MUC was ignored in this experiment. Therefore, a correct TYPE was credited if an entity was assigned the correct type, regardless of boundaries as long as there was an overlap.

5.4.3 Results and Discussions

5.4.3.1 Performance of Sentence Classification

The first experiment aims to evaluate the performance of the Bayes Classifier in eliminating the process-irrelevant sentences. It is noteworthy that, as the performance of the sentence filter has a direct impact on the performance of seed entity generation, the sentence classification needs to produce results of high accuracy.

Three Bayes classifiers, namely, Bayes_BOW, Bayes_NN and Bayes_VB_NN, are compared in Fig. 5.3. The three classifiers are distinguished from each other by the category of words used to vectorize a sentence, with Bayes_BOW using the bag of words, Bayes_NN using nouns only, and Bayes_VB_NN using both verbs and nouns. The influence of the training sample size on the prediction result is also investigated by varying the time gap g used to generate sentence samples for human annotation.

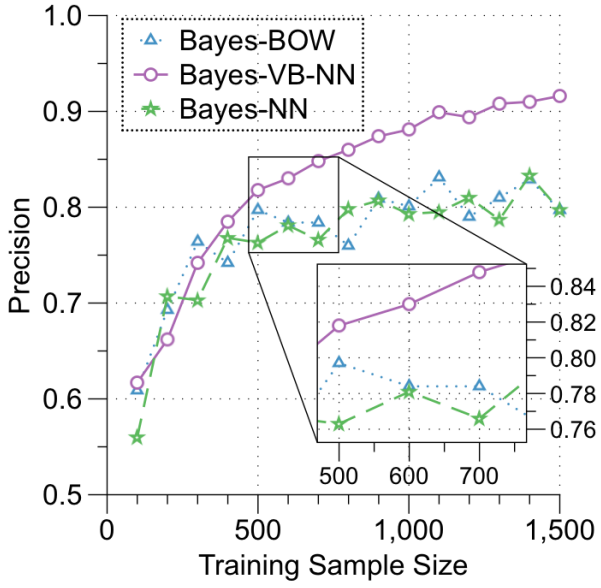


Figure 5.3 Performance of sentence classification

Figure 5.3 reports the precision results of the three classifiers by varying the training sample size. It can be observed that Bayes_VB_NN greatly outperforms the other two classifiers when the training sample size is larger than 400. Meanwhile, Bayes_BOW and Bayes_NN show a relatively matched performance, regardless of the training sample size. This comparison result reveals that both verbs and nouns play a much more significant role in bearing a sentence's semantic meanings than other types of words. Therefore, only using verbs and nouns can reduce the ambiguousness brought by other types of words, e.g., stop-words that usually refer to some extremely common but meaningless words in a language.

The impact of the training sample size on the prediction accuracy is reflected by the growing tendency of the precision lines. According to Fig. 5.3, the line resulting from Bayes_VB_NN shows that the precision keeps a fast increasing trend with a smaller number of annotated sentences, but steadily, this growing rate slows down after the training sample size is increased to about 600. Although the precision tendency reveals that a larger training sample size is more helpful to improve the performance, much more workload is required simultaneously. In order to balance the precision required by seed entity generation and the workload for labeling the training samples, the training sample size for the following experiments was selected as 600, with a precision about 84% and a workload about 3 hours of one person.

5.4.3.2 Performance of Seed Entity Generation

The second experiment aims to verify the hypothesis that the seed entities obtained by the speech act rules are of high precision. As defined in Section 5.2, seven classes of process relevant entities are considered, namely, design task TE, timestamp SE, person PE, organization OE, location LE, input/output information IE, and technique/tool ME. For each category, the performance is measured using precision, recall and F_1-value.

Table 5.2 Examples of speech act words

Entity Category	(# of VB, # of NN)	Examples of speech act verbs	Examples of speech act nouns
TE	(75, 20)	finish, set, check, revise, work, settle, develop, build, validate, modify, simulate, complete, design, improve, ...	application, simulation, optimization, collection, design, issue, task, problem, …
SE	(0, 48)	--	pm, am, Monday, …, January, February, …, week, tomorrow, …
PE	(0, 143)	--	Prof., professor, Dr., student, organizer, staff, Mr., …, names from email headers, …
OE	(0, 6)	--	department, organization, team, panel, LTA, ECE
LE	(1, 5)	locate	office, workshop, road, Clementi, Chang, crossroad
IE	(22, 15)	provide, forward, supply, output, submit, deliver, result, generate, …	input, data, information, requirement, output, result, report, form, file, …
ME	(7, 11)	use, apply, employ, propose, adopt, utilize, …	solution, method, technique, approach, algorithm, platform, tool, S3G, means, …

For better illustrating the "performative" function of the speech act words on each entity category, Table 5.2 lists some examples of the speech act nouns and verbs selected from the

600 manually annotated sentences. According to Table 5.2, all the selected words are of strong intentions. In addition, the numbers of the speech act words show that special verbs might be the preferred choice than nouns when engineers express their intentions of something done or to be done. On the contrary, some nouns of particular functional meanings are more likely to be used for expressing entities of time, person, organizations, and locations.

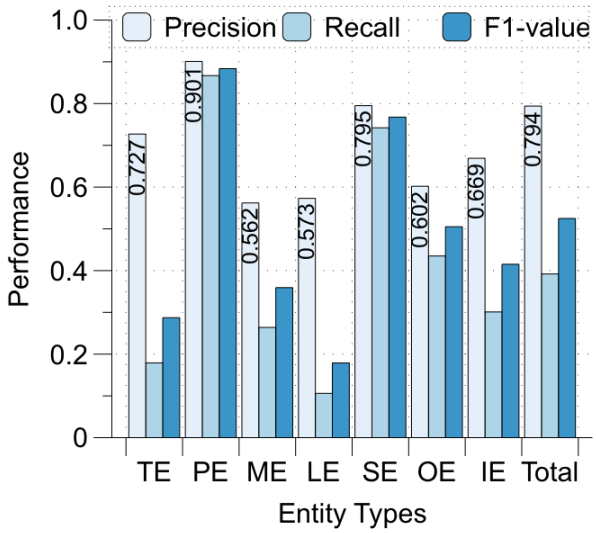


Figure 5.4 Performance of seed entity generation

Figure 5.4 reports the performance of the speech act rules in terms of precision, recall, and F_1-value. From Fig. 5.4, the first observation is that except for PE and SE, which have relatively close results on precision and recall, remaining entity types show a much higher precision score than recall. This phenomenon is well aligned with the conclusion drawn by existing studies on rule-based NER approaches. In addition, the performance difference between entity types reveals that the speech act rules performed much better on identifying four entity types, i.e., TE, PE, SE, and IE, with their precision scores above 0.7. In contrast,

relatively lower accuracy was obtained for ME (0.562) and LE (0.573). This imbalance situation might be caused by the less definitude of speech act words selected for the two categories of NEs.

The overall precision over the seven entity types, as shown in the last column in Fig. 5.4, is 0.794. This finding suggests that the seed entities are satisfied for being used as training data for expanding more general entities because the most concern in the step of seed entity generation is the overall performance.

5.4.3.4 Performance of Entity Expansion

Two significant aspects are investigated in this experiment: the ability of the SVM classifier for retrieving more entity instances and the ability of the proposed local dependency tree for capturing the discriminative features of the seed entities. Based on the proposed dependency tree kernel BMDTK, two kernelised machine learning approaches, K-nearest neighbors (KNN_BMDTK) and Support Vector Machine (SVM_BMDTK), were tested. For comparison purpose, this experiment also tested two baseline approaches that characterize a candidate NP by using the bag of words in its local context, named as KNN_BOW and SVM_BOW respectively. The parameters were set as $K = 7$ for KNN_BMDTK and KNN_BOW, and $\lambda = 0.75$ for the kernel function $k_{BMDTK}(T_1, T_2)$.

The entities expanded by all the above algorithms were compared to the seed entities generated by the speech act rules (SAR). Figure 5.5 plots the results in terms of precision, recall, and F_1-value.

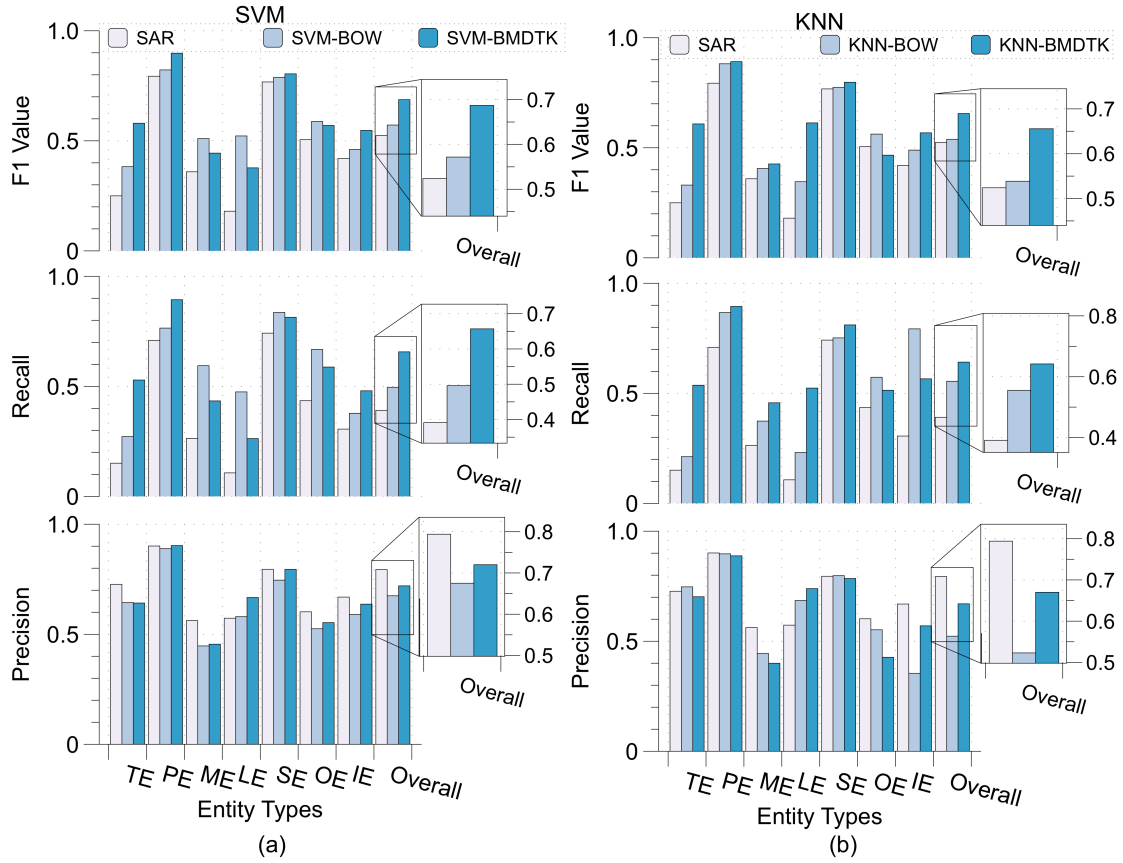


Figure 5.5 Performance of entity learning: (a) performance of SVM classifier, (b) performance of KNN classifier

From Fig. 5.5, it can be observed that no matter which feature is used, BOW or BMDTK, both KNN and SVM improve the overall F_1-value by increasing the recall, though the precision is slightly decreased. For example, compared to the SAR, SVM-BMDTK improves the overall recall from near 0.4 to slightly above 0.65, thus upgrades the overall F_1-value from about 0.52 to near 0.7. This observation demonstrates that by learning the local context feature of the seed entities, more general entity instances are successfully retrieved from those ones that can not be identified by speech act rules. However, as more general instances are founded, noises also are included, which influents the final accuracy

negatively. For example, compared to SAR, the overall precision score is decreased from about 0.8 to below 0.75 and around 0.67 by SVM-BMDTK and SVM-BOW respectively.

A close comparison between the performance of BMDTK and BOW reveals that the local dependency tree kernel performs better than BOW in capturing the discriminative features of the entities. This conclusion is evidenced by the increased overall performance of both machine learning algorithms, KNN and SVM. In the case of SVM, shown in Fig. 5.5 (a), DMDTK outperforms BOW in all the three performance measurements, with much higher F_1-value (near 0.7 vs. about 0.57), much higher Recall (slightly above 0.65 vs. around 0.5), and slightly higher precision (about 0.72 vs. about 0.67). The same case also happens to KNN, shown in Fig. 5.5 (b). However, when comparing the performance in each entity category, it is interesting to observe that BOW excels BMDTK on two entity types, i.e., ME (about 0.3 higher F1-value than SVM) and LE (about 0.6 higher F1-value than SVM). The cause for this phenomenon might be the lower precision of the seed entities of the two categories, which influence the quality of the SVM classifier when handling with the two type of entities.

5.4.3.5 Performance of Entity Clustering

Table 5.3 reports the number of the individual task entities before and after clustering. Two datasets are compared: one is the original dataset with the manually annotated entities, and one is the dataset automatically annotated by the proposed SVM-BMDTK classifier.

Table 5.3 Number of individual task entities

	# of Ms	**# of IMs**	**# of IMCs**
Annotated data	2289	1340	61
SVM-BMDTK	1948	1211	53

In Table 5.3, the second column (# of Ms) reports the total number of the entity mentions that are identified as task entity in both datasets. It is observed that the proposed approach generated a relatively equivalent volume of task entities, with 341 less than the manually-labeled ones. The number of individual entity mentions, as shown in the third column (# of IMs), is counted by ignoring repetitive entities. Here, the gap between the artificial and automated entities is further reduced to 129. The last column presents the number of the entity clusters. The further decreased gap in the IMCs column indicates that the proposed approach is able to find instances (or mentions) for most task entities (entity clusters), although it might be not competent enough to identify all the mentions for per task entity.

Figure 5.6 gives some examples of the mentions that are clustered in the same entity group. It can be easily observed that most mentions in the same group have a very close meaning. This observation indicates that the proposed approach is able to find the different linguistic expressions of the same named entity. Therefore, each cluster could be treated as a single entity for more complicated process analysis.

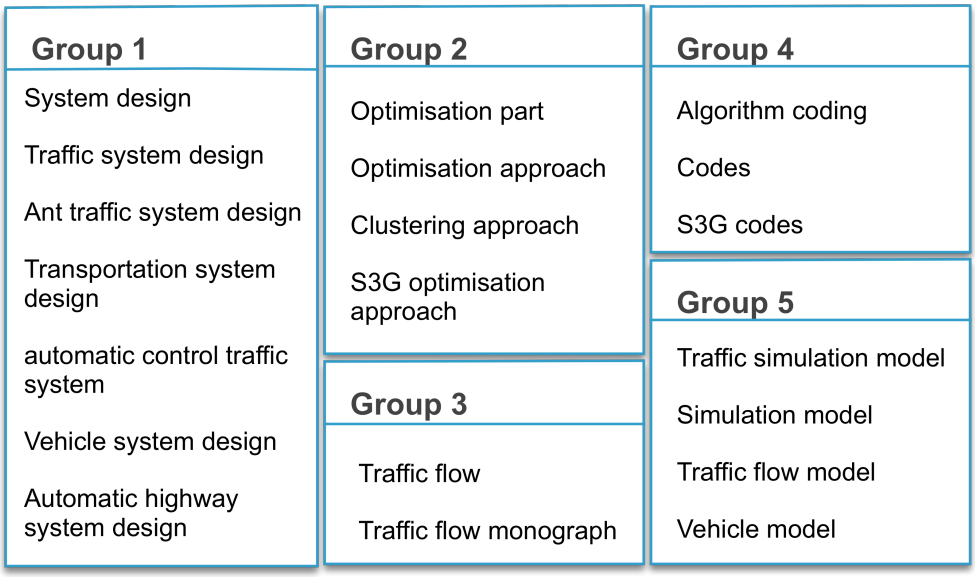


Figure 5.6 Examples of entity clusters

5.5 Summary

In this chapter, a hybrid NER approach was proposed to annotate design information in textual data at a fine-grained level. The annotated entities can be viewed as the fundamental elements that compose the underlying design process. By taking advantage of several timely techniques in machine learning, text mining, and natural language processing, the human intervention in creating training data has been controlled to an acceptable range. In addition, the comparison between the automated and artificial NEs shows an impressive performance with the proposed NER approach. Based on the detected NEs, more advanced, complex information extraction techniques can be applied to support decision makers in learning from the experience archived in the design documents.

CHAPTER 6 EVENT DETECTION BY ENTITY RELATION EXTRACTION

6.1 Introduction

In previous two chapters, the information extraction module of the PKDT system is discussed. So far, the extracted topics and named entities have been treated as independent metadata scattered in design documents. To go beyond the information that can be provided by such metadata, this chapter aims to detect design events from the extracted process information by finding their semantic relations. The detected design events could step toward a more structured representation of the target design process embedded in the design documents.

Design events in the product design process refer to observable occurrences of designers/engineers, tasks, locations, and times. From the viewpoint of relation extraction, a design event can be regarded as a higher-order relation referred from a set of binary relations among the involved entities. As discussed in Section 2.4, most of the existing higher-order entity relation extraction (ERE) approaches mainly consist of two steps: identifying confident binary relations using machine learning algorithms and assembling the binary relations based on prior knowledge. Therefore, these approaches have a heavy dependence on the training data used to learn the binary relation classifier, and the types of higher-order relations that can be detected are significantly constrained by the rules used to assemble binary relations. In this context, they are not suitable for design event detection as

the size of design events as well as the dependency strength within design events are ever changing due to the high flexibility of product design process.

To tackle the above issues, this chapter presents a graph partition based higher-order ERE approach to detect design events from design documents in an unsupervised manner. Unlike traditional higher-order ERE approaches, the proposed approach recognizes confident binary relations to the utmost according to the distance of the NE pairs, and then detects design events by finding the maximal NE cliques based on their binary relations. Therefore, noisy NEs that have weak relations to its neighbors could be eliminated by the graph density used to find the maximal NE cliques.

The symbolic representation of the relevant concepts is stated in Section 6.2. The proposed event detection approach is presented in Section 6.3. The experimental results as well as some examples of the detected design events are given in Section 6.4. Lastly, a simple conclusion is drawn in Section 6.5.

6.2 Problem Statement

On the basis of the entity types in Chapter 5, a design event is defined as a graph $EG = (V, v_0, t_s, t_e\ E)$, where

- V $\rightarrow$ Each vertex $v \in V$ denotes a named entity and the entity type of v belongs to $\{TE, PE, SE, OE, LE, IE, ME\}$;
- v_0 $\rightarrow$ The graph is centered at v_0, $v_0 \in V$, and the entity type of v_0 must be TE (task entity);

- $t_s, t_e \rightarrow t_s, t_e \in V$ are the starting and ending time of an event, and their entity types must be SE (time entity);
- $E \rightarrow$ Each edge $e \in E$ denotes a relation between a normal vertex and the center vertex v_0, therefore, $E \in \{v_0 \times V\}$.

Based on the above definition, the problem of design event detection is transformed into a higher-order ERE problem. The relations in a design event are jointly determined by at least two types of entities. Therefore, the design event detection differs from most relation extraction problems that focus on binary relations. More precisely, for each design document, the proposed ERE approach aims to extract a set of design events recorded, and all the entities in a design event have a direct or indirect relation to the central task entity. Moreover, it is nature that two design events can share the same entities. For example, a person can be in charge of two design activities at the same time. Therefore, it is also assumed that two design events extracted from the same document can overlap on some vertices except the central one.

6.3 A Graph Partition based ERE Approach

Figure 6.1 illustrates the workflow of the proposed higher-order ERE approach for design event detection. As shown in Fig. 6.1, the proposed ERE approach consists of three steps: direct binary relation detection, indirect higher-order relation detection, and post-processing. The first step aims to identify the binary relations between any two types of entities via matching patterns in the sentences. This would result in an intermediate graph

within which binary relations exist between any two connected entities. Next, the graph of binary relations is factorized into several event graphs by finding the maximal cliques centered at each task entity. The primary advantage of using graph partition is that it allows events to share the same entities in an unsupervised manner. Finally, the post-processing step selects the valid cliques and formats them in the form of the design event graph defined in Section 6.2.

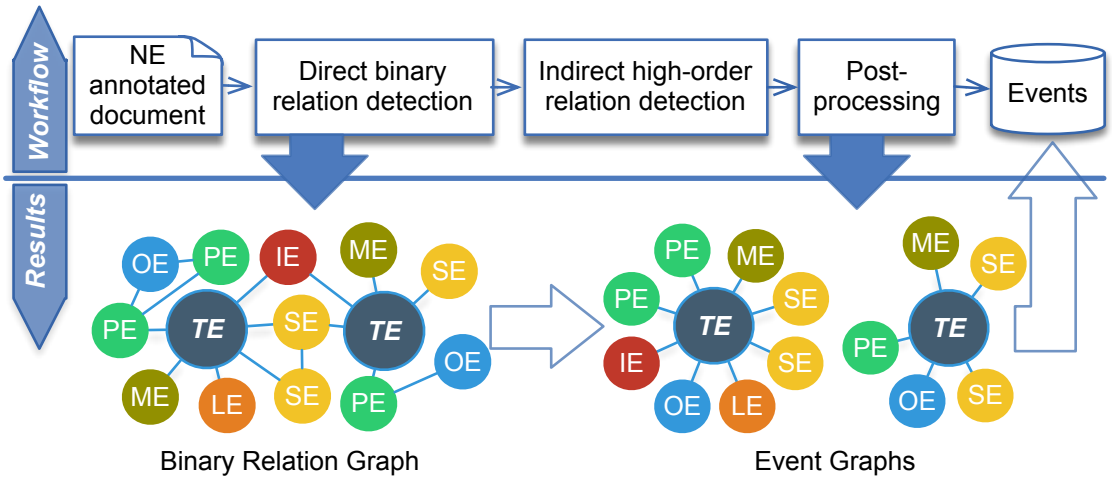


Figure 6.1 Workflow of event detection

6.3.1 Direct Binary Relation Detection

The proposed three-stage event detection approach starts by identifying pairs of entities that appear to have a binary relation of interest and high confidence. These entity pairs can then serve as edges in a graph, based on which more complex relations can be inferred.

A binary relation is defined as a tuple $br = (e_1, r, e_2)$, where e_1 and e_2 are entity mentions, $type\_of(e_1 or\ e_2) \in ET\ and\ ET = \{TE, PE, SE, OE, LE, IE, ME\}$, and r is the relation. Based on the definition of binary relation, the higher-order relation in the

previously defined event graph is factorized into a set of binary relations, and all the possible binary relations are listed in Table 6.1. The biggest advantage of this factorization operation is that the number of possible binary relations is dramatically reduced. More specifically, let $|et_i|$ be the number of entities under an entity type, the number of possible binary relations is $\sum_{et_i,et_j \in ET \ \& et_i \neq et_j} |et_i| * |et_j|$, which is much smaller than the number of possible higher-order relations, $\prod_{et \in ET} |et|$.

Table 6.1 Types of binary relations

Relation Type	Example	Relation Type	Example
TE_TE	(collect data, analyze data)	TE_PE	(collect data, John)
TE_SE	(collect data, 06/01/2011)	TE_OE	(collect data, Corp.)
TE_LE	(collect data, Queen Road)	TE_IE	(analyze data, vehicle data)
TE_ME	(analyze data, software)	PE_PE	(John, David)
PE_OE	(David, Corp.)	PE_ME	(David, software)
SE_SE	(06/01/2011, 08/01/2011)	OE_LE	(Corp., Queen Road)
OE_ME	(Corp., software)	LE_LE	(Queen Road, King Road)

A simple pattern search approach is used to match binary relations sentence by sentence. More generally, high-confident binary relations are likely to exist between entity pairs that satisfy all the following rules:

- Rule 1: Two entities must be mentioned in the same clause;
- Rule 2: Two entities are directly connected in the sentence dependency tree;
- Rule 3: The type of two entities must be consistent with one relation in Table 6.1;
- Rule 4: The sentence or clause is in present tense;

- Rule 5: No negative words (e.g., don't, not) exist between two entities.

It's worth to mention that Rule 4 is introduced to find events that are being done or will be done, and Rule 5 is for eliminating negative relations. An example of binary relation extraction is shown in Fig. 6.2. Referring to Fig. 6.2, four entities are detected in the example sentence. If binding any two of the four entities together, there might be six candidate binary relations. After matching the above rules on these candidate relations, two of the six relations violate Rule 2 because both are interrupted by a third entity in the dependency tree, and one candidate relation is filtered out by Rule 3. At last, only three of the six candidate relations satisfy all the five rules.

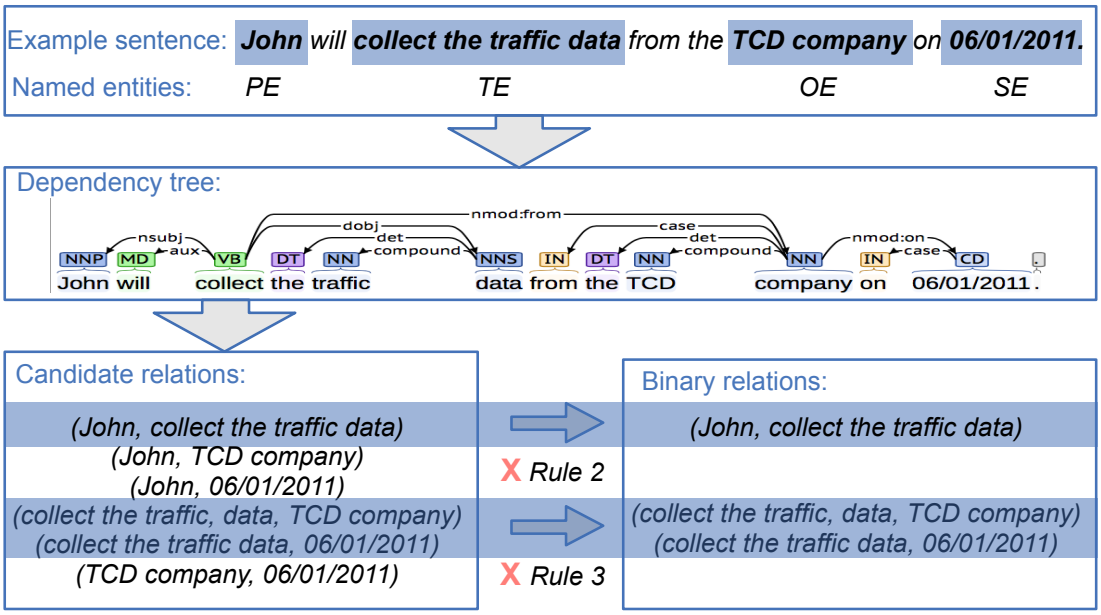


Figure 6.2 Example of binary relation detection

There are several advantages by using pattern matching other than training a classifier to classify all possible relations. Firstly, the pattern matching approach has no requirement for training data. Secondly, the computational cost of the pattern matching approach is linear

to the number of the sentences in a document. Last and most significantly, the goal of the binary relation detection in the proposed ERE approach is to find all possible binary relations other than to correctly classify all the extracted relations because the less-confident relations can be filtered out according to the clique density in the next step.

6.3.2 Indirect Higher-Order Relation Detection

The second step advances to recognize design events by growing higher-order relations from the binary relations obtained in the first step. To do so, all the entity pairs that have a binary relation are connected in an undirected graph $G = (V, E)$, where the vertices in V are the entities mentioned in each document, and the edges in E are the binary relations of the entities. In addition, the weight of each edge, $w(e \mid e \in E)$, is presented by the frequency that the corresponding binary relation is mentioned in a document. Based on this binary relation graph, design events are automatically detected by finding the maximal clique centered at each task entity.

Given a binary relation graph $G = (V, E)$, a clique G' that is centered at a task entity v_0 is a sub-graph of G in the form of $G' = (V', v_0, E')$, where $V' \subseteq V$, $E' \subseteq E$, $v_0 \subseteq V'$, $type(v_0) = TE$, and for each $v \in V' - v_0$, there is at least one path from v to v_0. A clique $MG = (V^{MG}, v_0, E^{MG})$ is the maximal clique centered at v_0 if there is no other clique $G' = (V', v_0, E')$ that $density(G') > density(MG)$. The function $density(G')$ shown in Eq. (6.1) computes the density of a clique using the mean of the edge weights.

$$density(G') = (\sum_{e \in E'} w(e))/|V'| \quad (6.1)$$

To find the maximal clique for each task entity, the simplest approach is enumerating all the possible cliques, computing their density, and then selecting the maximum clique. Unfortunately, the real problem is that the number of cliques grows exponentially with the number of entities that are directly or indirectly connected to a task entity. Enumerating all the possible cliques will result in a high time consumption.

To overcome the above problem, this step adopts a greedy strategy to find the maximal cliques approximatively. The main idea is to greedily expand an initial clique in the direction that the clique density increases. Figure 6.3 gives the proposed algorithm in detail. As shown is the 4th line of Algorithm 6.1, the algorithm starts by creating a small clique MG^o in which all the entities have a direct relation to the central task entity. Next, for each iteration shown in the 5th-7th lines, a new node is added to the target clique if it is connected to at least one node in MG^o, and the density of the new clique MG' is larger than the density of MG^o. By this means, nodes with strong connections to its neighbors would be added to the maximal clique, and nodes with weak connections to its neighbors would be eliminated. The whole algorithm stops when no more node can be included. Each maximal clique is an event as a whole.

The computational complexity of algorithm 6.1 is almost linear to the number of entities in the binary relation graph. However, one problem with this algorithm is that the final cliques might not be the maximal ones but approximations.

Algorithm 6.1 A event detection algorithm (Part I)
Inputs: $G(V, E)$ is the binary relation graph, δ is the stop criteria for eliminating a node

1: **Procedure** EVENT_CLIQUE_DETECTION(G, δ)
2: Initialization: $MG\_list \leftarrow \phi$
3: **For** each task entity $v_0 \in V$ and $type(v_0) = TE$ **do**
4: Create a initial clique $MG^0 = (V^0, v_0, E^0)$, where $V^0 \in V$, $v_0 \in V$, and $E^0 = \{(v', v_0) | v' \in V^0 - v_0 \; and \; (v', v_0) \in E\}$
5: **For** each $v^{new} \in \{v | (v, v') \in E \; and \; v \in V - V^0 \; and \; v' \in V^0\}$ **do**
6: Create a new clique $MG' = (V', v_0, E')$, where $V' = V^0 + v^{new}$ and $E' = E^0 + \{(v^{new}, v') | v' \in V^0 and \; (v^{new}, v') \in E\}$
7: $MG^0 \leftarrow MG'$ if $density(MG') - density(MG^0) >= \delta$
8: **End for**
9: Save a maximal clique by appending MG^0 to MG_list
10: **End for**

Figure 6.3 A graph decomposition algorithm for event detection

6.3.3 Post-processing

Another problem with the above algorithm is that it might output maximal cliques that have very small size in nodes because it only considers the local density of the cliques around each task entity. Generally, there are three situations that cause a maximal clique is very small: 1) there is no actual design event that can match with the maximal clique, 2) there is an actual design event matching with the maximal clique, but the maximal clique fails to capture insufficient information of the design event, 3) an actual design event itself is small, and the maximal clique correctly reflect the design event. The maximal cliques created in the first two situations would result in noisy or invalid design events.

To make sure each maximal clique represents a valid event of sufficient information, the post-processing operation shown in Fig. 6.4 is applied to filter the noisy and invalid cliques off. Firstly, each candidate event in the list of the maximal cliques is weighted by

the sum over the weight of the edges. Next, the 3rd line rearranges all the candidate events in a descending order. In the last line, the cutoff ρ is used to select cliques whose weights fall into the $\rho\%$ top rank. By this means, the algorithm in Fig. 6.4 only returns maximal cliques that have a large size in nodes or very strong edges as the valid events.

Algorithm 6.2 A event detection algorithm (Part II)
Inputs: MG_list is the list of the maximal cliques, ρ is the cutoff for eliminating noisy cliques in MG_list

1: **Procedure** EVENT_GRAPH_SELECTION(MG_list, ρ)
2: Weight each candidate event $MG_i \in MG\_list$ using following equation:
$$gw(MG_i) = \sum_{e \in MG_i} w(e)$$
3 Rearrange MG_list descendingly in according to $gw(MG_i)$
4: Save top ranked events into EG_list, making
$$\sum_{eg \in EG\_list} gw(eg) / \sum_{mg \in MG\_list} gw(mg) \approx \rho$$

Figure 6.4 The post-processing operation for event graph selection

Lastly, according to the definition of design event in Section 6.2, all the valid event graphs selected by Algorithm 6.2 are further normalized by replacing the path from the central task node to each indirectly-connected node by a single edge. The weight of the new edges is the smallest edge weight in the corresponding path. In addition, the starting and beginning times of each design event is simply set as the minimal and maximal time indicated by the time entity nodes. If no time entity node is included in a maximal clique, the creation time of the corresponding document is used in place.

6.4 Experimental Results and Discussions

6.4.1 Dataset and Performance Measures

The email dataset of the TWP project introduced in Section 3.3 was again used to test the proposed event detection method. 656 events were extracted from the 569 emails.

Considering the number of the extracted events and the flexibility of entities involved in each event, it is difficult to evaluate all the events manually. Therefore, 30 documents were randomly selected and manually annotated. The selected evaluation dataset contained 129 sentences, 406 entities, and 59 events.

It is important to note that a detected design event could consist of several entities of different types. In this context, a detected event was considered correct if and only if at least 50% of the entities in it were consistent with the entities in the manually-annotated events. Based on this, the performance results of the proposed design event detection method were reported in terms of precision, recall, and F-value.

6.4.2 An Example of Event Detection

To give an intuitive feeling of the events detected, an example is given in Fig. 6.5. The example document segment contains 12 sentences. Referring to the entities highlighted in bold, at least one entity is detected in each sentence. From the binary relation graph shown in Fig. 6.5, seven entities are recognized as task entities, namely, make group, report progress, redefine problem, be issue, adopt transportation system, target aspect, and shape project. After graph partition and graph selection, three of the seven task entities are top ranked and survived as valid events with sufficient information. From the three extracted events in Fig. 6.5, it can also be observed that entities within an event are usually mentioned in different sentences, rather than all in single sentence. This also indicates that using binary relation extraction alone can not successfully handle such complex relations.

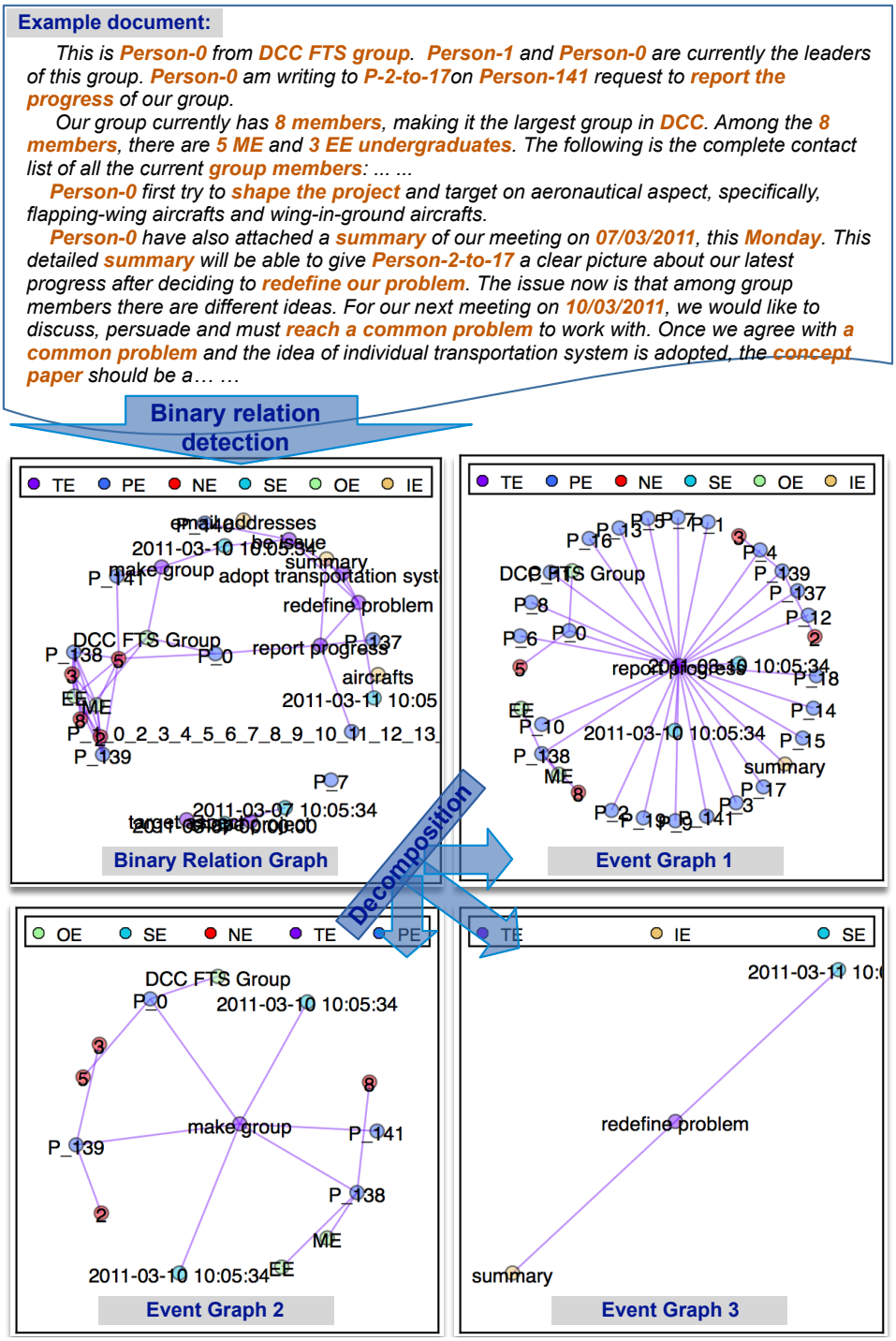


Figure 6.5 Example of event detection

6.4.3 Results

Three event detection approaches were compared:

- TRMEC: the proposed method in this chapter, which only uses the top ranked maximal cliques to construct event graphs.
- MEC: uses all the maximal cliques to construct event graphs.
- DREC: uses cliques in which all the entities have direct relation to the central task entity to construct event graphs.

The performance of the above three methods are reported in Fig. 6.6 in terms of precision, recall and F-value.

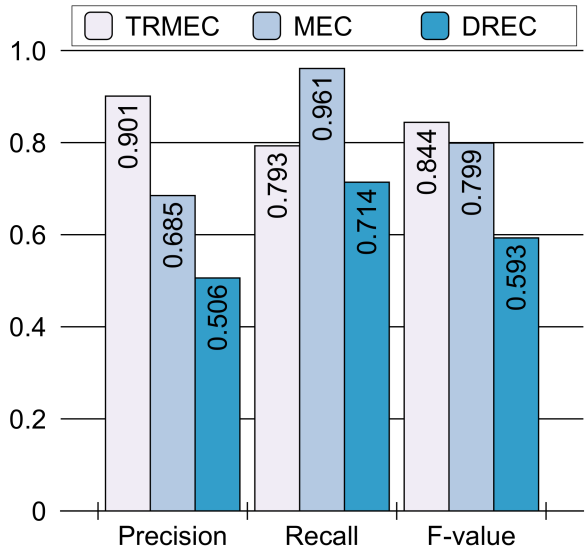


Figure 6.6 Event detection results

According to Fig. 6.6, it can be observed that the method based on top-ranked maximal cliques shows the highest precision and F-value, valued at 0.901 and 0.844 respectively. This result is very positive to indicate that the proposed event detection method can not only correctly eliminate noisy events, but also keep valid events efficiently. In contrast, the TRMEC method was defeated by the MEC method in term of recall. The reason is that the

MEC method recognized all the maximal cliques as events, which can result in relatively higher recall but poor precision. In addition, Fig. 6.6 shows that the method based on binary relations performed worst. This finding is consistent with the observation previously obtained from Fig.6.5 that the complex relations in an event are mostly mentioned in several sentences. Therefore, methods only based on binary relations can not handle such non-sentential relations.

6.5 Summary

In conclusion, this chapter presents a graph partition based higher-order ERE approach for detecting design events from design documents. The main idea of the proposed approach is to decompose the complex relations in an event into several binary relations and reconstruct the event by finding the maximal cliques centered at each task entity. There are several advantages by using graph decomposition. Firstly, it is unsupervised, without any requirement for training data. Secondly, the graph partition algorithm is simple, almost linear to the number of the entities in a document. Lastly, it well utilizes the local relations, which are more confident, to construct complex relations that are hidden in different sentences. The proposed method was tested on a real-life dataset, which showed very positive results.

CHAPTER 7 HIERARCHICAL PROCESS MODEL DISCOVERY

7.1 Introduction

Based on all the information that have been extracted in previous chapters, this chapter aims to automatically model the underlying design process recorded in the design documents from the viewpoint of workflow logic. The discovered workflows specify in which order the design activities have been executed on the basis of reality. Such a capability of discovering the realistic workflows can be a valid help in analyzing process performance, managing and reusing process knowledge.

It is noticeable that due to the inherent flexibility of the product design process, design activities are often carried out in a somewhat loose manner. Because there are no specific execution paths that are strictly defined in advance, designers often change or create new design activities at runtime. As a consequence, the flat models produced by most traditional process mining techniques tend to be large, complex, and difficult for understanding when they are used to model the less-structured product design processes. In order to reduce the complexity as well as to improve the understandability of the design process model, a hierarchical description that can provide different degrees of abstractions is typically desired.

To tackle the above problem, two hierarchical process mining approaches are presented in this chapter, i.e., bottom-up and top-down process mining. Both process mining approaches aim to automatically discover the product design process from the design

documents and to present the discovered process model in a hierarchy structure. The discovered hierarchy structure decomposes the entire design process into functional modules hierarchically. Moreover, in order to describe the design process with different degrees of details, modules can be refined into detailed transitions in a more specific layer, and small modules also can be merged into larger modules in a more abstract layer. To construct the hierarchy structure, the two proposed process mining approaches adopt two opposite strategies, with one from specification to generation and the other from generation to specification.

Section 7.2 defines the hierarchy structure of product design process. Section 7.3 and Section 7.4 present the bottom-up process mining and the top-down process mining respectively. Section 7.5 illustrates and compares the results of the two process mining approaches using a real-life case study. Section 7.6 concludes.

7.2 Problem Statement

This section introduces the formal representation of the hierarchical process model. Let P be a design process, the flat workflow graph of P is denoted as $\mathscr{W}(P)$. $\mathscr{W}(P)$ is a tuple (A, E, A^s, A^e, c), where A is a finite set of design tasks, A^s and A^e are the set of starting and ending tasks, and $E \subseteq (A - A^e) \times (A - A^s)$ denotes the precedence relations among the tasks in A. The edges in E define the potential sequence in which design tasks have been executed. In addition, it is the normal case that design tasks are enriched with some kind of restrictions or attributes, for example, the execution duration of a task, the

person in charge of a task, and the resources utilized in a task. Therefore, the function $c(a|a \in A)$ is introduced to add such attributes to design tasks.

Extending the above definition, a hierarchical process model is denoted by $\mathscr{H}(P)$, which is a tuple $(L, l^{top}, l^{bottom}, \mathscr{D})$, where $\mathscr{W}(L^i) = (A^i, E^i, A^{i,s}, A^{i,e}, c^i)$ is an abstracted workflow graph of P in the i[th] layer of the hierarchical process model, and $\mathscr{D}(A^i) \subseteq (A^i \times A^{i-1})$ is a function that decomposes a task in the i^{th} layer into a set of sub-tasks in the $(i-1)^{th}$ layer. In other words, each $a^i \in A^i$ can be seen as an abstraction of a set of smaller sub-tasks $C^{i-1} \subseteq A^{i-1}$ that are highly correlated in the $(i-1)^{th}$ layer. In turn, the sub-tasks in C^{i-1} add details to a^i. By this means, each abstraction layer describes the product design process with different degrees of details. In addition, l^{top} presents the most abstract view at the top of $\mathscr{H}(P)$, and l^{bottom} provides the most concrete details at the bottom of $\mathscr{H}(P)$.

Figure 7.1 shows an example of the hierarchical process model. Details on mining the hierarchical process model from design documents are presented in the rest of this chapter.

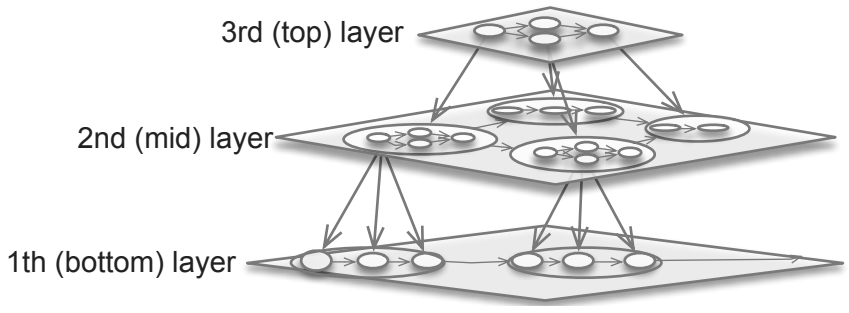


Figure 7.1 Example of hierarchical process model

7.3 Approach 1: Hierarchical Process Mining from Bottom to Top

The bottom-up process mining approach is sketched in Fig. 7.2. The whole system starts by detecting the design events from the design documents using the NER approach presented in Chapter 4 and the event detection approach presented in Chapter 5. Next, the detected design events are saved in a XML file chronologically, called as event log. Each design event corresponds to a record in the event log file and is associated with a set of attributes, e.g., event name, starting and ending time, and originators.

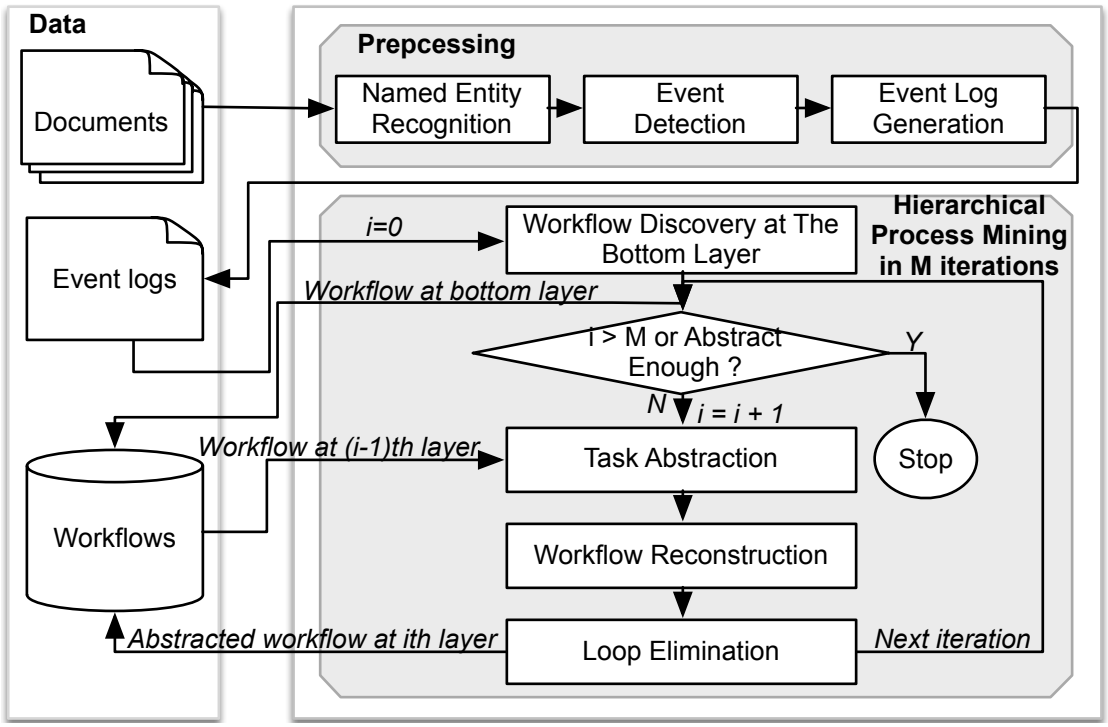


Figure 7.2 System architecture of bottom-up process mining

Given the event logs, the process mining module firstly constructs a flat workflow model that tries to capture all the execution sequences in the event logs at the bottom. Based on this flat model, the process mining module then iteratively creates a hierarchical process

model in three steps: task abstraction, workflow reconstruction, and loop elimination. As shown in Fig. 7.2, each iteration of the three steps could output an abstracted workflow model by merging highly-correlated sub-tasks in the lower layer. Finally, the whole system stops when the desired degree of abstraction has been achieved.

7.3.1 Workflow Discovery at The Bottom

The first step of the bottom-up process mining is to build a flat workflow model that covers all the possible behaviors in the event logs, without any abstraction. To do so, all the design events detected in the input documents are translated to the task nodes in the workflow graph, and the edges between pairs of events are determined by their time interval.

More specifically, let $EL = (e_1 \rightarrow e_2, \rightarrow \ldots, \rightarrow e_N)$ be the detected design events that are rearranged chronologically, and N be the event number. Based on the definition in Section 7.2, the workflow discovery attempts to produce a process model $\mathscr{W}(L^{bottom}) = (A^{bottom}, E^{bottom}, A^{bottom,s}, A^{bottom,e}, \mathcal{C}^{bottom})$, where each $a \in A^{bottom}$ corresponds to a unique event in EL, $A^{bottom,s}$ is a set of events that were carried out simultaneously with e_1, $A^{bottom,e}$ is a set of events that were carried out at the same time with e_N, and $E^{bottom} \subset (A^{bottom} \times A^{bottom})$ measures the possible precedence relations among all the events.

Given the execution time of two events (e.g., e_i and e_j), the possibility that a precedence relation exists is measured as:

$$pr(e_i \rightarrow_g e_j) = 1 - d/g \tag{7.1}$$

where g is the size of the time window, and d is the time interval between e_i and e_j. Based on $pr(e_i \rightarrow_g e_j)$, the relation between e_i and e_j is classified as:

- $e_i \neq\!\!\!\neq e_j$: there is no precedence relation if $pr(e_i \rightarrow_g e_j) \in (\propto, 0] \cup (1, \propto)$;
- $e_i == e_j$: e_i is parallel with e_j if $pr(e_i \rightarrow_g e_j)$ equals 0, which means two events were executed at the same time;
- $e_i \rightarrow e_j$: e_j is executed following e_i if $pr(e_i \rightarrow_g e_j) \in (0,1)$.

Only when the precedence relation $e_i \rightarrow e_j$ is detected, a corresponding edge is added into E^{bottom}. It is noteworthy that by using the time criteria, not only the direct relations $e_i \rightarrow e_j$ and $e_j \rightarrow e_r$ but also the long-distant relation $e_i \rightarrow e_r$ are taken into account as long as e_i and e_r are executed close enough. This is an important step to make sure that two events that are highly correlated but disturbed by a third event can be reconnected.

7.3.2 Task Abstraction

In each iteration shown in Fig. 7.2, the step of task abstraction merges small tasks that are highly correlated into bigger tasks in a higher abstraction layer based on the concept of aggregation and abstraction. Therefore, the composite task in a higher abstraction layer is the abstracted representation of the set of aggregated tasks. In turn, the set of small tasks in the lower layer can be seen as the same set of executions as the corresponding composite task, but in a more detailed way.

Before merging tasks, it is important to develop appropriate criterial that measure how closely two tasks are related. Let $\mathscr{W}(L^{i-1}) = (A^{i-1}, E^{i-1}, A^{i-1,s}, A^{i-1,e}, c^{i-1})$ be the workflow graph in a lower layer, the correlation among tasks in A^{i-1} is measured by two fundamental metrics: neighborhoodship and context similarity.

The first metric, Neighborhoodship, measures how closely two tasks are executed in time. As the edges in the workflow graph has already included the time information, the neighborhoodship is determined by the path connecting two tasks in the workflow graph. In detail, given the workflow graph $\mathscr{W}(L^{i-1})$, for each task node $a \in A^{i-1}$, its neighborhood is defined as:

$$neighbor(a) = \{a_i | \forall_{a_i \in (A^{i-1} - a)} \ path(a, a_i) \subseteq E^{i-1} \wedge |path(a, a_i)| < \alpha\} \qquad (7.2)$$

which is a set of nodes that are connected to a by paths shorter that α. All the task nodes in the neighborhood are treated as candidate tasks that might be merged with a.

The second metric, context similarity, measures the overlap of the attributes associated with two events. Examples of attribute overlap include two tasks were executed by the same person using the same tools or described by similar expressions, e.g., "buy simulation software" and "test simulation software". These attributes describe the execution context of a design task. More attributes are shared by two design tasks, higher possibility that they are correlated.

Based on the above two metrics, design tasks that are located in the same neighborhood as well as have similar execution contexts are merged into bigger composite tasks in a higher

layer. Figure 7.3 shows an example of task abstraction. For this example, the parameter of neighborhoodship is set as two. Fig. 7.3 (a) highlights the correlated nodes in the same color. Figure 7.3 (b) shows the abstracted workflow graph, in which the folder nodes indicate the composite tasks obtained by aggregation.

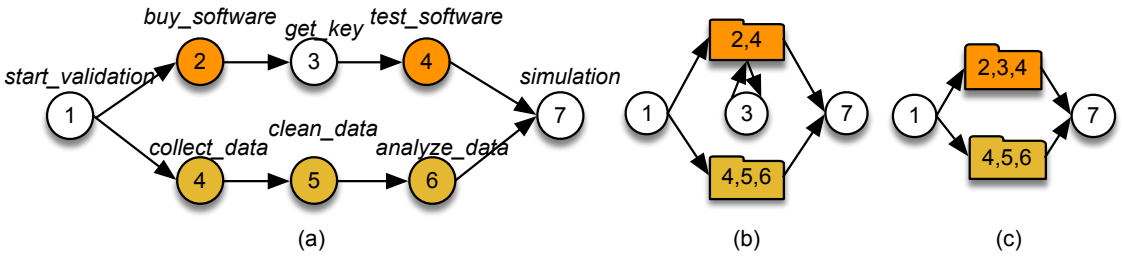


Figure 7.3 Example of task abstraction: a) $\mathscr{W}(L^{i-1})$, b) intermediate result, c) $\mathscr{W}(L^{i})$

7.3.3 Workflow Reconstruction

After getting the composite tasks via aggregation, this step advances to reconstruct the workflows for the composite tasks. A straightforward method is to let the abstracted workflow model include all the workflow patterns found in the lower layer.

Let $\mathscr{W}(L^{i-1})$ and $\mathscr{W}(L^{i})$ be the two adjacent layers, and L^{i} be the abstracted layer. Based on L^{i-1}, the workflows of L^{i} are constructed as:

- $E^{i} = \{(a_M, a_N) | \quad \forall_{a_M, a_N \in A^{i}} E_{a'_m, a'_n \in A^{i-1}}\ a_M \neq a_N \wedge (a'_m, a'_n) \in E^{i-1} \wedge (a_M, a'_m) \in \mathscr{D}(A^{i}) \wedge (a_N, a'_n) \in \mathscr{D}(A^{i})\}$
- $A^{i,s} = \{a | A_a \nexists_b\ a \in A^{i}\ \wedge b \in A^{i} \wedge\ (b, a) \in E^{i}\}$
- $A^{i,e} = \{a |\ A_a \nexists_b\ a \in A^{i} \wedge b \in A^{i} \wedge (a, b) \in E^{i}\}$

where all the edges connecting two child nodes in A^{i-1} are directly transferred to the corresponding composite nodes in A^i; starting nodes $A^{i,s}$ are the nodes without preceding nodes in A^i; and $A^{i,e}$ are the nodes without subsequent nodes. Take the workflow graph in Fig. 7.3 as an example, because Node 3 is connected to both Node 2 and Node 4 in opposite directions in Fig. 7.3 (a), two new edges are correspondingly created for connecting Node 3 to the composite node in Fig. 7.3 (b).

7.3.4 Loop Elimination

As shown in Fig 7.3 (b), the above workflow reconstruction operation tends to create loops in the abstracted workflow graph. Although loops resulting from unpredicted iterations is a key feature of real-life design processes, loops at a very detailed level would increase the complexity of the discovered design process model. Moreover, design tasks in a loop tend to be highly correlated. Therefore, for the sake of brevity, three operations are successively carried out to eliminate the loops created in three different scenarios. This would produce a further simplified workflow model. The main idea behind loop elimination is emerging the minor tasks into the strongest node in their neighborhood, or cutting off the weaker relation if two tasks are equally significant.

Examples of the three types of loops are shown in Fig. 7.4. They are:

- Case 1: as shown in Fig. 7.4 (a), one minor node and one composite node constitute a loop. Meanwhile, the minor node does not connect to any other composite nodes in its neighborhood;

- Case 2: as shown in Fig. 7.4 (b), one minor node and one composite node constitute a loop. Meanwhile, there are other composite nodes connected to the minor node;
- Case 3: as shown in Fig. 7.4 (c), two composite nodes constitute a loop.

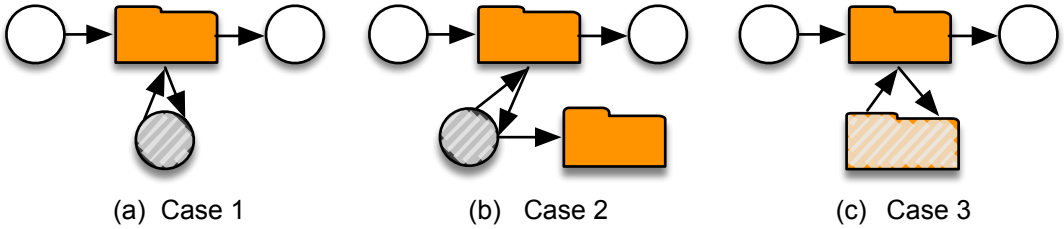


Figure 7.4 Example of loops

In the first case, it is assumed that the minor task node is less significant than the connected composite node. Therefore, the minor node is directly merged into the unique composite node. For example, the loop in Fig. 7.3 (b) just falls into this situation, thus, a new composite node consisting of nodes 2-4 is created in Fig. 7.3 (c).

Under the circumstance of Case 2, the minor node is merged into the composite node of the highest temporal significance. The temporal significance is calculated via dividing the number of the minor nodes in a composite node by its time duration. Therefore, composite nodes that are active frequently in a short period are of higher competitive power.

With respect to the last case, both composite nodes in the loop are significant. Under this circumstance, the loops are broken by cutting off the edges of weaker relations. This could result in a simplified model that remains the most normal behaviors to the greatest extent. The relation strength is estimated by how closely the minor tasks in one composite task are executed following the minor tasks in another.

7.4 Approach 2: Hierarchical Process Mining from Top to Bottom

7.4.1 System Architecture of Top-Down Process Mining

Compared to the above bottom-up approach, the top-down process mining introduced in this section works in the opposite direction, from generation to specification. Figure 7.5 depicts the system architecture of the top-down process mining approach. As shown in Fig. 7.5, the top-down process mining approach consists of two major steps: hierarchy construction and top-down sub-process modeling.

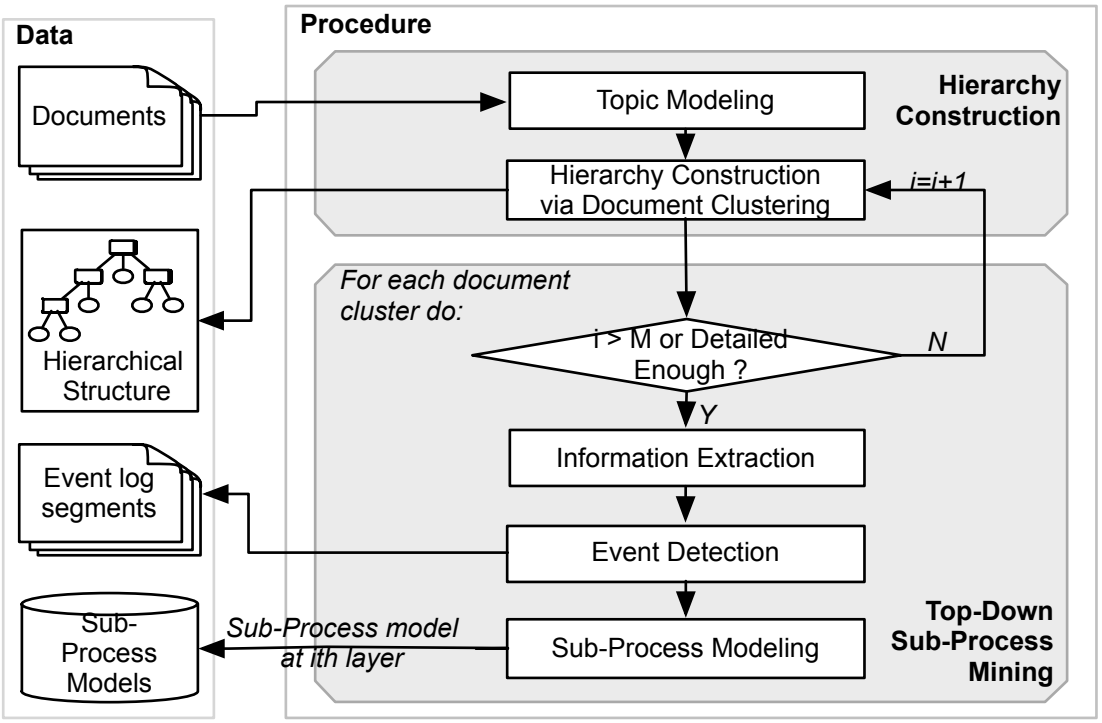


Figure 7.5 System architecture of top-down process Mining

The first major step aims to construct a tree structure that hierarchically tailors the whole design process into several functional modules. It starts at the the most abstract layer and views the underlying design process as a big black box. Next, the whole black box is

decomposed into smaller modules via document clustering based on per-document topic distributions. Hence, each module in the hierarchy tree corresponds to a document cluster, in which more homogeneous execution behaviors under the same function could be observed. Such a decomposition process can be repeated recursively until the desired degree of homogeneousness is achieved within each functional module.

The second major step in Fig. 7.5 aims to construct the sub-process model for each module from the corresponding document cluster. Within each module, process information is extracted using the NER approach presented in Chapter 5, design events are detected using the event detection approach presented in Chapter 6, and a sub-process model is constructed using the workflow discovery approach presented in Section 7.3.1. All the sub-processes together describe the whole process with different degrees of granularity.

7.4.2 Algorithm of Top-Down Process Mining

Figure 7.6 describes the algorithm of top-down process mining in detail. The meanings of some fundamental notions are defined as following:

- $D_i = (h_1, \ldots, h_H)$ is the topic representation of a document, where H is the number of topics detected, and $h_j \in [0,1]$ is the possibility that the jth topic appears in a document;
- C is a set of document clusters;
- M is a set of functional modules, each module (c, sw) corresponds to a workflow sw mined from a document cluster c;

- $T \subset \{M \times M\}$ is a tree that organizes M in a hierarchical structure.
- $homogeneity(c)$ is a function that measures how closely the contents of the documents in c are correlated. Let $\overline{D_c}$ be the center of c, the within-cluster homogeneity of c is determined by the average distance from all the documents to the center, $homogeneity(c) = 1 - \sum_{d \in c} dist(D_d, \overline{D_c}) / |c|$.

Algorithm 7.1 Top-down Process Mining
Inputs: D is the document collection, γ is the maximum homogeneousness and $\mathscr{P}$ indicates the maximum depth.

```
1:  Procedure TOPDOWN_MINING(D, γ, 𝒫)
      // Step 1: hierarchy construction
2:    For each D_i in D do:
3:      Topic modeling: D_i = (h_i, ..., h_H)
4     Initialization: C:= {D}, M:= {(D, ∅)}, T:= {∅}
5:    While |C| > 0 do:
6:      c^p := pop(C) //Select and delete a document cluster from C
7:      If homogeneity(c^p) < γ and depth(c^p) < 𝒫 do:
8:        C^new = HCA_clustering(c^p, D_{c^p}) //Cluster a module
9:        For each c^s in C^new do:
10:         m^new := (c^s, ∅), M := m^new ∪ M
11:         T := (m^new, m^{c^p}) ∪ T
12:       C := C^new ∪ C
      // Step 2: sub-process mining
13:   For each m = (c, ∅) in M do:
14:     If ∄m^p (m, m^p) ∈ T do:
15:       l_c := event_detection(c)
16:       sw := workflow_discovery(l_c)
17:       m := (c, sw)
```

Figure 7.6 Algorithm of top-down process mining

Referring to Fig. 7.6, the second to the twelfth lines describe the procedure for hierarchy construction. As shown in lines 2 to 3, the algorithm firstly transforms each document into its topic distribution by using the DBN-based topic modelling approach presented in Chapter 4. Based on the per-document topic representation, the algorithm starts constructing

the process hierarchy by initializing $C := \{D\}$ and $M := \{(D, \varnothing)\}$, which is the whole document set. Next, in each iteration shown in lines 5 to 8, one document cluster with the least homogeneous content is selected from $\boldsymbol{C}$ and clustered into smaller ones, within which more refined sub-process models can be discovered. For this purpose, the content relevance between any two documents is estimated by the cosine similarity over their topic distributions, as shown in Eq. (7.4). After each decomposition, a set of new clusters is appended to $\boldsymbol{C}$ (shown in line 12), and a set of new hierarchical relations is correspondingly created in $\boldsymbol{T}$ (shown in lines 10 to 11). The whole decomposition process can be iterated until all the leaf modules in $\boldsymbol{T}$ are homogeneous enough. At this point, each document cluster in the leaf node can be viewed as a functional module that contains desired details about the same task or activity.

$$sim_{cos}(D_i, D_j) = \frac{\sum_{n=1}^{n=H} h_n^{D^i} h_n^{D^j}}{\sqrt{\sum_{n=1}^{n=H} (h_n^{D^i})^2} \sqrt{\sum_{n=1}^{n=H} (h_n^{D^j})^2}} \tag{7.4}$$

Lines from 13 to17 describe the procedure for sub-process mining. For each leaf module in $\boldsymbol{T}$, a group of highly related design events are detected from the corresponding document clusters using the event detection approach in Chapter 6. Based on these design events, a sub-process model is then constructed to capture the underlying process behaviors using the workflow discovery approach in Section 7.3.1.

To give a more straightforward impression, Figure 7.7 depicts a hierarchical model that might be generated by Algorithm 7.1. The tree in Fig. 7.7 represents the hierarchy structure

organizing all the functional modules. Each leaf node corresponds to a sub-process mined from a set of correlated documents, and each parent node represents a bigger document cluster that projects the leaf nodes onto a higher abstraction level. The hierarchical model in Fig. 7.7 can also be easily transformed to the formation defined in Section 7.2.

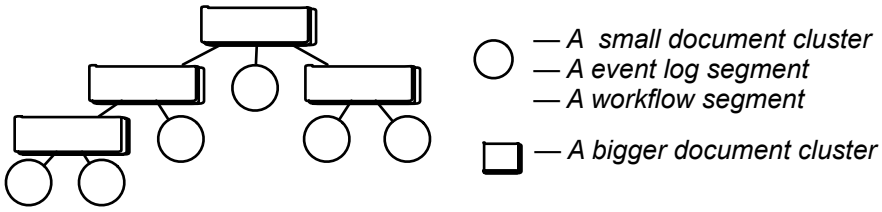


Figure 7.7 Hierarchical model of top-down process mining

7.5 Case Study

7.5.1 Dataset and Performance Measures

The process models discovered by both process mining approaches are illustrated and compared using the email dataset from the TWP project. As the project participants used emails as their major communication tool in this TWP project, the emails record the footprints of the entire design process. Before process mining, all the events were detected from the original email dataset. The detected events were then reordered and stored in a XML file, which used tags and markups to organize arbitrary data structures, such as the high-order relation among the attributes of an design event.

For validating the correctness of the discovered process models, one participant who played a admin role in this TWP project was interviewed to check the alignment between

the automated process models and his personal knowledge base. If the discovered process models are consistent with the expert knowledge, they have a good reflection of the reality.

7.5.2 Results of Bottom-Up Process Mining

7.5.2.1 Examples of Process Model in Bottom Layer

Figure 7.8 depicts a segment of the flat model discovered in the bottom layer, which attempts to give the most detailed description of the underlying process. 661 events were detected in the bottom layer.

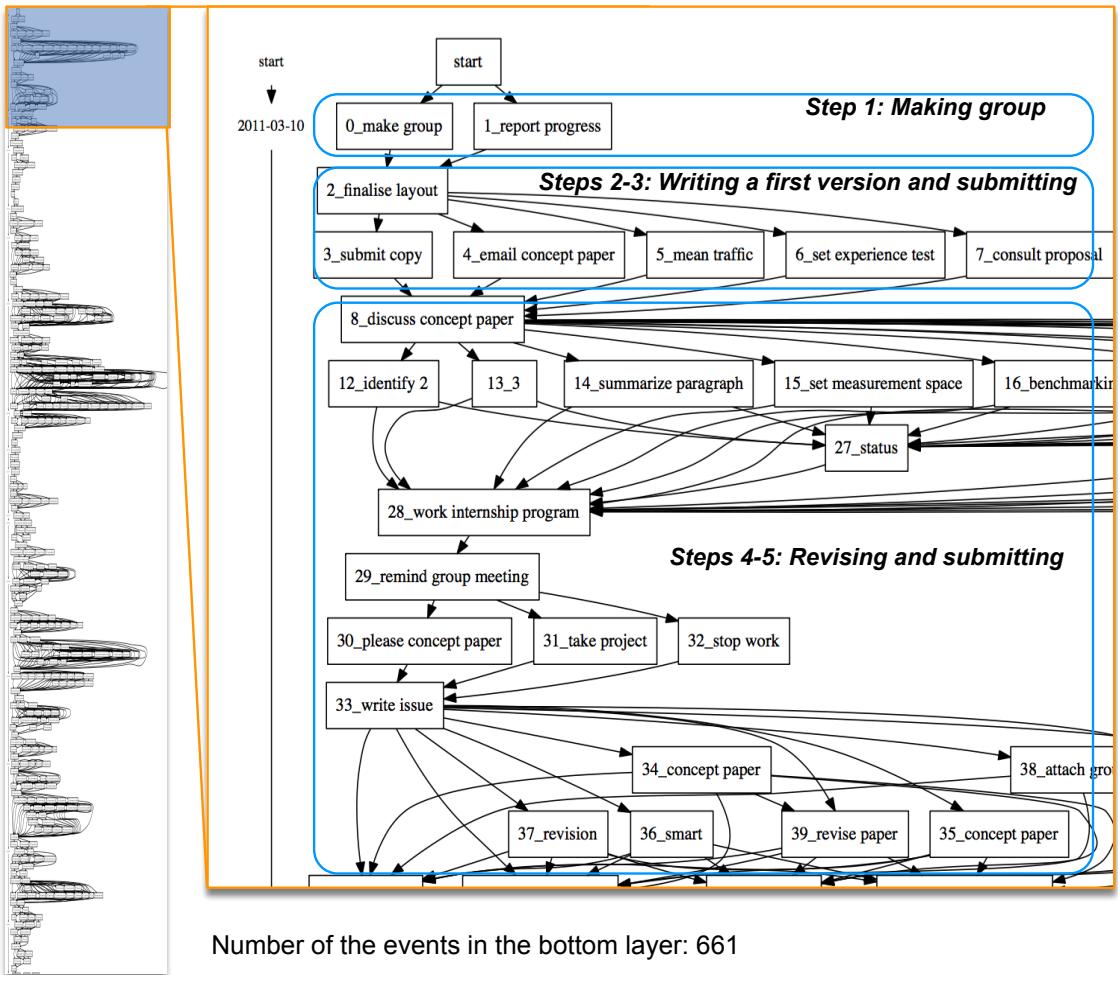


Figure 7.8 A segment of the process model in the bottom layer

According to Fig. 7.8, it is obvious that the flat model tries to capture every detail about the 661 events. For example, the magnified segment in Fig. 7.8 shows the design activities which aimed to write a concept paper when the project started out. As highlighted in Fig. 7.8, it can be clearly observed that the final concept paper was created after five main steps, namely making group, writing the first version in groups, submitting, revising, and submitting again. However, due to the enormous size of the events, such details do not allow decision makers to quickly get a clear insight into the underlying process. In this case, the discovered flat model becomes inefficient for use, even if it can be generated. Therefore, it is important to abstract and simplify the whole process by hiding undesired details, so as to improve the understandability of the discovered design process model.

7.5.2.2 Examples of Process Model in Abstraction Layer

Figure 7.9 (a) presents the process model of the TWP project in an abstraction layer, which is obtained after 15 iterations of aggregation and abstraction. In Fig. 7.9 (a), there are totally 48 composite tasks, which are integrated from several smaller sub-tasks or events. Each composite task is highlighted by a filled folder. In other words, each folder in Fig. 7.9 can also be seen as a module that corresponds to a subnet in a lower layer. The name of the composite tasks is automatically generated according to the frequency of words used to describe the subordinate events.

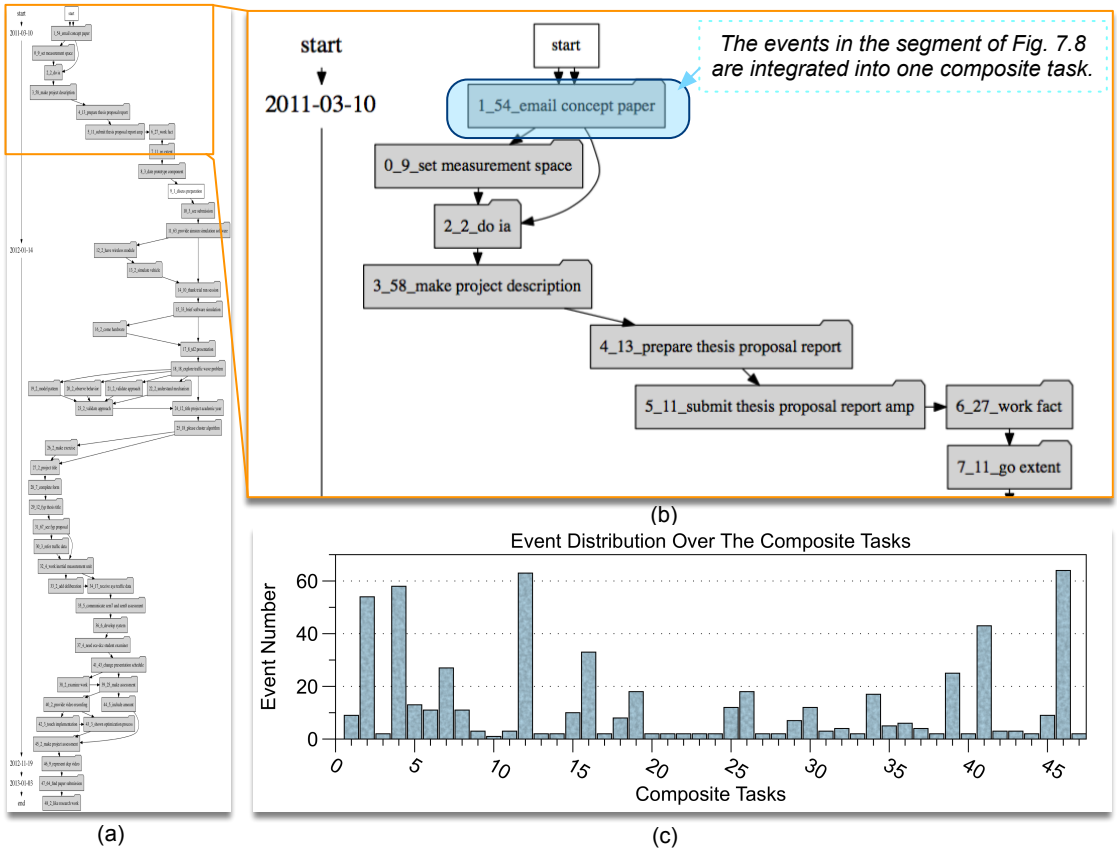


Figure 7.9 Process model in the top layer: a) the overall process model, b) a magnified segment, c) the event distribution over the composite tasks

To get a closer look at the abstracted process model, Fig. 7.9 (b) magnifies the upper part of Fig. 7.9 (a). Referring to the highlighted part of Fig. 7.9 (b), it is noteworthy that all the events shown in the segment of Fig. 7.8 are integrated to the same composite task, "email concept paper". In other words, the single node highlighted by a rectangle in Fig. 7.9 (b) represents a module including all the activities that were carried out to finish the concept paper. In a "zoom out" model, Fig. 7.9 (b) also shows a very clear workflow of what had been done at the beginning of this project. In detail, students firstly hypostatized their ideas into a concept paper. Under the guidance of the concept paper, they carried out some concrete activities, e.g., "set measurement space" and "do IA". Next, the project was

interrupted by an extemporaneous but urgent task, named as "make project description". Lastly, based on the works having been done, students wrote and submitted their thesis proposal report in groups. These findings indicate that the abstracted process model in the top layer is able to give a quite compact and brief reflection of the whole design process.

Figure 7.9 (c) plots the number of the events subordinated to the 48 composite tasks, which compose the workflow in Fig. 7.9 (a). From the event distribution in Fig. 7.9 (a), it is obvious that there are eight tasks which have relatively more event members than the others, above 20 events. To obtain a more correct understanding of the underlying process, it is necessary to look deeper into these composite tasks as these bigger tasks own the major proportion of the minor events.

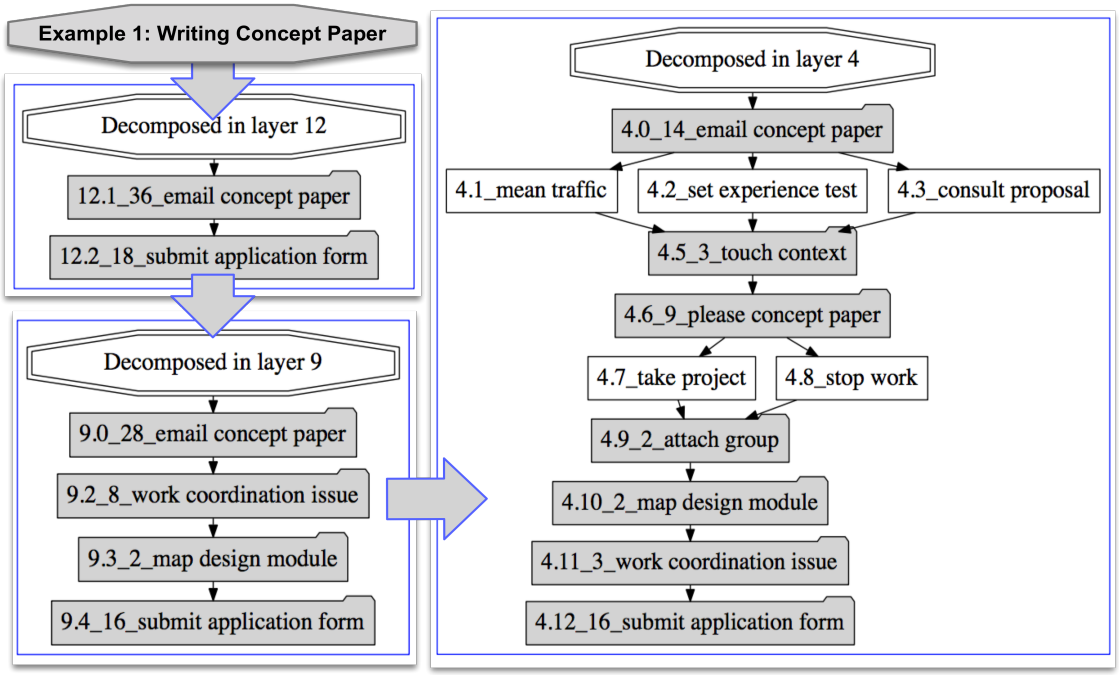


Figure 7.10 Example of decomposing the task of "writing concept paper"

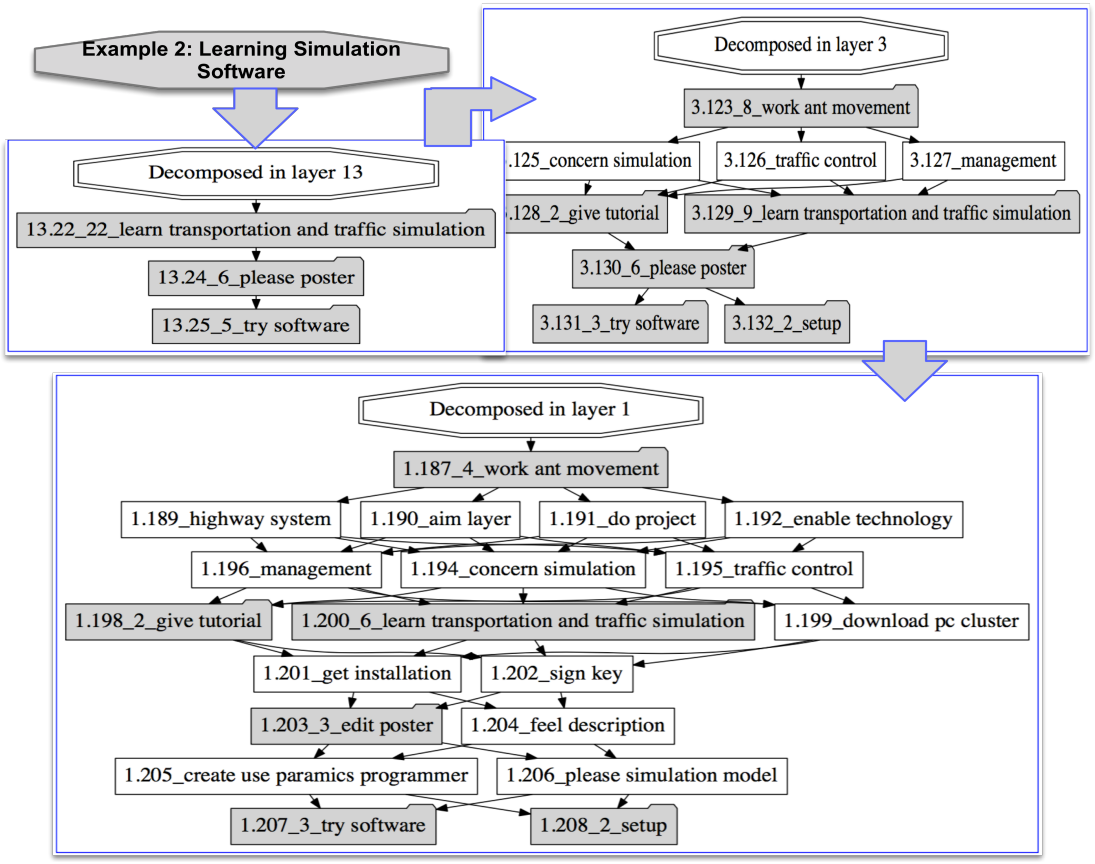


Figure 7.11 Example of decomposing the task of "learning simulation software"

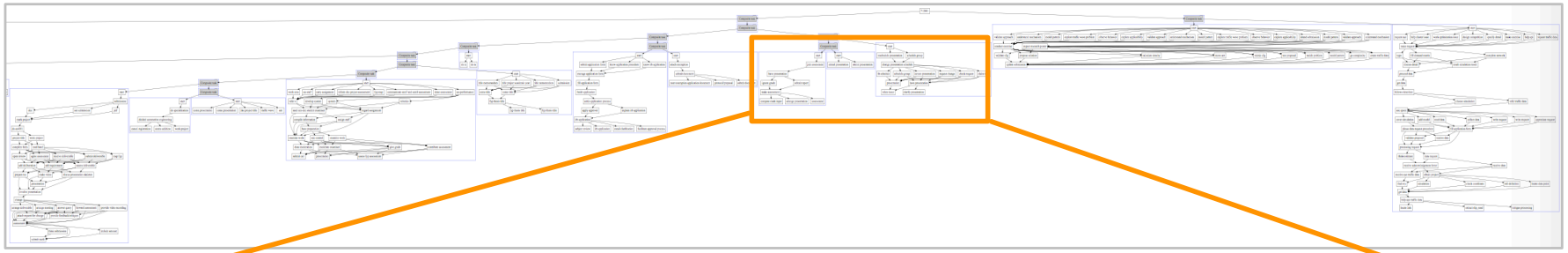
Regarding the above problem, Fig. 7.10 and Fig. 7.11 illustrate two examples of decomposing the same composite task at different abstraction levels. The first example shown in Fig. 7.10 is a composite task about writing concept paper. The second example shown in Fig. 7.11 is about learning simulation software. Each composite task is represented by a subnet of smaller tasks that are highly correlated in a lower layer. Each subnet itself can be composed of both minor and composite tasks, and the composite tasks can be further decomposed at another much lower level. This decomposition process can be executed iteratively until arriving at the bottom layer. Each decomposition tries to describe the target

task in a more detailed manner. In this way, a hierarchical description of the interesting composite tasks can be obtained, as shown in Fig. 7.10 and Fig. 7.11.

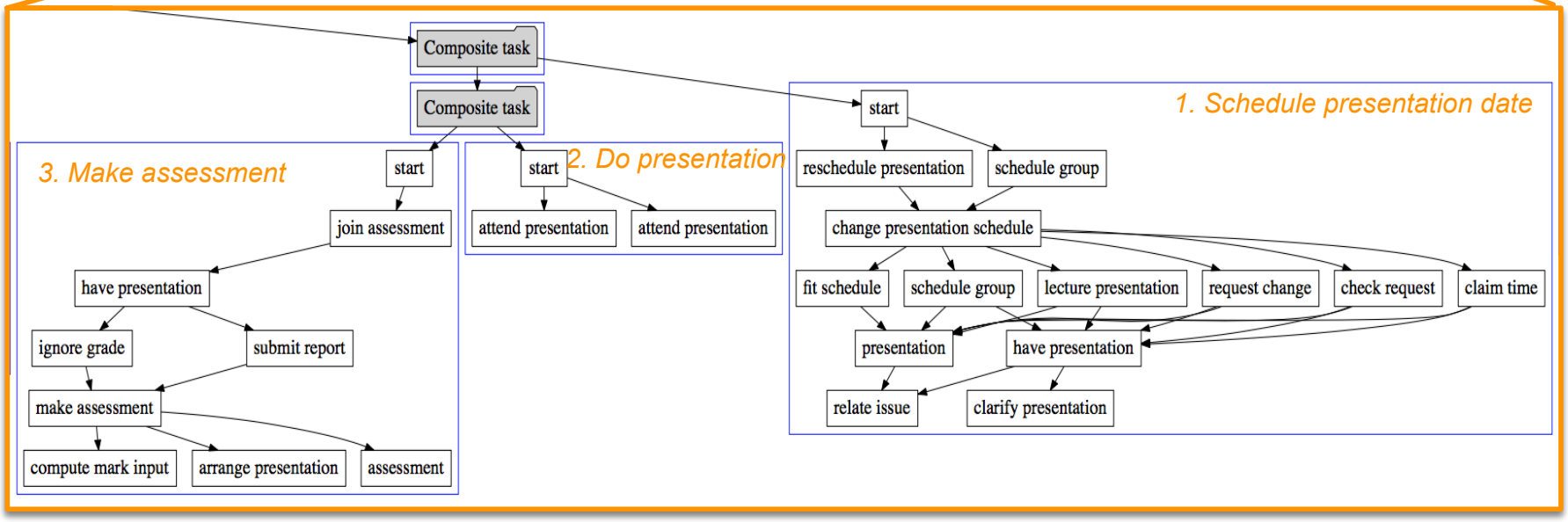
7.5.3 Results of Top-Down Process Mining

Figures 7.12 (a) and 7.13 (a) illustrate two segments of the hierarchical process model that were discovered by the top-down process mining approach from the same dataset of the TWP project after 40 decomposition iterations. For each iteration, one big document cluster was selected and decomposed into two smaller ones. Therefore, the hierarchy tree in Fig. 7.12 (a) and Fig. 7.13 (a) can be seen as a binary tree, in which the filled nodes correspond to document clusters that are decomposed into several smaller ones, and the leaf nodes denote sub-process models.

To give a more straightforward impression, Figures 7.12 (b) and 7.13 (b) illustrate two examples of the sub-process models represented by the leaf nodes. From Fig. 7.12 (b), it can be clearly observed that the three sub-processes are hierarchically connected, and all of them are related to the same task of “FYP presentation”. Among the three sub-processes, the first one shows a very clear workflow of scheduling and rescheduling presentation date, the second one is about doing the presentation, and the third one relates to the procedure of making assessment after the presentation. As shown in Fig. 13 (b), the second example is a sub-process of writing a conference paper. Most of the events in this sub-process are highly related to the task of “writing paper”, e.g., submit abstract, discuss and prepare the table of content, draft and combine sections, as well as refine and submit the full paper.

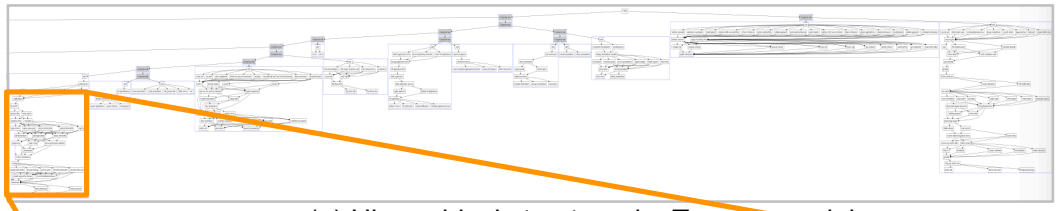


(a) Hierarchical structure by Top-down mining

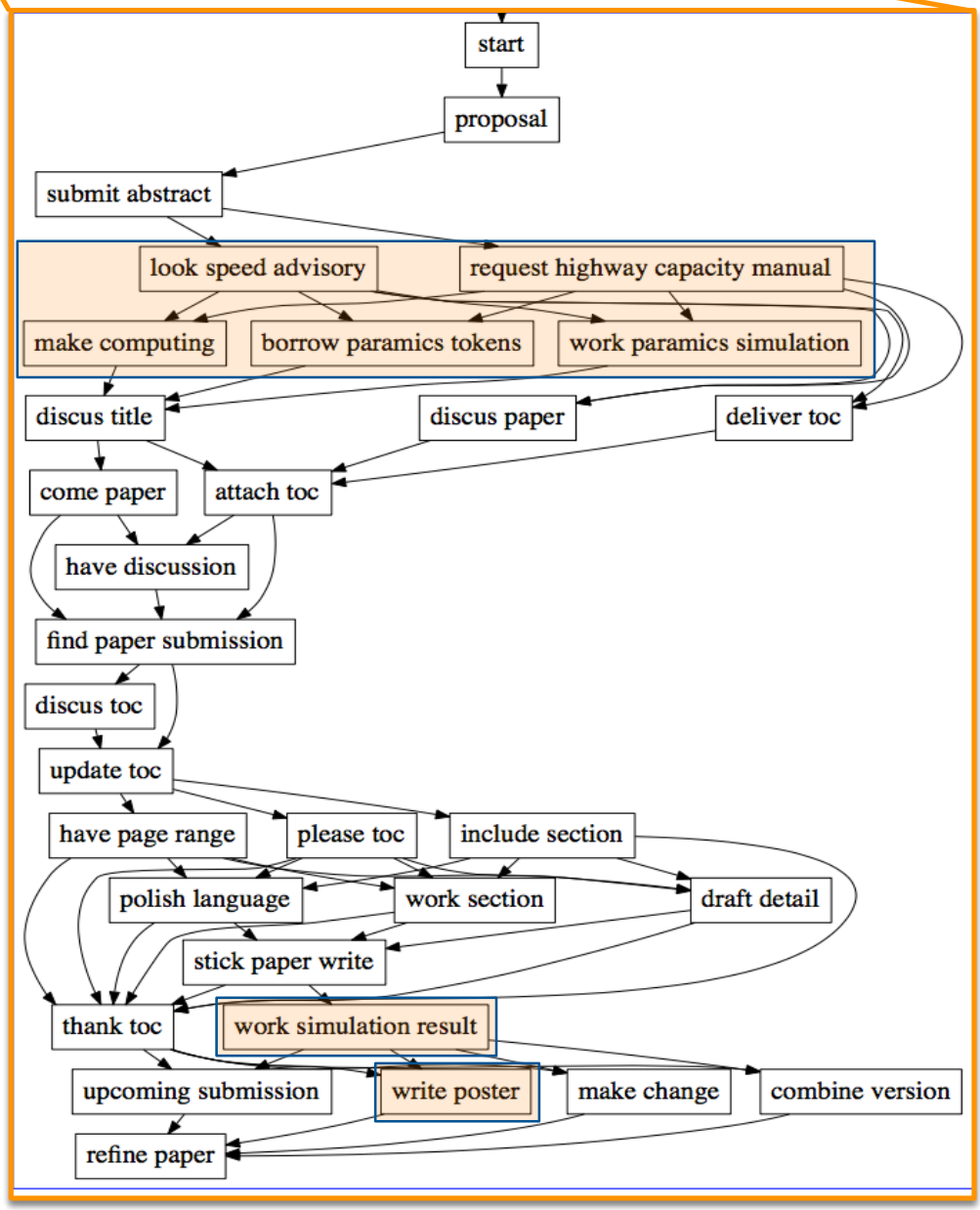


(b) Example of sub process models relating to “FYP presentation”

Figure 7.12 First example of hierarchical process model by top-down mining



(a) Hierarchical structure by Top-down mining



(b) Example of sub process models relating to “Writing paper”

Figure 7.13 Second example of hierarchical process model by top-down mining

The above findings indicate that the top-down process mining approach can not only discover the design process from the design documents, but also present the discovered process model in a user-friendly manner

However, Fig. 7.13 (b) also shows that some events are irrelevant to the task of writing paper but are included in this sub-process model. Figure 7.13 (b) highlights such irrelevant events using colored rectangles. Among these highlighted events, some (e.g., look speed advisory and request highway capacity manual) are related to data collection, while some (e.g., borrow paramics tokens and work simulation results) are about traffic simulation. The most essential reason for this confusing finding is that the top-down mining approach assumes that the events from the same document were carried out with the same goal. As a matter of fact, the most normal case is that people like to simultaneously discuss several heterogeneous issues in a single email. That is to say, the top-down mining approach may work poor when most documents mention events of different design tasks.

7.5.4 Discussions: Bottom-Up Vs. Top-Down

Two process mining approaches, bottom-up and top-down process mining, are proposed in this chapter to discover a hierarchical process model from the design documents. The two approaches work in two opposite directions with the former going from specification to generation and the latter going from generation to specification. The feedback from the interviewed participant indicated that: 1) As the hierarchical process model described a complex process with multiple levels of abstraction and refinement, it

was significantly helpful in reducing model complexity and improving the understandability of the discovered process model; 2) both approaches performed well in automatically mining such a hierarchical process model from the design documents.

In addition, as discussed in Section 2.6, one major difficulty of mining design process model is caused by the flexibility of product design processes. In the event logs of formal business processes, process behaviors are recorded by individual instances and each instance has a sequence of events. This characteristic of event logs enables the process mining algorithms to quantitatively calculate the causal dependence of business tasks. However, different from the events which are well organized in business event logs, design events detected from design documents have no deterministic boundary to split them into instance traces. In other words, all the design events are included in a single trace. Within this single trace, there might be iterative design events, but they never repeat in the same way. To overcome this problem, the two proposed process mining approaches adopt a heuristic strategy, which takes advantage of local information such as per-document topic distribution, time interval, and attribute overlap to create a hierarchical process model. The experiment results show that the discovered process models have a good reflection of the reality.

Besides the above positive findings, the discussion with the interviewed participant also reflects some disadvantages of both process mining approaches. From the time cost perspective, the top-down process mining approach is more agile than the bottom-up

approach. This difference is caused by the different ways in which the two approaches identify process modules. In detail, the bottom-up approach identifies modules by merging events with similar execution environment. Therefore, the time complexity is proportional to the number of events. However, the top-down approach treats documents with homogeneous contents as modules. Therefore, its time complexity is reduced to be proportional to the number of the input documents. From the accuracy perspective, the bottom-up approach outperforms the top-down approach. The reason is that the top-down approach considers that events which are recorded in the same document must be located in the same module. However, in most cases, people like to discuss and record issues relating to multiple tasks in the same documents. In this situation, some modules discovered by the top-down approach are mixed with irrelevant events, which may influence the further analysis based on these modules, e.g., the time line and resource allocation of an impure module.

For realistic applications, the strategy (bottom-up or top-down) should be selected according to the specific application goals. For example, if an application is more about document management and retrieval, the hierarchy structure constructed by the top-down approach can help to organize all the relevant documents in a more systematic manner. If an application is to identify the root causes of some problems, the hierarchical process model discovered by the bottom-up approach allows designers to obtain more accurate performance analysis.

7.6 Summary

To sum up, this chapter lays the attention on discovering process model based on the discrete information extracted in Chapters 4 to 6. Two process mining approaches are proposed. They are bottom-up mining and top-down mining. Both approaches aim to discover a hierarchical process model that represents a process with different levels of granularity and abstraction but use two opposite strategies. The bottom-up approach starts at the most concrete level then iteratively abstracts the target process by merging highly correlated events or sub-tasks. On the contrary, the top-down starts at the most abstract level then iteratively decomposes the target process into modules via document clustering. Results of the two approaches were discussed and compared using a real-life case study.

CHAPTER 8 MULTI-FACETED PROCESS KNOWLEDGE INTERPRETATION BY LINKING PROCESS INFORMATION TO PROCESS MODEL: A CASE STUDY

8.1 Introduction

The last component of the PKDT system is to distill multi-faceted knowledge patterns from the discovered process model. The discovered process knowledge reflects the experience learned from the past design projects and could be used to support decision making in current or future design projects.

As discussed in Chapter 2, most existing tools of reutilizing design knowledge have been focusing on reusing the geometric knowledge embedded in CAD models or the technology trend included in patents. The reutilization of such product knowledge is able to reduce the time required to reproduce some well-known components. However, it can not support the knowledge reutilization throughout the whole product design process. To overcome this problem requires to explore an integrated knowledge reutilization approach that include not only product knowledge, but also process knowledge such as task dependencies, organizational structure, and resource allocation. For this purpose, there are three important factors in the success of an integrated knowledge reutilization approach: making multi-faceted design knowledge reusable, storing the reusable design knowledge in a compactible manner, and providing the most relevant design knowledge in a user-friendly way. Considering the three factors, this chapter proposes an integrated design knowledge

reutilization framework in Section 8.2. In the proposed framework, the discovered hierarchical design process model serves as the central element of the design process knowledge. By applying different computational approaches on the process model, other types of design knowledge such product, organization, and temporal behaviors, can be integrated to present a product design process from multiple perspectives.

As the process model has already reflected the workflow perspective of the underlying design process, this chapter aims to enrich the discovered process model from other two perspectives: personnel perspective that focuses on the people involved and temporal perspective that focuses on the temporal behaviors of both the executed design tasks and the involved people. Section 8.3 introduces the organization mining from the personnel perspective, and Section 8.4 presents the statistical approaches for the temporal behavior analysis. The dataset of the TWP project described in Section 3.3 and the process model obtained by the bottom-up process mining are used to illustrate the discovered knowledge patterns. Furthermore, one participant, who played a admin role in this project, was interviewed to assess the accuracy of the discovered knowledge patterns.

8.2 An Integrated Design Knowledge Reutilization Framework

Figure 8.1 shows the proposed design knowledge reutilization framework. The design process model in the bottom layer of Fig. 8.1 severs as the center element to connect design knowledge from different perspectives. In detail, the design process model itself can be seen as a combination ontology, which provides links to design tasks, product components,

people, and resources. Therefore, by applying computational analysis approaches on the interaction within the design process model, three categories of design knowledge can be refined: product knowledge, process knowledge, and organization knowledge.

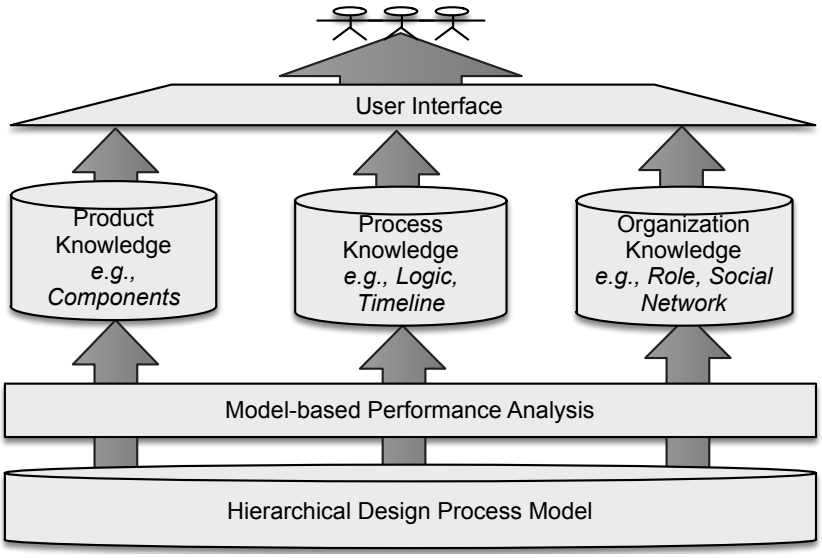


Figure 8.1 An integrated design knowledge reutilization framework

The first type of design knowledge, product knowledge (e.g., product structure and functional requirements), interacts with the process model via task objectives, e.g., creating some components. Product knowledge can be enriched by extracting product information from other files like CAD models and technical reports. The second type of knowledge, process knowledge such as task dependencies, task durations, and task waiting times, allows tracing task executions from the viewpoint of logical workflow. In addition, the design process model itself can be seen as a type of process knowledge. Lastly, organization knowledge, including organizational structure, cooperation patterns, and functional roles,

allows investigating the behavior of the designers through the design tasks they have been involved in.

Based on the three types of design knowledge, the user interface for knowledge reutilization enables decision makers to retrieve desired information for decision making. For example, a manager can check the performance of a specific person in the existing projects before he decides to assign this person to a new design task. In another situation, designers can predict the duration of a new design task, according to the durations of similar tasks in the existing projects. In addition, for the purpose of facilitating the knowledge reutilization procedure, it requires the user interface to visualize the desired design information in a user-readable, understandable manner.

Based on the data available in the TWP project, Section 8.3 and Section 8.4 illustrate two perspectives of the knowledge reutilization framework: organization mining and temporal process behavior analysis.

8.3 Organization Mining

This section puts emphasis on organization mining, which focuses on the involved people and their interactions embedded in the design process model. In general, organization mining can be divided into three categories: social network analysis, role mining, and human resource allocation. They either analyze the interaction patterns among activity performers or classify the performers in terms of roles and cooperation cliques. In the

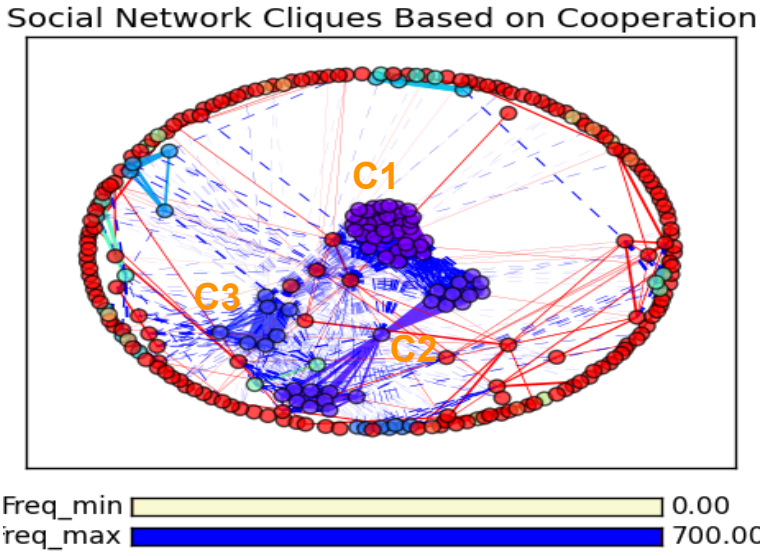
remaining parts of this section, the three perspectives of the organization mining will be studied and illustrated on the basis of the TWP project respectively.

8.3.1 Social Network Analysis

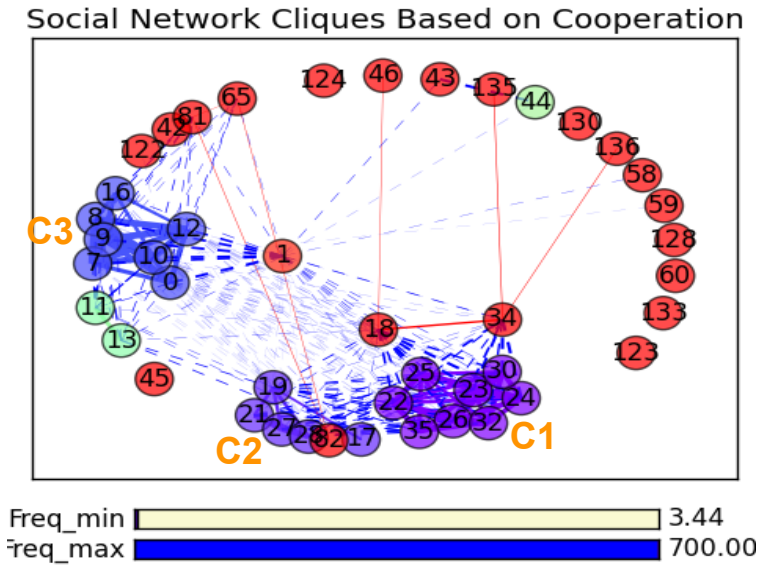
Social network analysis aims to find the relationship between activity performers and to infer the organizational structure based on their interactions within design tasks. In most cases, the interaction between two performers are measured using metrics such as joint activities, handover, and special event types [145]. Based on the interaction among performers, clustering analysis and graph partition in data mining are mostly used to discover organizational structure models [146, 147], in which people with similar skills and roles are clustered together.

On the basis of the hierarchical process model discovered by the bottom-up process mining, Figure 8.2 depicts the social network cliques of the TWP project by grouping all the participants according to their joint events. As shown in Fig. 8.2, more than 200 people were involved in this project, including the core members of this project and the external people from some third companies. Figure 8.2 (a) presents the interactions among all the participants. Figure 8.2 (b) is simplified from Fig. 8.2 (a) by hiding participants of a frequency less than 3.44. Furthermore, in Fig. 8.2, participants clustered into the same cliques are marked in the same color except the red nodes, which indicate participants who fail to join in any clique. The solid lines in Fig. 8.2 connect participants within the same

cliques, while the dash lines indicate the interactions across cliques. In addition, the thickness of the lines reflects the interaction strength of the connected participants.



(a)



(b)

Figure 8.2 Social network cliques based on cooperation

Referring to Fig. 8.2, three major cooperation cliques, labeled as C1, C2, and C3, are detected. Among the three cliques, C1 is the biggest, while C3 is the smallest. However, after removing less-frequent participants as shown in Fig. 8.2 (b), the size of both C1 and C2 are dramatically reduced, and only C3 remains unchanged in size. The interviewed participant explained this phenomenon that C3 reflected the true organization of the TWP project, while C1 and C2 were participants from two related sub-projects, which were under a same bigger project with the TWP project. Therefore, the core participants in C3 have higher frequency than the participants in C1 and C2. The feedback also revealed that the findings on cooperation patterns had a good alignment with the true situations.

8.3.2 Role Mining

Role mining aims to identify roles, rights, responsibilities, or skills of process participants via analyzing their interaction patterns [148]. Good insights into the roles and responsibilities of the project participants have a great help in supporting decision making about human resources, thus enable the whole workforce to work together to produce a qualified product.

In role engineering, there are two categories of roles: functional roles and organizational roles [149]. Process participants who execute similar activities tend to have similar skills, thus might be classified into similar functional roles. Social network analysis is often used to identify the functional roles for process performers. From this point of view, the three

cooperation cliques detected in Fig. 8.2 can also be seen as three types of functional roles, and people within each clique have the same functional role.

Organizational role is to identify participants of different permissions, responsibilities, and rights. Each participant has exactly one organizational role with a certain degree of permissions and rights. In this case study of the TWP project, we consider three levels of organizational roles:

- Managers - organize and direct the entire organization. They work closely with both the internal participants who might have different functional roles and the external customers.
- Leaders - work with managers to ensure that operators apply decisions and operations correctly. This requires them to have strong interactions to both managers and operators.
- Operators - have individual responsibility for low-level operations when creating products.

According to the above definitions, managers tend to have more frequent interactions with external customers and leaders from different groups or departments when compared to leaders and operators. Let $nei(p)$ be the participants who are directly connected to a people in the social network graph, $coop(p)$ be the set of participants who have the same functional role with p, $w(p, p')$ be the interaction strength of any two participants, function $fr\_num()$ calculate the number of unique functional roles in a set of participants,

and $\boldsymbol{P}$ include all the involved participants in the social network graph. The possibility that a participant is a manager is estimated as:

$$s\_manager(p) = \frac{fr\_num(nei(p) - coop(p))}{fr\_num(\boldsymbol{P})} * \sum_{p' \in nei(p) - coop(p)} w(p, p') \tag{8.1}$$

where, the sum over the external interactions is scaled by the diversity of the interacted functional roles. In other words, participants who frequently interact with more types of functional roles tend to be managers.

In contrast, leaders usually present the strongest internal interaction and the strongest external interaction among a small group of participants, e.g., departments and teams. Let the detected cooperation cliques in Fig. 8.2 be such functional groups, a leader pl is selected for each group $\boldsymbol{C}$ by finding the participant who has the highest degree $s\_leader(p)$ in the social network graph:

$$s\_leader(p) = \sum_{p' \in nei(p) \cap coop(p)} w(p, p') \tag{8.2}$$

$$pl(\boldsymbol{C}) = \max_{p \in \boldsymbol{C}} s\_leader(p) \tag{8.3}$$

The results of role mining are shown in Fig. 8.3, where the identified managers and leaders are highlighted by bigger nodes. According to Fig. 8.3, four participants, i.e., P1, P18, P19 and P34, are identified as managers, and three participants, i.e., P23, P21, and P0 are identified as the leaders of C1, C2, and C3 respectively. In Fig. 8.3, it can be clearly observed that the four participants play a critical role in connecting all the three cooperation cliques together. This finding is consistent with the feedback from the interviewed

participant. The participant explained that 1) P1, P18, P19, and P34 were four professors who had supervised different sub-projects in reality, 2) P1 was the supervisor of the TWP project, and 3) P0 was the student leader of this TWP project.

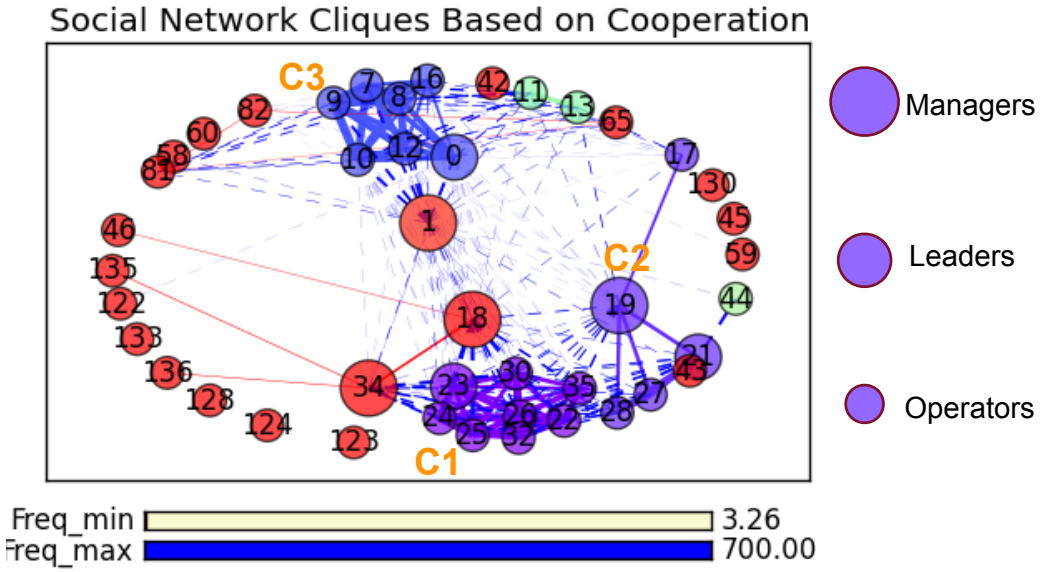
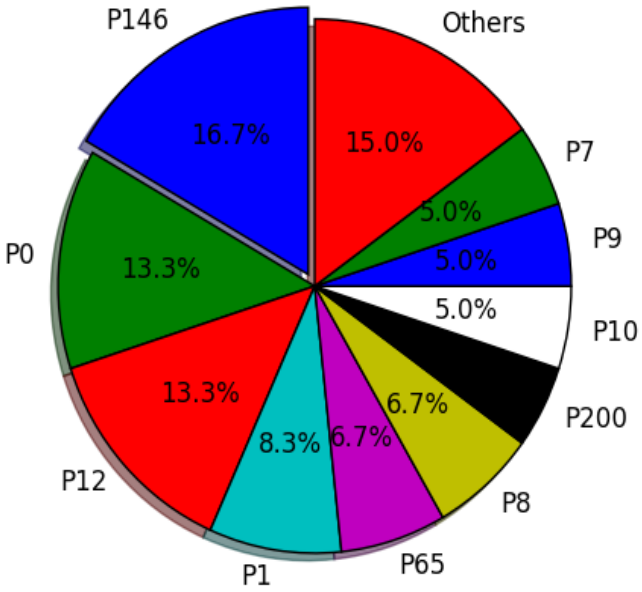


Figure 8.3 Results of role mining

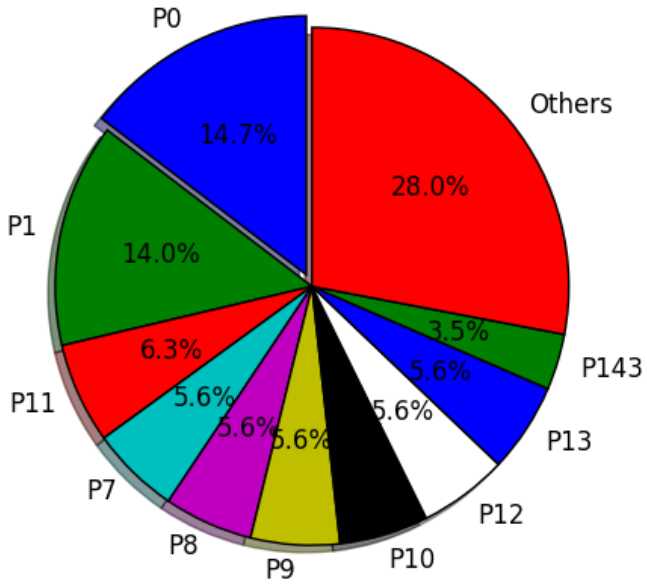
8.3.3 Human Resource Allocation

Human resource allocation aims to allocate the most appropriate people to execute different design activities. An effective human resource allocation can significantly improve productivity, maximize resource utilization, and reduce execution costs. A simple human resource allocation strategy is assigning a people to an activity whose requirements are consistent with the capability of the people [150]. However, this strategy does not consider the behavior and performance of the people in the past. To overcome this problem, one alternative strategy is finding design tasks which are similar to the new task from the past

design projects, and allocating human resources to the new task according to the resource behaviors in these historical design tasks.



(a) Example 1: Human resource utilization of *"Writing Concept Paper"*



(b) Example 2: Human resource utilization of *"Learning Simulation Software"*

Figure 8.4 Examples of human resource utilization

On the basis of the hierarchical process model obtained by the bottom-up process mining, Figure 8.4 depicts the human resource allocation and utilization of two composite tasks in the top abstraction layer. The contribution of a people to a composite task is estimated by the frequency that this people was involved in the events under this task. The pie charts in Fig. 8.4 show that participants P146 and P0 played the most crucial role in "writing concept paper" and "learning simulation software" respectively. Therefore, based on the concept of history-based human resource allocation, P146 could be the most probable candidate for executing a new task that is similar to "writing concept paper". Similarly, P0 might be the most suitable people to execute a new task that is similar to "learning simulation software".

8.4 Temporal Process Behavior Analysis

The temporal process behavior analysis aims to discover the temporal behaviors of both the executed design tasks and the involved people. The temporal behaviors include the duration, waiting time and severing time of the design tasks, as well as the temporal and overall frequencies of the task performers. Such information is helpful to answer questions like: “are there any irregular task executions or bottlenecks in the actual process?”, “who were always active throughout the entire design process?”, and “who only participated in some specific design events?”. The Gantt chart is used as the major tool to analyze the temporal behaviors based on the hierarchical process model discovered from the TWP project.

8.4.1 Temporal Behavior of Design Tasks

Figure 8.5 compares the 48 composite tasks in the top abstraction layer of the TWP process model using Gantt chart. The vertical axis of Fig. 8.5 (a) lists the 48 composite tasks in the hierarchical process model shown in Fig. 7.9, and each line corresponds to a composite task. The horizontal axis of the Gantt chart corresponds to the time dimension and rearranges the events under the same composite task in a chronological sequence. Therefore, each rectangle in a line refers to a continuous serving time of the corresponding task, spaces between pairs of rectangles represent waiting times, and a task starts at the first rectangle and ends at the last rectangle in the corresponding line.

The Gantt chart in Fig. 8.5 (a) visualizes the temporal status of the task executions in a very straightforward way. For example, it can be easily observed that the first task lasted about four months from Mar. 2011 to Jun. 2011, while the second task which was executed parallelly with Task 1 only lasted a few days.

Figure 8.5 (c) compares the relative durations of the 48 tasks by projecting the absolute task durations in Fig. 8.5 (a) onto the vertical axis. According to Fig. 8.5, design tasks that spent a long time or were interrupted frequently could be potential bottlenecks or irregular executions, which should be highlighted for deeper investigation. For example, the bar chart in Fig. 8.5 (c) shows that Task 11 had the longest duration and had been interrupted frequently during its execution. Therefore, the documents related to Task 11 should be further analyzed to find the root causes.

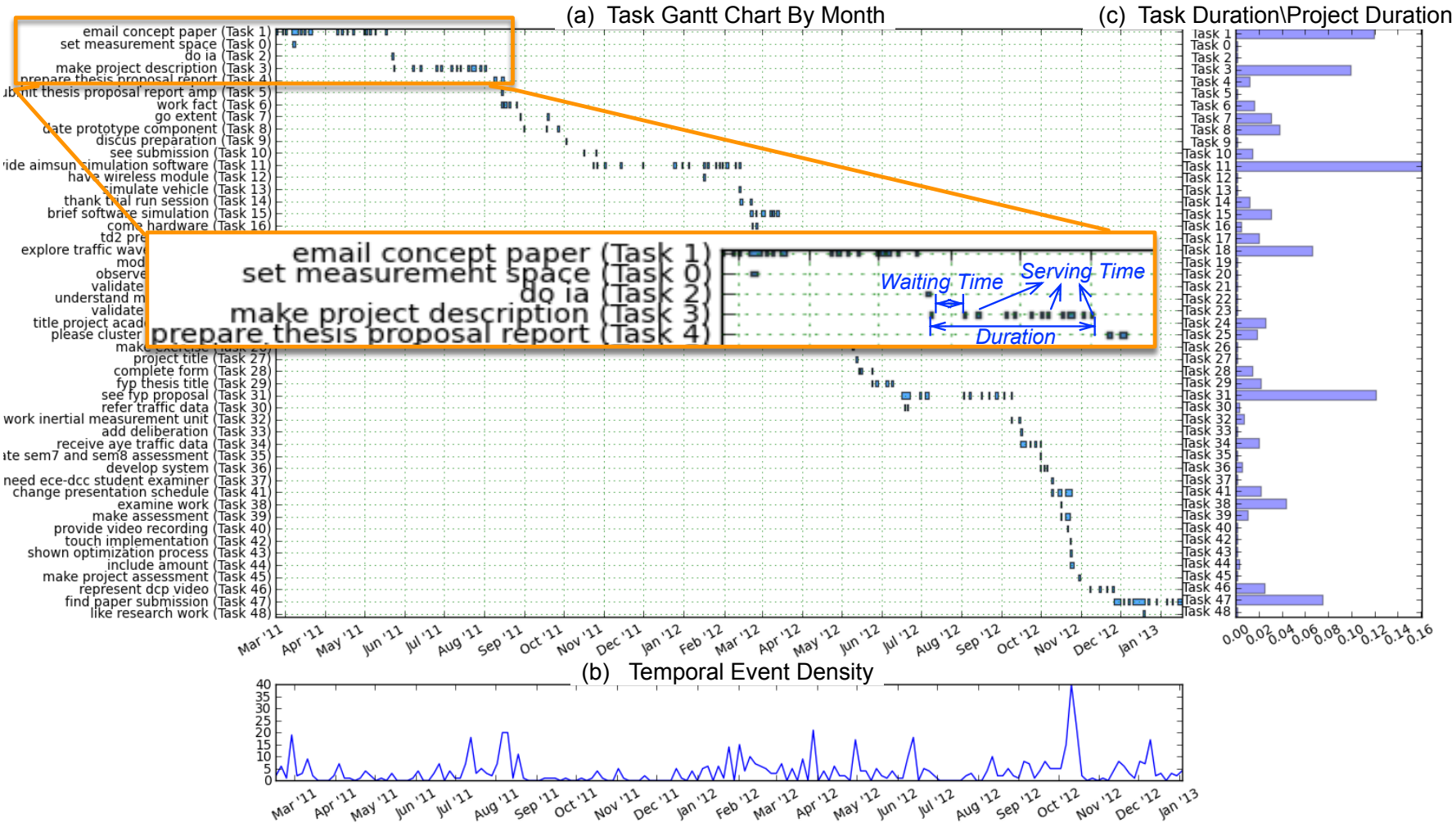


Figure 8.5 Temporal behavior of design tasks

By projecting the number of events that were executed in a short period onto the horizontal axis, Fig. 8.5 (b) plots the variation of the temporal event throughput. Time periods with a low event throughput might include potential bottlenecks that hinder the entire process. Under this assumption, the data in Fig. 8.5 (b) also reveals that Task 11 might be the cause that delayed the entire project as the numbers of executed events from Sep. 2011 to Dec. 2011 are relatively small, less than five.

8.4.2 Temporal Behavior of Human Resources

In the same way, the Gantt chart can also be used to analyze the human resource behavior, if replacing the tasks along the vertical axis of Fig. 8.5 (a) with the involved people. Figure 8.6 (a) depicts such a Gantt chart for temporal human resource behavior analysis. In Fig. 8.6 (a), each line corresponds to a people. The dots in each line indicate that the corresponding people was involved in an event at a certain time point. Similar to Fig. 8.5 (c), Fig. 8.6 (b) compares the relative contributions of all the involved participants to the entire project by projecting their overall frequency onto the vertical axis.

From the dot distribution on each line of Fig. 8.6, it can be observed that P0, P1, P7, P8, P9, P10, and P12 participated in almost all the events throughout the lifetime of this TWP project. Therefore, they could be recognized as the core participants of this project. This finding is consistent with the results obtained from the social network graph in Fig. 8.2. The dot distribution also shows that some participants only appeared at the beginning of this project, for example, P4 and P5, while some participants jointed in this project very late, for example, P235-P280.

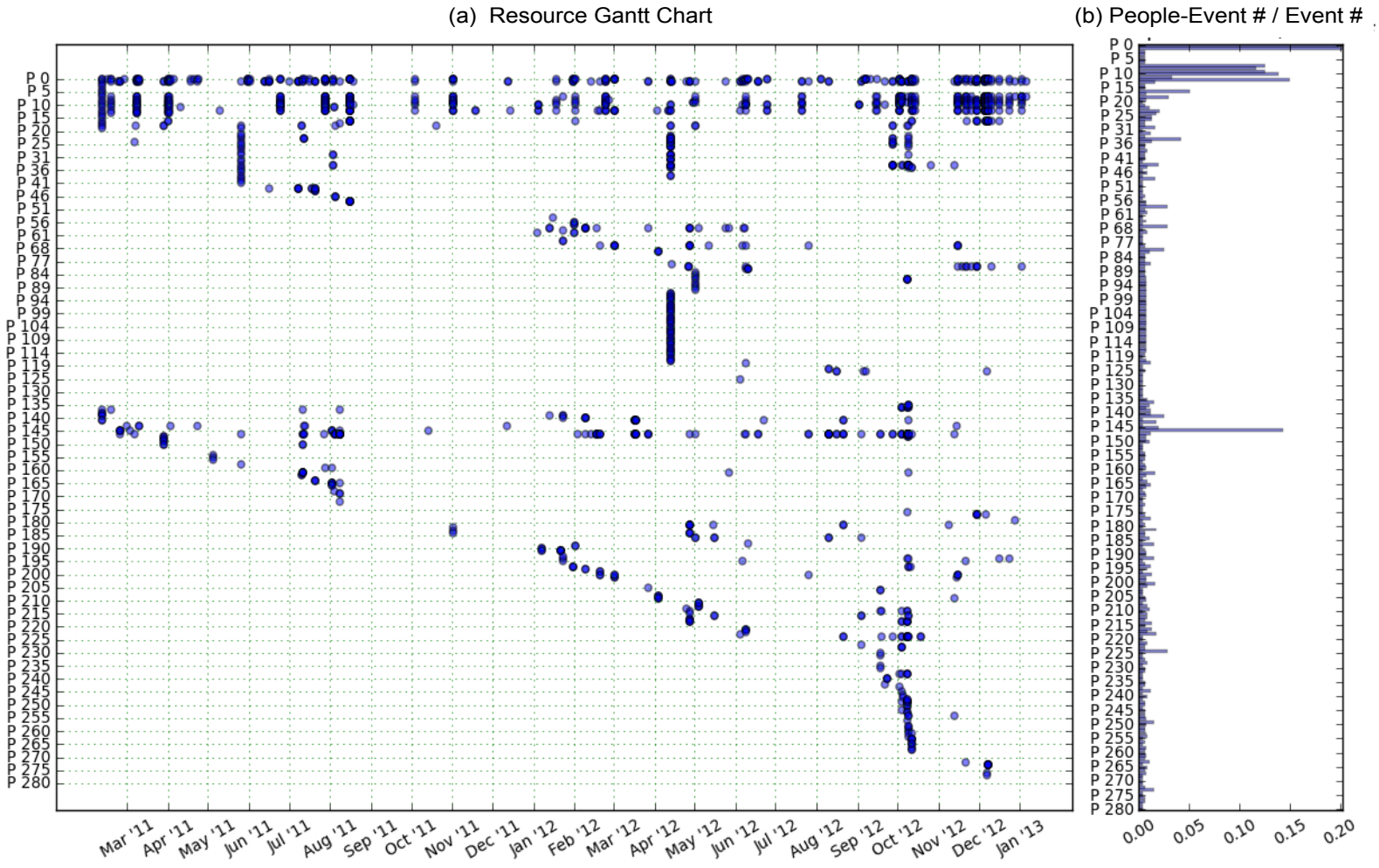


Figure 8.6 Temporal behavior of participants

8.5 Discussions

All the results obtained from the email dataset of the TWP project were assessed by one core member of this project. The feedback confirmed that the discovered process model and the knowledge patterns refined from it indeed represented the innate character of their processes. For the organization mining, the interviewed participant was somehow surprised that more than 200 people were involved throughout the project as there were only eight core members at the beginning. When given the social network graphs constructed from the discovered process model, the interviewed participant could name the different cliques, and recognize the clique consisted of the core members. The four people who were recognized as managers in the social network graph were also verified that they indeed had supervised this project. It is also interesting to point out that three of the four managers, i.e., P1, P18 and P34, failed to join any cliques in Fig. 8.3. This is because admins can have strong interactions with clique leaders, but less interactions with regular participants, whom have formed the major part of a project team. For the temporal behavior analysis, the interviewed participant stated that Task 11 shown in Fig. 8.5 indeed slowed down the whole project for several months. It was stated that Task 11 was about validating the developed traffic control system, and they spent several months to find the appropriate simulation software. Taken together, the expert feedback gave very positive comments to both the discovered process model and the information distilled from it.

8.6 Summary

Based on the hierarchical process model discovered in Chapter 7, this chapter shifts the attention to analyzing process performance from other perspectives for design knowledge interpretation and reutilization. An integrated design knowledge reutilization framework was proposed. Two significant perspectives of the proposed knowledge reutilization framework were illustrated through the case study of the TWP project. They are organization mining and temporal process behavior analysis. In the organization mining, the interaction patterns of the project participants were analyzed for cooperation clique discovery, role discovery, and human resource allocation. From the temporal perspective, both the task behaviors and the human resource behaviors were analyzed using Gantt charts. The results aligned well with the expert feedback.

CHAPTER 9 CONCLUSIONS AND RECOMMENDATIONS

9.1 Conclusions

The objective of this thesis is to develop a knowledge discovery system for extracting design process knowledge from design documents. The developed knowledge discovery system could be used as a tool to provide decision makers with right information in time, support decision makers in efficiently learning valuable experience from historical design projects, and help decision makers to reuse the learned experience in current or future design projects. Considering the characteristics of both the product design process and the design documents, approaches of extracting process information from textual data, detecting design events from the extracted process information, mining design process models from the detected events, as well as discovering multi-faceted design knowledge from the discovered process model have been developed and presented. The experimental results based on the TWP project indicate that:

1) The design documents collected from past design projects contain enough information to extract the design process model.

2) It is feasible to extract the design process model from the archival design documents using proper text mining and process mining techniques, although the current approaches have some limitations.

3) The knowledge distilled from the discovered process model include not only known knowledge but also unknown knowledge. Therefore, the discovered knowledge

could be used to assist decision makers by enhancing and extending their personal knowledge bases.

4) Based on the process model and the process knowledge discovered, it becomes possible to manage and compare design knowledge from several design projects in a structured, quantitative manner.

9.2 Contributions

As presented in Chapter 3, the developed process knowledge discovery system has three core components: process information extraction, process mining, and process knowledge interpretation. Based on the three core components, the major contributions of this work are summarized as follows.

In Chapter 4 and Chapter 5, two information extraction approaches have been proposed to extract unstructured process information from design documents.

- A DBN based topic modeling approach has been presented to extract topics that are relevant to design task executions from the design documents. With the task-relevant topics, the interaction patterns of the design tasks can be estimated from the co-occurrence frequency of the corresponding topics, and the dynamic changes of the process status can be reflected by the changes of the corresponding topics. In addition, the presented topic modeling approach is totally unsupervised. Therefore, there is no requirement for manual annotation.

- A hybrid NER approach has been proposed to identify special writing terms or phrases that refer to physical objects, which have been involved in the underlying design process. The proposed NER approach takes advantage of both the ruled-based and the machine learning-based NER techniques. Therefore, it reduces the human intervention required by most of the traditional approaches to a minimum. In addition, to improve the accuracy of recognizing the process-relevant entities, a local dependency tree has been designed to capture the linguistic features of the entities in terms of tree structure.

Based on the extracted process information, Chapter 6 and Chapter 7 focus on constructing process model via design event detection and process mining.

- A higher-order ERE approach has been proposed to detect design events from design documents by extracting the higher-order relations among the process-relevant entities. The main idea is decomposing the higher-order relations in an event into several binary relations, and then reconstructing the event by finding the maximum cliques centered at each task entity. While most the traditional higher-order ERE approaches heavily rely on the binary classifier, the graph partition based ERE approach proposed in this work is much simpler by using graph density to eliminate both noisy entities and noisy events.

- A bottom-up process mining approach has been developed to discover a hierarchical process model on the basis of the detected design events. To deal with the flexibility

of product design processes, design events with similar execution context are iteratively merged into bigger design tasks. This aggregation operation would result in several abstracted process models, which is much simpler and easier for understanding when compared to the flat models produced by most of the existing process mining approaches. In addition, the experimental results indicate that this bottom-up process mining approach outperforms the top-down approach in terms of accuracy.

- A top-down process mining approach has been developed to discover a hierarchical process model from generation to specification on the basis of the extracted topics and the detected design events. This approach treats the entire process as a big black box, which can be recursively decomposed into several modules based on the document content determined by the per-document topic distribution. This approach outperforms the bottom-up process mining in terms of time cost.

Based on the discovered process model, Chapter 8 aims to discover design knowledge from other perspectives and to reuse the discovered design knowledge for decision making.

- An integrated design knowledge reutilization framework has been proposed. To overcome the problem that most of the existing design knowledge reutilization systems are not compatible with the whole design process, the proposed framework treats the discovered process model as the central element of the design knowledge

and links other types of design knowledge such as product and organization to the process model.

- The personnel perspective of the proposed knowledge reutilization framework has been illustrated using a real-life case study. A series of organization mining approaches were introduced to analyze the cooperation patterns of the project participants, the participant roles, and the human resource allocation.
- The temporal perspective of the proposed knowledge reutilization framework has also been illustrated using the same real-life case study. The Gantt chart was used to analyze the temporal behaviors of both the design tasks and the project participants.

In addition, it is noteworthy that the above approaches are not restricted to the discovery of product design processes. Other types of processes that involve discovering tasks and workflows from textual data can also be suited to use the above approaches.

9.3 Limitations

The experiment results also revealed several limitations of the developed process-oriented knowledge discovery system.

From the data perspective, this research used an email dataset collected from a real-life design project to discover the underlying design process. Although the experiment results indicated that the proposed approaches were able to discover the underlying design process and the discovered process model indeed had a good reflection of the reality, the expert

feedback also revealed that the details of some tasks were missed in the discovered process model. This is because the emails are just one type of the design documents that have been accumulated during design processes, and there are many other types of documents such as progress reports, minutes, technique reports, and CAD files. These documents might record information that are not included in the emails. Therefore, to improve the quality of the discovered process model, more types of design documents should be taken into account.

From the process model perspective, one weakness is that the focus of the current process-oriented knowledge discovery system is more on discovering design process, without connecting the designed product to the discovered process model. As a result, the discovered process model lacks an ability of providing detailed information about the created product. It is noteworthy that among the archival design documents, there is a large proportion of documents which embed detailed information of product itself, such as geometrical structures and component graphics. Extracting and integrating such product information into the procedure of design process mining would be helpful to generate more powerful process models that connect product and design processes together.

From the technical perspective, the NER approach proposed in Chapter 5 only considered seven types of named entities. In reality, design processes are usually more complex than the TWP project used in the case study, and much more types of named entities are involved. Although the proposed NER approach can be easily expanded for recognizing more types of named entities, it needs to be tested on more complex product

design projects. In addition, the two process mining approaches proposed in Chapter 7 have eliminated the loops in the outputted hierarchical process model for the sake of brevity. However, in most cases, loops caused by uncertainties are a key feature of product design processes. Therefore, the discovered process model would not reflect the actual design processes well if they have many loops.

Lastly, from the validation perspective, all the approaches integrated in the developed knowledge discovery system were tested on a single case study. As mentioned in Section 3.3, the selected case study is a university-hosted design project, which has the common characteristics of a typical design process. Therefore, it can be used to test the feasibility of the developed knowledge discovery systems. However, the generality needs to be further tested on design documents collected form different design projects.

9.4 Recommendations for Future Work

This research work has proved the feasibility of discovering process model and process-oriented knowledge from design documents using the proposed approaches. This feasibility and the limitations discussed in Section 9.3 also open multiple possibilities for future extensions, which may lead to develop more efficient, comprehensive, and reliable design knowledge discovery systems. Some of these possibilities are listed below:

- Use a variety of data – The experiment dataset used in this research work are emails collected from a real-life design project. It would be interesting to explore how the proposed approaches perform on other types of textual data, or how the information

recorded in other types of design documents can be extracted and utilized to enrich the process model.

- Test the generality on more design projects – Based on the TWP project described in Section 3.4, this research work has proved the feasibility of discovering process model and process-oriented knowledge from archival design documents. However, the generality of the proposed approaches need to be further validated on more design projects in the future.

- Identify loops – In product design, loops caused by all kinds of uncertainties are a key feature of product design processes. To capture this characteristic of the product design process, the knowledge discovery system presented in thesis will be improved with the ability of identifying loops in the underlying design process.

- Connect product to process – The discovered process model can also be improved by connecting the created product to the design process creating it. In practice, there is a large proportion of design documents which embed detailed information of product itself; such as geometrical structures and component graphics. Extracting and integrating such product information into the procedure of design process mining would be helpful to generate more powerful process models that connect product and design processes together.

- Integrate prior knowledge with the proposed approaches for process optimization – Based on the process-oriented knowledge discovered, the current system can also be

extended to support decision making about design process optimization. For example, the process model can be used to quantitatively setup the optimization parameters (e.g., task dependencies, execution durations, and personnel skills) of optimization functions, which are built by prior knowledge and aim to reduce the time-to-market, or to optimize resource allocation.

- Develop ontology-based knowledge management and retrieval systems – The proposed process mining approaches also introduce the possibility of managing and retrieving past design documents in a structured, graphic manner. For example, by representing a design project as the process model uncovered from its archival documents, past design projects can be compared according to the structure similarity of their process models. This is a critical step to search for and reutilize interesting information from large quantities of past design projects.

REFERENCES

[1] ElMaraghy, H., et al., 2009, "Managing Variations in Products, Processes and Manufacturing Systems," CIRP Annals-manufacturing technology, **58**(1), pp. 441-446.

[2] Karl T. Ulrich, S. D. E., 2012, *Product Design and Development*. 5th ed ed. New York, NY: McGraw-Hill/Irwin.

[3] Braha, D., 2013, *Data Mining for Design and Manufacturing: Methods and Applications*. Vol. 3: Springer Science & Business Media.

[4] Fayyad, U. M., et al., 1996, *Advances in Knowledge Discovery and Data Mining*. Vol. 21: AAAI press Menlo Park.

[5] Liew, A., 2007, "Understanding Data, Information, Knowledge and Their Inter-Relationships," Journal of Knowledge Management Practice, **8**(2), pp. 1-16.

[6] ElMaraghy, W., 2009, "Knowledge Management in Collaborative Engineering," International Journal of Collaborative Engineering, **1**(1-2), pp. 114-124.

[7] Van der Aalst, W., et al., 2004, "Workflow Mining: Discovering Process Models from Event Logs," IEEE Transactions on Knowledge and Data Engineering, **16**(9), pp. 1128-1142.

[8] Noh, H., et al., 2015, "Keyword Selection and Processing Strategy for Applying Text Mining to Patent Analysis," Expert Systems with Applications, **42**(9), pp. 4348-4360.

[9] Yoon, B., and Y. Park, 2004, "A Text-Mining-Based Patent Network: Analytical Tool for High-Technology Trend," The Journal of High Technology Management Research, **15**(1), pp. 37 - 50.

[10] Wang, J., et al., 2015, "A Two-Level Parser for Patent Claim Parsing," Advanced Engineering Informatics, **29**(3), pp. 431-439.

[11] Yu, W. D., and J. Y. Hsu, 2013, "Content-Based Text Mining Technique for Retrieval of Cad Documents," Automation in Construction, **31**, pp. 65-74.

[12] Bai, J., et al., 2010, "Design Reuse Oriented Partial Retrieval of Cad Models," Computer-Aided Design, **42**(12), pp. 1069-1084 %@ 0010-4485.

[13] Bosche, F., and C. Haas, 2008, "Automated Retrieval of 3d Cad Model Objects in Construction Range Images," Automation in Construction, **17**(4), pp. 499-512.

[14] Park, Y., and S. Lee, 2011, "How to Design and Utilize Online Customer Center to Support New Product Concept Generation," Expert Systems with Applications, **38**, pp. 10638-10647.

[15] Jin, J., et al., 2015, "Translating Online Customer Opinions into Engineering Characteristics in Qfd: A Probabilistic Language Analysis Approach," Engineering Applications of Artificial Intelligence, **41**, pp. 115-127.

[16] Rozinat, a., and W. M. P. van der Aalst, 2008, "Conformance Checking of Processes Based on Monitoring Real Behavior," Information Systems, **33**, pp. 64-95.

[17] van der Aalst, W., et al., 2012, "Replaying History on Process Models for Conformance Checking and Performance Analysis," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, **2**(2), pp. 182-192.

[18] da Cruz, J. I. B., and D. D. Ruiz, 2011, "Conformance Analysis on Software Development: An Experience with Process Mining," International Journal of Business Process Integration and Management, **5**(2), pp. 109-120.

[19] Rozinat, A., and Wil MP van der Aalst, 2005, "Conformance Testing: Measuring the Fit and Appropriateness of Event Logs and Process Models," In International Conference on Business Process Management, Nancy, France, pp. 163-176.

[20] Mans, R. S., et al., 2008, "Application of Process Mining in Healthcare–a Case Study in a Dutch Hospital," Biomedical Engineering Systems and Technologies, **25**, pp. 425-438.

[21] Rozinat, A., et al., 2009, "Workflow Simulation for Operational Decision Support," Data & Knowledge Engineering, **68**(9), pp. 834 - 850.

[22] Liu, Y., et al., 2012, "Workflow Simulation for Operational Decision Support Using Event Graph through Process Mining," Decision Support Systems, **52**(3), pp. 685-697.

[23] Ur-Rahman, N., and J. a. Harding, 2012, "Textual Data Mining for Industrial Knowledge Management and Text Classification: A Business Oriented Approach," Expert Systems with Applications, **39**(5), pp. 4729-4739.

[24] Browning, T. R., et al., 2006, "Key Concepts in Modeling Product Development Processes," Systems Engineering, **9**(2), pp. 104-128.

[25] Sullivan, D., 2001, *Document Warehousing and Text Mining: Techniques for Improving Business Operations, Marketing, and Sales*: John Wiley & Sons, Inc.

[26] Tan, A.-H., 1999, "Text Mining: The State of the Art and the Challenges," In Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases, pp. 65-70.

[27] Bouckaert, R. R., 2002, "Low Level Information Extraction: A Bayesian Network Based Approach," In Proc of the 12 th Int. Conference on Machine Learning, pp. 194-202.

[28] Li, Y., et al. 2005. "Svm Based Learning System for Information Extraction." In *Deterministic and Statistical Methods in Machine Learning*, 185-1192. Springer.

[29] Sarawagi, S., and W. W. Cohen, 2004, "Semi-Markov Conditional Random Fields for Information Extraction," Advances in neural information processing systems, pp. 1185-1192.

[30] Petrović, J., et al., 2012, "Optimization of Matrix Tablets Controlled Drug Release Using Elman Dynamic Neural Networks and Decision Trees," International Journal of Pharmaceutics, **428**(1), pp. 57-67.

[31] Choy, K. L., et al., 2005, "A Knowledge-Based Supplier Intelligence Retrieval System for Outsource Manufacturing," Knowledge-based systems, **18**(1), pp. 1-17 %@ 0950-7051.

[32] Venugopal, V., and T. T. Narendran, 1992, "Neural Network Model for Design Retrieval in Manufacturing Systems," Computers in industry, **20**(1), pp. 11-23 %@ 0166-3615.

[33] Trappey, A. J. C., et al., 2009, "A Fuzzy Ontological Knowledge Document Clustering Methodology," IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, **39**(3), pp. 806-814.

[34] Murphy, J., et al., 2014, "Function Based Design-by-Analogy: A Functional Vector Approach to Analogical Search," Journal of Mechanical Design, Transactions of the ASME, **136**(10), pp. 101-102.

[35] Xue, D., and Z. Dong, 1997, "Coding and Clustering of Design and Manufacturing Features for Concurrent Design," Computers in Industry, **34**(1), pp. 139-153 %@ 0166-3615.

[36] Kim, Y. G., et al., 2008, "Visualization of Patent Analysis for Emerging Technology," Expert Systems with Applications, **34**(3), pp. 1804-1812 %@ 0957-4174.

[37] Trappey, C. V., et al., 2011, "Using Patent Data for Technology Forecasting: China Rfid Patent Analysis," Advanced Engineering Informatics, **25**(1), pp. 53-64 %@ 1474-0346.

[38] Trappey, C. V., et al., 2010, "Clustering Patents Using Non-Exhaustive Overlaps," Journal of Systems Science and Systems Engineering, **19**(2), pp. 162-181 %@ 1004-3756.

[39] Liang, Y., et al., 2012, "Learning the "Whys": Discovering Design Rationale Using Text Mining—an Algorithm Perspective," Computer-Aided Design, **44**(10), pp. 916-930.

[40] Lan, L., et al., 2015, "Automatic Discovery of Design Task Structure Using Deep Belief Nets," In ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Boston, Massachusetts, USA.

[41] Lam, P., and M. Rinard, 2003, "A Type System and Analysis for the Automatic Extraction and Enforcement of Design Information," In European Conference on Object-Oriented Programming, pp. 275-302.

[42] Li, Z., and K. Ramani, 2007, "Ontology-Based Design Information Extraction and Retrieval," AI EDAM: Artificial Intelligence for Engineering Design, Analysis, and Manufacturing, **21**(02), pp. 137-154 %@ 1469-1760.

[43] Jin, G., et al., 2015, "Technology-Driven Roadmaps for Identifying New Product/Market Opportunities: Use of Text Mining and Quality Function Deployment," Advanced Engineering Informatics, **29**(1), pp. 126-138.

[44] Rajpathak, D. G., 2013, "An Ontology Based Text Mining System for Knowledge Discovery from the Diagnosis Data in the Automotive Domain," Computers in Industry, **64**(5), pp. 565-580.

[45] Efthymiou, K., et al., 2015, "On Knowledge Reuse for Manufacturing Systems Design and Planning: A Semantic Technology Approach," CIRP Journal of Manufacturing Science and Technology, **8**, pp. 1-11.

[46] Blei, D. C., Lawrence; Dunson, David, "Probabilistic Topic Models," IEEE Signal Processing Magazine, **27**(6), pp. 55-65.

[47] Griffiths, T. 2002. "Gibbs Sampling in the Generative Model of Latent Dirichlet Allocation." http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.138.3760.

[48] Blei, D. M., and J. D. Lafferty, 2006, "Dynamic Topic Models," In Proceedings of the 23rd international conference on Machine learning, pp. 113-120.

[49] Weinshall, D., et al., 2013, "Lda Topic Model with Soft Assignment of Descriptors to Words," In 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, United states, pp. 1748-1756.

[50] Yanning, Z., and W. Wei, 2014, "A Jointly Distributed Semi-Supervised Topic Model," Neurocomputing, **134**, pp. 38-45.

[51] Gehler, P. V., et al., 2006, "The Rate Adapting Poisson Model for Information Retrieval and Object Recognition," In ICML 2006: 23rd International Conference on Machine Learning, Pittsburgh, PA, United states, pp. 337-344.

[52] Hinton, G. E., and R. Salakhutdinov, 2009, "Replicated Softmax: An Undirected Topic Model," In 23rd Annual Conference on Neural Information Processing Systems, NIPS 2009, Vancouver, BC, Canada, pp. 1607-1614.

[53] Bengio, Y., 2009, "Learning Deep Architectures for Ai," Foundations and trends in Machine Learning, **2**(1), pp. 1-27.

[54] Abdelbary, H. A., et al., 2014, "Utilizing Deep Learning for Content-Based Community Detection," In 2014 Science and Information Conference, SAI 2014, London, United kingdom, pp. 777-784.

[55] Ko, K. E., and K. B. Sim, 2010, "Development of a Facial Emotion Recognition Method Based on Combining Aam with Dbn," In Proceedings - 2010 International Conference on Cyberworlds, CW 2010, pp. 87-91.

[56] Ravyse, I., et al., 2006, "Dbn Based Models for Audio-Visual Speech Analysis and Recognition," In Advances in Multimedia Information Processing - Pcm 2006, Proceedings, pp. 19-30.

[57] Marrero, M., et al., 2013, "Named Entity Recognition: Fallacies, Challenges and Opportunities," Computer Standards & Interfaces, **35**(5), pp. 482-489.

[58] Saggion, H., et al. 2007. "Ontology-Based Information Extraction for Business Intelligence." In *The Semantic Web*, 843-856. Springer Berlin Heidelberg.

[59] Alani, H., et al., 2003, "Automatic Ontology-Based Knowledge Extraction from Web Documents," IEEE Intelligent Systems, **18**(1), pp. 14-21 %@ 1541-1672.

[60] Cimiano, P., and J. Völker, 2005, "Towards Large-Scale, Open-Domain and Ontology-Based Named Entity Classification," In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP), pp. 66-166.

[61] Rizzo, G., and R. Troncy, 2012, "Nerd: A Framework for Unifying Named Entity Recognition and Disambiguation Extraction Tools," In Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics, pp. 73-76.

[62] Rau, L., 1991, "Extracting Company Names from Text," The Seventh IEEE Conference on Artificial Intelligence Application, **1**, pp. 29-32.

[63] Settles, B., 2004, "Biomedical Named Entity Recognition Using Conditional Random Fields and Rich Feature Sets," In Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications, pp. 104-107.

[64] Isozaki, H., and H. Kazawa, 2002, "Efficient Support Vector Classifiers for Named Entity Recognition," In Proceedings of the 19th international conference on Computational linguistics, pp. 1-7.

[65] Riloff, E., and R. Jones, 1999, "Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping," AAAI/IAAI, pp. 474-479.

[66] Nothman, J., et al., 2013, "Learning Multilingual Named Entity Recognition from Wikipedia," Artificial Intelligence, **194**, pp. 151 - 175.

[67] Cucchiarelli, A., and P. Velardi, 2001, "Unsupervised Named Entity Recognition Using Syntactic and Semantic Contextual Evidence," Computational Linguistics, **27**(1), pp. 123-131.

[68] Yang, T. Y., 2009, "Simple Bayesian Binary Framework for Discovering Significant Genes and Classifying Cancer Diagnosis," Computational Statistics & Data Analysis, **53**(3), pp. 1743-1754.

[69] Chen, Y., et al., 2015, "A Study of Active Learning Methods for Named Entity Recognition in Clinical Text.," Journal of biomedical informatics, **58**, pp. 11-8.

[70] Dong, X., et al., 2016, "A Multiclass Classification Method Based on Deep Learning for Named Entity Recognition in Electronic Medical Records," In 2016 New York Scientific Data Summit (NYSDS), pp. 1-10.

[71] Korkontzelos, I., and D. Piliouras, 2015, "Boosting Drug Named Entity Recognition Using an Aggregate Classifier," Artificial intelligence in medicine, **65**(2), pp. 145.

[72] Rocktäschel, T., et al., 2012, "Chemspot: A Hybrid System for Chemical Named Entity Recognition.," Bioinformatics (Oxford, England), **28**(12), pp. 1633-40.

[73] Freitas, C., et al., 2009, "Relation Detection between Named Entities: Report of a Shared Task," In Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions, pp. 129-137.

[74] Lafferty, J., et al., 2001, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," In Proceedings of the eighteenth international conference on machine learning, ICML, pp. 282-289.

[75] Zheng, S., et al., 2016, "Joint Learning of Entity Semantics and Relation Pattern for Relation Extraction," In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 443-458.

[76] Kambhatla, N., 2004, "Combining Lexical, Syntactic, and Semantic Features with Maximum Entropy Models for Extracting Relations," In Proceedings of the ACL 2004 on Interactive poster and demonstration sessions, pp. 22.

[77] Freitag, D., and A. McCallum, 2000, "Information Extraction with Hmm Structures Learned by Stochastic Optimization," AAAI/IAAI, **2000**, pp. 584-589.

[78] Wang, T., et al. 2006. "Automatic Extraction of Hierarchical Relations from Text." In *The Semantic Web: Research and Applications*, 215-229. Springer Berlin Heidelberg.

[79] Lodhi, H., et al., 2002, "Text Classification Using String Kernels," Journal of Machine Learning Research, **2**(Feb), pp. 419-444.

[80] Mooney, R. J., and R. C. Bunescu, 2006, "Subsequence Kernels for Relation Extraction," In Advances in neural information processing systems, pp. 171-178.

[81] Zelenko, D., et al., 2003, "Kernel Methods for Relation Extraction," Journal of machine learning research, **3**(Feb), pp. 1083-1106.

[82] GuoDong, Z., et al., 2005, "Exploring Various Knowledge in Relation Extraction," In Proceedings of the 43rd annual meeting on association for computational linguistics, pp. 427-434.

[83] Bunescu, R. C., and R. J. Mooney, 2005, "A Shortest Path Dependency Kernel for Relation Extraction," In Proceedings of the conference on human language technology and empirical methods in natural language processing, pp. 724-731.

[84] Nguyen, T. H., et al., 2015, "Semantic Representations for Domain Adaptation: A Case Study on the Tree Kernel-Based Method for Relation Extraction," In ACL, pp. 635-644.

[85] Sun, L., and X. Han. 2014. "A Feature-Enriched Tree Kernel for Relation Extraction." https://www.aclweb.org/anthology/P/P14/P14-2011.pdf.

[86] Fundel, K., et al., 2007, "Relex—Relation Extraction Using Dependency Parse Trees," Bioinformatics, **23**(3), pp. 365-371.

[87] Brin, S., 1998, "Extracting Patterns and Relations from the World Wide Web," In International Workshop on The World Wide Web and Databases, pp. 172-183.

[88] Agichtein, E., and L. Gravano, 2000, "Snowball: Extracting Relations from Large Plain-Text Collections," In Proceedings of the fifth ACM conference on Digital libraries, pp. 85-94.

[89] Etzioni, O., et al., 2005, "Unsupervised Named-Entity Extraction from the Web: An Experimental Study," Artificial intelligence, **165**(1), pp. 91-134.

[90] Yates, A., et al., 2007, "Textrunner: Open Information Extraction on the Web," In Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations, pp. 25-26.

[91] de Abreu, S. C., et al., 2013, "A Review on Relation Extraction with an Eye on Portuguese," Journal of the Brazilian Computer Society, **19**(4), pp. 553-571.

[92] Angeli, G., et al., 2014, "Combining Distant and Partial Supervision for Relation Extraction," In EMNLP, pp. 1556-1567.

[93] Min, B., et al., 2013, "Distant Supervision for Relation Extraction with an Incomplete Knowledge Base," In HLT-NAACL, pp. 777-782.

[94] Krause, S., et al., 2012, "Large-Scale Learning of Relation-Extraction Rules with Distant Supervision from the Web," In The Semantic Web–ISWC 2012, pp. 263-278.

[95] Hasegawa, T., et al., 2004, "Discovering Relations among Named Entities from Large Corpora," In Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, pp. 415.

[96] Etzioni, O., et al., 2011, "Open Information Extraction: The Second Generation," In Twenty-Second International Joint Conference on Artificial Intelligence, pp. 3-10.

[97] Wu, F., and D. S. Weld, 2010, "Open Information Extraction Using Wikipedia," In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 118-127.

[98] Nguyen, T. H., et al., 2016, "Joint Event Extraction Via Recurrent Neural Networks," In Proceedings of NAACL-HLT, pp. 300-309.

[99] Ritter, A., et al., 2012, "Open Domain Event Extraction from Twitter," In Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1104-1112.

[100] Li, J., et al., 2014, "Major Life Event Extraction from Twitter Based on Congratulations/Condolences Speech Acts," In EMNLP, pp. 1997-2007.

[101] Hogenboom, F., et al., 2016, "A Survey of Event Extraction Methods from Text for Decision Support Systems," Decision Support Systems, **85**, pp. 12-22.

[102] Jans, M., et al., 2011, "A Business Process Mining Application for Internal Transaction Fraud Mitigation," Expert Systems with Applications, **38**(10), pp. 13351-9.

[103] Aalst, W. V. D., et al., 2012, "Replaying History on Process Models for Conformance Checking and Performance Analysis," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, **2**(2), pp. 182-192.

[104] van der Zee, D. J., 2011, "Building Insightful Simulation Models Using Petri Nets — a Structured Approach," Journal of Decision Support Systems, **51**(1), pp. 53-64.

[105] Luengo, D., and M. Sepúlveda, 2012, "Applying Clustering in Process Mining to Find Different Versions of a Business Process That Changes over Time Extending Trace Clustering Techniques to Include The," In International Conference on Business Process Management, pp. 153-158.

[106] Rozinat, A., et al., 2009, "Workflow Simulation for Operational Decision Support," Data & Knowledge Engineering, **68**(9), pp. 834-850.

[107] Di Francescomarino, C., et al., 2016, "Predictive Business Process Monitoring Framework with Hyperparameter Optimization," In International Conference on Advanced Information Systems Engineering, Cham, pp. 361-376.

[108] Vergidis, K., et al., 2015, "An Automated Optimisation Framework for the Development of Re-Configurable Business Processes: A Web Services Approach," International Journal of Computer Integrated Manufacturing, **28**(1), pp. 41-58.

[109] Agrawal, R., et al., 1998, "Mining Process Models from Workflow Logs," In Proceedings of the 6th International Conference on Extending Database Technology: Advances in database Technology, Valencia, Spain, pp. 469-483.

[110] van der Aalst, W. M. P., et al., 2004, "Workflow Mining: Discovering Process Models from Event Logs," IEEE Transactions on Knowledge and Data Engineering, **16**, pp. 1128-1142.

[111] Wen, L., et al., 2009, "A Novel Approach for Process Mining Based on Event Types," Journal of Intelligent Information Systems **32**(2), pp. 163-190.

[112] Huang, H.-m., and Y. Zhang, 2008, "Process Mining Algorithm to Discover Non-Certain Choice and Parallel Relation," Journal of Computer Applications, **28**(11), pp. 2922-5.

[113] Tiwari, A., et al., 2008, "A Review of Business Process Mining: State-of-the-Art and Future Trends," Business Process Management Journal, **14**(1), pp. 5-22.

[114] Li, J., et al., 2011, "A Heuristic Genetic Process Mining Algorithm," CIS, **2011**, pp. 15-19.

[115] Schimm, G., 2004, "Mining Exact Models of Concurrent Workflows," Computers in Industry, **53**(3), pp. 265-81.

[116] Schonig, S., et al., 2012, "Adapting Association Rule Mining to Discover Patterns of Collaboration in Process Logs," In Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on, pp. 531-534.

[117] De Leoni, M., and d. A. W. M. P. Van, 2016, "Data-Aware Process Mining: Discovering Decisions in Processes Using Alignments," In Proceedings of the 28th annual ACM symposium on applied computing, pp. 1454-1461.

[118] Repta, D., et al., 2017, "Towards the Development of Semantically Enabled Flexible Process Monitoring Systems," International Journal of Computer Integrated Manufacturing, **30**(1), pp. 96-108.

[119] Gunther, C. W., and W. M. P. van der Aalst, 2007, "Fuzzy Mining - Adaptive Process Simplification Based on Multi-Perspective Metrics," In Business Process Management. 5th International Conference, BPM 2007, 24-28 Sept. 2007, Berlin, Germany, pp. 328-43.

[120] Maggi, F. M., et al., 2011, "User-Guided Discovery of Declarative Process Models," In Symposium Series on Computational Intelligence, IEEE SSCI2011 - 2011 IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, April 11, 2011 - April 15, 2011, Paris, France, pp. 192-199.

[121] Di Ciccio, C., and M. Mecella, 2013, "A Two-Step Fast Algorithm for the Automated Discovery of Declarative Workflows," In 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), 16-19 April 2013, Piscataway, NJ, USA, pp. 135-42.

[122] Maggi, F. M., et al., 2013, "Online Process Discovery to Detect Concept Drifts in Ltl-Based Declarative Process Models," In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Berlin, Germany, pp. 94-111.

[123] Diamantini, C., et al., 2016, "Behavioral Process Mining for Unstructured Processes," Journal of Intelligent Information Systems, **47**(1), pp. 5-32.

[124] Rozinat, A., et al., 2009, "Discovering Simulation Models," Information Systems, **34**(3), pp. 305-27.

[125] Liu, T., et al., 2012, "Mining Event Logs to Support Workflow Resource Allocation," Knowledge-Based Systems, **35**, pp. 320-331.

[126] Fern, C., et al., 2014, "Temporal Abstractions to Enrich Activity-Based Process Mining Corpus with Clinical Time Series ." In Biomedical and Health Informatics (BHI), 2014 IEEE-EMBS International Conference, pp. 785-788.

[127] Zhang, H., et al. 2010. "A Novel Approach of Process Mining with Event Graph." In *Knowledge-Based and Intelligent Information and Engineering Systems*, 131 - 140. Springer Berlin Heidelberg.

[128] Aalst, W. V. D., et al., 2012, "Process Mining Manifesto," Information Systems, **37**(3), pp. 288-290.

[129] Weijters, A. J. M. M., and Wil MP van der Aalst, 2001, "Process Mining Discovering Workflow Models from Event-Based Data " In Belgium-Netherlands Conference on Artificial Intelligence.

[130] De Weerdt, J., et al., 2013, "Process Mining for the Multi-Faceted Analysis of Business Processes—a Case Study in a Financial Services Organization," Computers in Industry, **64**(1), pp. 57-67.

[131] Seung-kyung, L., et al., 2013, "Mining Transportation Logs for Understanding the after-Assembly Block Manufacturing Process in the Shipbuilding Industry," Expert Systems with Applications, **40**(1), pp. 83-95.

[132] Caron, F., et al., 2013, "A Comprehensive Investigation of the Applicability of Process Mining Techniques for Enterprise Risk Management," Computers in Industry, **64**(4), pp. 464-475.

[133] Dotoli, M. F., M.P; Mangini, A.M, 2008, "Fault Detection of Discrete Event Systems Using Petri Nets and Integer Linear Programming," IFAC Proceedings Volumes, **41**(2), pp. 6554-6559.

[134] Rebuge, A., and D. R. Ferreira, 2012, "Business Process Analysis in Healthcare Environments: A Methodology Based on Process Mining," Information Systems, **37**(2), pp. 99-116.

[135] Rebuge, Á. F., Diogo R, 2012, "Business Process Analysis in Healthcare Environments: A Methodology Based on Process Mining," Information Systems, **37**(2), pp. 99 - 116.

[136] Kumar, A., and S. Rahman, 2014, "Rfid-Enabled Process Reengineering of Closed-Loop Supply Chains in the Healthcare Industry of Singapore," Journal of Cleaner Production, **85**, pp. 382 - 394.

[137] Rolland, C., 1998, "A Comprehensive View of Process Engineering," In International Conference on Advanced information Systems Engineering, pp. 1 - 24.

[138] Bose, R. P. J. C., et al., 2012, "Discovering Hierarchical Process Models Using Prom," In CAiSE Forum 2011 on IS Olympics: Information Systems in a Diverse World, June 20, 2011 - June 24, 2011, London, United kingdom, pp. 33-48.

[139] Wei Wang; Barnaghi, P. M. B., Andrzej, 2010, "Probabilistic Topic Models for Learning Terminological Ontologies," IEEE Transactions on Knowledge and Data Engineering, **22**(7), pp. 1041-4347.

[140] Steyvers, M., and T. Griffiths. 2007. "Probabilistic Topic Models." In *Handbook of Latent Semantic Analysis*, edited by K Thomas and Danielle Landauer, 427-448. Lawrence Erlbaum Associates.

[141] Barbieri, N., et al., 2013, "Probabilistic Topic Models for Sequence Data," Machine learning, **93**(1), pp. 5-29.

[142] Hinton, G. E., et al., 2006, "A Fast Learning Algorithm for Deep Belief Nets.," Neural computation, **18**(7), pp. 1527-1554.

[143] Marina, A. J. L. U. J. O. S., 1975, *How to Do Things with Words*. Edited by Cambridge. 2d ed. ed. Vol. 1955: Harvard University Press.

[144] Chinchor, N., and P. Robinson, 1997, "Muc-7 Named Entity Task Definition," In Proceedings of the 7th Conference on Message Understanding, pp. 29.

[145] Van der Aalst, W. M. P., and M. Song, 2004, "Mining Social Networks: Uncovering Interaction Patterns in Business Processes," In International Conference on Business Process Management, pp. 244-260.

[146] Song, M., and W. M. P. Van der Aalst, 2008, "Towards Comprehensive Support for Organizational Mining," Decision Support Systems, **46**(1), pp. 300-317

[147] Abdelkafi, M., and L. Bouzguenda, 2010, "Discovering Organizational Perspective in Workflow Using Agent Approach: An Illustrative Case Study," In Proceedings of the 6th International Workshop on Enterprise & Organizational Modeling and Simulation, pp. 84-98.

[148] Zhao, W., and X. Zhao. 2014. "Process Mining from the Organizational Perspective." In *Foundations of Intelligent Systems*, 701-708. Springer.

[149] Kuhlmann, M., et al., 2003, "Role Mining-Revealing Business Roles for Security Administration Using Data Mining Technology," In Proceedings of the eighth ACM symposium on Access control models and technologies (pp. 179-186.

[150] Arias, M., et al., 2015, "A Framework for Recommending Resource Allocation Based on Process Mining," In International Conference on Business Process Management, pp. 458-470.

APPENDIX

Appendix A. Example Event Logs for Process Mining

The example file below shows the event logs in XML format. Each record tagged by "Event" corresponds to a design event detected from the given design documents.

```
<?xml version='1.0' encoding='UTF-8'?>
<Process><!--event logs-->
     <Event Doc_id="0" Frequency="0.4737" Id="0" Name="report progress">
          <Originator Organization="DCC FTS Group" R_strength="0.5000" R_type="Direct" Value="
          <Originator R_strength="0.3333" R_type="Direct">137</Originator>
          <Originator Organization="EE" R_strength="0.2000" R_type="Indirect" Value="8">138</(
          <Originator R_strength="0.2000" R_type="Indirect" Value="2">139</Originator>
          <Originator R_strength="0.2000" R_type="Indirect">141</Originator>
          <Data R_strength="0.3333" R_type="Direct">summary</Data>
          <Originator R_strength="0.3333" R_type="S_Indirect">1</Originator>
          <Start R_strength="1.0000" R_type="Indirect">2011-03-10 10:05:34</Start>
          <End R_strength="1.0000" R_type="Indirect">2011-03-10 10:05:34</End>
     </Event>
     <Event Doc_id="0" Frequency="0.4561" Id="1" Name="making group">
          <Originator Organization="DCC FTS Group" R_strength="0.2500" R_type="Indirect" Value
          <Start R_strength="0.3333" R_type="Direct">2011-03-10 10:05:34</Start>
          <Originator Organization="EE" R_strength="0.2000" R_type="Indirect" Value="8">138</(
          <Originator R_strength="0.2000" R_type="Indirect" Value="2">139</Originator>
          <Originator R_strength="0.3333" R_type="Direct">141</Originator>
          <End R_strength="0.3333" R_type="Indirect">2011-03-10 10:05:34</End>
     </Event>
     <Event Doc_id="1" Frequency="1.0" Id="2" Name="finalise layout">
          <Start R_strength="0.3333" R_type="Direct">2011-03-15 01:49:35</Start>
          <Originator R_strength="0.3333" R_type="Direct">0</Originator>
          <End R_strength="0.3333" R_type="Direct">2011-03-16 01:49:35</End>
     </Event>
     <Event Doc_id="2" Frequency="0.4286" Id="3" Name="email concept paper">
          <Start R_strength="1.0000" R_type="Indirect">2011-03-17 03:11:54</Start>
          <Originator R_strength="0.5000" R_type="Direct">0</Originator>
          <Data R_strength="0.3333" R_type="Direct">draft</Data>
          <End R_strength="1.0000" R_type="Indirect">2011-03-17 03:11:54</End>
          <Originator R_strength="0.3333" R_type="S_Indirect">7</Originator>
          <Originator R_strength="0.3333" R_type="S_Indirect">8</Originator>
          <Originator R_strength="0.3333" R_type="S_Indirect">9</Originator>
          <Originator R_strength="0.3333" R_type="S_Indirect">10</Originator>
          <Originator R_strength="0.3333" R_type="S_Indirect">11</Originator>
          <Originator R_strength="0.3333" R_type="S_Indirect">12</Originator>
          <Originator R_strength="0.3333" R_type="S_Indirect">13</Originator>
     </Event>
     <Event Doc_id="2" Frequency="0.4286" Id="4" Name="submit copy">
          <End R_strength="0.3333" R_type="Indirect">2011-03-17 03:11:54</End>
          <Originator R_strength="0.3333" R_type="Direct">137</Originator>
          <Start R_strength="0.3333" R_type="Direct">2011-03-17 03:11:54</Start>
          <Originator R_strength="0.3333" R_type="Direct">0</Originator>
     </Event>
```

Appendix B. Example Dot File for Visualizing Process Models Discovered by Bottom-Up Process Mining

The example dot files below show the data structure for visualizing the process model discovered by the bottom-up process mining approach. Each doc file corresponds to a workflow model in an abstraction or bottom layer of the hierarchical process model.

```
digraph workflow_at_0th_layer{        Bottom layer
        ranksep = 0.2;
        {
                node [shape=plaintext, fontsize=16];
                start ->"2011-03-10" ->"2011-04-09" ->"2011-05-09"…          }

        node [shape = box];
        -1 [label = "start", weight = 0.1];      Each node corresponds to a design event
        0 [label = "0_make group" weight = 1];
        1 [label = "1_report progress" weight = 1];
        2 [label = "2_finalise layout" weight = 1];
        3 [label = "3_submit copy" weight = 1];
        4 [label = "4_email concept paper" weight = 1];
        5 [label = "5_mean traffic" weight = 1];
        6 [label = "6_set experience test" weight = 1];
        7 [label = "7_consult proposal" weight = 1];
        8 [label = "8_discuss concept paper" weight = 1];
        9 [label = "9_concept paper" weight = 1];
        ……

        -1 -> 0 [weight = 1.0000];
        -1 -> 1 [weight = 1.0000];
        0 -> 2 [weight = 0.0500];    Each edge corresponds to a workflow
        1 -> 2 [weight = 0.0500];
        2 -> 3 [weight = 0.3833];
        2 -> 4 [weight = 0.3833];
        2 -> 5 [weight = 0.3833];
        2 -> 6 [weight = 0.3833];
        2 -> 7 [weight = 0.3833];
        3 -> 8 [weight = 0.0500];
        4 -> 8 [weight = 0.0500];
        5 -> 8 [weight = 0.0500];
        6 -> 8 [weight = 0.0500];
        7 -> 8 [weight = 0.0500];
        8 -> 9 [weight = 0.7167];
        ……
}
```

```
digraph workflow_at_15th_layer{   Abstraction layer
        ranksep = 0.2;
        {
                node [shape=plaintext, fontsize=12];
                start -> "2011-03-10" -> "2012-02-03" -> "2012-12-29" -> "2013-01-21" -> end;
        }

        node [shape = box];   Each node corresponds to a composite design task
        -1 [label = "start", weight = 0.1];
        0 [label = "0_20_email concept paper", shape = folder, style = filled, fillcolor = lightgrey,
          weight = 20];
        1 [label = "1_20_please concept paper", shape = folder, style = filled, fillcolor = lightgrey,
           weight = 20];
        2 [label = "2_2_attach group", shape = folder, style = filled, fillcolor = lightgrey, weight = 2];
        3 [label = "3_2_map design module", shape = folder, style = filled, fillcolor = lightgrey,
           weight = 2];
        4 [label = "4_3_work coordination issue", shape = folder, style = filled, fillcolor = lightgrey,
           weight = 3];
        5 [label = "5_16_submit application form", shape = folder, style = filled, fillcolor = lightgrey,
            weight = 16];
        6 [label = "6_2_do ia", shape = folder, style = filled, fillcolor = lightgrey, weight = 2];
        7 [label = "7_18_find irb exemption application", shape = folder, style = filled, fillcolor =
            lightgrey, weight = 18];
        8 [label = "8_2_optimize design", shape = folder, style = filled, fillcolor = lightgrey, weight = 2];
        9 [label = "9_24_make project description", shape = folder, style = filled, fillcolor = lightgrey,
            weight = 24];
        ……

        -1 -> 0 [weight = 1.0000];
        -1 -> 0 [weight = 1.0000];   Each edge corresponds to a workflow
        0 -> 1 [weight = 0.8833];
        1 -> 2 [weight = 0.1000];
        1 -> 3 [weight = 3.6333];
        2 -> 4 [weight = 0.0500];
        3 -> 4 [weight = 0.2000];
        4 -> 5 [weight = 0.9667];
        5 -> 6 [weight = 0.0500];
        6 -> 7 [weight = 1.4333];
        7 -> 8 [weight = 0.3000];
        7 -> 9 [weight = 0.1000];
        8 -> 9 [weight = 0.1000];
        9 -> 10 [weight = 0.0500];
        9 -> 13 [weight = 1.5333];
        10 -> 11 [weight = 0.0500];
        10 -> 12 [weight = 0.0500];
        ……
}
```

Appendix C. Example Dot File for Visualizing Process Model Discovered by Top-Down Process Mining

```
digraph TopDownModel{
        ranksep = 0.2;
        node [shape = box, fontsize=20];

        subgraph cluster712{
                color=blue;
                712.Start [label = "start"];
                712.0 [label = "report progress"];
                712.1 [label = "make group"];
                712.2 [label = "finalise layout"];
                712.3 [label = "email concept paper"];
                712.4 [label = "submit copy"];
                712.5 [label = "mean traffic"];
                712.6 [label = "set experience test"];
                712.7 [label = "consult proposal"];
                712.22 [label = "discuss concept paper"];
                712.9 [label = "concept paper"];
                712.10 [label = "provide summary"];
                ……
    }

    subgraph cluster707{
                color=blue;
                707.Start [label = "start"];
                707.322 [label = "model pattern"];
                707.323 [label = "explore traffic wave problem"];
                707.324 [label = "observe behavior"];
                707.325 [label = "explore applicability"];
                707.326 [label = "validate approach"];
                707.327 [label = "understand mechanism"];
                707.332 [label = "model pattern"];
                …….
                707.322 -> 707.379 [weight = 0.0500];
                707.323 -> 707.379 [weight = 0.0500];
                707.324 -> 707.379 [weight = 0.0500];
                707.325 -> 707.379 [weight = 0.0500];
                ……
    }
    ……

    cluster712 -> cluster707;
    cluster712 -> cluster711;
    cluster711 -> cluster650;
    cluster711 -> cluster710;
    ……
}
```

Each subgraph node corresponds to a module and a sub-process model

Each node corresponds to a design event within a module

Each edge corresponds to a workflow within a module

Each relation corresponds to a decomposition relation between two modules