A Planning Approach to Migrating Domain-specific Legacy Systems into Service Oriented Architecture

Ph.D Thesis

Zhuo Zhang

Software Technology Research Laboratory

De Montfort University

2012

To my husband, Shaochun Zhong

my daughter,   Jiayi Zhong

and my parents, Weihua Zhang and Guizhi Li

for their love and support.

Declaration

I declare that the work described in this thesis was originally carried out by me during the period of registration for the degree of Doctor of Philosophy at De Montfort University, U.K., from April 2007 to September 2012. It is submitted for the degree of Doctor of Philosophy at De Montfort University. Apart from the degree that this thesis is currently applying for, no other academic degree or award was applied for by me based on this work.

Acknowledgements

I want to thank some people who gave me more help during the planning and development of my thesis. The success of my thesis would be impossible if it was not for their encouragement and direction.

First, I want to show my great gratitude to my supervisor Prof. Hongji Yang, for his valuable direction, encouragement and suggestion during the course of my study. Especially, he taught me more on scientific research methods, which will benefit my future research work.

Second, many thanks must go to Prof. Dongdai Zhou, for his innovative and valuable contribution to the thesis, together with his excellent technical support in the working environment and encouragement in completing this thesis.

Third, many thanks will be given to my classmates at De Montfort University for their useful discussion and communication during the course of my study.

Finally, I want to express my gratitude to my parents, my husband and my daughter for their continuous support, encouragement, and love in the years of my study. All my work is dedicated to them.

Abstract

The planning work prior to implementing an SOA migration project is very important for its success. Up to now, most of this kind of work has been manual work. An SOA migration planning approach based on intelligent information processing methods is addressed to semi-automate the manual work. This thesis will investigate the principle research question: “How can we obtain SOA migration planning schemas (semi-) automatically instead of by traditional manual work in order to determine if legacy software systems should be migrated to SOA computation environment?”.

The controlled experiment research method has been adopted for directing research throughout the whole thesis. Data mining methods are used to analyse SOA migration source and migration targets. The mined information will be the supplementation of traditional analysis results. Text similarity measurement methods are used to measure the matching relationship between migration sources and migration targets. It implements the quantitative analysis of matching relationships instead of common qualitative analysis. Concretely, an association rule and sequence pattern mining algorithms are proposed to analyse legacy assets and domain logics for establishing a Service model and a Component model. These two algorithms can mine all motifs with any *min-support* number without assuming any ordering. It is better than the existing algorithms for establishing Service models and Component models in SOA migration situations. Two matching strategies based on keyword level and superficial semantic levels are described, which can calculate the degree of similarity between legacy components and domain services effectively. Two decision-making methods based on similarity matrix and hybrid information are investigated, which are for creating SOA migration planning schemas. Finally a simple evaluation method is depicted.

Two case studies on migrating e-learning legacy systems to SOA have been explored. The results show the proposed approach is encouraging and applicable. Therefore, the SOA migration planning schemas can be created semi-automatically instead of by traditional manual work by using data mining and text similarity measurement methods.

Table of Contents

Declaration ..... **I**

Acknowledgements ..... **II**

Abstract ..... **III**

Table of Contents ..... **IV**

List of Figures ..... **IX**

List of Tables ..... **XII**

List of Acronyms ..... **XIV**

Chapter 1 Introduction **1**

1.1 Problem Statement 1
1.2 Research Objectives and Research Methods 3
1.3 Research Questions and Hypotheses 4
1.3.1 Research Questions 4
1.3.2 Research Hypotheses 6
1.4 Original Contributions 6
1.5 Success Criteria 7
1.6 Organisation of Thesis 8

Chapter 2 Research Background and Related Work **10**

2.1 Legacy System Analysis 10
2.1.1 Software Reengineering 10
2.1.2 Legacy Software System Migration 12
2.1.3 Legacy Software System Understanding and Decomposition 14
2.1.4 Component Identification and Measurement 16
2.2 Domain Business Analysis 19
2.2.1 Domain Analysis 19
2.2.2 Business Process Modelling 22
2.3 Service Oriented Architecture (SOA) 23
2.3.1 Overview 23
2.3.2 SOA Reference Model 28
2.3.3 Service Identification and Composition 31
2.3.4 SOA Migration Strategies 32

2.3.5 SOA Migration Practices 35
2.4 Data Mining Techniques for SOA Migration Environment 39
2.4.1 Data Mining Techniques and Software Engineering Data 39
2.4.2 Application of Data Mining Method in Legacy Migration Situation 41
2.4.3 Analysis of Traditional Data Mining Algorithms 41
2.5 Text Retrieval Techniques for SOA Migration Environment 45
2.5.1 Vector Space Model (VSM) 46
2.5.2 Granularity of Text Similarity Calculation 47
2.6 Summary 48

Chapter 3 A Proposed SOA Migration Planning Approach 50

3.1 Key Factor Analysis 50
3.2 The Framework of SOA Migration Planning Approach 52
3.2.1 Analysis of Legacy Assets and Domain Requirements with Data Mining Techniques 55
3.2.2 Matching of Legacy Components and Domain Services with Text Retrieval Techniques 55
3.2.3 Decision-making of SOA Migration Planning Schemas 56
3.2.4 Evaluation of SOA Migration Planning Schemas 57
3.3 SOA Migration Toolkit --- SOAMT 58
3.4 Summary 59

Chapter 4 Domain Service Model Establishment with Data Mining Techniques 61

4.1 Service Model Establishment 61
4.1.1 The Process of Establishing a Service Model 61
4.1.2 Hierarchical Directed Acyclic Graph (HDAG) 64
4.1.3 The Representation of a Service Model 64
4.2 A Proposed Sequence Pattern Mining Algorithm for Service Composition 65
4.2.1 Concepts and Notations 65
4.2.2 Theorems and Algorithms 69
4.2.3 Examples 74
4.3 Performance Analysis on Proposed Sequence Pattern Mining Algorithm 79
4.3.1 Data Resource 80
4.3.2 Experiment Environment 80
4.3.3 Experiment Results 80
4.3.4 Conclusion 82
4.4 Supporting Tool: SEquence Pattern Miner (SEPAM) 82
4.5 Summary 85

Chapter 5 Legacy Component Model Establishment with Data Mining Techniques 86

5.1 Component Model Establishment 86

5.1.1 The Process of Establishing a Component Model ... 86
5.1.2 The Representation of a Component Model ... 88
5.2 A Proposed Association Rule Mining Algorithm for Determining Relationships of Legacy Components ... 91
5.2.1 Concepts and Notations ... 91
5.2.2 Theorems and Algorithms ... 94
5.2.3 Examples ... 99
5.3 Performance Analysis on the Proposed Association Rule Mining Algorithm ... 105
5.4 Supporting Tool: Association Rule Miner (ARM) ... 105
5.5 Summary ... 107

Chapter 6 Matching Strategies between Legacy Components and Domain Services with Text Similarity Measurement Techniques ... 109

6.1 Matching Strategies Based on Text Similarity Measurement Method ... 110
6.1.1 Keywords-based Level Matching Strategy ... 110
6.1.2 Superficial Semantic-based Level Matching Strategy ... 112
6.2 A Matching Algorithm Based on Keyword Level ... 113
6.2.1 The Weight Calculation Method of Nodes in a HDAG ... 113
6.2.2 Notations Definition ... 113
6.2.3 Algorithm SMA-Keyword ... 115
6.2.4 An Example on Algorithm SMA-Keyword ... 116
6.3 A Matching Algorithm based on Superficial Semantic Level ... 119
6.3.1 Analysis ... 119
6.3.2 Notation Definition ... 121
6.3.3 Algorithm SMA-Semantic ... 123
6.3.4 An Example on Algorithm SMA-Semantic ... 126
6.4 Supporting Tool ... 127
6.4.1 Matching Tool ... 127
6.4.2 Indexing Function ... 127
6.4.3 Similarity Calculation Function ... 128
6.5 Summary ... 129

Chapter 7 Creation and Evaluation of SOA Migration Planning Schemas ... 131

7.1 SOA Migration Planning Schemas ... 131
7.1.1 Analysis of SOA Migration Planning Schema ... 131
7.1.2 User's Direction ... 132
7.1.3 Methods on Creating SOA Migration Planning Schemas ... 133
7.1.4 Architecture of Creation and Evaluation of SOA Migration Planning Schemas ... 134

7.2 A Decision-making Method based on Similarity Matrix ... 135
7.2.1 Symbol Definition ... 136
7.2.2 A Proposed Method ... 137
7.2.3 An Example ... 140
7.3 An Optimal Decision-making Method Based on Hybrid Information ... 142
7.3.1 Analysis ... 142
7.3.2 A Proposed Method ... 143
7.3.3 An Example of the Proposed Method ... 143
7.4 Evaluation of SOA Migration Planning Schemas ... 148
7.4.1 Implementation Cost Estimation on SOA Migration Planning Schemas ... 149
7.4.2 Performance Evaluation of SOA Migration Planning Schemas ... 150
7.5 Supporting Tools ... 152
7.5.1 Decision-making Tool ... 152
7.5.2 Evaluation Tool ... 153
7.6 Summary ... 154

Chapter 8 Case Studies ... 156

8.1 Case Study I: Improving Service Model and Component Model with SEPAM and ARM ... 157
8.1.1 Overview ... 157
8.1.2 A Service Composition Method for MELS by Using SEPAM ... 157
8.1.3 Data Pre-process Phase ... 159
8.1.4 Pattern Discovery Phase ... 163
8.1.5 Pattern Analysis Phase ... 164
8.1.6 Evaluation ... 164
8.2 Case Study II: Creating SOA Migration Planning Schemas with SOAMT ... 165
8.2.1 Overview ... 165
8.2.2 Legacy Assets ... 166
8.2.3 A New Service Oriented Architecture ... 170
8.2.4 Creating SOA Migration Planning Schemas ... 174
8.2.5 Evaluation ... 178
8.3 Conclusion ... 178
8.4 Summary ... 179

Chapter 9 Conclusions ... 181

9.1 Summary of Thesis ... 181
9.2 Revisiting Original Contributions ... 182
9.3 Evaluation ... 183
9.3.1 Answering Research Questions ... 183

9.3.2 Revisiting Research Hypotheses 186
9.3.3 Revisiting the Measure of Success Criteria 187
9.4 Limitations 188
9.5 Future Work 189

References 191

Appendix A Templates and Examples of Related XML Files 206

Appendix B List of Publications 223

List of Figures

Figure 2-1. *Stages of Domain Analysis [83].* 20

Figure 2-2. *Information Sources for Domain Analysis [83].* 21

Figure 2-3. *Domain Analysis Support Software Development [83].* 22

Figure 2-4. *Correlations of Services, Activities and Business Process [122].* 23

Figure 2-5. *The Relationship among Service Providers, Consumers and Brokers [1].* 25

Figure 2-6. *Service Composition [71].* 26

Figure 2-7. *A Basic Service Oriented Architecture* 27

Figure 2-8. *IBM SOA Reference Model [15].* 29

Figure 2-9. *IBM SOA Solution Stack [15]* 29

Figure 2-10. *An Example of Service Definition Hierarchy [102].* 31

Figure 2-11. *The Apriori Algorithm.* 42

Figure 2-12. *The Apriori-generate Algorithm.* 42

Figure 2-13. *The AprioriAll Algorithm.* 44

Figure 2-14. *The AprioriAll-generate Algorithm.* 44

Figure 2-15. *The AprioriSome Algorithm [110].* 45

Figure 3-1. *An SOA Migration Planning Approach.* 54

Figure 3-2. *A Main Interface of SOAMT.* 59

Figure 4-1. *The Process of Establishing a Service Model.* 62

Figure 4-2. *An Example of Hierarchical Directed Acyclic Graph (HDAG).* 64

Figure 4-3. *A Node in a Domain HDAG.* 65

Figure 4-4. *An Example of Service Model Represented by HDAG.* 65

Figure 4-5. *A Transaction Database.* 67

Figure 4-6. *The Motifs Mined by the Proposed Sequence Mining Algorithm.* 79

Figure 4-7. *The Memory Performance for Finding Frequent Patterns.* 81

Figure 4-8. *The Runtime Performance for Finding Frequent Patterns.* 81

Figure 4-9. The Memory Performance for Finding Motifs. ... 81

Figure 4-10. The Runtime Performance for Finding Motifs. ... 82

Figure 4-11. The Main Interface of SEPAM. ... 83

Figure 4-12. Frequent Patterns with Min-support=20% and length=2 ... 84

Figure 4-13. The Information on Supporting Group. ... 84

Figure 5-1. The Process of Establishing a Component Model. ... 88

Figure 5-2. A Node in a Legacy HDAG. ... 90

Figure 5-3. An Example of Component Model Represented by HDAG. ... 90

Figure 5-4. The Representation of a Service Model and a Component Model. ... 90

Figure 5-5. The Application Example of Components' Association Relations ... 91

Figure 5-6. The Main Interface of ARM. ... 106

Figure 5-7. Frequent Patterns with Min-support=10% and size=3. ... 107

Figure 5-8. The Information on Supporting Group. ... 107

Figure 6-1. The Corresponding Relationships between SOA Migration and Text Retrieval ... 111

Figure 6-2. An Example of Component model and Service Model. ... 117

Figure 6-3. Similarity Matrix (Ma_Keyword). ... 117

Figure 6-4. An Example of the Combination of Legacy Components. ... 118

Figure 6-5. The Similar Relationship between a Node and a Document. ... 121

Figure 6-6. The Interface of the Indexing Function. ... 128

Figure 6-7. The Interface of Similarity Calculation Function. ... 129

Figure 7-1. The Architecture of Creation and Evaluation of SOA Migration Planning Schemas. ... 135

Figure 7-2. An Example for the Dimensionality Reduction (k=3) for Similarity Matrix Ma. ... 136

Figure 7-3. The Interface of Decision-making Tool. ... 153

Figure 7-4. An Evaluation Report on SOA Migration Planning Schemas. ... 154

Figure 8-1. The Framework of a Service Composition Method. ... 158

Figure 8-2. Architecture of an E-learning System. ... 159

Figure 8-3. A Screenshot on Object Relationship Mapping (ORM) Configuration File. ... 168

Figure 8-4. A Screenshot of Legacy Asset on Component Relationship. ... 169

Figure 8-5. *A Screenshot of a Component Model.* 170

Figure 8-6. *Migration Target -- The New Service Oriented Architecture.* 172

Figure 8-7. *A Screenshot of the Service Model.* 173

Figure 8-8. *The Flow Chart of Creating SOA Migration Planning Schemas based on SOAMT.* 175

Figure 8-9. *The Key Part of SOA Migration Planning Schemas in the Experiment.* 176

Figure 8-10. *A Screenshot of Service Implementation.* 177

List of Tables

Table 4-1. *A Sequence Database U.* *67*

Table 4-2. *Patterns with Length 1 and Min-support 2 over U.* *75*

Table 4-3. *Patterns with Length 2 and Min-support 2 over U.* *75*

Table 4-4. *Patterns with Length 3 and Min-support 2 over U.* *76*

Table 4-5. *Patterns with Length 4 and Min-support 2 over U.* *76*

Table 4-6. *Patterns with Min-support 2 over U.* *77*

Table 4-7. *Patterns with Min-support 3 over U.* *78*

Table 4-8. *Patterns with Min-support 4 over U.* *78*

Table 4-9. *Patterns with Min-support 5 over U.* *78*

Table 4-10. *maxM(U, 2).* *79*

Table 4-11. *maxM(U,3).* *79*

Table 4-12. *maxM(U,4).* *79*

Table 4-13. *maxM(U,5).* *79*

Table 5-1. *Transaction Database U.* *93*

Table 5-2. *Patterns with Size 1 and Min-support 1 over U* *99*

Table 5-3. *Motifs with Min-support 1 over U* *100*

Table 5-4. *Patterns with Size 2 and Min-support 2 over U* *100*

Table 5-5. *Patterns with Size 3 and Min-support 2 over U* *101*

Table 5-6. *Patterns with Size 4 and Min-support 2 over U.* *101*

Table 5-7. *Patterns with Min-support 2 over U.* *102*

Table 5-8. *Patterns with Min-support 3 over U.* *103*

Table 5-9 *Patterns with Min-support 4 over U.* *103*

Table 5-10. *Patterns with Min-support 5 over U.* *103*

Table 5-11. *Motifs maxM(U,2).* *104*

Table 5-12. *Motifs maxM(U,3.)* *104*

Table 5-13. *Motifs maxM(U,4).* *104*

Table 5-14. *Motifs maxM(U,5).* *104*

Table 5-15. *Association Rules from Motifs maxM(U,2).* *104*

Table 6-1. *The Matching Relationships Based on Keyword Level*.* *118*

Table 6-2. *The Matching Relationships Based on Keyword and Semantic level*.* *126*

Table 6-3. *The SimItems between S_{31} and its Matched Objects.* *127*

Table 6-4. *The Format of Indexing Files.* *129*

Table 7-1. *The Matching Relationship Table of Figure7-2.* *136*

Table 7-2. *The Matching Relationships Based on Keyword and Semantic level*.* *138*

Table 7-3. *The Mapping Relation between the Implementation Means and the User’s Direction.* ... *139*

Table 7-4. *An Example on Keyword Level.* *140*

Table 7-5. *An Example on Semantic Level.* *141*

Table 7-6. *CC-method.* *145*

Table 7-7. *CCA-method.* *145*

Table 7-8. *CCAQ of the Example Shown in Table 7-5.* *146*

Table 7-9. *The Re-ranked Matching Relationships of Target $S_{31.}$* *147*

Table 7-10. *The Final SOA Migration Planning Schemas for Target $S_{31.}$* *147*

Table 7-11. *The Final SOA Migration Planning Schemas of Figure 6-2 (Performance-First).* *148*

Table 7-12. *An SOAMS-EX and An SOAMS-HI.* *150*

Table 7-13. *An Evaluation Report.* *152*

Table 8-1. *The Collected Original Data.* *162*

Table 8-2. *Sequence Database.* *163*

Table 8-3. *Performance Comparison.* *178*

List of Acronyms

BPC	*Business Process Choreography*
BPM	*Business Process Modelling*
CBD	*Component-Based Development*
CIM	*Computation-Independent Model*
CORBA	*Common Object Request Broker Architecture*
DCOM	*Distributed Component Object Model*
DM	*Data Mining*
DSKs	*Domain Specific Kits*
EA	*Enterprise Architecture*
ERD	*Entity-Relationship Diagrams*
ESA	*Enterprise Services Architecture*
ESB	*Enterprise Service Bus*
FODA	*Feature-Oriented Domain Analysis*
FORM	*Feature-Oriented Reuse Method*
HDAG	*Hierarchical Directed Acyclic Graph*
MDA	*Model Driven Architecture*
MELS	*Migrating E-learning Legacy systems to SOA*

SEPAM	*SEquence Pattern Miner*
SMART	*Service-oriented MigrAtion and Reuse Technique*
SOA	*Service Oriented Architecture*
SOAMA	*SOA Migration Planning Approach*
SOAMS	*SOA Migration planning Schemas*
SOAMT	*SOA Migration Toolkit*
SWA	*Salvaging & Wrapping Approach*
TSM	*Text Similarity Measurement*
UDDI	*Universal Description Discovery and Integration*
UML	*Unified Modelling Language*
VSM	*Vector Space Model*
XML	*eXtensible Markup Language*

Chapter 1 Introduction

Objectives

- To state the problems on migrating legacy assets to service oriented architecture.
- To present the research objectives and choose the research method.
- To determine research questions and develop research hypotheses.
- To explain original contributions and determine the success criteria.
- To describe the organisation of the thesis.

1.1 Problem Statement

With the rapid development of technology, the software field has been becoming a much more complex field. Some of the software companies bear more pressures from marketing and competition. At the same time, it is known that more changes have been happening in the aspects of business requirements and objectives. In modern society, current satisfied business solutions will soon become legacy systems. At present, some kinds of legacy systems exist. It is expensive and difficult to modify them to meet the user’s demands. Therefore, it may lead to a loss of business opportunities [79]. However, some legacy systems cannot be discarded directly since they contain a number of business experiences and valuable business processes. In this case, they should be reused wholly or partly. It is necessary to find a new approach for solving this situation. The new approach should be able to design and implement good quality systems, which are good for reusing maximally the valuable components and systems’ maintenance and application.

Recently, Service-Oriented Architecture (SOA) has been more and more popular. A service-oriented architecture refers to a collection of services that own a common communication model and well-described interfaces [2]. SOA can reuse legacy components to achieve the constantly changing needs of business requirements. The research on migrating existing legacy systems into SOA environment (in short, SOA migration) is triggered. The migration of a software system refers to the movement to a different development platform, architecture, system hardware and software. Some problems on system application, such as complex degree, non-reused and redundant code, bad interfaces and some business logic problems, can be solved by doing SOA migration. SOA migration has been completed in some fields. At present, the earlier practitioners are reviewing their experiences and the degree of satisfaction runs high. According to a recent report by Forrester, nearly 70% of SOA users agree to increasing their use.

Among the existing literatures on SOA migration, most of them pay attention to questions like “how to do SOA migration?”. There is less attention to the question “is it worth triggering an SOA migration project?”. The latter question can be answered by planning schemas of an SOA migration project. It is known, an SOA migration project is risky. The prediction is important before starting an SOA migration project. If there are not enough predictions on the project’s function, quality and so on, the SOA migration projects may fall into failure. Moreover, it may cause a catastrophic loss of money, time and resources. SOA migration planning schemas are preconditions for doing predictions in order to avoid undesirable results. It is necessary to create SOA migration planning schemas before starting an SOA migration project. Up to now, most of this kind of work is manual work. Therefore, semi-automatically creating an approach on SOA migration planning schemas should be investigated.

In an SOA migration situation, the function or sub-function of legacy systems can be wrapped as services. Not only the explicit but also the implicit legacy assets should be taken into account. Thus, how to reuse legacy assets maximally is one of the most important questions, such as, “how to understand and decompose legacy assets for reusing them in the new domain services”, “how to calculate the matching relationships between domain services and legacy components”, “how to create migration planning schemas”, etc. Another most important question is how to mine the hidden information from the application data of legacy systems, such as, how to discover the hidden business process and rules from the user's usage behaviour log files and user' feedbacks, etc.

1.2 Research Objectives and Research Methods

The main goal of this study is to present an SOA migration planning approach to plan and deploy an SOA migration project, in which questions like "which legacy component(s) can be reused", "how to reuse it or them in the new system" and "how about the cost and performance of this deployment", etc. can be solved. Concretely, the research objectives presented in this study are as follows.

- Propose an SOA migration planning approach.
- Apply data mining methods to analyse legacy assets and domain logics for improving Component/Service model.
- Apply text similarity measurement methods to establish the matching strategies between legacy components and domain services.
- Apply heuristic decision making methods to create SOA migration planning schemas.
- Evaluate the proposed approach by two case studies.

The empirical methodology is playing a key role in the software engineering field [39]. According to [121], there are five research methods available for the software engineering field. They are controlled experiment, case studies, survey research, ethnography and action research.

A controlled experiment is a study on a hypothesis that can be tested. In a controlled experiment, some independent variables are created to assess their roles on some dependent variables. The question "how many variables should be considered in the study and how to measure them?" will be solved during the process of the experimental establishment. The experimental design should be directed by the research question and hypothesis.

In this thesis, the controlled experiment research method has been adopted for directing research throughout the whole study. A research question and a research hypothesis are established in Chapter 1. A theory involving the research question and hypothesis is built in Chapter 3. The experimental steps guided by this theory have been detailed in Chapter 4, 5, 6, 7 and 8. The concrete research methods in the experimental steps are as follows.

- Mathematical proof method.

  It uses formal proofs to reason about the validity of a hypothesis given some evidence. The proposed theorems on association rule mining algorithm and sequence pattern mining algorithm are validated by this method.
- Qualitative analysis method.

  Is for the collection and analysis of qualitative data. The qualitative data refers to the data that is not in numeric format such as interview recordings or transcripts, questionnaires, etc. In this study, the qualitative analysis method has been used for the key factors analysis, benchmark determination, indicator selection of component quality, etc.
- Quantitative analysis method.

  Is for the collection and analysis of quantitative data, which refers to the data that is in numeric format. The quantitative analysis method has been used for the identified key factors measurement, weight assignment scheme, similarity calculation between legacy components and domain services, decision-making methods and so on. Normally, a qualitative method is the application base of a quantitative method.

1.3 Research Questions and Hypotheses

1.3.1 Research Questions

A research question is a statement of the research goal. Research questions should state what the study will explore. The principal research question in this study is:

How to obtain SOA migration planning schemas
(semi-)automatically instead of by traditional manual work
in order to determine if legacy software systems should be

migrated to an SOA computation environment?

For answering the principal question, a collection of research questions is described as follows.

RQ1: Why SOA migration planning schemas are needed in SOA migration projects?

RQ2: What is a proposed SOA migration planning approach?

- What are the key factors and their relationships in a proposed SOA migration planning approach?
- What kinds of legacy software system can be processed by the proposed approach?
- What are the final returned results by the proposed approach?

RQ3: How to analyse legacy systems and domain logics?

- What techniques will be used to analyse legacy assets and domain requirements from the aspect of application data of legacy systems?
- What are the analysis results?

RQ4: How to measure the matching relationships between legacy components and domain targets?

- How to represent legacy components and domain targets?
- What techniques can be adopted to calculate the matching degrees between legacy components and domain targets?
- What are the matching strategies between legacy components and domain targets?

RQ5: How to create SOA migration planning schemas?

- What is a SOA migration planning schema?
- How many methods can be used to create SOA migration planning schemas?
- How to evaluate SOA migration planning schemas?

RQ6: How to evaluate the proposed approach?

1.3.2 Research Hypotheses

After establishing these research questions, a series of research hypotheses based on them are developed. The underlying hypothesis of this thesis is:

Data mining techniques and text similarity measurement methods can be used to create migration planning schemas in SOA migration projects.

This hypothesis can be validated through an SOA migration planning approach, which mainly consists of the preparation stage, analysis stage (including domain analysis, legacy analysis and data mining method), matching stage, decision-making stage and evaluation stage. A set of hypotheses is derived from the underlying one:

RH1: Data mining techniques can be used to analyse legacy assets and domain requirements from the application data of legacy systems for perfecting the quality of Component model and Service model. This proposition can be tested by developing concrete data mining algorithms and applying them to the application data of legacy systems.

RH2: Text similarity measurement methods can be utilised to calculate matching degrees between legacy components and domain targets. This proposition can be tested by establishing the corresponding relationships between the representation of a document and the representation of a component/service.

RH3: User’s direction is necessary during the process of creating SOA migration planning schemas. Moreover, the solution is obtained by iteration. This proposition can be tested by developing a small SOA migration prototype system.

RH4: Not only functional factors but also non-functional factors should be considered during the process of creating SOA migration planning schemas. This proposition can be tested by comparing the quality of SOA migration planning schemas created by these two methods individually.

1.4 Original Contributions

A general SOA migration planning approach is proposed, which can offer planning and implementing directions for decision-makers in both sides of the application domain and information technology development field. Some domain logic analysis methods, software reengineering methods and intelligent information process methods (such as knowledge representation, data mining, information retrieval, etc.) are deployed to determine migration planning schemas of SOA migration projects. The following are original contributions.

C1. Data mining algorithms for service identification and composition as well as for existing legacy assets comprehension and decomposition are developed, which can be of great benefit to the establishment of a Service model and a Component model.

C2. Matching strategies based on text similarity measurement methods (from keyword level to superficial semantic level) between domain targets and legacy components are investigated, which are the foundation for creating SOA migration planning schemas.

C3. Decision-making methods for creating SOA migration planning schemas are explored. Both functional factors and non-functional factors are taken into account in the decision making stage.

1.5 Success Criteria

The overall success measurement of an SOA migration planning approach is how well it supports an SOA migration project. The success criteria for the research described in this thesis are as follows.

1. What kinds of legacy system can be processed by the proposed approach?
2. What type of data mining techniques can be used to analyse legacy assets and domain logics?
3. How can text similarity measurement methods be applied to establish the matching strategies between legacy components and domain targets?
4. How is the performance of this proposed approach?
5. How about the implementation of the proposed algorithms, strategies and methods?

E.g., is it possible to develop a toolkit according to the proposed approach?

1.6 Organisation of Thesis

In Chapter 1, the research objectives, methods, questions and hypotheses are introduced. Furthermore, the original contributions and success criteria are presented.

In Chapter2, the research background and related work are presented. First, some techniques and approaches on legacy system analysis are introduced. Second, domain analysis and business modelling are described. Third, Service Oriented Architecture (SOA) and SOA migration situation are shown. Fourth, data mining techniques for SOA migration environment are presented. Fifth, text retrieval techniques for SOA migration environment are presented.

In Chapter 3, an SOA migration planning approach has been proposed. This approach includes 5 stages, which are the preparation stage, analysis stage, matching stage, decision-making stage and evaluation stage. The main work on each stage is described in this Chapter. Moreover, the related tools are developed for supporting this proposed approach.

In Chapter 4, a sequence pattern mining algorithm for service/component identification and composition is proposed. It can perfect the quality of a Service/Component model. A SEquence PAttern Miner (SEPAM) has been designed and implemented.

In Chapter 5, an association rule mining algorithm for determining the association relationships of components/services is proposed. It can perfect the quality of a Component/Service model. An Association Rule Miner (ARM) has been designed and implemented.

In Chapter 6, two matching strategies based on text similarity measurement methods are proposed. Concretely, a keyword-based matching algorithm and a superficial semantic-based matching algorithm are presented, which can be used to calculate the similarity degrees between legacy components and domain targets. A matching tool has been designed and implemented.

In Chapter 7, two methods for creating SOA migration planning schemas have been investigated. One is a similarity matrix-based method. Another one is a method based on hybrid information. Moreover, a simple evaluation method is proposed. A decision-making and an evaluation tool have been designed and implemented.

In Chapter 8, two case studies for evaluating the proposed algorithms, strategies and methods are presented. In case study 1, the proposed data mining algorithms have been used to improve the Service model and Component model by using SEPAM and ARM. The performance is promising. In case study 2, the proposed matching strategies and decision-making methods are used to migrate components in an education administration legacy system for primary and secondary schools to SOA for completing the work on student’s management. The performance is acceptable.

In Chapter 9, the work that has been done in this study is concluded. It includes a summary of the thesis, original contributions, evaluation, limitations and future work.

Appendix A is templates and examples of related *XML* files.

Appendix B is the author’s publications during the study for PhD.

Chapter 2 Research Background and Related Work

Objectives

- To introduce legacy software system analysis.
- To introduce domain business analysis.
- To conclude SOA migration strategies and approaches.
- To present data mining techniques for SOA migration environment.
- To present text retrieval techniques for SOA migration environment.

2.1 Legacy System Analysis

With the development of information technology, the proportion of legacy software systems is becoming greater. Legacy software systems [112] refer to old application programs, which are difficult to integrate, modify or extend with the changes of business requirements. They reduce the software organisation’s competitive abilities in the dynamic business world. However, they cannot be discarded simply since they hold some important business logics and related data [72, 73]. Sometimes, legacy software systems are worth reusing wholly or partly, such as a legacy system which can be reused with other business functions to get better performance [112] and some business reasons drive a legacy system to be integrated into new solutions [65], etc. In this case, it is necessary to do the work on legacy software reengineering.

2.1.1 Software Reengineering

The existing software systems have to face work on their maintenance, modernisation and replacement. Software reengineering refers to changing the old software systems to meet the new needs on the aspects of maintenance, application and replacement. The objectives of software reengineering are to facilitate the maintenance of existing software products, to recover and extract reusable components from the legacy system, to improve reliability, to establish a base for future software evolution or migration, and so on.

The software reengineering process is composed of the reverse engineering stage, the function restructuring stage and the forward engineering stage [60].

- In the reverse engineering stage, the main task is to recover the old requirements, design, structure, and so on, from the existing system. The concrete method is to analyse the existing legacy assets, such as, source codes, user usage behaviour log, requirement specifications and some kinds of documentation. Meanwhile, some proved information and rules on business logic are also retrieved. The quality of reverse engineering will affect the performance of the reengineering system.
- In the function restructuring stage, the main work is to rebuild new functions according to new requirements and delete those unnecessary functions.
- In the forward engineering stage, the main work is to redesign and implement the new target system. It is actually a forward movement according to the standard software development process.

The software reengineering approaches include a “big bang” approach, incremental approach and so on [60].

- “Big bang” approach changes the old system to the target system at one time. For large systems, this approach may cost too many resources and time before the target system is produced.
- Incremental approach (also known as “phase-out”) means that the legacy system is divided into sections. These sections are reengineered and integrated incrementally to meet the new requirements. In practice, this approach has been applied widely.

The ways of software reengineering include software system extension, transformation, integration, migration, etc. Nowadays, many software organisations want to migrate legacy systems to new environments that are good for the maintenance and deployment of new domain requirements. Thus, it is important and valuable to investigate the question of legacy software system migration.

2.1.2 Legacy Software System Migration

Up-to-date software systems can change with the changing of domain business, the customer requirements, hardware and environment. However, in practice, many software systems cannot satisfy the ever-changing environments and requirements. They become the legacy software systems.

Legacy software system migration means that the legacy software system is migrated into a new operating system, software architecture, hardware equipment or development platform. Legacy software system migration involves some research fields [19] and it is not simple work. Usually, it faces a number of challenges [65]:

- Documentations: most legacy systems are facing problems on documentation, such as, uncompleted or outdated requirements and specifications, undocumented pre- and post-implementation changes, etc.
- Skills and schedule: developers have little skills in business processes and technologies that the legacy system is based on. It is difficult to identify what parts can be cost-effectively reused. In addition, schedule overrun is also a challenge to be faced.
- Cost and feasibility: the ratio among cost, benefit and the technical feasibility for migrating the applications to new solutions should be assessed first. After making sure there are no problems in finance and technology, the project can be started.
- Management: management becomes a problem when different development groups are responsible for different applications, technologies or functional areas. Tools and strategies for maintaining effective cooperation are necessary factors for a legacy system migration project.

There are five stages in a generic legacy software system migration process [67]:

- Justification stage

It is the planning stage before triggering some application project.

- Understanding legacy system stages

  It is the basis for ensuring the success of a legacy migration project.

- Developing a target system stage

  It is a key phase of some migration projects. The selected target environment should be good enough for the application requirements of domain targets.

- Testing stage

  It is a necessary process during the legacy system migration.

- Cutting over stage

  It refers to the movement from the old environment to a new environment.

For reducing the migration risk, more attention should be paid to the justification stage of legacy software system migration. The abundant planning and predicable work [56] is the foundation of SOA migration success. Otherwise, it may cause a great deal of loss in money and time.

Some different approaches for migrating legacy systems into new environment exist. Usually, typical solutions of legacy software system migration [7] include wrapping the sub-system(s) or function(s) as component(s) of the target system, modifying legacy assets to adapt to the new requirements, redeveloping a replacement system, etc. These solutions can be adopted individually or by their combination. The choice of migration approaches [6, 80] will base on the domain value of the legacy system and software quality [50].

For evaluating the legacy system's quality, legacy system analysis must be done. Legacy system analysis can be implemented through some reverse engineering methods. The main work includes identifying the legacy components and their relationships with each other, and establishing abstraction representations for the components in a higher level [35].

The quality of legacy system analysis is related to the development approach of the legacy system. If the legacy programs have been implemented by a Component Based

Development (CBD) approach, it will be easier for legacy software system analysis.

Component-based development [23] is a reuse-based software engineering approach. It builds loosely coupled and independent software components into systems. CBD owns advantages such as flexibility, reusability, scalability, high quality, cost savings and faster time-to-market [46]. Clearly, it is necessary to replace traditional procedural programming approaches by CBD [23]. The advantages of CBD can be of great benefit for the software product and its sponsors for a long time.

One of the main purposes of legacy software system migration is to reuse the existing legacy assets maximally. Sometimes, migrating legacy systems to service-oriented architecture (for short, SOA migration) is a good choice. SOA provides a mechanism for reusing legacy assets [150] despite their development environment. Reusability is a key advantage of SOA. The legacy system's value is to reuse their functionality or subsets by the ways of service. Understanding how a legacy system works and how it is used is essential for reuse. Therefore, the starting point should be to find what exactly exists in a legacy software system. Two important things have to be done before identifying legacy components and measuring their quality, namely, legacy system understanding and decomposition.

2.1.3 Legacy Software System Understanding and Decomposition

Legacy system understanding is necessary for moving an old system to a new environment. If the quality of the legacy system understanding is poor, it may lead to a failed migration project.

A verified and effective approach for legacy system understanding has been described in [72, 124]. Usually, it can be done by reverse engineering based on existing assets. For doing so, the supporting artefacts include specifications of software, documents of design and requirements or code comments; user guides, screenshots, test samples, release notes, defect reports and improvement requests; user's needs, priorities and complaints related to the legacy system, as well as ways to improve it [64].

There are some tools available in the legacy system understanding stage [64], such as, a reverse-engineer can use IBM Rational XDE to understand application code and

RDBMS functionality, UML models can be generated automatically by using IBM Rational XDE, etc. These tools can benefit legacy system understanding. Although the existing tools can help legacy system's understanding, it is still difficult to complete understanding of a legacy system's structure automatically [131].

Another work of legacy system analysis for the purpose of migration is to cut legacy assets to match the target's requirements maximally. Legacy systems decomposition is a preliminary step for legacy migration. The quality of legacy system migration is up to its decomposability. If the decomposability of the system is less, the migration should be more difficult [89].

Normally, a software system is composed of database, interface and application components. According to the components' separation and identification methods, three kinds of legacy systems exist, namely, a decomposable legacy system, a semi-decomposable legacy system or a non-decomposable legacy system. In non-decomposable legacy systems, the three types of component are inseparable. In semi-decomposable legacy systems, interface components are isolated from domain logic and database components. In decomposable legacy systems, the three types of component are all separable [89].

Each program of legacy systems may consist of three types of components. The purpose of decomposing a legacy system is to identify and re-organise the different level's components for reuse. Some decomposition approaches and techniques should be investigated. According to [41, 55], there are mainly three types of legacy program decomposition approaches.

- The procedural approach.

  In this approach, a program is regarded as a directed graph. In this graph, nodes are used to represent decisions and edges are used to represent branches. The complex graphs are divided into sub-graphs according to the rule of minimum interconnections. In addition, program slicing techniques are exploited for program restructuring [52] or for identifying re-usable functions [38].
- The functional approach.

  In this approach, a program is decomposed in terms of concrete functions. The program is hierarchical, which consists of high-level control nodes and low-level basic nodes. The matching relationships between each business rule and some high-level node(s) have been established [3, 53, 86].

- The data type approach.
  In this approach, a program is regarded as a collection of objects. Some modules are established with the collection of operations on the known entity or data type.

In addition, some approaches for the identification of object-oriented architectures from legacy systems [22, 43, 104] are also contributed. They can benefit the legacy system decomposition.

2.1.4 Component Identification and Measurement

A component refers to a software or hardware unit. A **s**oftware component refers to a web service, a module or a software package that contains some related functions or data. Hardware components are devices with their software drivers [30].

Reusability and substitutable ability are the key characteristics of a good-quality component. The developed software components in different systems should have high reusability. If a new component satisfies the needs of the given component, the new one may replace the old one without breaking the system where the component runs. Components communicate with each other via interfaces, which will be regarded as the component’s signature [139]. The user can reuse it directly even he/she knows nothing about how to implement it. Components can be regarded as the starting platform for service-oriented application.

Legacy component identification is necessary for implementing legacy assets reuse. Components will be identified from some kind of legacy codes, which include procedural codes, object-oriented codes, component-based codes, etc. [8, 27, 42]. Some related reverse engineering approaches will be used to do more analysis.

The component granularities include the fine-grained, medium-grained and coarse-grained [9]. The process of legacy component identification is divided into three phases: object identification from function-oriented codes [42], IT component identification from object-oriented codes [8] and business component identification or composition (aligning IT with business process) from component-based codes [115, 153]. A business component refers to an IT component that implements some services in a business domain. Some work on reusing source programs to establish services have already been explored [4, 54, 61]. A bottom-up method is usually adopted.

A component's reusability means the degree of reuse of this component. If the reusability of a component is lower, software engineers have to use more effort to understand it before it can be reused. Thus, it is necessary to measure legacy components' reusabilities [59,101, 85] for reusing them maximally.

According to ISO 9126, usually coupling degree, cohesion degree and adaptability measurements can be chosen as the key factors of component reusability measurement. Coupling [126] and cohesion [140] are two important attributes related to decompose a system into smaller subsystems. Coupling and cohesion always occur together. Low coupling and high cohesion are often co-occurred and they are desirable characteristics of a component. Adaptability refers to a system's ability of adapting to changed circumstances [140]. High adaptability will be of great benefit to the component's deployment in the new target system. Therefore, the metrics on the coupling degree, cohesion degree and adaptability of migrated legacy software components will be taken into account when the SOA migration planning schemas are created. The existing measure methods on these indicators are as follows.

- Coupling degree.

  It refers to the dependable degree among modules. A lot of work has been done on coupling measurement, such as, measures on structural coupling [84,117, 118, 144], logical and evolutionary coupling [51, 136] and dynamic coupling [34]; coupling measures based on information entropy approach [33] and information retrieval approach [18, 29]; coupling measures for knowledge-based systems [123], aspect-oriented systems [77] and others [45].

  Low coupling means that the interactions between any two modules are through a stable and simple interface and there are no relationships with the implementation of other modules. This kind of component can always provide high readability and maintainability. The disadvantage of high coupled systems might be that they are harder to reuse. Clearly, the lower coupling is preferred for a SOA migration situation.

- Cohesion degree.

  It refers to a measure of how well the lines of source code within a module work together. Methods of measuring cohesion include qualitative measures and quantitative measures. Usually, the qualitative measure is expressed as “high cohesion” or “low cohesion”; the quantitative measure is to calculate a numerical value through analysing the contents of the programs. The concrete approaches for cohesion metric in object-oriented system have been addressed in [93, 117, 118]. The advantages of high cohesion of modules/components include robustness, reliability, reusability and understandability. The disadvantages of low cohesion include that they are hard to test, to maintain, to understand and to reuse. Clearly, the higher cohesion of a single module/component is preferred for a SOA migration situation.

- Adaptability.

  Components generally should be modified to some extent to adapt to a new application environment. An adaptable component should be able to cope with both functional and non-functional changes. The adaptability of a component is not only influenced by internal factors, but also by the adaptability of the architecture. In [10, 68, 141], some adaptability evaluation methods (the metric for making quantitative or qualitative measurement and analysis.) have been concluded to support decision making for choosing the best targets among candidate components.

An SOA migration solution will always involve reusing high-quality legacy components through identifying them as services, operations, business processes and business rules. Concretely, existing components are factored into sets of discrete services that represent a group of related operations. Normally, a bottom-up analysis approach is adopted in legacy system analysis. Moreover, business processes and rules are abstracted from them into a separate Business Process Model (BPM), managed by a business choreography model [102]. Business modelling plays an important role in legacy system understanding and reuse. Business modelling can be of great benefit to understand the behaviour of a legacy system and the relationship of a legacy system to the enterprise architecture. Especially, when there are some problems on reverse-engineering legacy systems or some major components, a business model may understand the relationship between the system's functionality and business logic, and can obtain the change of requirements and communication. Therefore, it is necessary to do domain analysis and business modelling for SOA migration.

2.2 Domain Business Analysis

A service model is an important part of establishing an SOA system. Service identification starts with a domain analysis. Moreover, business rules and process logics are key factors for service identification. Thus, domain analysis and business modelling are necessary for the success of an SOA system.

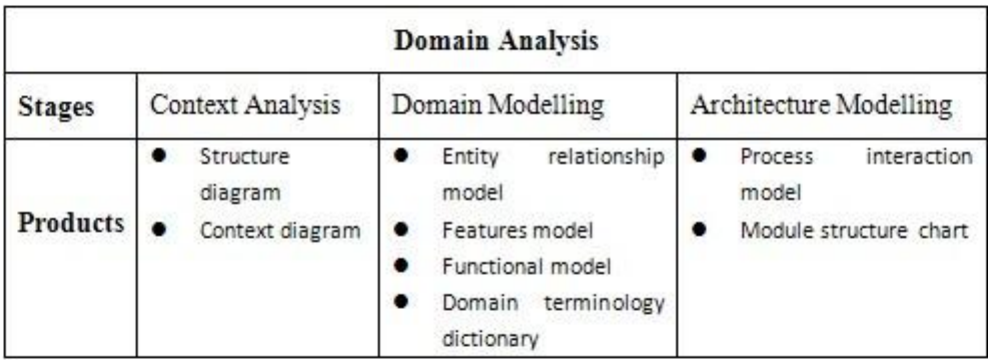
2.2.1 Domain Analysis

Domain analysis refers to collection and refinement of domain knowledge [75] to form a domain model and it is used as a single system development. As software development techniques advance, the domain analysis task is becoming more complex. Domain analysis information comes from not only domain knowledge but also other resources [83]. The roles of domain analysis are to identify common architectures, design reusable components and determine domain-oriented terminology in the application domain. And then they are encoded into generic requirement descriptions and approaches for their implementation. The main output of a domain analysis is a domain model. In a domain model, the data, objects, functions and their relationships are defined and they are usually represented in Unified Modelling Language (UML) format. In SOA, a domain model can be the index of a service registry to make selection and retrieval easer.

In SOA migration, business modelling can direct legacy system understanding. Furthermore, legacy system can be leveraged as an information source to derive business process logics and business rules. Legacy systems are a kind of information resource for domain analysis. The content and principle of legacy systems can be mined by analysing legacy artefacts and through consulting with domain experts. However, it is important to be independent from the legacy systems design and architecture when doing a domain analysis. Placing more reliance upon practices obtained from the development of a legacy system may damage new developments [49].

Generally, three stages exist during the process of domain analysis: the context analysis stage, domain modelling stage and architecture modelling stage, which are shown in Figure 2-1.

In the context analysis stage, the structure and context of analysed contents should be determined. For selecting reasonable analysis scope, the domain's bounds should be determined by domain experts, domain analysts, users, professionals and so on. Rather, information sources used for the analysis are collected by the analyst. Usually, some kinds of information sources are available for doing domain analysis. Figure 2-2 provides an overview of information sources used in the domain analysis [83].

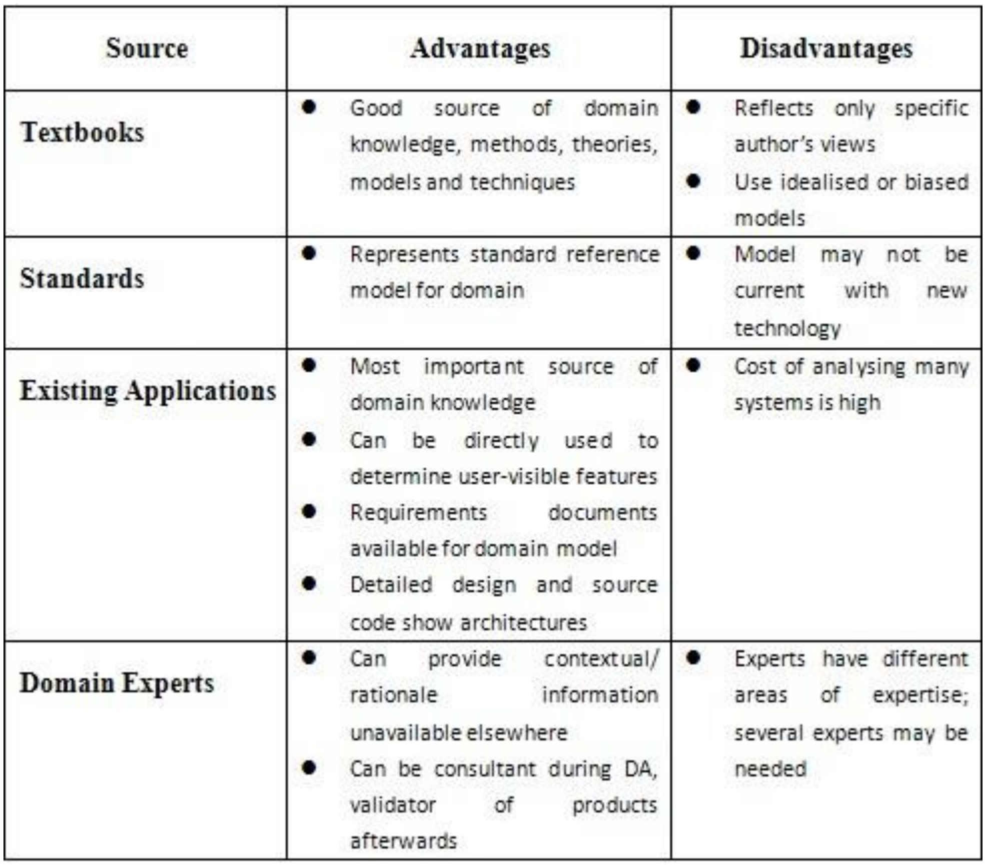


Domain Analysis			
Stages	Context Analysis	Domain Modelling	Architecture Modelling
Products	● Structure diagram ● Context diagram	● Entity relationship model ● Features model ● Functional model ● Domain terminology dictionary	● Process interaction model ● Module structure chart

Figure 2-1. Stages of Domain Analysis [83].

In the domain modelling stage, the domain's problems which have been exposed by software will be modelled, such as, entity relationship model, domain terminology dictionary, etc. A domain model is established by the domain analyst through analysing information sources and products of the context analysis. The requirement analysts, domain experts and users will review this model.

In the architecture modelling stage, software architectures for solving the domain's problems will be created. The domain analyst will address the architecture model through referencing the established domain model. The requirement analyst, domain expert and software engineer will review this model.



Source	Advantages	Disadvantages
Textbooks	● Good source of domain knowledge, methods, theories, models and techniques	● Reflects only specific author's views ● Use idealised or biased models
Standards	● Represents standard reference model for domain	● Model may not be current with new technology
Existing Applications	● Most important source of domain knowledge ● Can be directly used to determine user-visible features ● Requirements documents available for domain model ● Detailed design and source code show architectures	● Cost of analysing many systems is high
Domain Experts	● Can provide contextual/ rationale information unavailable elsewhere ● Can be consultant during DA, validator of products afterwards	● Experts have different areas of expertise; several experts may be needed

Figure 2-2. Information Sources for Domain Analysis [83].

Domain analysis is a key way for software reuse. Some mature domain analysis methods exist [37, 82, 83, 124]. The general review of domain analysis methods is introduced in [83].

Recently, domain analysis methods based on features are becoming valuable such as, a domain analysis method named Feature-Oriented Domain Analysis (FODA) [83]. This method is created on real-life experiences and years of expertise by SEI CMU [130]. Its application to the Army Movement Control Domain is shown in [49]. Another method is Feature-Oriented Reuse Method (FORM), which creates a reusable and adaptable domain model through analysing and modelling commonalities and differences in an application domain [82]. These applications could be good examples when similar work may be done.

In SOA, the domain analysis process is divided into two steps, namely, the identification of sub-domain and its analysis. The identification of sub-domain includes the determination of sub-domains' scopes and decomposition of problem domains. The output of all domain analysis is a set of domain models [146]. Sometimes, the analysis results are object-oriented models or data models. Usually, the former can be represented by UML and the latter can be represented by entity-relationship diagrams (ERD). They are useful for designing and implementing software architectures and domain applications [32]. The obtained domain models can be used for further software development. Figure 2-3 illustrates the application of domain analysis products. Clearly, domain analysis can support software development [83].

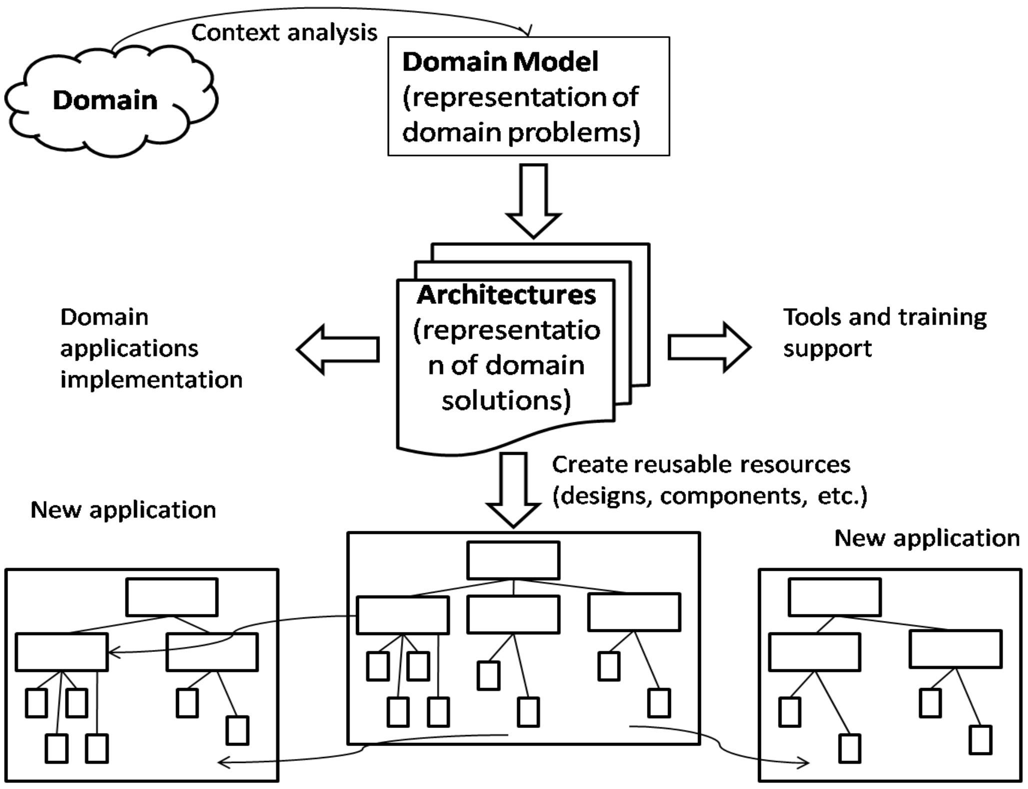

Context analysis
Domain
Domain Model (representation of domain problems)
Architectures (representation of domain solutions)
Domain applications implementation
Tools and training support
Create reusable resources (designs, components, etc.)
New application
New application

Figure 2-3. Domain Analysis Support Software Development [83].

2.2.2 Business Process Modelling

SOA architectural style aims to extend or change enterprise business solutions with the requirement change. Practice from SOA implementation proves that the abstraction level should be raised up to the business domains.

A Business Process (BP) is a sequence of activities or tasks that are valuable in this business domain [63]. Business activities and tasks are elements used to decompose business processes. Business Process Modelling (BPM) is about modelling, deploying, optimizing, simulating, monitoring, managing and running business processes [63].

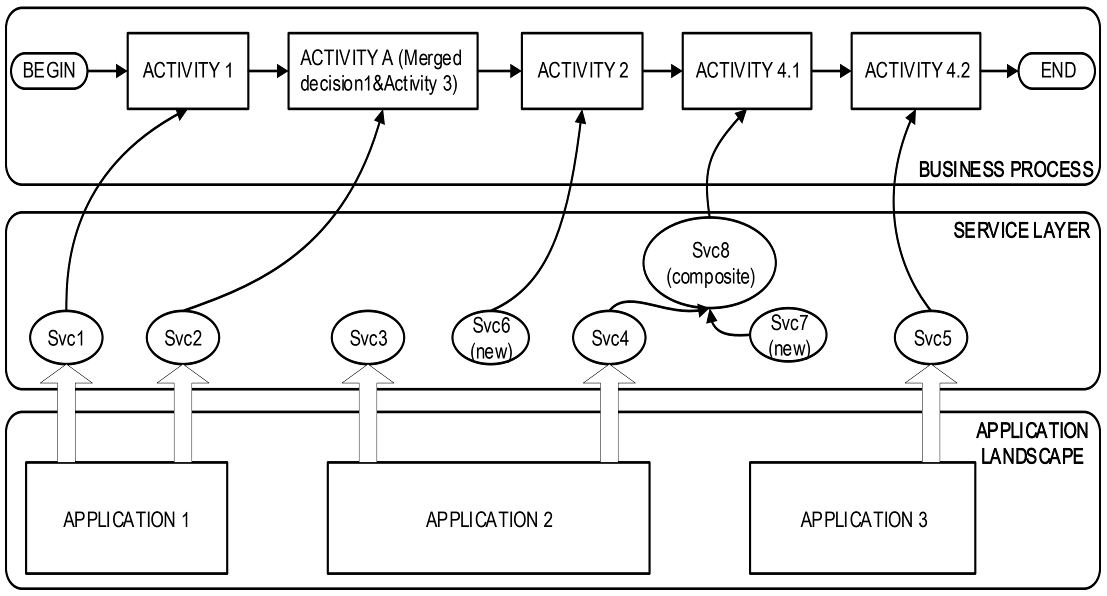


Figure 2-4. Correlations of Services, Activities and Business Process [122].

Another concept that should be mentioned in business processes is business process choreography, which means the flow of business processes. Business process flows can be obtained by choreographing identified services or processes into compound applications. It includes non-interruptible or interruptible flow. A non-interruptible flow refers to receiving input and generating output. An interruptible flow refers to taking some time to complete, or being interrupted for human interaction, or being shut down or restarted.

Correlations of services, activities and business process are shown in Figure 2-4 [122]. Clearly, business process is about an end-to-end view on functional units of work. Business processes encompass multiple service invocations.

2.3 Service Oriented Architecture (SOA)

2.3.1 Overview

SOA is to integrate a software system for providing a set of linked services. These services may be end-user applications or some discoverable and published interfaces [133]. In the following sections, more descriptions of them will be presented.

2.3.1.1 Service

The term service is used widely and is the core of SOA. Different organisations have different opinions on service's definition, which always leads to a lot of arguments. In this study, the service definition from IBM is supported. No matter which definition will be accepted, service is always concerned with the following concepts and characteristics.

- Service provider.

The maker of a service application is called a service provider. Normally, the interface signature of a service is published through a Web Services Description Language (WSDL) [36]. Service providers advertise their services to service registries.

- Service consumer (user, requestor, client).

The user of a published service is called a service consumer. Service consumers question service registries to find required services.

- Service registry**.**

Service registry is a specific service provider, which allows service consumers to search the interfaces of the service provider and the locations of service. Service consumers invoke services only based on registered information.

- Service broker**.**

Service broker refers to a specific service provider, which can transfer service requests to some other service providers, to help services consumers find services supporting their operations. Broker is optional since consumers can obtain services directly from a provider. Figure 2-5 shows the relationship among service providers, consumers and brokers [1].

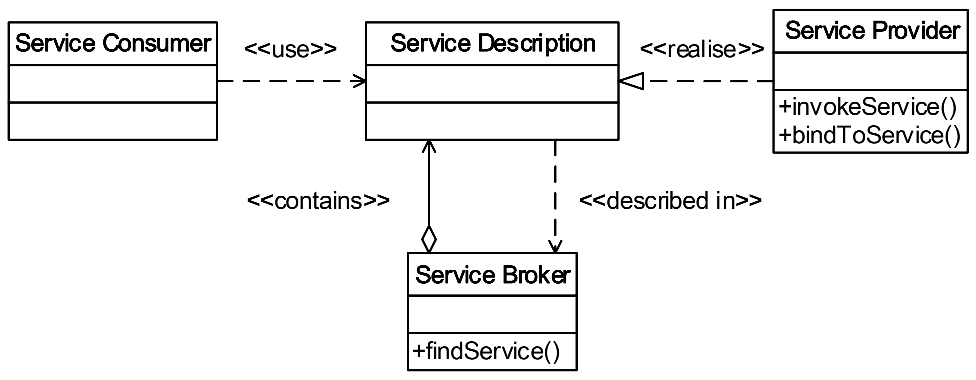


Figure 2-5. The Relationship among Service Providers, Consumers and Brokers [1].

- Service granularity.

The functionality scope exposed by a service is defined as service granularity. Different granularities exist for services themselves, such as coarse-grained and fine-grained. The normally recognised granularities consist of technical functions (such as, logon), business functions (such as, getCustomerInfo), business transactions (such as, closeAccount) and business processes (such as, processOrder) [12]. Usually, it is necessary to do service composition among these granularities because an application can consist of different grained services. Figure 2-6 shows the detailed illustration of service composition [71].

The main features of service's application [2] are as follows.

- Coarse-grained.

Coarse-grained services can fulfil more powerful functionality and can be applied on huge sizes of data sets to meet specific business needs. Thus, the complexity of a coarse-grained service is higher.

- Discoverable**.**

Services should be discovered at any time (such as, run and design time) and by any identity (such as, unique identity, interface identity and service identity).

- Loosely coupled**.**

It refers to a qualitative measure of the dependency between two components. The connection methods amongst services, clients and other services are isolated, standard, and message-based decoupled methods, such as the exchanging of XML file.

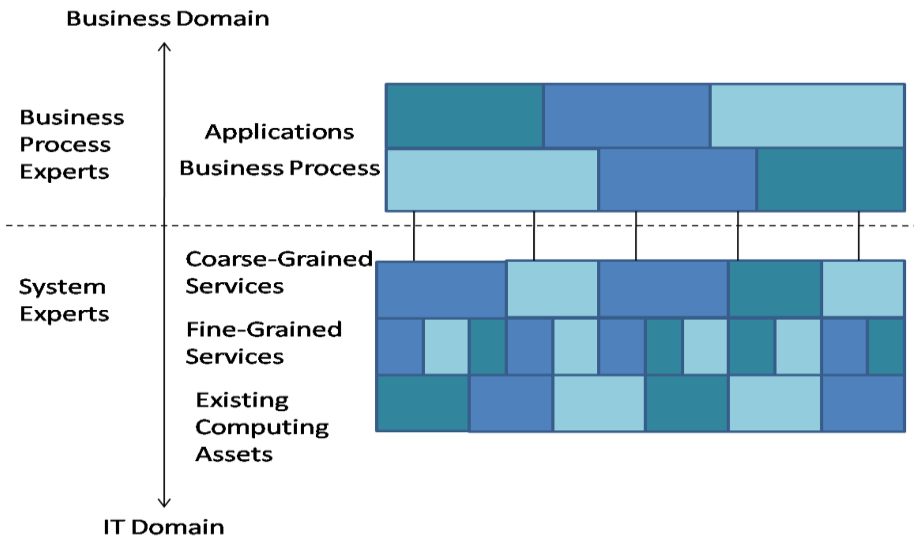


Figure 2-6. Service Composition [71].

- Interface-based design.

The designed interfaces can be developed separately. Its advantage is that a common interface can be developed by some services and vice versa.

- Asynchronous**.**

The message transferring data among services is asynchronous. Asynchrony reduces the dependency between components by allowing a component not to wait for another one.

- Single instance**.**

Any one of the services is a single instance. Usually, it is a working instance that communicates to some others.

From an implementation viewpoint, a service should be logical and self-contained. There are satisfied interfaces and contracts among services. The services communicate to each other for transferring data or for some coordination of business process. The services will be mapped into business processes. Web technologies can be regarded as abundant toolkits for services delivery. Usually, services are delivered as wrappers for concrete applications.

From an application viewpoint, the establishment of service is up to the business needs and is not up to IT capabilities. The routing information of service can be obtained from business requirements. Business functions can be published by services. Therefore, services orientation is a good choice to set up enterprise-scale applications.

2.3.1.2 SOA

SOA is not a novelty. The first SOA is introduced in [105]. Different companies or individuals have a different understanding for SOA. Roughly, an SOA refers to a framework model. Its core is a collection of services. Figure 2-7 shows a basic SOA. The interface descriptions of these services can be published and discovered. In SOA, resources in one network are available to other network members by the means of independent services. The data transmission or coordinating activities among services can be completed by communication. The concrete techniques and practices on component-based systems development will be applied to implement an SOA. Compared to traditional system architectures, SOA provides a loosely coupled interoperable structure and independent standard interfaces. SOA becomes more important with the emergence of web services technology.

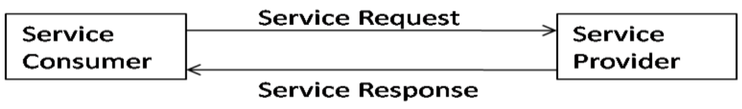


Figure 2-7. A Basic Service Oriented Architecture

The SOA's characteristics are as follows [40].

- In SOA, components have become services, which communicate to each other through well-defined protocols like SOAP.
- In SOA, communication infrastructures are protocol-independent.
- In SOA, the interdependencies among services are lower.
- In SOA, the service granularity provides flexible and agile business processes. The coarse-grained business services are preferred.

The advantages of SOA [40] are as follows,

- The location independence is provided. Services are not bound with fixed users or places. Services can be executed on any consumers and anywhere.
- Protocol-independent communication frameworks enables code reusability.

- Loosely coupled system architecture makes service composition and business process choreography easier.
- SOA can benefit run-time deployment and perfect service management and application development.
- SOA can contribute with a fast response and good adaptability when changes happen in business requirements.
- It can offer functions on authorisation and authentication, and other security functionalities through well-defined interfaces, which is better than closely-coupled situations.
- SOA allows retrieving and linking dynamically to available services.
- SOA can improve an enterprise's management abilities. By analysing the communication among services in SOA, when and what business process is effectively implemented can be captured and this enables administrators, or analysts, to optimise business process.

It has been over years of history that some large software organisations and institutes have built and deployed SOA applications successfully. Some SOA development practices and tools developed by IBM [16, 102], SEI CMU [130], BPTrends [122] and CBDI [24, 87] are useful for SOA application. In the following sub-section, an SOA reference model from IBM [15, 17] will be described.

2.3.2 SOA Reference Model

A Reference Model (RM) means abstract definitions of core concepts within an application domain. Its purpose is to ensure consistency when these concepts are used elsewhere. The elements and constructs in a RM are non-implemented. A RM might also be considered an ontology representing the semantics and schema for concepts within a domain.

An SOA reference model represents the key capabilities needed to support service-oriented architecture. A clear and unambiguous definition of service concept is expected in an SOA RM. A reference architecture (RA) is always expected after a reference model is established [130]. Reference architectures are often regarded as scenario specific (addressing a specific requirement). Currently, some SOA reference models and architectures have been released by IT organisations and institutes. In the following, some work from IBM [1, 15] will be introduced.

Figure 2-8 shows the IBM SOA reference model. The incremental SOA deployment method is allowed in this model with the changing of business requirements. It means that the SOA deployment starts from small size parts, and gradually extends to the whole project. In this way, the risk will be reduced.

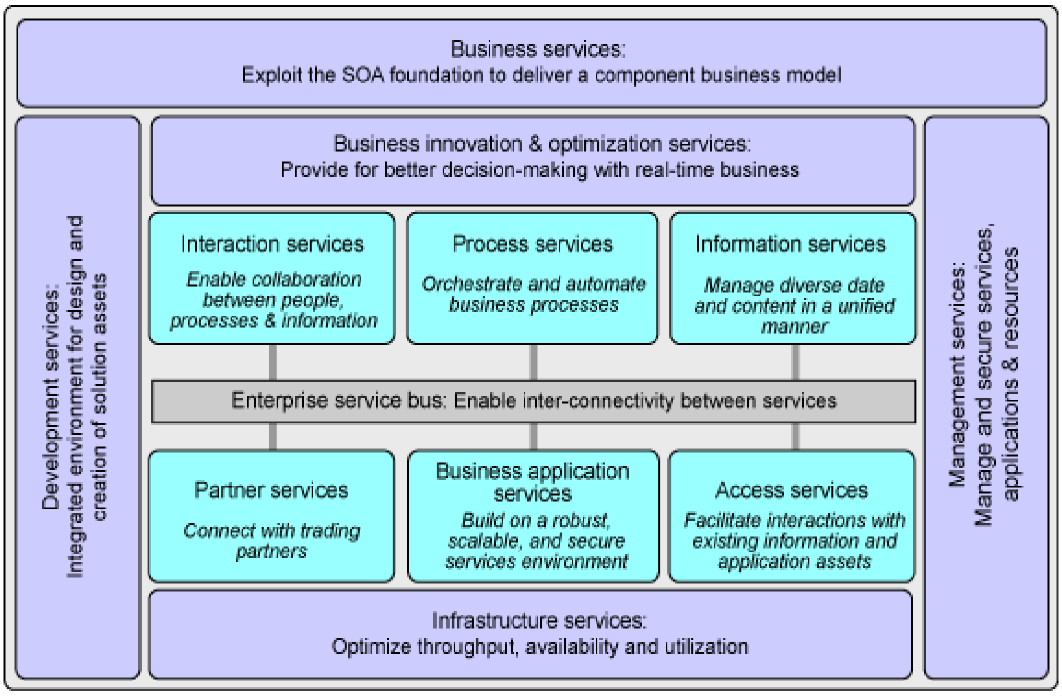

Business services:
Exploit the SOA foundation to deliver a component business model
Development services:
Integrated environment for design and creation of solution assets
Business innovation & optimization services:
Provide for better decision-making with real-time business
Interaction services
Enable collaboration between people, processes & information
Process services
Orchestrate and automate business processes
Information services
Manage diverse date and content in a unified manner
Enterprise service bus: Enable inter-connectivity between services
Partner services
Connect with trading partners
Business application services
Build on a robust, scalable, and secure services environment
Access services
Facilitate interactions with existing information and application assets
Infrastructure services:
Optimize throughput, availability and utilization
Management services:
Manage and secure services, applications & resources

Figure 2-8. IBM SOA Reference Model [15].

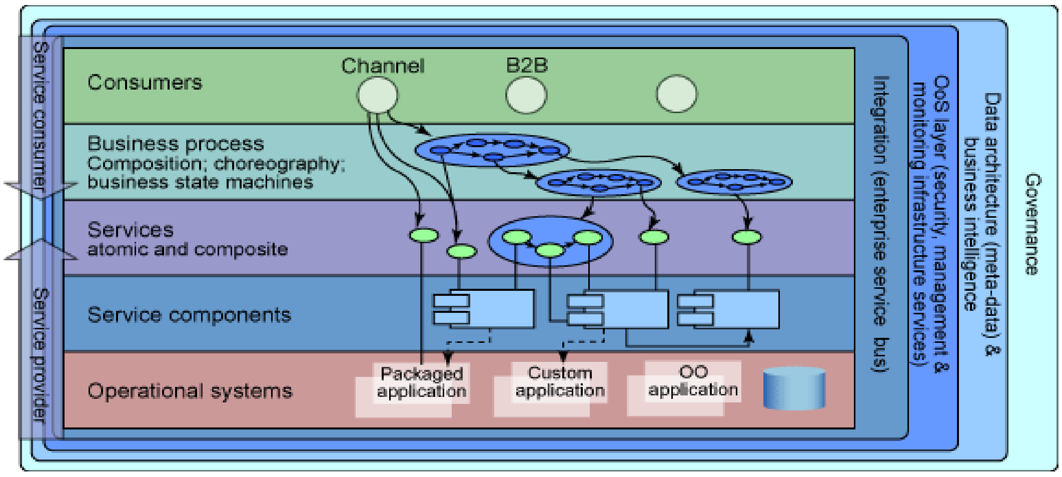

Service consumer
Service provider
Consumers
Channel
B2B
Business process
Composition; choreography; business state machines
Services
atomic and composite
Service components
Operational systems
Packaged application
Custom application
OO application
Integration (enterprise service bus)
QoS layer (security, management & monitoring infrastructure services)
Data architecture (meta-data) & business intelligence
Governance

Figure 2-9. IBM SOA Solution Stack [15].

Figure 2-9 introduces the IBM SOA solution stack [15], which is a part of SOA RM. It is the high-level abstraction from the conceptual viewpoint of an SOA solution. It is also called layered service oriented architecture. It illustrates five functional layers. They are defined as follows.

- Operational system layer.

It presents legacy assets, such as CRM and ERP packaged applications, custom applications, OO applications and business intelligence applications. Furthermore, it shows that IT investments are wonderful and can be triggered and integrated into an SOA environment.

- Service component's layer.

It realises service components by using existing applications in the operational systems layer. Some of the legacy components can be reused in the SOA environment. The service components will be elected to realise or manage functionality and maintain the quality of service. Services can be used to access components by business processes and consumers.

- Service's layer.

It exposes the deployed services to the new application environment. The exposed services can be found, invoked and bound. Moreover, the composite services will be composed with them.

- Business Process layer.

It shows which processes should be represented as services' choreographies and which will be deployed into applications. Services are bound together to form a concrete application, which can be of great benefit to specific business processes and use cases.

- Consumer's layer.

It exposes the channels of access and presentation. They can be used to access services, business processes and applications.

Among these five layers, services layer is the basis of application, which can be accessed directly. An example of service definition hierarchy [102] is illustrated in

Figure 2-10.

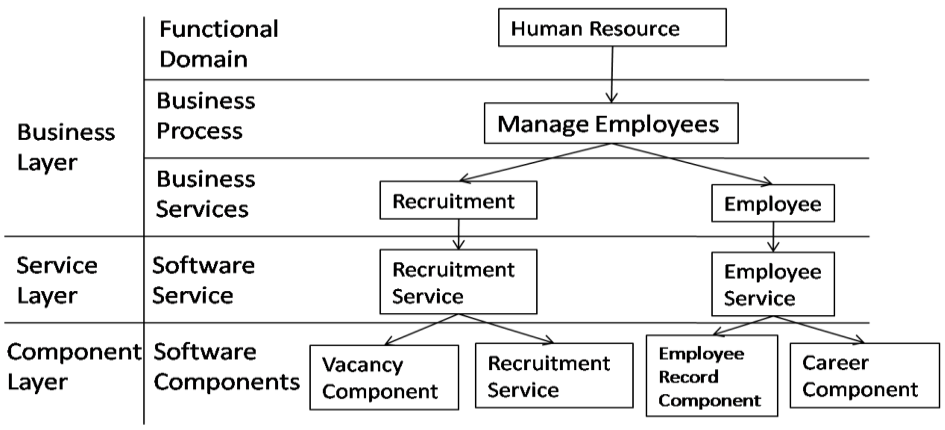


Figure 2-10. An Example of Service Definition Hierarchy [102].

For ensuring the quality of service, these built services must be supervised and managed. Thus, non-functional requirements are also attached. They are an Integration layer, Service quality layer, Information architecture layer and Governance layer.

2.3.3 Service Identification and Composition

Service identification is the key activity of service oriented analysis. The main steps are as follows,

- Service identification in a service domain.

  It includes the business-driven manner for value-chain analysis and uses a case driven manner for composite applications [122]. A top-down approach, which decomposes business domains into functional areas, sub-systems, processes, sub-processes, use cases and so on, is always adopted in this step. High-level business process and use cases are externalised for large-grained services.
- Service identification through leveraging an existing system.

  A bottom-up approach is always adopted in this step. The main work is to analyse legacy systems for selecting some candidates to form potential service functionality that can benefit the business process. Usually, fine-grained services are identified in this step. They are the foundation for implementing the coarse-grained and composite services.
- Service identification through goal-service modelling.

A middle-out approach is adopted in this step. The main work is to discover and verify other services that are not obtained by a bottom-up or top-down approach.

Some granularity's services exist, such as coarse-grained or fine-grained. Usually, service composition methods on different granularities will be needed because an application can be the mixture of varying-grained services. The detailed illustration of service composition [71] is shown in Figure 2-4.

Nowadays, three main service composition methods exist [120], namely, workflow-based method, artificial intelligence method and ontology-based method [91]. These normal service composition methods are often applied to the service composition invoked by service consumers [98]. However, in a SOA migration situation, the information obtained from traditional business analysis methods is not enough for service composition. There are a lot of data assets in a legacy system. Currently, the collected Software Engineering (SE) data has been applied to domain analysis. In order to utilise this data efficiently, data mining methods have been applied to the software engineering field [74]. Furthermore, some experienced and practical information on service composition has been hidden in user's behaviour data. It is known that the ways that fine-grained services are discovered, selected, and appended into composite services are outside the scope of these traditional service composition methods. Therefore, how to implement service composition in a SOA migration situation by using the user's behaviour data is becoming a wonderful research question. Some work has been done for this research question in this study.

2.3.4 SOA Migration Strategies

For reusing the existing legacy assets, SOA migration is a good choice. SOA provides a framework for reusing legacy assets despite their development environment [150]. The higher reusability is a key advantage of SOA. The work on SOA migration is complex. Up to now, some SOA migration strategies have been proposed and used to direct some SOA migration projects' implementation.

SOA migration work is analysed from the following viewpoints: service provider, SOA architect or service customer [146]. From the viewpoint of the service provider, normal

SOA migration strategies can be classified into three categories [145],

- Black-box strategy.
  This is to wrap the legacy component to the web service. First, the legacy system's interfaces are analysed. Afterwards, the analysed results will be compared with the SOA domain targets. The matched pairs will be wrapped into services by service-oriented technology. Therefore, services will be composed of the legacy components and new web service interfaces. It is hard to evolve for the re-built systems by adopting a black-box strategy. It can satisfy a temporary need. However, it will complicate the management and maintenance of legacy systems in the long term.
- Business logic strategy (White-box strategy).
  This is to migrate a legacy system to a new environment in terms of business logic [28]. First, reverse engineering methods are applied to discover the business logic from legacy assets. Secondly, a new system is designed and developed according to the recovered business logic. This kind of approach can be of great benefit to the efficiency of the maintenance process. However, the system comprehension is costly. Sometimes, it is impossible to discover the business logic correctly and completely.
- Grey-box strategy.
  This is the combination of black-box and white-box strategies. Only some parts of legacy assets contain valuable business logic. At the same time, many legacy functions are wrapped as components. Component orientation supports the implementation of service oriented computation [116]. These components can be reused by the technologies of SOA and web services. The legacy assets will provide more benefits to the services.

These strategies have strengths and weaknesses respectively [145]. Currently, the grey-box strategy is a popular one. Concretely,

- first, top-down analysis of domain requirements;
- second, goal-service modelling;
- third, bottom-up analysis of legacy assets.

- fourth, the matching relationship of services/business processes and IT components provides a candidate solution for SOA migration.

In addition, from an enterprise architecture (EA) point of view, a migration strategy from old enterprise architecture to service oriented enterprise architecture is presented in [12]. It points out Enterprise Service Bus (ESB) can be used as a mechanism for routing messages as well as integrating and managing services between service consumers and providers. ESB provides connectivity among services, service registry and service choreography. Moreover, the work on publishing, discovering and triggering services can be implemented through ESB Gateways.

Some progress has been made on migrating legacy systems into an SOA environment. The details of some existing SOA migration strategies [116] are as follows.

- Software Engineering Institute of Carnige Melon University contributes to the SOA migration strategy focusing on Service-oriented MigrAtion and Reuse Technique (SMART) [47]. In SMART, the main work includes identifying and analysing services for the purpose of SOA migration.
- Harry Sneed proposes the SOA migration strategy focusing on a Salvaging & Wrapping Approach (SWA) [58]. The main steps in SWA are:
    - first, salvaging the existing software programs;
    - second, wrapping the salvaged programs [57];
    - third, exposing the wrapped programs as Web services.
- Ziemann presents the SOA migration strategy focusing on an Enterprise Model Driven Migration Approach (EMDMA) [78]. In EMDMA, more attention is paid to enterprise modelling and SOA migration by adopting a business driven approach.
- SOA migration strategy from IBM [81] describes not only the appropriate practices with reused legacy assets but also the content on SOA governance.
- Oracle presents an Oracle Modernisation Framework (OMF) [103]. It has set up an Application Portfolio Analysis (APA). Afterwards, some kind of modernisation techniques (for example, re-architecting, automated migration, COTS replacement, enablement, re-hosting, etc.) will be adopted to do the related work.

- Microsoft [99] suggests that SOA strategy should be designed according to domain requirement and be implemented by adopting an incremental and iterative approach.
- SAP provides the Enterprise Services Architecture (ESA) Adoption Program [129]. Customers can use its Web-based development and integration platform to build their SOA.

Most of these strategies emphasise the role of the migration technique itself and lack of SOA implementing principles [102]. Moreover, only some work on the architectural reconstruction or reverse engineering of source program levels have been done. However, this kind of context-sensitive attempt is not suitable for large scale migration projects and it is not easier to generalise them [116]. Currently, a holistic method, which considers both legacy system assets and new business targets, is needed to accomplish SOA migration systematically.

2.3.5 SOA Migration Practices

Some SOA migration projects have been completed successfully by large IT organisations and companies such as banking applications [96, 138], electronic payment applications [76] and development tools [119]. Some detailed approaches and techniques will be presented in this sub-section.

In [153], it contributes an approach on migrating legacy system to SOA. The details are as follows.

Step1: Evaluating legacy assets. The main work is to evaluate the quality of the existing legacy asset and identifying its phase in the lifecycle. This assessment is used to make reengineering decisions. The evaluated aspects are connected together by a decision tree. The Options Analysis for Reengineering (OAR) and other techniques are utilised to direct the decision-making process.

Step 2: Recovering architecture. The main work is applying reverse engineering methods to discover architecture and design information maximally. The quality of legacy assets will be used to choose analysis methods, such as the dynamic and static analysis techniques. Some quality indicators (for example, coupling, cohesion, reliability and so on) will be taken into account for identifying hidden components and connectors. Its output is the input of the next Step.

Step 3: Identifying service. The main work includes: first, identifying the concrete business functionalities that will be completed by the domain target services; and second, extracting components from the legacy assets for reusing them in domain target services.

Step 4: Wrapping Service. The main work is to wrap legacy components to meet the needs of the identified domain services. The integration process subjects to the architecture.

Step 5: Publishing and choreographing service. Universal Description Discovery and Integration (UDDI) will be used to register services.

In [116], an approach named MASHUP is represented. It includes the following six steps,

Step 1: Business modelling. In order to obtain the target’s requirements, the domain requirements are modelled.

Step 2: Analysing legacy assets. The valuable data and architecture information can be recovered through legacy assets analysis. The information can be used to design Domain Specific Kits (DSKs) and investigate legacy components’ reuse potential.

Step 3: Establishing the mapping relationships. According to the legacy analysis results and business modelling results, deciding how to obtain services. The ways for obtaining services include wrapping (match) or improving (part of match) or implementing (unmatchable).

Step 4: Applying DSKs to design the architecture of a Mashup server. The work on architecture modelling is subject to architectural aspects and quality targets for running domain specific kits and choreography rules.

Step 5: Defining Service Level Agreements (SLA). In a service library, SLA is defined. Usually, SLA includes not only functional but also non-functional information such as QoS.

Step 6: Implementing and deploying services. There are four ways for implementing and deploying services, namely,

- Wrapping: wrapping legacy components as services;
- Reconstruction: modifying legacy components as services;
- Development: implementing new services from scratch;
- COTS: integrating COTS components as services.

In [109], an SOA migration approach is represented in which a top-down analysis method is applied to SOA design and a bottom-up analysis method is used to legacy system understanding and application portfolio assessment. Two projects in the finance industry and in manufacturing have been completed by adopting this approach. A similar approach is also described in [1].

In [12], it shows some techniques on domain separation. A complex enterprise, which owns a number of departments, should be divided into some separated domains. The smaller-size makes it easier to do further processes. The boundaries between domains can be established by gateways and firewalls. Transparent/Proxy gateways expose domain services so that consumers can interact with business functions. Meanwhile, firewalls are good for implementing domain encapsulation.

In [44], a wrapping SOA migration approach is described. Its main work is to migrate interactive functionalities of legacy systems to SOA. Finite State Machines are established to do wrapping work. In this approach, a black-box SOA migration strategy and some reverse engineering techniques are presented for wrapping legacy systems' interactive functionalities as services.

In [7], an incremental wrapping migration strategy is presented. In order to determine the migration strategy, the work on evaluating legacy systems from the aspects of software quality and domain value has been performed. For migrating multi-user COBOL legacy systems to a multi-tier web-based architecture, the Migration Environment for Legacy Information System (named MELIS) has been designed and implemented.

In [13], several opinions on the reuse potential in SOA migration situation are outlined. A strategic decision model for SOA reengineering projects is depicted. In this model, the decision-making method for integration versus migration takes technical and strategic factors and cost benefit analysis into account. There are four parts in this model: strategic analysis, architecture analysis, solution development and implementation. However, it just focuses on the migration techniques themselves and lacks the detailed implementation schema. It is still a high level paper for planning an SOA migration project.

SOA migration is a part of legacy system migration. It should follow the generic system migration process. According to the above reviews, it seems most of related researches have just focused on the last four stages of a generic legacy system migration process. Namely, much more research has been done for answering a question like “how to do a SOA migration project?”, less work has been done for answering a question like “is it worth triggering an SOA migration project?”.

The current researches did not present details on the justification stage of an SOA migration project. In fact, this kind of work (planning and prediction) is very important for avoiding undesired results in the SOA migration situation. Adequate planning and prediction are important before starting an SOA migration project. Otherwise, once it fails, it will be an expensive mistake. Some SOA migration details should be explored to ensure that project decision-makers can obtain comprehensive information and make a sound decision. It is necessary to explore the feasibility of an SOA migration project.

Therefore, it is necessary to address a general and comprehensive SOA migration planning approach to supervise the planning and deployment of an SOA migration project. Furthermore, this approach should pay more attention to the implementation details instead of just methodological strategies. According to this, some planning and prediction tasks can be done (semi)-automatically instead of through manual work. In this study, more research will be done on an SOA migration planning approach. In the proposed approach, data mining and text retrieval techniques will be adopted to create SOA migration planning schemas, which is the foundation for planning and prediction of an SOA migration project.

2.4 Data Mining Techniques for SOA Migration Environment

Generally, some intelligent information processing technologies and approaches can be of great benefit to SOA migration task. Concretely, data mining techniques and text retrieval techniques can be applied to a SOA migration situation. This research question has been investigated in this study.

2.4.1 Data Mining Techniques and Software Engineering Data

2.4.1.1 Data Mining Process

Data mining is a field relating to a number of domains, such as database technology, artificial intelligence, statistics, information retrieval, machine learning, neural networks, knowledge acquisition and so on [70]. Techniques and methods in artificial intelligence, machine learning, statistics and database systems can be applied in the data mining field.

A simply data mining process includes three stages,

- Pre-processing stage.
  Pre-processing is the foundation of data mining. Techniques on data integration and data cleaning can be adopted in this stage for ensuring the mining quality.
- Data mining stage.
  It is the core of the data mining process. There are some typical methods and algorithms available for completing some specific functions, for example, outlier/change/deviation detection, association analysis, clustering, classification, evolution, summarisation, etc.
- Results validation stage.
  It is to verify the patterns produced by the data mining algorithms. Some measurements and interactions will be deployed in the data mining process for keeping the interesting patterns. If the mined patterns go through the validation, then the final step is to interpret them and represent them into knowledge.

2.4.1.2 Data Mining Application

With the development of information technology, a great deal of software data has been collected by some organisations and companies. Some data mining approaches on SE data [95] have been investigated [137] since the late 90’s.

According to [135], data mining techniques applied on SE data are classification, clustering, association rules, frequent patterns, matching/searching, abstraction-based analysis, concept analysis, automaton/grammar learning, template-based analysis and so on. The mined SE data includes execution traces, software change history, static code bases, profiled program states, bug reports/natural languages, profiled structural entities, deployment logs and so on. SE tasks which have benefited from DM techniques include programming, testing, debugging, static defect detection, maintenance, etc.

SOA is an IT architectural approach that is business-oriented. Moreover, it helps integrating business logics as iterative business work or services. Reusing legacy assets is very important for the success of an SOA migration solution. Clearly, legacy system understanding and reuse is an important part in an SOA migration environment. For comprehending legacy assets completely, some DM methods can be used for software engineering data. The purposes of mining software engineering data are to transfer static record-keeping SE data to active data and make SE data actionable by recovering hidden patterns and motifs. Mining software engineering data can predict, plan and understand more details of existing applications and further benefit future management and development activities [135].

Legacy system understanding approaches by using data mining techniques [14] include dynamic traces analysis and static traces analysis, etc. Dynamic traces are utilised to determine service identification and composition as well as migration strategy [132]. Static traces are adopted to validate the program modification in a SOA migration situation [127]. These two are also utilised to extract the descriptions of service interface.

2.4.2 Application of Data Mining Method in Legacy Migration Situation

In an SOA migration situation, a great deal of data assets is available in a legacy system. The information coming from domain analysis and superficial legacy assets analysis is not enough for implementing SOA migration. For reusing existing legacy assets efficiently and adequately, not only the explicit assets, but also the implicit assets, should be utilised. Clearly, it is an important task to discover the hidden information from the legacy assets.

It is known that service identification and composition are the key tasks in an SOA migration situation. Practical and experienced information on service identification and composition had been hidden in the usage behaviour data of legacy assets. The usage behaviour data resource is the collection of real operation sequences requested by users. The hidden patterns/motifs in this data collection will provide users' practical experiences, which can benefit the service identification and composition in the SOA migration situation. The motif refers to the set of maximal and frequently occurring patterns. These mined frequent patterns/motifs will be the candidate coarse-grained services in the target system. In this study, this question is triggered. The investigation results show the usage behaviour data can be mined to discover the potential concerns that can provide benefits to SOA migration projects. The mined hidden information can be used for perfecting service models and component models of a SOA migration environment.

2.4.3 Analysis of Traditional Data Mining Algorithms

The data mining algorithms include an association rule mining algorithm, sequence pattern mining algorithm, classification algorithm, clustering algorithm, etc. In this study, more attention has been paid on the association rule mining algorithm and the sequence pattern mining algorithm.

2.4.3.1 Association Rule Mining Algorithms

The traditional algorithms on association rule mining include Apriori algorithm and its variations [108,114] as well as FP-growth algorithm [70]. The former is proposed by

Agrawal et al. The latter is proposed by Han et al.

The purpose of association rule mining is to mine the patterns/motifs over a transaction database. Algorithm Apriori is to discover patterns/motifs with a fixed *min-support* number over a transaction database. Apriori assumes items in a transaction set are totally ordered. Furthermore, items within an itemset are sorted in lexicographic order.

Figure 2-11 describes the Apriori Algorithm.

```
Algorithm A   Apriori for finding motifs with a fixed min-support number over transaction database $\mathcal{U}$.
A1.[Find $L_1$] $L_1$={large 1-sequence}   //finding $L_1$ from transaction database $\mathcal{U}$
A2.[Find $L_l$ for $l \geq 2, L_{l-1} \neq \emptyset$] for $(l = 2; L_{l-1} \neq \emptyset; l++)$ do
    $C_l = apriori\text{-}generate(L_{l-1})$ //call algorithm apriori-generate
            to generate candidate set $C_l$ from $L_{l-1}$
    for each $w \in C_l$ compute support number $|w^{\mathcal{U}}|$
    $L_l$ = Candidates in $C_l$ with min-support number
A3.[Find motifs] Answer = Maximal transactions in $\cup_l L_l$ .  // the motifs of transaction database $\mathcal{U}$
```

Figure 2-11. The Apriori Algorithm.

The Apriori-generate Algorithm is presented in Figure 2-12.

```
Algorithm G   apriori-generate($L_{l-1}$) for finding set $C_l$ of candidate patterns.
G1.[join] join $L_{l-1}$ with $L_{l-1}$
    insert into $C_l$
    select $p.litemset_1, p.litemset_2, ..., p.litemset_{l-1}, q.litemset_{l-1}$
    from $L_{l-1}$  $p$,  $L_{l-1}$  $q$
    where $p.litemset_1 = q.litemset_1, ..., p.litemset_{l-2} = q.litemset_{l-2}$
G2.[ pruning] delete all itemsets $c \in C_l$ such that some $(l-1)$-sub-itemsetof $c$ is not in $L_{l-1}$
    forall sequences $c \in C_l$ do
        forall $(l-1)$-sub-itemsets $s$ of $c$ do
            if $(s \notin L_{l-1})$ then delete c from $C_l$
```

Figure 2-12. The Apriori-generate Algorithm.

2.4.3.2 Sequence Pattern Mining Algorithms

The purpose of sequence pattern mining is to discover patterns/motifs of a sequence database. Currently, many sequence pattern mining algorithms have been available for some kinds of applications. The most famous sequence pattern mining algorithms include AprioriAll [110] and AprioriSome algorithms [113,114], which are proposed by

Agrawal and Srikant. Some similar algorithms are also presented in [5, 90, 92]. The quantities associated with items have not been taken into account in the typical sequence mining algorithm. At present, this factor has been added into the sequence mining algorithm. These kinds of methods are named quantitative sequence mining approaches [21]. Some fuzzy techniques are utilised in this field [142, 143].

AprioriSome and AprioriAll are used to find motifs on a fixed *min-support* number. The main work includes generating the set of candidate patterns and doing further inspection to discover motifs.

There are five phases in the sequence patterns mining process proposed by Rakesh Agrawal and Ramakrishnan Srikant:

1. Sort Phase.

In this phase, the old transaction database will be changed to a customer sequence database. In the new database, the major key is the column of customer-id and the minor key is the column of transaction-time.

2. Large itemset (Litemset) Phase.

In this phase, all large 1-sequences will be found.

3. Transformation Phase.

In this phase, the Litemset will be coded into some contiguous integers. The transformed 1-sequences will be obtained.

4. Sequence Phase.

In this phase, the sequence pattern mining algorithm is applied to find the desired sequences. The main work includes: generate candidate set; prune and scan database for checking support frequency, etc.

5. Maximal Phase.

In this phase, motifs will be found.

Figure 2-13 presents the Algorithm AprioriAll. Figure 2-14 describes the apriori-generate function of Algorithm AprioriAll/Some.

```
**Algorithm A    AprioriAll** for finding motifs with a fixed min-support number
over sequence database $\mathcal{U}$.
    A1.[Find $L_1$] $L_1$={large 1-sequence}    //finding $L_1$ from sequence database $\mathcal{U}$
    A2.[Find $L_l$ for $l \geq 2, L_{l-1} \neq \emptyset$] for $(l = 2; L_{l-1} \neq \emptyset; l++)$ do
        $C_l = apriori\text{-}generate(L_{l-1})$ //call algorithm apriori-generate
                to generate candidate set $C_l$ from $L_{l-1}$
        for each $w \in C_l$ compute support number $|w^{\mathcal{U}}|$
        $L_l$ = Candidates in $C_l$ with min-support number
    A3.[Find motifs] Answer = Maximal sequences in $\cup_l L_l$ . // the motifs of sequence
                database $\mathcal{U}$
```

Figure 2-13.   The AprioriAll Algorithm.

```
**Algorithm G    apriori-generate**$(L_{l-1})$ for finding set $C_l$ of candidate patterns.
    G1.[join] join $L_{l-1}$ with $L_{l-1}$
        insert into $C_l$
        select $p.litemset_1, p.litemset_2, ..., p.litemset_{l-1}, q.litemset_{l-1}$
        from $L_{l-1}$  $p$,  $L_{l-1}$  $q$
        where $p.litemset_1 = q.litemset_1, ..., p.litemset_{l-2} = q.litemset_{l-2}$
    G2.[ pruning] delete all sequences $c \in C_l$ such that some $(l-1)$-subsequence of $c$
is not in $L_{l-1}$
        forall sequences $c \in C_l$ do
            forall $(l-1)$-subsequences $s$ of $c$ do
                if $(s \notin L_{l-1})$ then delete c from $C_l$
```

Figure 2-14.   The AprioriAll-generate Algorithm.

The Algorithm AprioriSome [110] is shown in Figure 2-15.

```
// Forward Phase
$L_1$ = {large 1-sequences}; // Result of the litemset phase
$C_1 = L_1$; // so that we have a nice loop condition
$last = 1$; // we last counted $C_{last}$
for ( $k = 2$; $C_{k-1} \neq \emptyset$ and $L_{last} \neq \emptyset$; $k$++ ) do
    begin
    if ($L_{k-1}$ known) then
        $C_k$ = New candidates generated from $L_{k-1}$;
    else
        $C_k$ = New candidates generated from $C_{k-1}$;
    if ( $k$ == next($last$) ) then begin
        foreach customer-sequence $c$ in the database do
            Increment the count of all candidates in $C_k$ that are contained in $c$.
        $L_k$ = Candidates in $C_k$ with minimum support.
        $last = k$;
        end
    end

// Backward Phase
for ( $k$--; $k >= 1$; $k$-- ) do
    if ( $L_k$ was not determined in the forward phase ) then begin
        Delete all sequences in $C_k$ contained in some $L_i$, $i > k$;
        foreach customer-sequence $c$ in $\mathcal{D}_T$ do
            Increment the count of all candidates in $C_k$ that are contained in $c$.
        $L_k$ = Candidates in $C_k$ with minimum support.
        end
    else begin // $L_k$ already known
        Delete all sequences in $L_k$ contained in some $L_i$, $i > k$.
        end

Answer = $\bigcup_k L_k$;
```

Figure 2-15. The AprioriSome Algorithm [110].

The Apriori and AprioriAll/Some algorithms can be improved in a domain-specific application. For the e-learning field, some improved algorithms based on them can be proposed according to the data features of the e-learning field. In Chapter 4 and Chapter 5, two improved algorithms based on Apriori [151] and AprionAll/Some [152, 154] will be depicted, which can be utilised in the establishment of a Component model and a Service model for SOA migration in the e-learning field.

2.5 Text Retrieval Techniques for SOA Migration Environment

Information retrieval refers to the process of retrieving the relevant information from a set of information resources. Text retrieval is a branch of information retrieval where the information is stored in the form of text.

A document is the unit that will build a retrieval system. The group of documents over which retrieval work will be performed is named as collection or corpus [26]. A query refers to the representation of a user’s question for retrieving the wanted information through a computer.

For accelerating the retrieval speed, it is necessary to establish the index for the collection of documents in advance. An index refers to the mapping relationships between terms and its’ positions in documents. An index term is any word that appears in the text. Usually, an index term is a keyword (or group of related words) which has some real meaning. A document is a set of index terms.

Three types of models exist in the information retrieval field [111],

- Boolean model.
  The set theory is the foundation of the Boolean model. In this model, query and document are represented as sets of index terms.
- Vector space model.
  The algebra theory is the foundation of the vector space model. In this model, query and document are represented as vectors.
- Probabilistic model.
  The probability theory is the foundation of the probabilistic model.

2.5.1 Vector Space Model (VSM)

In VSM, queries and documents are represented as vectors. An index term, which is usually defined as a single word, keyword or longer phrase, represents a dimension of vector space. Some methods on assigning term weight have been proposed. *tf***idf* (*tf* means term frequency, *idf* means inverse document frequency) method is a famous weight assigning method [111], which has been widely applied in some information retrieval systems. Similarity is a defined concept applied in an information retrieval field. Normally, it is used to measure the similarity degree between documents. More similarity measurement methods in information retrieval field have been concluded in

[94].

For a document *d*,

- the frequency of a term *t* is the occurrence's number of term *t*, denoted as $tf_{t,d}$.
- df is the document frequency, df_t is the number of documents in the corpus that contain a term *t*.
- the *idf* of a term *t* is: $idf_t = log\ (\frac{N}{df_t})$, where N refers to the total number of documents in a corpus.
- the *tf*idf* weight of a term *t* is: $tf\text{-}idf_{t,d} = tf_{t,d} * idf_t$.

The general text similarity calculation methods in VSM include dot products (or inner product) method, cosine similarity method, etc.

The cosine similarity calculation method between documents d_1 and d_2 is,

$$sim(d_1, d_2) = \frac{\vec{V}(d_1)\cdot\vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|} \ ,$$

where $\vec{V}(d_1)$ and $\vec{V}(d_2)$ are the vector representations of documents d_1 and d_2. $sim(d_1, d_2)$ represents the similarity degree between documents d_1 and d_2.

2.5.2 Granularity of Text Similarity Calculation

Indexing granularity refers to the document unit for indexing. Each document or passage can be taken as a mini-document (unit). Normally, the trade-off between recall and precision should be taken into account. If the text granularity becomes too fine, it will miss important passages since index terms are cut into mini-documents. However, if the text granularity becomes too coarse, it is difficult to find the relevant information [26]. Hence, the granularity of text similarity calculation can be document, passage, sentence, chunk, phrase, etc.

In this study, the descriptions on legacy components and domain services will be treated as documents or passages. Text similarity calculation methods will be utilised to recognise the matching relationship between legacy components and domain services. These matching relationships are the important parts of SOA migration planning schemas.

2.6 Summary

Some research background and literature have been introduced in this Chapter.

- The approaches and process of software reengineering are reviewed simply. The legacy software system can be improved or transformed through software reengineering.
- Legacy assets can be reused through extension, transformation, integration and migration. Legacy system migration means moving legacy assets to a new operating system, hardware, middleware platform or system architecture. The work on legacy system analysis includes legacy system understanding and decomposition, and component identification and measurement.
- Business modelling plays an important role in legacy system understanding and reuse. Domain analysis and business modelling are necessary for the success of an SOA migration.
- The concepts and characteristics of service and service-oriented architecture are introduced. IBM reference model and SOA solution stack are also introduced. Service identification is the key activity of service-oriented analysis. The main steps of service identification include: service identification in service domain, service identification through leveraging an existing system and service identification through goal-service modelling. Nowadays, three main service composition methods exist, namely, workflow-based method, artificial intelligence method and ontology-based method. In fact, the service composition methods in an SOA migration situation are outside the scope of these traditional service composition methods. More studies will be done in this thesis.
- Three categories of SOA migration strategies exist: Black-box strategy, White-box strategy and Grey-box strategy. These strategies have strengths and weaknesses respectively. Some concrete migration approaches are reviewed. Meanwhile, some existing problems are pointed out.

- Data mining is a field relating to a number of domains. Applying data mining techniques to SE data, the mined software engineering data and software engineering tasks benefited from data mining techniques have been concluded. The traditional association rule mining algorithm Apriori and sequence pattern mining algorithm AprioriAll/Some are reviewed.
- Information retrieval refers to the process of retrieving the relevant information from a set of information resources. Some related concepts and basic information retrieval models have been introduced.

Chapter 3 A Proposed SOA Migration Planning Approach

Objectives

- To analyse key factors that should be contained in an SOA migration planning approach.
- To address the framework of an SOA migration planning approach.
- To explain each stage in this proposed approach.
- To introduce the implemented support tools.

3.1 Key Factor Analysis

In an SOA migration project, it is necessary to decompose and identify legacy assets into components and then reuse them to satisfy the new domain requirements maximally. Obviously, legacy assets are the migration source and domain requirements are the migration target. In order to ensure the quality of SOA migration, some analysis methods on legacy assets and domain requirements will be involved. In order to create SOA migration planning schemas (more details are described in Section 7.1), the representation on the analysis results of legacy assets and domain requirements should be addressed. The above information can be concluded into a *Service model* and a *Component model*. In this study, the two models are defined.

Definition 3.1 A *Service model* is a domain logic model, which consists of the work on domain analysis, service identification and composition as well as the representation of final results. A service model is the relationship graph of services in a domain aspect. It will be denoted as XML_{Domain}.

Definition 3.2 A *Component model* is a legacy asset model, which consists of the work on legacy analysis, component identification and the representation of final results. A component model is the relationship graph of components in a legacy aspect. It will be denoted as XML_{Legacy}.

Clearly, a Service model and a Component model are necessary factors in an SOA migration planning approach. They are the basis for creating SOA migration planning schemas.

In different theoretical support frames, there are different representations and decision-making methods for evaluating the matching relationships between legacy assets and domain requirements. Thus, the theoretical support frame is a main factor of an SOA migration planning approach. For example, if the vector space theory is taken as a theory support frame, this kind of approach can be named as a vector SOA migration planning approach; if mixed theories are taken as the theory support frame, this kind of approach can be named as a hybrid SOA migration planning approach, etc.

In order to reuse legacy assets maximally, the matching relationship between legacy assets and domain requirements should be measured. The concrete matching strategies should be addressed.

The trade-off between system performances and system cost should be taken into account. Usually, the more perfect the system performance, the more expensive the system cost. Thus, a decision-making part for creating SOA migration planning schemas is necessary.

In addition, one of the goals of an SOA migration project is to meet the users' new needs. Thus, the user's direction should be included in an SOA migration planning approach.

Finally, the obtained SOA migration planning schemas should be evaluated. Some evaluation work needs to be done so that the migration planning schemas can give more directions to the managers and implementers of an SOA migration project.

According to the above-mentioned, an SOA Migration planning Approach (SOAMA) is concluded to be a 7-tuple, namely,

$$SOAMA = < SM, CM, TSF, MS, DM, UD, MSE >$$

where, *SM* refers to a Service model; *CM* refers to a Component model; *TSF* refers to the Theoretical Support Frame; *MS* refers to Matching Strategy between legacy assets (sources) and domain requirements (targets); *DM* refers to Decision-Making method; *UD* refers to User's Direction. *MSE* refers to the SOA Migration planning Schema Evaluation.

3.2 The Framework of SOA Migration Planning Approach

The key factors in an SOA migration planning approach have been determined. In this Section, the relationships amongst these key factors and some related descriptions will be presented.

The main goal of this proposed planning approach is to create SOA migration planning schemas for directing the prediction and deployment of SOA migration projects. To reach this goal, the work can be divided into five stages, namely, preparation stage, analysis stage, matching stage, decision-making stage and evaluation stage.

Tasks in the preparation stage include calling professional persons (application domain and development field) together; collecting sources on both sides of the domain and legacy; double checking the financial state and so on.

In the analysis stage, some work on legacy and domain analysis, service and component identification, knowledge representation and so on, will be done. In this stage, a Service model and a Component model will be established.

The legacy components and domain services will meet in the matching stage. The matching relationships between them will be quantified (calculated) in this stage, which is the basis for creating SOA migration planning schemas.

SOA migration planning schemas will be created in the decision-making stage. The factor of "user's direction" will be embodied in this stage. The means of "user's direction" should be detailed.

The created SOA migration planning schemas will be evaluated in the evaluation stage. The concrete work is to integrate evaluation reports, which include all useful information for decision-makers and developers. Finally, through analysing the evaluation reports, it can be determined if the SOA migration project should be triggered.

According to the order of these five stages, the framework of the proposed SOA migration planning approach is shown in Figure 3-1. Some descriptions on the main stages are as follows.

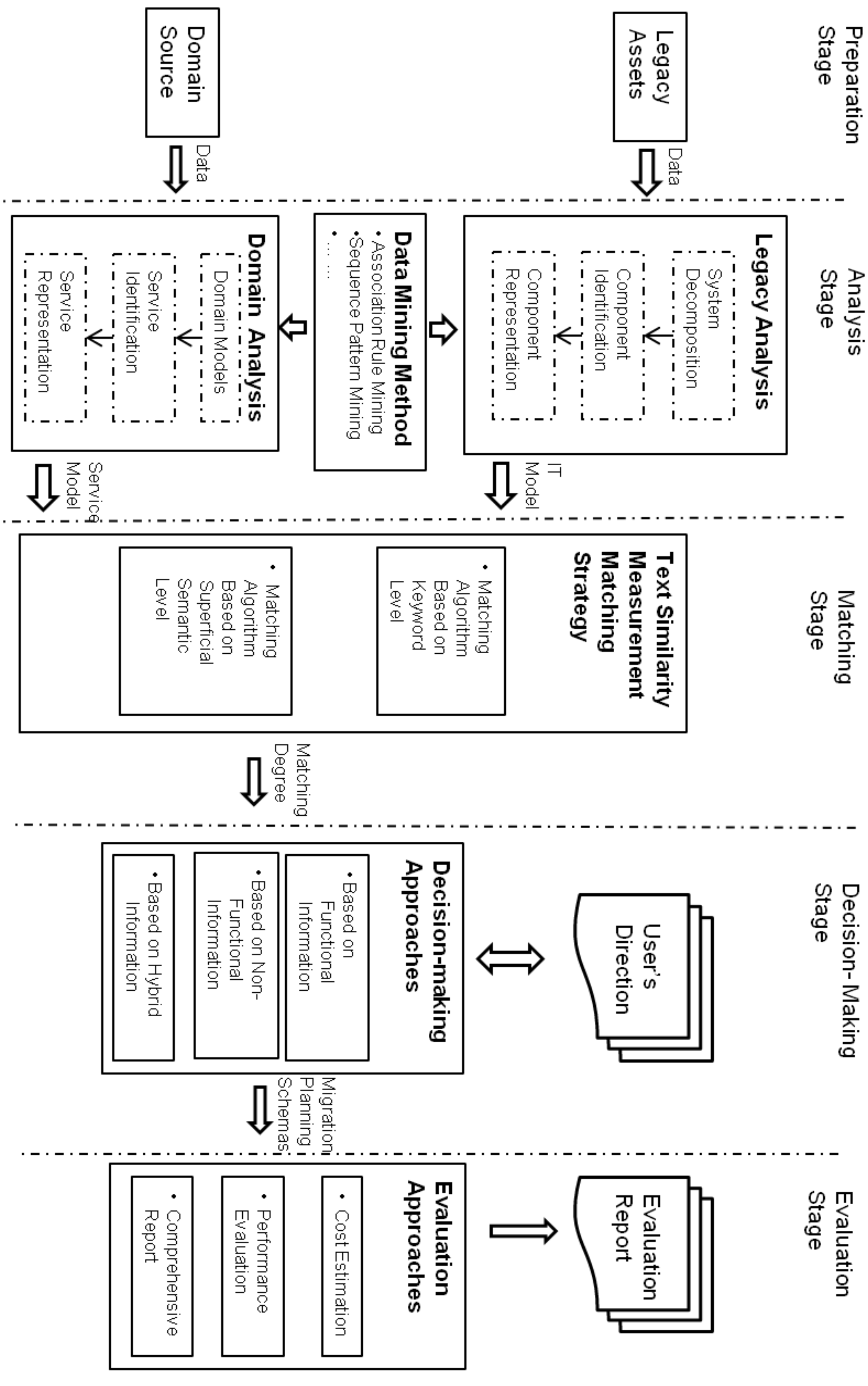


Figure 3-1. An SOA Migration Planning Approach.

3.2.1 Analysis of Legacy Assets and Domain Requirements with Data Mining Techniques

The purpose of legacy asset analysis and domain requirement analysis is to establish the Component model and Service model which can change legacy assets and domain requirements into the computational environment. The Component model (denoted as XML_{Legacy}) and Service model (denoted as XML_{Domain}) are the foundation for measuring the matching relationships between legacy components and domain services in the matching stage.

In legacy systems, a usage behaviour data resource can provide more help for legacy systems comprehension and domain business logic analysis from the angle of application practice. Thus, it is necessary to discover the hidden information from the collected usage behaviour data.

In this study, the DM techniques have been adopted to establish the Service model and Component model. Concretely, a sequence pattern mining algorithm for service/component identification and composition (see Chapter 4) and an association rules mining algorithm for determining the association relationships of components/services (see Chapter 5) are presented in this thesis.

3.2.2 Matching of Legacy Components and Domain Services with Text Retrieval Techniques

The matching stage is the key stage in this SOA migration planning approach. In this stage, legacy components in the Component model will meet domain services in the Service model. In order to decide which service can be implemented by reusing which components or the combination of them, the similarity between the domain service and legacy component(s) should be calculated. The concrete theoretical support frames include a probability theory, algebra theory, evidence theory and so on. In this study, the similarity between legacy components and services of the domain requirements will be saved as an XML file, denoted as XML_{Match}. The formula 3.1 expresses the relationship among XML_{Legacy} (Component model), XML_{Domain} (Service model), XML_{Match} and the matching algorithm.

$$XML_{Match} = Algorithm_{Match}(XML_{Domain},\ XML_{Legacy}) \qquad (3.1)$$

Chapter 6 describes more details on matching strategies based on keyword level and superficial semantic level by using the text similarity measurement method.

3.2.3 Decision-making of SOA Migration Planning Schemas

The main task in the decision-making stage is to create SOA migration planning schemas, which describe how to reuse identified components to implement services. Moreover, SOA migration planning schemas can provide planning and predictable functions to persons who are facing problems such as whether or not to trigger an SOA migration project. In order to assure the quality of SOA migration planning schemas, a decision-making stage is needed.

In this stage, the main solved questions include: should an existing component become a service? If not, how to integrate them together to compose a service?. There are three types of migration schemas between legacy components and domain services, namely, migration in whole, migration in part and migration nothing. The corresponding solutions are Wrapper, Modification and Redevelopment. The three solutions can work combinationally or individually. The decision-making method on adopting which solution or combination of solutions is subject to the matching degree between legacy components and domain services, as well as the user's direction.

Both some functional factors and some non-functional factors should be considered during the decision-making stage. If some conflicts exist among them, then work on trade-off or negotiations has to be done.

Sometimes, the migration process really needs human involvement although automation is many researchers' pursuit. The decision-making process is an uncertainty reasoning process. For example, what to do in the case of some IT components which match with the domain services 60% or 70%? The solution can be: integrating them into the new system or redeveloping them for the new system? Obviously, user's direction is necessary. In this study, the user's directions include functional direction and non-functional directions. The non-functional directions refer to cost aspect and performance aspect. Three types of user's direction are defined in the proposed approach: cost-first, function-first and performance-first. The cost-first direction means users prefer the cheapest cost to the others. The function-first direction means users prefer more functions to the others. The performance-first direction means users prefer the best performance to the others. According to the user's direction, the different thresholds will be determined for creating SOA migration planning schemas. In this study, the user's direction will be expressed as $XML_{\text{Direction}}$.

Information on cost estimation and performance evaluation can be of great benefit to the user's direction. In the beginning, user's direction is experiential and subjective. After obtaining the related information, user's direction can be more reasonable and functional. Therefore, SOA migration is an incremental process.

The output of the decision-making stage is an SOA migration planning schema. It is denoted as XML_{Schema}. The formula 3.2 expresses the relation among XML_{Match}, $XML_{\text{Direction}}$, XML_{Schema} and decision-making algorithm.

$$XML_{Schema} = Algorithm_{Decision-making}(XML_{Match}, \ XML_{Direction}) \quad (3.2)$$

Some investigations on solving software migration problems by using artificial intelligence theory and methods will be the key tasks in this stage. Two decision-making methods will be addressed in Chapter 7.

3.2.4 Evaluation of SOA Migration Planning Schemas

SOA migration planning schemas are created after passing the decision-making stage. In order to provide all useful information to domain and information technology decision-makers, further analysis and evaluation work on the created SOA migration planning schemas should be done. The evaluation approaches on the created SOA migration planning schemas should be investigated. An evaluation report that combines all useful information together will be presented. The formula 3.3 shows an example of the combined contents in an evaluation report.

$$XML_{EvaluationReport} = Algorithm_{Combination}(XML_{Schema}, XML_{Cost}, XML_{Performance}) \quad (3.3)$$

A simple evaluation method on SOA migration planning schema will be presented in Section 7.4.

Up to now, the framework of an SOA migration planning approach has been addressed.

In the following, the question “what kinds of legacy software system can be processed by the proposed approach?” will be answered.

The proposed approach is suitable for a legacy system that can be decomposed into different grained components. In this study, the granularities of components are divided into three categories: fine-grained, medium-grained and coarse-grained. According to this classification, there are three kinds of components. The application component that concerns multi-business logics and completes specific functionality is a coarse-grained component. The business component consists of fine-grained components and logic unit belongs to a medium-grained component. The atomic component that is with minimum logic belongs to a fine-grained component.

3.3 SOA Migration Toolkit --- SOAMT

In this thesis, an SOA Migration Toolkit (SOAMT) based on the techniques of data mining and text similarity measurement has been designed and implemented to support the SOA migration planning approach.

Figure 3-2 shows the main interface of SOAMT. There are five kinds of tool available in this toolkit, namely, a sequence mining tool, an association rule mining tool, a matching tool, a decision-making tool and an evaluation tool. More details on each tool will be described in the following Chapters.

A sequence mining tool named SEquence PAttern Miner (SEPAM) is introduced in Chapter 4 for supporting the proposed sequence pattern mining algorithm. In Chapter 5, an association rule mining tool named Association Rule Miner (ARM) is developed for supporting the proposed association rule mining algorithm. In Chapter 6, a matching tool is implemented based on the proposed matching strategies. In Chapter 7, a decision-making tool is implemented according to the proposed decision-making methods and an evaluation tool is implemented for evaluating the created SOA migration planning schemas. How to apply these tools to create SOA migration planning schemas will be presented in Chapter 8.

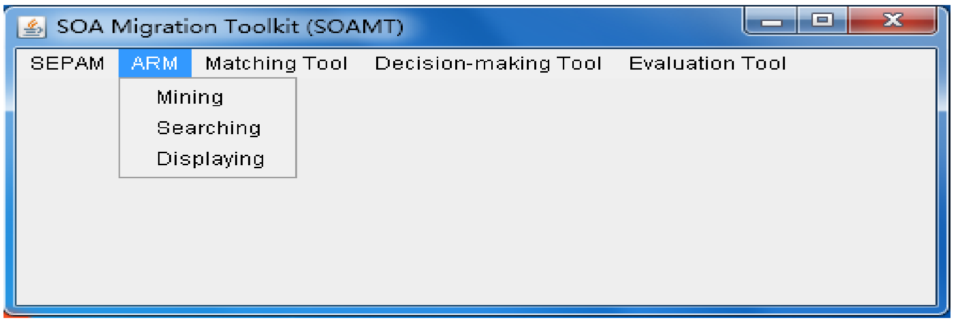


Figure 3-2. A Main Interface of SOAMT.

3.4 Summary

A general SOA migration planning approach has been proposed in this Chapter. This approach includes five stages, which is a preparation stage, an analysis stage, a matching stage, a decision-making stage and an evaluation stage.

- The main work of the preparation stage includes calling professional persons (application domain and development field) together; collecting sources on both sides of domain and legacy, double checking the financial state and so on.
- In the analysis stage, a Service model and a Component model are established. A Service model includes the work on domain analysis, service identification, final result's representation, etc. Some methods on domain analysis are the basis for establishing a Service model. A Component model includes the work on legacy analysis, component identification and final result's representation. Some related reverse engineering methods will be used to do more analysis. DM techniques are utilised to perfect the quality of a Service model and a Component model.
- The matching stage is the key stage in this SOA migration planning approach. In this stage, legacy components in the Component model will meet domain services in the Service model. The similarity between domain service and legacy component(s) will be calculated.
- The main work in the decision-making stage is to create SOA migration planning schemas, which describe how to reuse identified components to implement services. There are three types of migration schemas between legacy components and domain services. The corresponding solutions are Wrapper, Modification and

Redevelopment. The three solutions can work combinationally or individually. The decision-making method on adopting which solution or combination of solutions is subject to the matching degree between legacy components and domain services as well as the user’s direction.

- SOA migration planning schemas are created after passing the decision-making stage. In order to provide more information to domain and information technology decision-makers, further analysis and evaluation work on the created migration planning schema will be done in the SOA migration evaluation stage.

Chapter 4 Domain Service Model Establishment with Data Mining Techniques

Objectives

- To show the process for establishing a Service model.
- To propose a sequence pattern mining algorithm for service identification and composition.
- To evaluate the proposed sequence pattern mining algorithm.
- To present a sequence pattern mining tool named SEquence PAttern Miner (SEPAM).

4.1 Service Model Establishment

4.1.1 The Process of Establishing a Service Model

In this study, the establishment of a Service model includes the work on domain analysis, service identification, the representation of final results, etc. The process of establishing a Service model consists of three phases: first, collecting the related information source; secondly, adopting the suitable methods and techniques to solve the concerns which is the key phase of this process; and finally, selecting a knowledge representation method for the final results. Some methods on domain analysis are the foundation for establishing a Service model. Figure 4-1 shows the process for establishing a Service model.

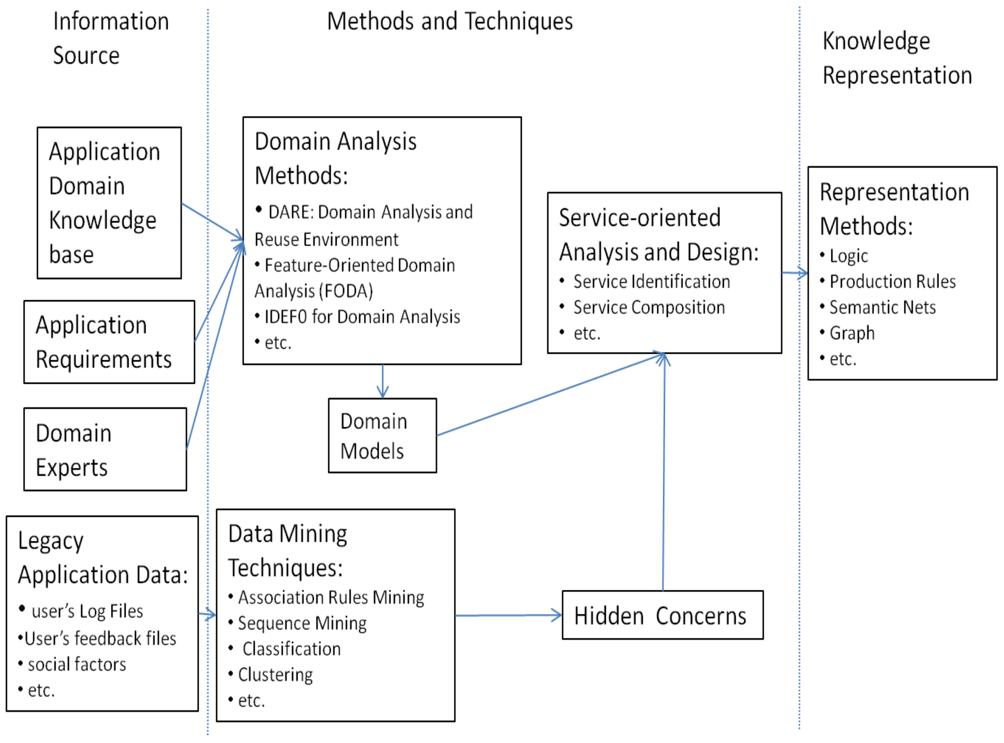


Figure 4-1. The Process of Establishing a Service Model.

1. Information source.

Gathering information for the domain analysis is the first step of establishing a Service model. In this study, the selected information sources for establishing a Service model include application domain knowledge base, existing application data, domain experts, etc. The application domain knowledge base consists of domain knowledge, theories, methods, techniques, models and so on. The existing application data refers to application requirements, log files and so on.

2. Domain analysis methods.

Domain analysis refers to distinguishing and determining a collection of reusable assets for domain specific systems. The process of domain analysis includes identifying sub-domains and analysing the identified sub-domains. Domain experts and a domain knowledge base are important sources for analysing application requirements. Generally, the results of domain analysis are domain models with a wider business context, which are the core for service identification. Some existing domain analysis methods can be used in this part.

3. Data mining techniques.

Data mining techniques are utilised to mine the implicit patterns from the application data (business level). The mined information can be of great benefit to establishing a Service model. The concrete data mining techniques include sequence pattern mining, association rules mining, classification, clustering and so on.

4. Service identification and composition.

Service identification in a service domain includes the business-driven manner for value-chain analysis and a use case driven manner for composite applications. Service identification can normally be implemented in a top-down manner, which decomposes a business domain into functional areas, sub-systems, processes, sub-processes, use cases, etc. The abstract level that services will be defined should be higher than the abstract level that objects will be defined. In this case, a service definition can be mapped into an object-oriented system, such as J2EE, .NET, and the like.

Some attention should be paid to service granularity since it is very important for services' flexibility and reuse. In this study, the main purpose of service identification and composition is for SOA migration. Thus, the fine-grain services are preferred since they are good for matching with legacy components. A lot of different orchestrations can be implemented by reusing some fine-grain services. In addition, the granularity of service identification and composition should be cooperated with the granularity of legacy decomposition for better component-to-service matching.

Some existing methods on service identification and composition can be applied in this part. Service description is based on *XML* technology. The data type and structure of services can be expressed by *XML* schemas.

5. Knowledge representation methods.

In order to reuse legacy assets maximally, not only qualitative research but also quantitative research are needed in this study. Thus, the identified services should be machine-readable. The purpose of establishing knowledge representation is to put the identified services into the computation environment, which is the foundation for measuring the matching relationship between services and components.

4.1.2 Hierarchical Directed Acyclic Graph (HDAG)

Through analysing features of legacy components and domain services, it is found that the hierarchical representation method is preferred since different granularities in legacy components and domain services exist; the directed graph representation method is a better one since some relationships exist among components or among services; the cyclic graph is not preferred since it is difficult to process. Therefore, in this study, a Hierarchical Directed Acyclic Graph (HDAG) is adopted to represent domain components and domain services. Each node in a HDAG is a composite node, which is composed of some items. This richer representation is extremely useful to improve the performance of similarity measurement between legacy components and domain services.

Figure 4-2 presents an example of HDAG. In this example, there are three levels in the HDAG; there are two nodes in level1; there are four items in each node; there are some feature descriptions in each item; etc.

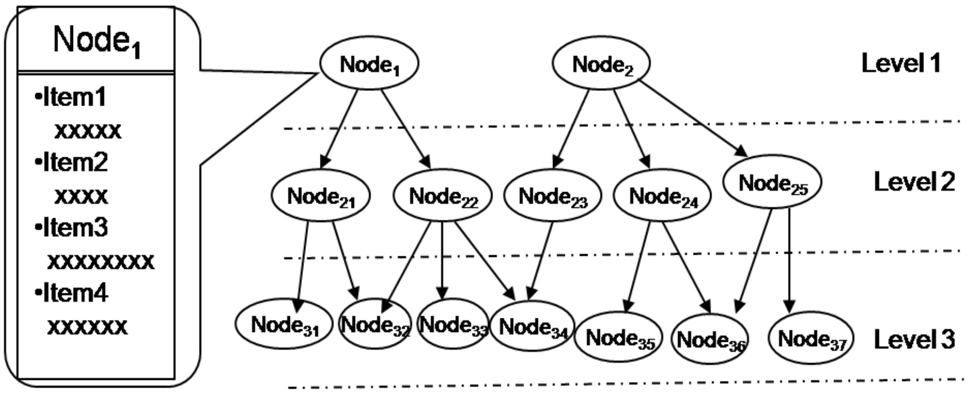


Figure 4-2. An Example of Hierarchical Directed Acyclic Graph (HDAG).

4.1.3 The Representation of a Service Model

Domain services are represented as a domain HDAG, which includes three levels; namely, application level, business process level and service level (fine, medium and coarse). Each node in a domain HDAG can represent an atomic service, a composite service, a business process or an application, etc. It should be a composite node. In the support of algebra theory, a node in a domain HDAG shown in Figure 4-3 is defined as follows.

Definition 4.1 Each *Node* in a domain HDAG is a vector, which is composed of

$$Item_{\text{father}},\ Item_{\text{function}},\ Item_{\text{data}} \text{ and } Item_{\text{son}}\ .$$

where, $Item_{\text{father}}$ refers to the father node of a node; $Item_{\text{function}}$ refers to the information from functional description; $Item_{\text{data}}$ refers to the information from data analysis; $Item_{\text{son}}$ refers to the son node of a node.

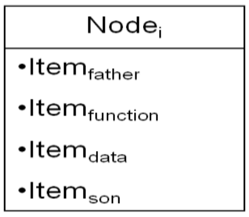


Figure 4-3. A Node in a Domain HDAG.

Figure 4-4 presents an example of Service model that are represented by a HDAG. The nodes S_{21} and S_{23} are created by the data mining technique.

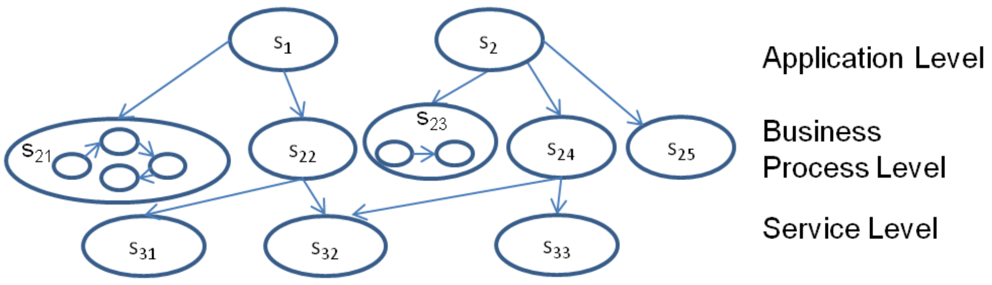


Figure 4-4. An Example of Service Model Represented by HDAG.

The representation method of a Service model and a Component model should be identical since these two models will meet in the matching stage for a further process.

4.2 A Proposed Sequence Pattern Mining Algorithm for Service Composition

Sequence pattern mining can be utilised for service identification and composition in SOA migration situation. The mined frequent patterns can be services or composite services in SOA.

4.2.1 Concepts and Notations

Some useful information can be hidden in sequences data. The common sequence data includes web click streams, shopping sequences, DNA sequences and so on. In this section, some concepts, notations and relations among them are established for further discussion.

4.2.1.1 Sequences and Items

For a set *T*, its element number is called its size and denoted by |*T*|. Given a non-empty set *T*, elements in it are called *items* and set *T* is called *transaction set*. A non-empty subset of transaction set *T* is called *itemset*.

A *sequence s* over *T* is an ordered list of non-empty subsets (terms) of *T*, expressed as $s = A_1A_2...A_n...A_N$, where $\emptyset \subset A_n \subseteq T, 0 < |A_n| \leq |T|$, for n = 1, 2, ... , N and $N > 0$. An item can occur only once in a term of a sequence, but can occur multiple times in different elements. An itemset is considered to be a sequence with a single term [69]. Given a sequence $s = A_1A_2...A_\mathrm{n}...A_\mathrm{N}$, *N* is called the length of sequence *s* and denoted as |*s*| = *N*.

The set of sequences over *T* is denoted by S_T and the set of sequences over *T* with length *N* is denoted by $S_{T,N}$. There exist $|S_{T,N}| = (2^{|T|} - 1)^N$ since each A_n $(n = 1,2,\ldots,N)$ can take one of $(2^{|T|} - 1)$ non-empty subsets of *T*.

A non-empty set composed of a group of sequences over *T* is named as *sequence database* over *T*. Sequence pattern mining means to discover knowledge from sequence database. An example is given to explain the concepts mentioned above.

Suppose that transaction set *T* = {Service1, Service2, Service3}.

The itemsets of *T* are 7 non-empty subsets of *T*, namely, {Service1}, {Service2}, {Service3}, {Service1, Service2}, {Service1, Service3}, {Service2, Service3}, {Service1, Service2, Service3}.

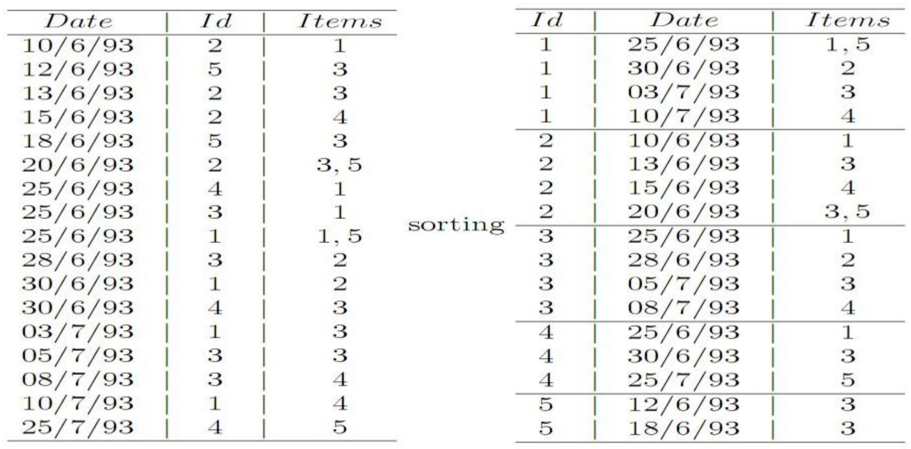
The set of sequences over *T* with length 2 is:

$$S_{T,2} = \{A_iA_j | \mathrm{A}_i, \mathrm{A}_j \in 2^{|T|}, \text{and } \mathrm{A}_i, \mathrm{A}_j \neq \emptyset \}$$

$$|S_{T,2}| = \left(2^{|T|} - 1\right)^2 = 7^2 = 49$$

Namely, {Service1}{Service1}, {Service1,Service2,Service3}{Service1}, {Service2}{Service3}, {Service1}{Service3}, {Service1,Service2,Service3} {Service1, Service2,Service3}, etc. Here, an itemset can occur in the same sequence more than one time.

With the support of information technology, much more original data are collected each day. After doing pre-process, sequences databases can be obtained. For example, Figure 4-5 presents an original transaction database and its variance (sorting according to customer Id and Date).



Date	*Id*	*Items*
10/6/93	2	1
12/6/93	5	3
13/6/93	2	3
15/6/93	2	4
18/6/93	5	3
20/6/93	2	3, 5
25/6/93	4	1
25/6/93	3	1
25/6/93	1	1, 5
28/6/93	3	2
30/6/93	1	2
30/6/93	4	3
03/7/93	1	3
05/7/93	3	3
08/7/93	3	4
10/7/93	1	4
25/7/93	4	5

sorting

Id	*Date*	*Items*
1	25/6/93	1, 5
1	30/6/93	2
1	03/7/93	3
1	10/7/93	4
2	10/6/93	1
2	13/6/93	3
2	15/6/93	4
2	20/6/93	3, 5
3	25/6/93	1
3	28/6/93	2
3	05/7/93	3
3	08/7/93	4
4	25/6/93	1
4	30/6/93	3
4	25/7/93	5
5	12/6/93	3
5	18/6/93	3

Figure 4-5. A Transaction Database.

Through combination, Table 4-1 shows a sequence database *U* based on the transaction database shown in Figure 4-5, where *T* = {1, 2, 3, 4, 5}.

Table 4-1. A Sequence Database *U*.

Customers *i*	Sequences u_i	Length $l_i = \lvert u_i \rvert$
1	{1,5}{ 2}{3}{4}	4
2	{1}{3}{4}{3,5}	4
3	{1}{2}{3}{4}	4
4	{1}{3}{5}	3
5	{3}{3}	2

In the Sub-section 4.2.3, this sequence database *U* will be taken as an example to run the proposed sequence pattern mining algorithms.

4.2.1.2 Containing Relations between Sequences

Given two sequences a and b, $a = A_1A_2 \dots A_n$ $(n > 0)$, $b = B_1B_2 \dots B_m$ $(m > 0)$, it is said that sequence a is s*upported by* sequence b; or sequence a is a *sub-sequence* of sequence b; or sequence b is a *super* sequence of sequence a if there exist n integers $1 \le i_1 < i_2 < \dots < i_n \le m$ such that,

$$A_1 \subseteq B_{i_1}, A_2 \subseteq B_{i_2}, \dots, A_n \subseteq B_{i_n}$$

Moreover, it can be denoted by $a \le b$. For example, there exist $u_5 < u_2$ and $u_4 < u_2$ in sequence database *U* (see Table 4-1), namely,

$$\{3\}\{3\} < \{1\}\{3\}\{4\}\{3,5\}; \quad \{1\}\{3\}\{5\} < \{1\}\{3\}\{4\}\{3,5\}$$

4.2.1.3 Patterns and Their Support Sequence Groups

Given a sequence $a = A_1A_2 \dots A_n \dots A_N$ of sequence database *U*, let *w* be any sub-sequence of *a*. Sub-sequence *w* is called a *pattern* of sequence *a*. By the definition of patterns, it is known that a pattern is also a sequence. Thus, pattern *w* has *length* |*w*|. A pattern with length 1 is called as an itemset of *U*.

Given a non-empty sequence database $U = \{u_1, u_2, \dots, u_k \dots u_K\}, 0 < k \le K = |U|$, the pattern of any sequence in *U* are called *patterns of U*. Let *L* be the maximal length of sequences of *U*, namely, $L = max\,(|u_1|, |u_2|, \dots, |u_K|)$. For any pattern *w*, there is $0 < |w| \le L$.

A particular pattern can be contained (or can "co-occur") in many sequences of a sequence database as their common sub-sequence. For a pattern *w* of *U*, it needs to know which sequences in *U* containing *w*. The set of sequences in *U* containing *w* is $M = \{u | u \in U, w \le u\}$, denoted by w^U. Subset w^U consists of sequences in group *U* in which sequence *w* is contained, and is called the *support* or *occurrence group* of pattern *w*.

The set of patterns over *U* with support number being not less than *h* is denoted by,

$$M(U, h) = \{w | |w^U| \ge h\}.$$

The set of patterns over *U* with length *l* and support number being not less than *h* is denoted by,

$$M(U,h,l) = \{w \mid |w^U| \geq h \text{ and } |w| = l\}.$$

The set of patterns over U with support number h is denoted by,

$$G(U,h) = \{w \mid |w^U| = h\}.$$

The set of patterns over U with length l and support number h is denoted by,

$$G(U,h,l) = \{w \mid |w^U| = h \text{ and } |w| = l\}.$$

There are,

$$G(U,h) = M(U,h) - M(U,h+1), h = 1,2,\dots,|U|-1;$$

$$G(U,|U|) = M(U,|U|);$$

$$G(U,h) = G(U,h,1) \cup G(U,h,2) \cup \dots \cup G(U,h,L);$$

$$M(U,h) = M(U,h,1) \cup M(U,h,2) \cup \dots \cup M(U,h,L);$$

$$M(U,1) \supseteq M(U,2) \supseteq \cdots \supseteq M(U,|U|).$$

Given a non-empty sequence database $U = \{u_1, u_2, \dots, u_k \dots u_K\}$, the set $maxM(U,h)$ of maximal patterns with support number being not less than *min-support* number h is called *motifs* over U.

4.2.2 Theorems and Algorithms

In this Sub-section, some theorems and algorithm are established for finding motifs with any *min-support* number h from sequence databases.

4.2.2.1 Motifs with Min-support $h = 1$ (maxM$(U, h = 1)$)

1-length patterns are called *units* or *unit patterns*. Notice that a 1-length pattern $w \in (2^T - \{\emptyset\})$ is in $M(U,h,1)$ if and only if $|w^U| \geq h$. This fact can be used to find $M(U,h,1)$, which is useful for finding $M(U, h=2)$.

In fact, there is a simple way to find motifs with *min-support* $h = 1$. Motifs with *min-support* 1 are the maximal sequences of $M(U,1)$, denoted by $maxM(U,1)$.

Theorem 4.1 Given a sequence database *U*, motifs with *min-support* 1 are the maximal sequences of *U*, and vice versa. That is, $maxM(U,1) = maxU$.

Proof.

1. $U \subseteq M(U,1)$; suppose $u \in U$ to prove $u \in M(U,1)$.

   Let $u \in U$, there is $u \leq u$ according to the definition of containing relation between patterns. Moreover, according to the definition of support group, there exist $u \in u^U$ and $|u^U| \geq 1$. Thus, $u \in M(U,1)$.

2. $maxU \subseteq maxM(U,1)$; suppose $u \in maxU$ to prove $u \in maxM(U,1)$.

   (2.1) $u \in M(U,1)$

   From $u \in maxU$, there is $u \in U$; by (1), there is $u \in M(U,1)$.

   (2.2) $u \in maxM(U,1)$

   Suppose $u \notin maxM(U,1)$, there exists $w \in M(U,1)$ and $w > u$; there is $u' \in U$ such that $w \leq u'$ and $w > u$; there exists $u' \in U$ such that $u < w \leq u'$; there is $u' \in U$ such that $u < u'$. This is contrary to assumption $u \in maxU$. Thus, $u \in maxM(U,1)$.

3. $maxM(U,1) \subseteq maxU$. Assume $w \in maxM(U,1)$ to prove $w \in maxU$.

   (3.1) $w \in U$

   From $w \in maxM(U,1)$, there is $w \in M(U,1)$ and there exists $u \in U$ such that $w \leq u$. It can prove that $w = u$ and so $w \in U$. Otherwise, $w < u$, by (1) there is $u \in M(U,1)$. It is contrary to assumption $w \in maxM(U,1)$.

   (3.2) $w \in maxU$

   If $w \notin maxU$, there exists $u \in U$ such that $w < u$. This is contrary to assumption $w \in maxM(U,1)$. Thus, $w \in maxU$.

The proof is completed.

By theorem 4.1, the motifs $maxM(U,1)$ over sequence database U can be found immediately by finding $maxU$ from U itself.

4.2.2.2 Patterns with Min-support $h = 2$ ($\mathbf{M}(U, h = 2)$)

The set of patterns with support number being not less than 2 over U is denoted by

$M(U,2)$. Generally, there are the following theorems.

Theorem 4.2 Every subsequence of h-pattern is also an h-pattern. That is, if $w \in M(U,h)$ and $y \leq w$ then $y \in M(U,h)$.

Proof.

(1) If $y \leq w$ then $w^U \subseteq y^U$. Suppose $u \in w^U$ to prove $u \in y^U$.

Since $u \in w^U$, there is $w \leq u$; since $y \leq w$ and $w \leq u$, there is $y \leq u$; since $y \leq u$, there is $u \in y^U$.

(2) By (1), there is $|w^U| \leq |y^U|$.

(3) Since $w \in M(U,h)$, $|w^U| \geq h$ can be proved. By (2), there is $|y^U| \geq h$, and, $y \in M(U,h)$.

The proof is completed.

Theorem 4.3 If $w \in M\ (U,h)$ and $|w| > 1$ then w can be expressed as a concatenation of two patterns x and z, where x has *min-support* h and length $|w|-1$ and z has *min-support* h and length 1. That is, $w= xz$, where $x \in M(U,h,|w|-1)$ and $z \in M(U,h,1)$.

Proof.

Since $|w| > 1$, a pattern z with length 1 can be cut from the right, such that $w = xz$, where $x = |w|-1$, $|z| = 1$ and $x, z \leq w$. By theorem 4.2 and $w \in M(U,h)$, there are $x, z \in M(U,h)$, more precisely, $x \in M(U,h,|w|-1)$ and $z \in M(U,h,1)$.

The proof is completed.

Now, considering a case of *min-support* 2 and discussing how to find $M(U,2)$. Then theorem 4.3 can be expressed as follows:

$$M(U,2,l) \subseteq M(U,2,l-1)M(U,2,1) \qquad \text{when } l > 1$$

Therefore, $M(U,2,l)$ has a set of candidates $M(U,2,l-1)M(U,2,1)$, which is called *the set of right candidates* and denoted as:

$$C^R(U,2,l) = M(U,2,l-1)M(U,2,1) \qquad \text{when } l > 1$$

Similarly, there is,

$$M(U,2,l) \subseteq M(U,2{,}1)M(U,2,l-1) \quad \text{when } l > 1$$

Therefore, $M(U,2,l)$ has a set of candidates $M(U,2{,}1)M(U,2,l-1)$, which is called *the set of left candidates* and denoted as:

$$C^L(U,2,l) = M(U,2{,}1)M(U,2,l-1) \quad \text{when } l > 1$$

In the following, the set of candidates will be improved and reduced.

For length =2 or l = 2, there are,

$$C^R(U,2{,}2) = M(U,2,1)M(U,2{,}1)$$

$$C^L(U,2{,}2) = M(U,2{,}1)M(U,2,1)$$

The set of right candidates is the same as the set of left candidates.

For length >2 or l > 2, since

$$M(U,2,l) \subseteq M(U,2{,}1)M(U,2,l-1)$$

There is,

$M(U,2,l) \subseteq M(U,2{,}1)M(U,2,l-1)_{[first\ l-2\ terms]}M(U,2,l-1)_{[last\ term]}$ by enlarging $M(U,2,l-1)$ to concatenation $M(U,2,l-1)_{[f\ irst\ l-2\ terms]}M(U,2,l-1)_{[last\ term]}$.

However, the $l-1$ terms in concatenation $M(U,2{,}1)M(U,2,l-1)_{[first\ l-2\ terms]}$ as the first $l-1$ term candidates of $M(U,2,l)$, have support being no less than 2 by theorem 5.5 and should be in $M(U,2,l)$. Thus, the following theorem can be proved.

Theorem 4.4 If l > 2, then $M(U,2,l) \subseteq M(U,2,l-1)M(U,2,l-1)_{[\text{last term}]}$.

Theorem 4.4 can be used to find pattern $w \in M(U,2,l)$ by concatenation $M(U,2,l-1)$ and $M(U,2,l-1)_{[\text{last term}]}$. Here the set of unit patterns $M(U,2,l-1)_{[\text{last term}]}$ changes as length l. The candidate set generated from pattern $w = xz$ where $x \in M(U,2,l-1)$ and $z \in M(U,2,1)_{[\text{last term}]}$ is called *the set of dynamic candidates* and denoted by:

$$C^D(U,2,l) = M(U,2,l-1)M(U,2{,}1)_{[\text{last term}]} \quad \text{when } l > 2$$

Notice that changing the unit pattern candidate term in $M(U,2{,}1)$ to $M(U,2,l-$

$1)_{[\text{last term}]}$ reduces the candidate set since $M(U,2,l-1)_{[\text{last term}]} \subseteq M(U,2,1)$.

The detailed algorithm for finding the set $M(U,2)$ of patterns with *min-support* 2 is as follows:

Algorithm *SH₂A* for finding $M(U,2)$ // **S**equence pattern with **h**=**2** **A**lgorithm

```
SH_2A1. Find all 1-length (or unit) patterns $M(U,2,1)$ from sequence database $U$ by
        checking its occurring or support number $h \geq 2$.   //find $M(U,2,1)$
SH_2A2. For each $x \in M(U,2,1)$ do      //  Find $M(U,2,2)$ from $M(U,2,1)$
            For each $z \in M(U,2,1)$ do
                $x$ right concatenates $z$ to generate $w = xz$, and put $w$ to the candidate set
                                                   // obtain candidate set with length 2
            if $(|w^U| \geq 2)$ then $w \in M(U,2,2)$ //check support number of the candidate
                                                   patterns and get $M(U,2,2)$
SH_2A3. For $(l = 3; M(U,2,l-1) \neq \emptyset;\ l++)$ do     // Find $M(U,2,l)$ from
                                                   $M(U,2,l-1), l > 2$.
            For each $x \in M(U,2,l-1)$ do
                For each $z \in M(U,2,l-1)_{[\text{last term}]}$ do
                    $x$ right concatenates $z$ to generate $w = xz$, and put $w$ to the candidate set
                                                   // obtain candidate set with length l
            if $(|w^U| \geq 2)$ then $w \in M(U,2,l)$ //check the support number of the
                                                   candidate patterns and get $M(U,2,l)$
SH_2A4. Output the set $M(U,2)$ of patterns with min-support 2.   //Answer
            $M(U,2) = M(U,2,1) \cup M(U,2,2) \cup \ldots \cup M(U,2,l)$
```

The algorithm ***SH₂A*** is completed.

4.2.2.3 Patterns with Min-support $h > 2$ ($M(U, h > 2)$)

Patterns $M(U,h>2)$ can be found by the following formula,

$$M(U, h+1) = M(U, h) - G(U, h) \quad \text{for } h = 1,2,3, \dots, |U| - 1$$

Therefore, the following algorithm SHG_2A can be concluded for finding patterns with *min-support h* > 2.

Algorithm $\boldsymbol{SHG_2A}$ // Sequence pattern with **h G**reater than **2 A**lgorithm

```
$SHG_2A$ 1. Call algorithm $SH_2A$.
$SHG_2A$ 2. For $(h = 2; M(U, h+1) \neq \emptyset; h++)$ do
                $M(U, h+1) = M(U, h) - G(U, h)$
```

4.2.2.4 Algorithm for Finding Motifs (maxM$(\boldsymbol{U}, \boldsymbol{h} \geq \mathbf{2})$)

Let *V* be a non-empty partially ordered set by less than (<) relation. An element *w* of set *V* is a maximum in *V* if there exist no $v \in V - w$ such that $v > w$.

Algorithm *SMX* for finding *maxV* from *V*

```
$maxV = \emptyset$
For each $w \in V$
    For each $v \in V - w$
        { counter=0
          If $v > w$ then counter++}
If counter==0, then $(maxV) = (maxV) \cup \{w\}$ //there exist no $v \in V - w$ such
                that $v > w$. So w is a maximum in V and put $w \in maxV$.
```

The algorithm ***SMX*** is completed.

4.2.3 Examples

In order to understand well the proposed theorems and algorithms, a sequence database *U* shown in Table 4-1 is taken as an example to run them.

4.2.3.1 For Finding maxM$(\boldsymbol{U}, \boldsymbol{h} = \mathbf{1})$

Taking sequence database *U* (see Table 4-1) as an example, there are $U = \{u_1, u_2, u_3, u_4, u_5\}$. Patterns with length 1 and *min-support* 2 over *U* shown in

Table 4-2 is useful for finding $M(U, h = 2)$, which is denoted by $M(U, 2,1)$.

Table 4-2. Patterns with Length 1 and *Min-support* 2 over *U*.

1-length patterns w	Support Group w^U	Support Number $\lvert w^U \rvert$
$\{1\}$	$\{u_1, u_2, u_3, u_4\}$	4
$\{2\}$	$\{u_1, u_3\}$	2
$\{3\}$	$\{u_1, u_2, u_3, u_4, u_5\}$	5
$\{4\}$	$\{u_1, u_2, u_3\}$	3
$\{5\}$	$\{u_1, u_2, u_4\}$	3

According to theorem 4.1, it is easier to find $maxM(U, 1)$. Since $u_3 < u_1$, $u_4 < u_2$, $u_5 < u_2$, there is $maxM(U, 1) = maxU = \max\{u1, u2, u3, u4, u5\} = \{u1, u2\}$.

4.2.3.2 For Finding $\mathbf{M}(\boldsymbol{U}, \boldsymbol{h} = \mathbf{2})$

Taking sequence database *U* as an example to run algorithm *SH₂A*.

1. Finding $M(U, 2, 2)$.

   $M(U, 2,1)$ shown in Table 4-2 can be obtained from $M(U, 1,1)$ by running step *SH₂A*1 of algorithm *SH₂A*. $M(U, 2, 2)$ shown in Table 4-3 can be obtained from $M(U, 2,1)$ by running step *SH₂A*2 of algorithm *SH₂A*.

Table 4-3. Patterns with Length 2 and *Min-support* 2 over *U*.

2-length patterns w	Support Group w^U	Support Number $\lvert w^U \rvert$
$\{1\}\{2\}$	$\{u_1, u_3\}$	2
$\{1\}\{3\}$	$\{u_1, u_2, u_3, u_4\}$	4
$\{1\}\{4\}$	$\{u_1, u_2, u_3\}$	3
$\{1\}\{5\}$	$\{u_2, u_4\}$	2
$\{2\}\{3\}$	$\{u_1, u_3\}$	2
$\{2\}\{4\}$	$\{u_1, u_3\}$	2
$\{3\}\{3\}$	$\{u_2, u_5\}$	2
$\{3\}\{4\}$	$\{u_1, u_2, u_3\}$	3
$\{3\}\{5\}$	$\{u_2, u_4\}$	2

2. Finding $M(U,2,3)$.

According to $M(U,2,2)$, there is $M(U,2,2)_{[\text{last term}]} = \{\{2\},\{3\},\{4\},\{5\}\}$. Then, $M(U,2,3)$ shown in Table 4-4 can be obtained by running step *SH*$_2$*A*3 of algorithm *SH*$_2$*A* at $\mathrm{l}=3$.

Table 4-4. Patterns with Length 3 and *Min-support* 2 over *U*.

3-length patterns w	Support Group w^U	Support Number $\|w^U\|$
$\{1\}\{2\}\{3\}$	$\{u_1, u_3\}$	2
$\{1\}\{2\}\{4\}$	$\{u_1,u_3\}$	2
$\{1\}\{3\}\{4\}$	$\{u_1, u_2,u_3\}$	3
$\{1\}\{3\}\{5\}$	$\{u_2, u_4\}$	2
$\{2\}\{3\}\{4\}$	$\{u_1, u_3\}$	2

3. Finding $M(U,2,4)$.

According to $M(U,2,3)_{[\text{last term}]} = \{\{3\},\{4\},\{5\}\}$, there are candidate set with length 4,namely, C_4,

$$C_4 = \{\ \{1\}\{2\}\{3\}\{3\},\{1\}\{2\}\{3\}\{4\},\{1\}\{2\}\{3\}\{5\},\{1\}\{2\}\{4\}\{3\},\{1\}\{2\}\{4\}\{4\},$$
$$\{1\}\{2\}\{4\}\{5\},\{1\}\{3\}\{4\}\{3\},\{1\}\{3\}\{4\}\{4\},\{1\}\{3\}\{4\}\{5\},\{1\}\{3\}\{5\}\{3\},$$
$$\{1\}\{3\}\{5\}\{4\},\{1\}\{3\}\{5\}\{5\},\{2\}\{3\}\{4\}\{3\},\{2\}\{3\}\{4\}\{4\},\{2\}\{3\}\{4\}\{5\}\ \}$$

and $M(U,2,4)$ shown in Table 4-5 by running step *SH*$_2$*A*3 of algorithm *SH*$_2$*A* at $\mathrm{l}=4$.

Table 4-5. Patterns with Length 4 and *Min-support* 2 over *U*.

4-length patterns w	Support Group w^U	Support Number $\|w^U\|$
$\{1\}\{2\}\{3\}\{4\}$	$\{u_1, u_3\}$	2

4. Finding $M(U,2,5)$.

From $M(U,2,4)_{[\text{last term}]} = \{\{4\}\}$, there are $C_5 = \{\{1\}\{2\}\{3\}\{4\}\{4\}\}$; $M(U,2{,}5) = \emptyset$ by running step *SH*$_2$*A*3 of algorithm *SH*$_2$*A* at $\mathrm{l}=5$. Finally, step *SH*$_2$*A*3 of algorithm *SH*$_2$*A* terminates at $\mathrm{l}=6$.

5. Finding $M(U,2)$

$M(U,2)$ can be found by running step *SH*2*A*4 of algorithm *SH*2*A*. Namely,

$$M(U,2) = M(U,2,1) \cup M(U,2,2) \cup \ldots \cup M(U,2,l)$$

The result is shown in Table 4-6.

Table 4-6. Patterns with *Min-support* 2 over *U*.

2-length patterns *w*	Support Group w^U	Support Number $\|w^U\|$
{1}	$\{u_1, u_2, u_3, u_4\}$	4
{2}	$\{u_1, u_3\}$	2
{3}	$\{u_1, u_2, u_3, u_4, u_5\}$	5
{4}	$\{u_1, u_2, u_3\}$	3
{5}	$\{u_1, u_2, u_4\}$	3
{1}{2}	$\{u_1, u_3\}$	2
{1}{3}	$\{u_1, u_2, u_3, u_4\}$	4
{1}{4}	$\{u_1, u_2, u_3\}$	3
{1}{5}	$\{u_2, u_4\}$	2
{2}{3}	$\{u_1, u_3\}$	2
{2}{4}	$\{u_1, u_3\}$	2
{3}{3}	$\{u_2, u_5\}$	2
{3}{4}	$\{u_1, u_2, u_3\}$	3
{3}{5}	$\{u_2, u_4\}$	2
{1}{2}{3}	$\{u_1, u_3\}$	2
{1}{2}{4}	$\{u_1, u_3\}$	2
{1}{3}{4}	$\{u_1, u_2, u_3\}$	3
{1}{3}{5}	$\{u_2, u_4\}$	2
{2}{3}{4}	$\{u_1, u_3\}$	2
{1}{2}{3}{4}	$\{u_1, u_3\}$	2

4.2.3.3 For Finding $\mathbf{M(U, h > 2)}$

$M(U, 3)$ shown in Table 4-7, $M(U, 4)$ shown in Table 4-8, and $M(U, 5)$ shown in Table 4-9 can be obtained by running step SHG_2A2 of algorithm SHG_2A.

Table 4-7. Patterns with *Min-support* 3 over *U*.

patterns w	Support Group w^U	Support Number $\lvert w^U \rvert$
$\{1\}$	$\{u_1, u_2, u_3, u_4\}$	4
$\{3\}$	$\{u_1, u_2, u_3, u_4, u_5\}$	5
$\{4\}$	$\{u_1, u_2, u_3\}$	3
$\{5\}$	$\{u_1, u_2, u_4\}$	3
$\{1\}\{3\}$	$\{u_1, u_2, u_3, u_4\}$	4
$\{1\}\{4\}$	$\{u_1, u_2, u_3\}$	3
$\{3\}\{4\}$	$\{u_1, u_2, u_3\}$	3
$\{1\}\{3\}\{4\}$	$\{u_1, u_2, u_3\}$	3

Table 4-8. Patterns with *Min-support* 4 over *U*.

patterns w	Support Group w^U	Support Number $\lvert w^U \rvert$
$\{1\}$	$\{u_1, u_2, u_3, u_4\}$	4
$\{3\}$	$\{u_1, u_2, u_3, u_4, u_5\}$	5
$\{1\}\{3\}$	$\{u_1, u_2, u_3, u_4\}$	4

Table 4-9. Patterns with *Min-support* 5 over *U*.

patterns w	Support Group w^U	Support Number $\lvert w^U \rvert$
$\{3\}$	$\{u_1, u_2, u_3, u_4, u_5\}$	5

4.2.3.4 For Finding Motifs

By running algorithm SMX, $maxM(U, 2)$ shown in Table 4-10, $maxM(U, 3)$ shown in Table 4-11, $maxM(U, 4)$ shown in Table 4-12 and $maxM(U, 5)$ shown in Table 4-13 can be found.

For example, by running the proposed sequence mining algorithm on the legacy system application data, the composite services and business processes shown in Figure 4-6 can be discovered.

Table 4-10. *maxM*(*U*, 2).

patterns w	Support Group w^U	Support Number $\|w^U\|$
{3}{3}	$\{u_2, u_5\}$	2
{1}{3}{5}	$\{u_2, u_4\}$	2
{1}{2}{3}{4}	$\{u_1, u_3\}$	2

Table 4-11. *maxM*(*U*,3).

patterns w	Support Group w^U	Support Number $\|w^U\|$
{5}	$\{u_1, u_2, u_4\}$	3
{1}{3}{4}	$\{u_1, u_2, u_3\}$	3

Table 4-12. *maxM*(*U*,4).

patterns w	Support Group w^U	Support Number $\|w^U\|$
{1}{3}	$\{u_1, u_2, u_3, u_4\}$	4

Table 4-13. *maxM*(*U*,5).

patterns w	Support Group w^U	Support Number $\|w^U\|$
{3}	$\{u_1, u_2, u_3, u_4, u_5\}$	5

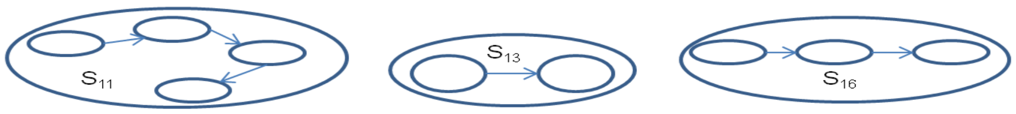


Figure 4-6. The Motifs Mined by the Proposed Sequence Mining Algorithm.

4.3 Performance Analysis on Proposed Sequence Pattern Mining Algorithm

The performance analysis on proposed algorithms includes the correctness and the applicability. In order to analyse them, some experiments by applying the *AprioriAll* algorithm and proposed sequence pattern mining algorithm *SM-AprioriAll* should be done. The performance comparisons on *AprioriAll* algorithm and *SM-AprioriAll* algorithm will be presented in this Section.

4.3.1 Data Resource

These algorithms will be applied to two kinds of data resource: one is artificial data from a data generator and another one is practical data from a real legacy system.

First, artificial data will be generated. The hidden association rules or sequence patterns in artificial data will be defined. The defined information can be regarded as an answer key. The mined results by applying the proposed algorithms can be compared with this answer key. The correctness can be proved.

If the correctness obtained from the artificial data is accepted, the proposed algorithms can be applied to the practical (real) data, which is collected from the real application system.

4.3.2 Experiment Environment

In order to prove the correctness and applicability of the proposed sequence pattern mining algorithm (*SM-AprioriA*ll, for short), the following experiments were designed and implemented. The system was developed under the following hard and software environment:

CPU is Pentium4 2.0GHz; memory is 2G; Window XP operating system; programming language is Visual Basic.

4.3.3 Experiment Results

4.3.3.1 Artificial Data

The following sequence data was generated by IBM Quest Market-Basket Synthetic Data Generator [62]:

```
Number of transactions in database = 200000
Average transaction length = 10
Number of items = 9
Number of patterns = 10000
Average length of pattern = 4
```

The experiment results returned by *SM-AprioriAll* meet the answer key from artificial data. Thus, the correctness of *SM-AprioriAll* is proved.

The performance comparisons on Memory and Runtime between *SM-AprioriAll* and traditional *AprioriAll* are as follows:

The Memory performance for finding frequent patterns with *min-support* is equal to 0.02%, 0.03%, 0.04% and 0.05% is shown in Figure 4-7.

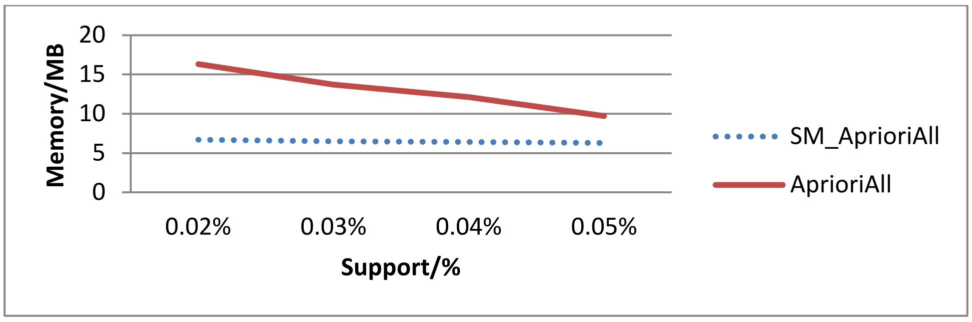


Figure 4-7. The Memory Performance for Finding Frequent Patterns.

The Runtime performance for finding frequent patterns with *min-support* is equal to 0.02%, 0.03%, 0.04% and 0.05% is shown in Figure 4-8.

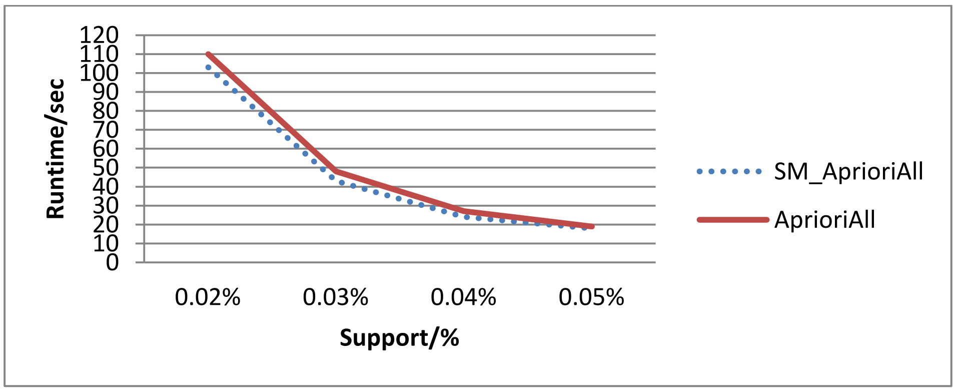


Figure 4-8. The Runtime Performance for Finding Frequent Patterns.

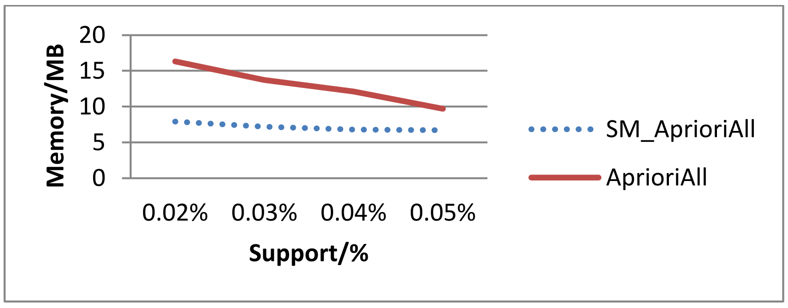


Figure 4-9. The Memory Performance for Finding Motifs.

The Memory performance for finding Motifs with *min-support* is equal to 0.02%, 0.03%, 0.04% and 0.05% is shown in Figure 4-9.

The Runtime performance for finding Motifs with *min-support* is equal to 0.02%, 0.03%, 0.04% and 0.05% is shown in Figure 4-10.

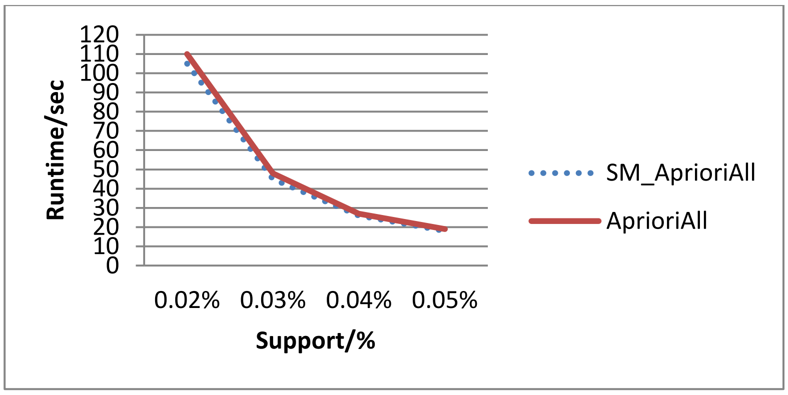


Figure 4-10. The Runtime Performance for Finding Motifs.

4.3.3.2 Practical Data

The practical data is collected from the usage’s log files of an e-learning system. The performance comparisons on Memory and Runtime between the proposed sequence pattern mining algorithm *SM-AprioriAll* and traditional *AprioriAll* are similar to the case of artificial data. More details are shown in Section 8.1.

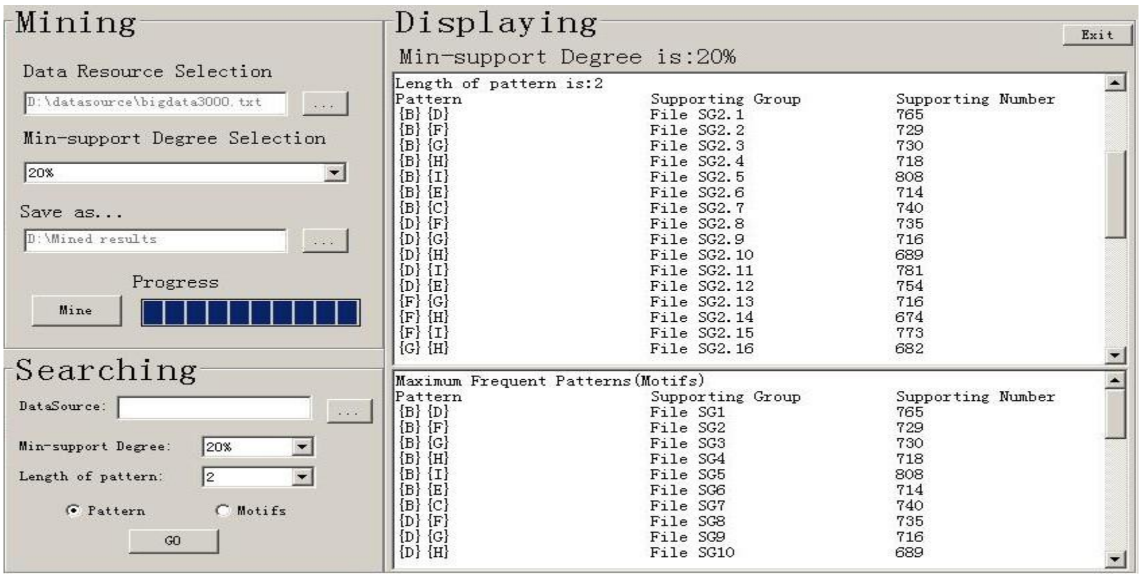
4.3.4 Conclusion

On artificial data, the experimental data shows the Memory performance of the proposed algorithm *SM-AprioriAll* is better than traditional *AprioriAll* algorithm. Moreover, the experimental data shows the Runtime performance of the proposed algorithm *SM-AprioriAll* is similar to (a little bit better) traditional *AprioriAll* algorithm. On practical data, the results are similar to the artificial data. Therefore, the proposed sequence pattern mining algorithm SM-*AprioriAll* is applicable.

4.4 Supporting Tool: SEquence Pattern Miner (SEPAM)

To support the method on applying data mining method to improve Service model and Component model, a SEquence PAttern Miner (SEPAM) based on the proposed sequence pattern mining algorithm has been designed and implemented.

There are three functions in SEPAM, namely, mining function, searching function and displaying function. Figure 4-11 presents the main interface of SEPAM.



Data Resource Selection: D:\datasource\bigdata3000.txt

Min-support Degree Selection: 20%

Save as...: D:\Mined results

DataSource:

Min-support Degree: 20%

Length of pattern: 2

[✓ Pattern] Motifs

Min-support Degree is:20%

```
Length of pattern is:2
Pattern                  Supporting Group          Supporting Number
 {B} {D}                 File SG2.1                765
 {B} {F}                 File SG2.2                729
 {B} {G}                 File SG2.3                730
 {B} {H}                 File SG2.4                718
 {B} {I}                 File SG2.5                808
 {B} {E}                 File SG2.6                714
 {B} {C}                 File SG2.7                740
 {D} {F}                 File SG2.8                735
 {D} {G}                 File SG2.9                716
 {D} {H}                 File SG2.10               689
 {D} {I}                 File SG2.11               781
 {D} {E}                 File SG2.12               754
 {F} {G}                 File SG2.13               716
 {F} {H}                 File SG2.14               674
 {F} {I}                 File SG2.15               773
 {G} {H}                 File SG2.16               682
```

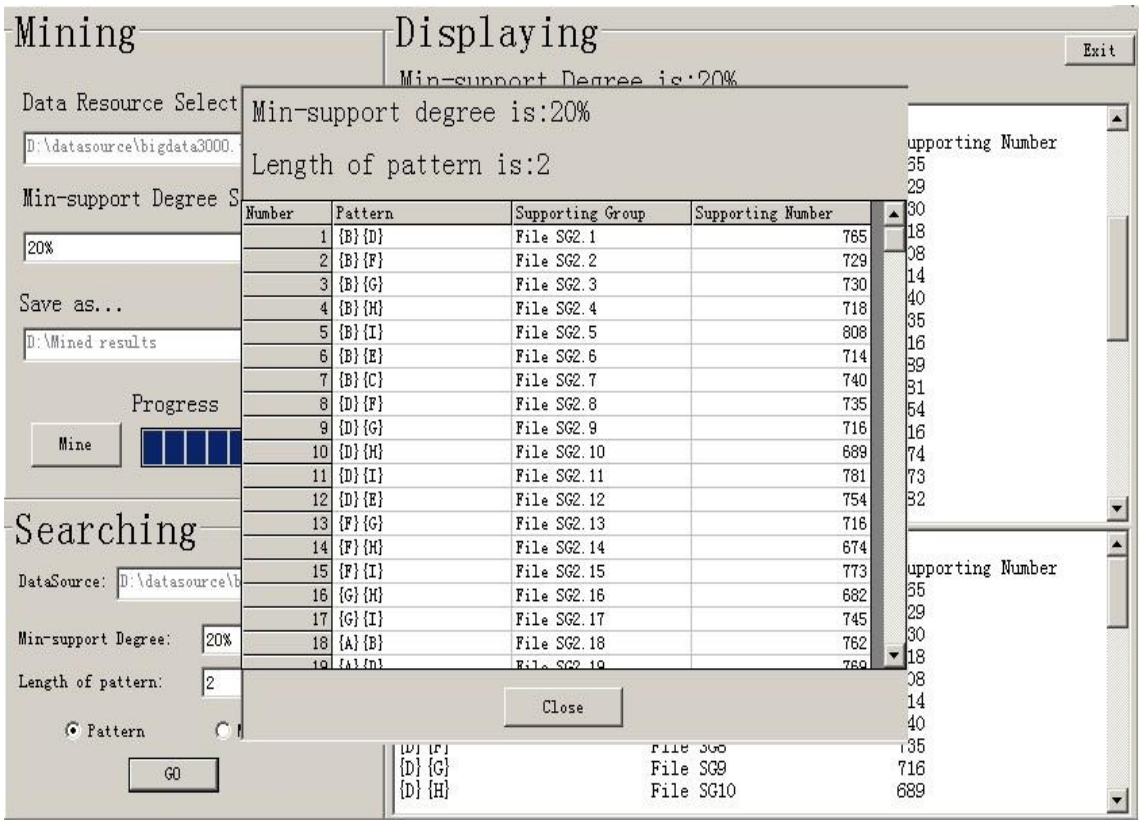
```
Maximum Frequent Patterns(Motifs)
Pattern                  Supporting Group          Supporting Number
 {B} {D}                 File SG1                  765
 {B} {F}                 File SG2                  729
 {B} {G}                 File SG3                  730
 {B} {H}                 File SG4                  718
 {B} {I}                 File SG5                  808
 {B} {E}                 File SG6                  714
 {B} {C}                 File SG7                  740
 {D} {F}                 File SG8                  735
 {D} {G}                 File SG9                  716
 {D} {H}                 File SG10                 689
```

Figure 4-11. The Main Interface of SEPAM.

Mining function is the key function, which reflects the performance of the proposed algorithm. The time complexity of this algorithm is subject to the length of each sequence, the number of items and the size of records in sequence database.

The "*saved as*" item in mining function is to save the useful data (some middle and final results) onto the hard disk for utilising in the searching and displaying functions.

The searching function and displaying function are the auxiliary functions, which can contribute to the analysis and process of the mined patterns. The searching function can browse frequent patterns, supporting group and supporting number/degree at any level. For example, the *min-support* number/degree is denoted as *min-support* and pattern's length is denoted as *length*. The frequent patterns with *min-support*=20% and *length*=2 is shown in Figure 4-12. Figure 4-13 shows the information on supporting group.



D:\datasource\bigdata3000.

20%

D:\Mined results

DataSource: D:\datasource\b

Min-support Degree: 20%

Length of pattern: 2

(•) Pattern

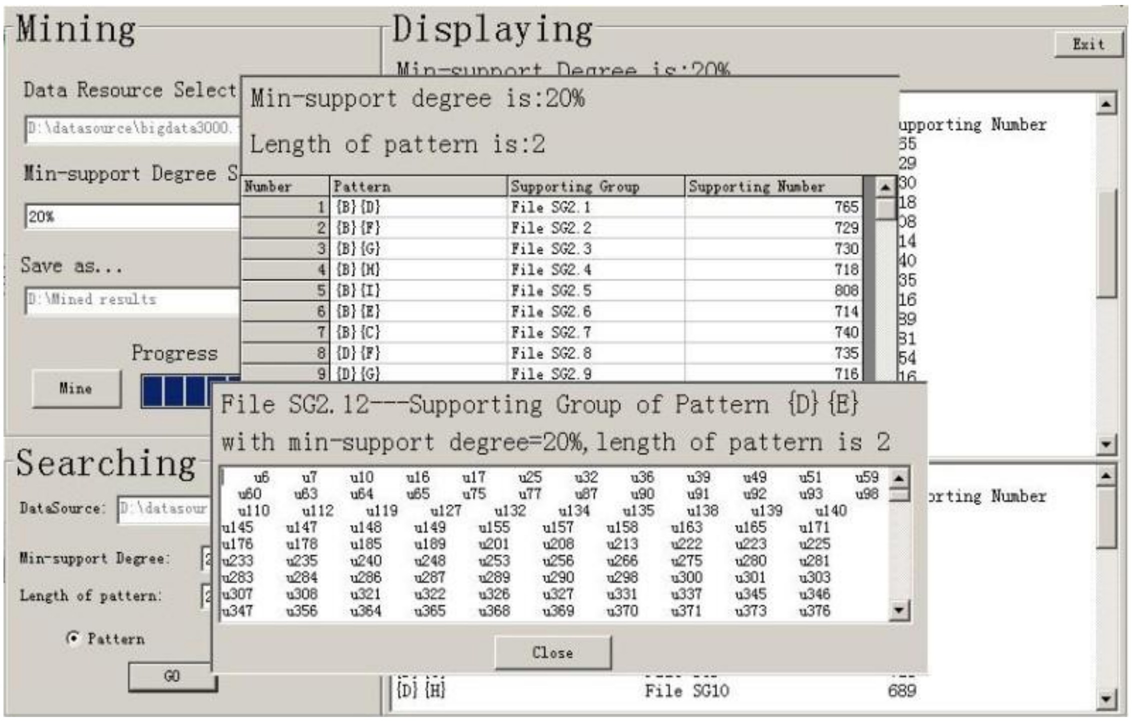
Min-support degree is:20%

Length of pattern is:2

Number	Pattern	Supporting Group	Supporting Number
1	{B}{D}	File SG2.1	765
2	{B}{F}	File SG2.2	729
3	{B}{G}	File SG2.3	730
4	{B}{H}	File SG2.4	718
5	{B}{I}	File SG2.5	808
6	{B}{E}	File SG2.6	714
7	{B}{C}	File SG2.7	740
8	{D}{F}	File SG2.8	735
9	{D}{G}	File SG2.9	716
10	{D}{H}	File SG2.10	689
11	{D}{I}	File SG2.11	781
12	{D}{E}	File SG2.12	754
13	{F}{G}	File SG2.13	716
14	{F}{H}	File SG2.14	674
15	{F}{I}	File SG2.15	773
16	{G}{H}	File SG2.16	682
17	{G}{I}	File SG2.17	745
18	{A}{B}	File SG2.18	762

```
{D} {G}          File SG9       716
{D} {H}          File SG10      689
```

Figure 4-12. Frequent Patterns with *Min-support*=20% and *length*=2.



Min-support degree is:20%

Length of pattern is:2

Number	Pattern	Supporting Group	Supporting Number
1	{B}{D}	File SG2.1	765
2	{B}{F}	File SG2.2	729
3	{B}{G}	File SG2.3	730
4	{B}{H}	File SG2.4	718
5	{B}{I}	File SG2.5	808
6	{B}{E}	File SG2.6	714
7	{B}{C}	File SG2.7	740
8	{D}{F}	File SG2.8	735
9	{D}{G}	File SG2.9	716

File SG2.12---Supporting Group of Pattern {D}{E}

with min-support degree=20%, length of pattern is 2

```
    u6     u7     u10    u16    u17    u25    u32    u36    u39    u49    u51    u59
  u60    u63    u64    u65    u75    u77    u87    u90    u91    u92    u93    u98
  u110     u112     u119     u127     u132     u134     u135     u138     u139     u140
u145   u147   u148   u149   u155   u157   u158   u163   u165   u171
u176   u178   u185   u189   u201   u208   u213   u222   u223   u225
u233   u235   u240   u248   u253   u256   u266   u275   u280   u281
u283   u284   u286   u287   u289   u290   u298   u300   u301   u303
u307   u308   u321   u322   u326   u327   u331   u337   u345   u346
u347   u356   u364   u365   u368   u369   u370   u371   u373   u376
```

```
{D} {H}          File SG10      689
```

Figure 4-13. The Information on Supporting Group.

4.5 Summary

Sequence mining techniques are used to software engineering data. The investigation results show the usage behaviour data can be mined to discover the potential concerns that can benefit SOA migration. The mined hidden information can be used for improving Service model and Component model.

- The process of establishing a Service model is presented. The hierarchical directed acyclic graph is adopted to represent a Service model.
- A sequence pattern mining algorithm for service composition is proposed for migrating e-learning legacy systems to SOA environment. First, some concepts and notations are defined. Then, some theorems and algorithms are described. Finally, some examples are presented for a better understanding of the proposed theorems and algorithms.
- The performance of proposed algorithm is analysed. The experimental data shows this sequence pattern mining algorithm is applicable.
- A sequence pattern mining tool (SEPAM) has been developed based on the proposed algorithm. It is good for improving the quality of Service model and Component model.

Chapter 5 Legacy Component Model Establishment with Data Mining Techniques

Objectives

- To show the process of establishing a Component model.
- To propose an association rule mining algorithm for determining relationships of legacy components.
- To evaluate the proposed association rule mining algorithm.
- To build an association rule miner based on the proposed algorithm.

5.1 Component Model Establishment

5.1.1 The Process of Establishing a Component Model

The work on the establishment of a Component model includes legacy asset analysis, component identification and the representation of final results. The process of establishing a Component model also consists of three phases. The first phase is to collect the related information source. The second phase is to apply the reasonable methods and techniques to solve the concerns (it is the key phase of this process). The third phase is to select knowledge representation method for the analysis results. Figure 5-1 describes the process of establishing a Component model.

1. Information source.

In this study, the selected information sources for establishing a Component model include legacy source codes, legacy application data, IT professionals and so on.

2. Legacy system analysis.

To establish a most reasonable Component model, an assessment of the legacy systems will be performed. Legacy systems should be decomposed into some sub-modules and then into some smaller and manageable components. Usually, there are three types of legacy systems: decomposable legacy system, semi-decomposable legacy system and non-decomposable legacy system. In this study, the non-decomposable legacy systems will be put aside. The other two will be processed further.

Domain application data and knowledge base are useful for analysing legacy systems. Such as, services, rules and business processes can be extracted from development documentation, source code, domain experts, UML files, user feedbacks and system logs. Some existing legacy system analysis methods can be applied in this part.

3. Data mining techniques.

Data mining techniques are utilised to mine the implicit patterns from application data (implementation level). The mined information can be of great benefit to establishing a Component model. The concrete data mining techniques include sequence pattern mining, association rules mining, classification, clustering and so on.

4. Component identification.

Legacy component identification is necessary for implementing new services. The granularity of component identification should be coincided with services' identification granularity since the two will meet for component-to-service matching later.

5. Knowledge representation methods.

The purpose of establishing knowledge representation is to put the identified components into the computation environment, which is the foundation for measuring the matching relationship between services and components. The representation method for services and components should be identical since they will meet in the matching stage for a further process.

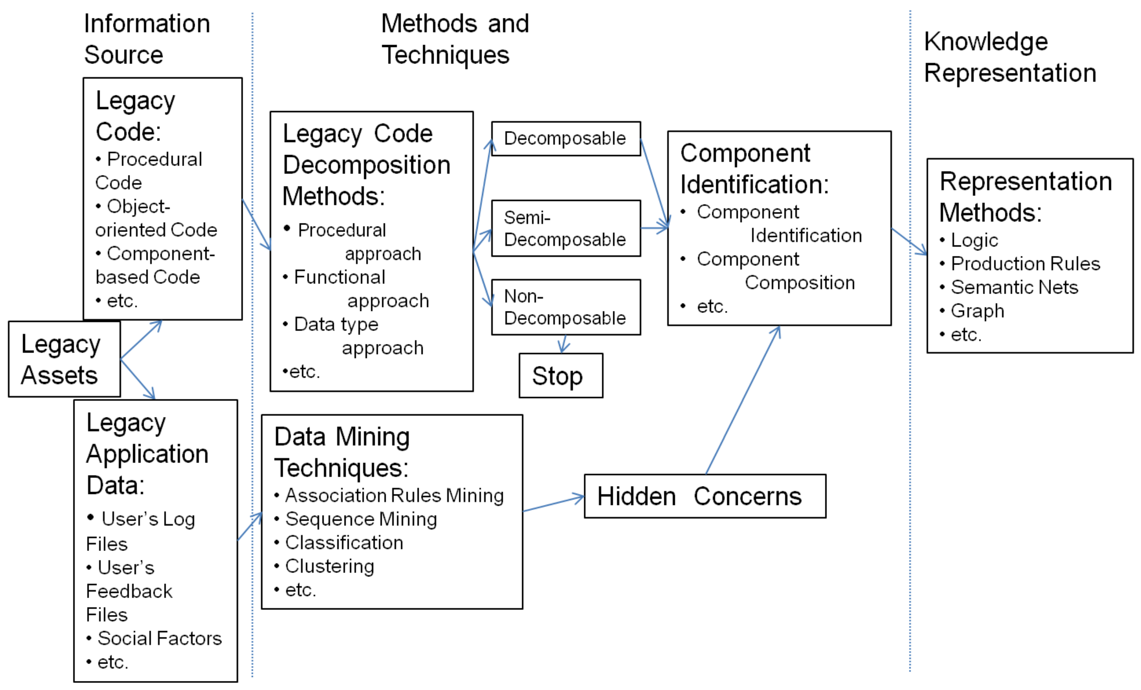


Figure 5-1. The Process of Establishing a Component Model.

5.1.2 The Representation of a Component Model

Legacy components are represented as a legacy HDAG, which also includes three levels: atomic component level, business component level and application component level. Each node in a legacy HDAG can represent an atomic component, a business component and an application component. It should be a composite node.

In this study, the legacy system is analysed from five levels, namely, text level (specification files), data level, code level, practical level (usage behaviour logs) and logic level (UML files).

- Text level – Analysing specification (Function) information written in natural language.

  The concerned factors include: domain knowledge base, domain thesaurus, similarity calculation between documents, etc.

- Data level -- Analysing database information.

  The concerned factors include: Relation (data, service), Relation (data, Function), Relation (data, Class), Relation (data, Component), business rules, etc.

- Code level – Analysing source code information.

  Component and sub-function are a relatively atomic level of abstraction. The concerned factors include: the name of Class, Component, Sub-function and Function; the number and type of Parameter and Statement, etc.

- Practical level –Analysing users' usage behaviour information.

  Some potential service composition and business process information can be found. The data mining techniques can be applied.

- Logic level –Analysing UML information.

  The concerned factors include: Class graph, Use case, a hierarchical tree or graph structure, etc.

In the support of algebra theory, each node in a legacy HDAG shown in Figure 5-2 is defined as follows.

Definition 5.1 Each *Node* in a legacy HDAG is a vector, which is composed of $Item_{\text{father}}$, $Item_{\text{function}}$, $Item_{\text{data}}$, $Item_{\text{code}}$, $Item_{\text{mining}}$, $Item_{\text{uml}}$, $Item_{\text{son}}$.

where, $Item_{\text{father}}$ refers to the father node of this node; $Item_{\text{function}}$ refers to the information from specification analysis; $Item_{\text{data}}$ refers to the information from data analysis; $Item_{\text{code}}$ refers to the information from code analysis; $Item_{\text{mining}}$ refers to the information from usage's log analysis by using data mining techniques; $Item_{\text{uml}}$ refers to the information from UML analysis; $Item_{\text{son}}$ refers to the son node of this node.

Node information is similar to the semantic interface of an atomic service/component or a composite service (component) or a business process (function), etc. For domain requirement analysis, normally, a top-down analysis approach will be applied. For legacy system analysis, a bottom-up approach may be adopted.

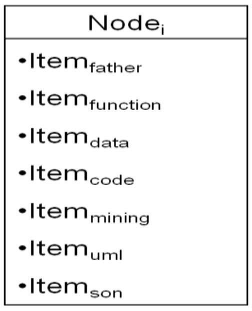


Figure 5-2. A Node in a Legacy HDAG.

Figure 5-3 presents an example of Component model represented by a hierarchical directed acyclic graph (see Section 4.1.2).

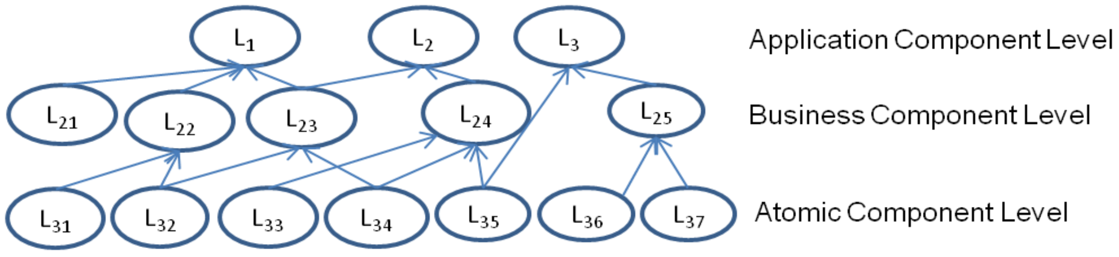


Figure 5-3. An Example of Component Model Represented by HDAG.

Figure 5-4 shows an example of a Component model and a Service model represented by the hierarchical directed acyclic graph.

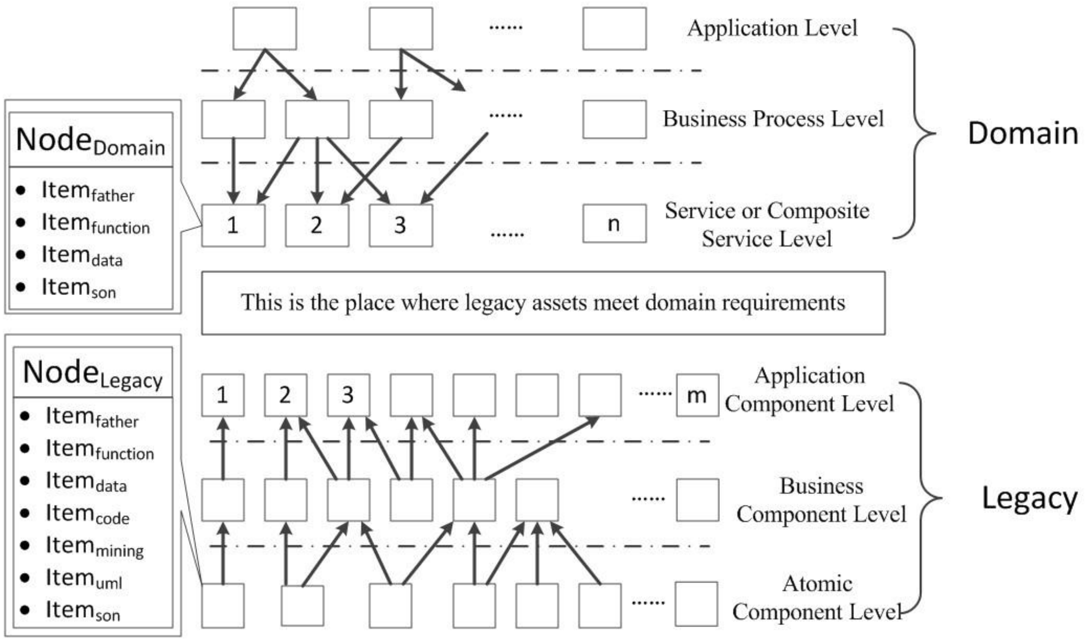


Figure 5-4. The Representation of a Service Model and a Component Model.

5.2 A Proposed Association Rule Mining Algorithm for Determining Relationships of Legacy Components

According to the application data features, an association rule mining algorithm is proposed. The proposed algorithm is suitable for the data set whose size is medium and whose number of items is less than 20. In addition, Apriori algorithm assumes that items in a transaction set are totally ordered. Normally, items within an itemset are sorted in lexicographic order in Apriori algorithm. The proposed algorithm can find all motifs with any *min-support* number without assuming any ordering.

The mined association relations can be utilised to determine the relationships of legacy components or the relationships of domain services. If the components (For example, C_1, C_2, C_3 in Figure 5-5) have a close association relation, they will be organised into the same unit in the relationship graph of legacy components. Sometimes, they may construct a new composite component (e.g., C_{123} in Figure 5-5).

In this Section, first, some related concepts and notations are described. Then, the proposed theorems and algorithm will be presented. Finally, for the best understanding of the theorems and algorithm, some small examples will be shown.

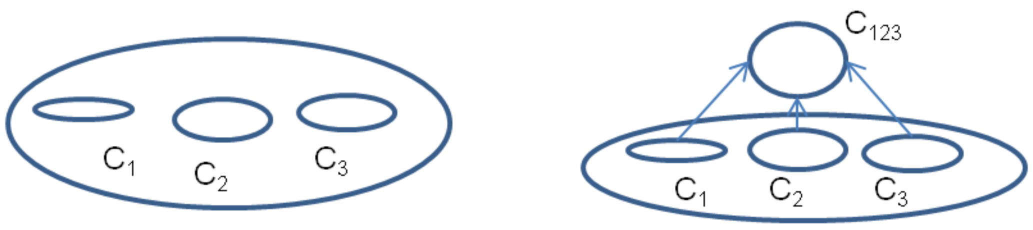


Figure 5-5. The Application Example of Components' Association Relations

5.2.1 Concepts and Notations

Investigation of item sets is based on the containing relation between itemsets. In this Sub-section, some concepts, notations and relations are established for further discussion.

5.2.1.1 Transaction Sets and Items

For a finite set T, the number of elements is named its *size* and denoted by $|T|$. Suppose a non-empty set T, its elements is named *items* and the set T is called *transaction set.* A non-empty subset of transaction set *T* is called *itemset*.

A *transaction* u over *T* refers to a non-empty subset of *T*, where $\emptyset \subset u \subseteq T$, $0 < |u| \leq |T|$ and $u \in 2^{|T|}$, namely, a *transaction* is an itemset. Thus, there are $2^{|T|} - 1$ itemsets over transaction set *T*.

A non-empty set U composed of a group of itemsets $\{u_1, u_2, \ldots, u_n\}$ over *T* is named as a *transaction database* over T. Association rules mining means to discover association rules from transaction database $U = \{u_1, u_2, \ldots, u_n\}$ and $0 < |U| = n \leq 2^{|T|} - 1$. An example is given to explain the concepts mentioned above.

Example:

Suppose that transaction set T = {Component1, Component2, Component3}. Then, 'Component1', 'Component2' and 'Component3' are called items in transaction set *T*. The item number or element number is $|T| = 3$.

The elements of powerset $2^{|T|}$ are the subsets of T, its element number is $2^{|T|} = 2^3 = 8$, namely,

$$2^{|T|} = \{A | A \subseteq T\}$$
$$= \{\emptyset, \{\text{Component1}\}, \{\text{Component2}\}, \{\text{Component3}\}, \{\text{Component1}, \text{Component2}\}, \{\text{Component1}, \text{Component3}\},$$
$$\{\text{Component2}, \text{Component3}\}, \{\text{Component1}, \text{Component2}, \text{Component3}\}\}$$

Thus, the itemsets of T are 7 non-empty subsets of T as follows: $\{\text{Component1}\}, \{\text{Component2}\}, \{\text{Component3}\}, \{\text{Component1}, \text{Component2}\}, \{\text{Component1}, \text{Component3}\},$

$\{\text{Component2}, \text{Component3}\}, \{\text{Component1}, \text{Component2}, \text{Component3}\}.$

The set of itemsets over *T* with size $l = 2$ is:

$$S_{T,2} = \{A \in 2^{|T|} \big| \, |A| = 2\}; \quad |S_{T,2}| = 3$$

namely,

$$S_{T,2} = \{\{\text{Component1}, \text{Component2}\}, \{\text{Component1}, \text{Component3}\}, \{\text{Component2}, \text{Component3}\}\}.$$

With the support of information technology, some original data are collected each day. After doing pre-process, Table 5-1 presents a transaction database *U*. In the following Sub-section, this transaction database *U*, where $\{A, B, C, D, E\} \subseteq T$, will be taken as an example to run the proposed association rule mining algorithms.

Table 5-1. Transaction Database *U*.

Database U	Transaction u_i	Size $l_i = \|u_i\|$
u_1	{A, B, C, D}	4
u_2	{ B, C, E}	3
u_3	{ A, B, C, E }	4
u_4	{ B, D, E }	3
u_5	{ A, B, C, D }	2

5.2.1.2 Containing Relations between Items

Given two itemsets u and v, $u = \{A_1, A_2, \dots, A_n\}$ ($n = |u| > 0$), $v = \{B_1, B_2, \dots, B_m\}$ ($m = |v| > 0$), itemset u is said to be contained in itemset v, or occurring in itemset v, or supported by itemset v, or u is a sub-itemset of v, or v is a super itemset or an extension of u when $u \subseteq v$. For example, in transaction database U (see Table 5-1), there exist,

$$u_2 \subset u_3: \{B, C, E\} \subset \{A, B, C, E\} \text{ and } \{E\} \subset \{B, E\} \subset \{B, C, E\} \subset \{A, B, C, E\}.$$

5.2.1.3 Patterns and Their Support Transaction Groups

Given a transaction $u = \{A_1, A_2, \dots, A_n, \dots A_N\}$ of transaction database U, let w be any sub-transaction of u. Sub-itemset w is called a *pattern* (*supported by or occurring in*) transaction u. By the definition of patterns, it is known that a pattern is also a transaction. Thus, pattern w has size$|w|$. A pattern with size 1 is an item of T. Given a non-empty transaction database $U = \{u_1, u_2, \dots, u_n, \dots u_N\}$, $0 < k \le K = |U|$, the pattern of any transaction in U are called patterns of U. Let L be the maximal size of transactions of U, namely, L =max($|u_1|$,$|u_2|$,…, $|u_K|$). For any pattern w, having $0 < |w| \le L$.

A particular pattern can be contained (or can "co-occur") in many transactions of a transaction database as their common sub-itemset. For a pattern w of U, it needs to know which transactions in U containing w. The set of transactions in U containing w is $M = \{u | u \in U, w \subseteq u\}$, denoted by w^U. Subset w^U consists of transactions in group U in which sequence w is contained, and is called the *support* or *occurrence group* of pattern w. The set of patterns over U with support number being not less than h is denoted by,

$$M(U,h) = \{w|\ |w^U| \geq h\}.$$

The set of patterns over U with size l and support number being not less than h is denoted by,

$$M(U,h,l) = \{w|\ |w^U| \geq h \ \text{ and } \ |w| = l\}.$$

The set of patterns over U with support number h is denoted by,

$$G(U,h) = \{w|\ |w^U| = h\}.$$

The set of patterns over U with size l and support number h is denoted by,

$$G(U,h,l) = \{w|\ |w^U| = h \ \text{ and } \ |w| = l\}.$$

There are,

$$G(U,h) = M(U,h) - M(U,h+1), h = 1,2,\dots,|U|-1;$$

$$G(U,|U|) = M(U,|U|);$$

$$G(U,h) = G(U,h,1) \cup G(U,h,2) \cup \dots \cup G(U,h,L);$$

$$M(U,h) = M(U,h,1) \cup M(U,h,2) \cup \dots \cup M(U,h,L);$$

$$M(U,1) \supseteq M(U,2) \supseteq \cdots \supseteq M(U,|U|).$$

Given a non-empty transaction database$U = \{u_1, u_2, \dots, u_k, \dots, u_K\}$, the set *maxM*($U$, h) of maximal patterns with support number being not less than *min-support* number h is called *motifs* over U.

5.2.2 Theorems and Algorithms

In this Sub-section, some theorems are established for finding patterns and a new algorithm is introduced for finding motifs with any *min-support* number h.

5.2.2.1 Motifs with Min-support $h = 1$ (maxM$(U, h = 1)$)

1-size patterns are called *units* or *unit patterns.* Notice that a 1-size pattern $w \in (2^T - \{\emptyset\})$ is in $M(U,h,1)$ if and only if $|w^U| \geq h$. This fact can be used to find $M(U,h,1)$, namely, units or unit patterns, which is useful for finding $M(U,h=2)$.

The same way can be used to find the other size patterns, such as 2-size, 3-size, etc. A lot of work should be done in this way. In fact, there is a simple way to find motifs with *min-support* $h = 1$.

Motifs with *min-support* 1 are the maximal transactions of $M(U, 1)$, denoted by *maxM*(*U*,1). Theorem 5.1, which shows a simple way for finding $maxM(U, 1)$, is shown as follows.

Theorem 5.1 Given a transaction database *U*, motifs with *min-support* 1 are the maximal transactions of *U*, and vice versa. That is *maxM*(*U*, 1) = *maxU*.

Proof.

(1) $U \subseteq M(U, 1)$; suppose $u \in U$ to prove $u \in M(U, 1)$.

Let $u \in U$, there is $u \subseteq u$; also, according to the definition of support group, there exist $u \in u^U$ and $|u^U| \geq 1$. Thus, $u \in M(U, 1)$.

(2) $maxU \subseteq maxM(U, 1)$; suppose $u \in maxU$ to prove $u \in maxM(U, 1)$.

(2.1) $u \in M(U, 1)$

From $u \in maxU$, there is $u \in U$; by (1), there is $u \in M(U, 1)$.

(2.2) $u \in maxM(U, 1)$

Suppose $u \notin maxM(U, 1)$, there exists $w \in M(U, 1)$ and $u \subset w$; there is $u' \in U$ such that $w \subseteq u'$ and $u \subset w$; there exists $u' \in U$ such that $u \subset w \subseteq u'$; there is $u' \in U$ such that $u \subset u'$. This is contrary to assumption $u \in maxU$. Thus, $u \in maxM(U, 1)$.

(3) $maxM(U, 1) \subseteq maxU$; assume $w \in maxM(U, 1)$ to prove $w \in maxU$.

(3.1) $w \in U$

From $w \in maxM(U, 1)$, there is $w \in M(U, 1)$; there exists $u \in U$ such that $w \subseteq u$. It can prove that $w = u$ and so $w \in U$. Otherwise, $w \subset u$, by (1) there is $u \in M(U, 1)$. This is contrary to assumption $w \in maxM(U, 1)$.

(3.2) $w \in maxU$

If $w \notin maxU$, there exists $u \in U$ such that $w \subset u$. This is contrary to assumption $w \in maxM(U, 1)$. Thus, $w \in maxU$.

The proof is completed.

5.2.2.2 Patterns with Min-support $\boldsymbol{h = 2}$ $(\boldsymbol{M(U, h = 2)})$

The set of patterns with support number being not less than 2 over U is denoted by $M(U, 2)$. Generally, there are the following theorems.

Theorem 5.2 Every sub-itemset of an h-pattern is also an h-pattern. That is, if $w \in M(U, h)$ and $y \subseteq w$ then $y \in M(U, h)$.

Proof.

(1) If $y \subseteq w$ then $w^U \subseteq y^U$. Suppose $u \in w^U$ to prove $u \in y^U$.

Since $u \in w^U$, there is $w \subseteq u$; since $y \subseteq w$ and $w \subseteq u$, there is $y \leq u$; since $y \subseteq u$, there is $u \in y^U$.

(2) By (1), there is $|w^U| \leq |y^U|$.

(3) Since $w \in M(U, h)$, $|w^U| \geq h$ can be proved. By (2), there is $|y^U| \geq h$, thus, $y \in M(U, h)$.

The proof is completed.

Theorem 5.3 If $w \in M(U, h)$ and $|w| > 1$ then *w* can be expressed as an union of two patterns *x* and *z*, where *x* has *min-support h* and length $|w| - 1$and *z* has *min-support h* and length l. That is, $w = x \cup z$, where $x \in M(U, h, |w| - 1)$and $z \in M(U, h, 1)$.

Proof.

Since $|w| > 1$, a pattern z with size 1 from w can be separated such that $w = x \cup z$, where $|x| = |w| - 1$, $|z| > 1$ and $x, z \subseteq w$. By theorem 5.2 and $w \in M(U, h)$, there is $x, z \in M(U, h)$, more precisely, $x \in M(U, h, |w| - 1)$ and $z \in M(U, h, 1)$.

The proof is completed.

Now, considering a case of *min-support* 2 and discussing how to find $M(U, 2)$. Then theorem 5.3 can be expressed as follows:

$$M(U, 2, l) \subseteq M(U, 2, l - 1) \cup M(U, 2,1), \quad \text{when } l > 1$$

Therefore, $M(U, 2, l)$ has a set of candidates $M(U, 2, l - 1) \cup M(U, 2,1)$.

For *size*=2 or *l*=2, there is candidate set,

$$C(U,2,2) = M(U,2,1) \cup M(U,2,1)$$

For *size* > 2 or *l* >2, there is,

$$M(U,2,l) \subseteq M(U,2,l-1) \cup M(U,2,1)$$

The following is the detailed algorithm for finding the set $M(U,2)$ of patterns with *min-support* $h = 2$.

Algorithm *AH*₂*A* for finding $M(U,2)$ //**A**ssociation rules with **h**=**2** **A**lgorithm

```
AH_2A1. Find all 1-size (or unit) patterns $M(U,2,1)$ from sequence database $U$ by checking its occurring or support number $h \geq 2$.   //finding $M(U,2,1)$
AH_2A2. Find $M(U,2,2)$ from $M(U,2,1)$
        for each $x \in M(U,2,1)$ do
            for each $z \in M(U,2,1)$ do
                union of x and z to generate $w = x \cup z$, and put w to the candidate set
                                                  //obtain candidate set with size2
        if($|w^U| \geq 2$) then $w \in M(U,2,2)$   //check support number of the candidate
                                                  patterns and get $M(U,2,2)$.
    AH_2A3. Find $M(U,2,l)$ from $M(U,2,l-1)$, $l > 2$
        for ($l = 3; M(U,2,l-1) \neq \emptyset; l++$) do
            for each $x \in M(U,2,l-1)$ do
                for each $z \in M(U,2,1)$ do
                    union of x and z to generate $w = x \cup z$, and put w to the
                                    candidate set //obtain candidate set with size l
        if($|w^U| \geq 2$) then $w \in M(U,2,l)$   //check support number of the candidate
                                                  patterns and get $M(U,2,l)$.
AH_2A4. Output the set $M(U,2)$ of patterns with min-support 2.
            $M(U,2) = M(U,2,1) \cup M(U,2,2) \cup \ldots \cup M(U,2,l)$   //answer
```

The algorithm ***AH*₂*A*** is completed.

5.2.2.3 Patterns with Min-support $\boldsymbol{h} > 2$ ($\mathbf{M}(\boldsymbol{U}, \boldsymbol{h} > 2)$)

After having found $M(U, 2)$ over transaction database U, the others, such as $M(U, 3)$, $M(U, 4)$, …, $M(U, |U|)$, can be found by the following formula,

$$M(U, h+1) = M(U, h) - G(U, h) \qquad \text{for } h = 2,3,\ldots,|U|-1$$

Therefore, the algorithm ***AHG₂A*** for finding patterns with *min-support h*>2 is shown in the following.

Algorithm ***AHG₂A*** for finding patterns with *min-support h*>2 //**A**ssociation rules with **h G**reater than **2 A**lgorithm

```
AHG2A1. Call algorithm AH2A //find $M(U, 2)$
AHG2A 2. For ($h = 2; M(U, h+1) \neq \emptyset; h++$) do
    $M(U, h+1) = M(U, h) - G(U, h)$ //delete pattern $w$ with $|w^U| = h$
        from $M(U, h)$, the remainder belongs to
        $M(U, h+1)$.
```

5.2.2.4 Motifs with $\boldsymbol{h} \geq \mathbf{2}$ ($\mathbf{maxM}(\boldsymbol{U}, \boldsymbol{h} \geq \mathbf{2})$)

Let *V* be a non-empty partially ordered set by less than (<) relation. An element *w* of set *V* is a maximum in *V* if there exist no $v \in V - w$ such that $v > w$.

Algorithm ***AMX*** for finding *maxV* from *V*

```
$maxV = \emptyset$
For each $w \in V$
    For each $v \in V - w$
        { counter=0
            If $v > w$ then counter++}
If counter==0, then $(maxV) = (maxV) \cup \{w\}$ //there exist no $v \in V - w$ such
        that $v > w$. So w is a maximum in V and put $w \in maxV$.
```

The algorithm *AMX* is completed.

5.2.2.5 Association Rules from Motifs

After having found Motifs, association rules can be found from motifs. An association rule is $x \dashrightarrow y$, which means if x co-occurs then y co-occurs. And the rule has confidence. An example is shown in Sub-section 5.2.3.6.

5.2.3 Examples

In order to describe the proposed theorem and algorithms clearly, transaction database *U* (see Table 5-1) is taken as an example to run them.

5.2.3.1 For Finding Unit Patterns (Units)

Table 5-2 shows the results of $M(U, 1,1)$. The same way can be used to find other size patterns, such as 2-size, 3-size, etc.

Table 5-2. Patterns with Size 1 and *Min-support* 1 over *U*

1-size patterns w	Support Group w^U	Support Number $\lvert w^U \rvert$
$\{A\}$	$\{u_1, u_3, u_5\}$	3
$\{B\}$	$\{u_1, u_2, u_3, u_4, u_5\}$	5
$\{C\}$	$\{u_1, u_2, u_3, u_5\}$	4
$\{D\}$	$\{u_1, u_4, u_5\}$	3
$\{E\}$	$\{u_1, u_3, u_4\}$	3

$$M(U, 1) = M(U, 1,1) \cup M(U, 1,2) \cup M(U, 1,3) \cup M(U, 1,4) \cup M(U, 1,5)$$

5.2.3.2 For Finding $\boldsymbol{maxM(U, h = 1)}$

By theorem 5.1, the motifs *maxM* (*U*, 1) with *min-support* 1 over transaction database *U* can be found immediately by finding *maxU* from *U* itself. Since $u_2 \subset u_3$, there is *maxM*(*U*, 1) = *maxU* =*max*{u_1, u_2, u_3, u_4, u_5} ={ u_1, u_3, u_4, u_5}. Table 5-3 represents deailed results.

Table 5-3. Motifs with *Min-support* 1 over *U*

Patterns w	Support Group w^U	Support Number $\|w^U\|$
$\{A, B, C, D\}$	$\{u_1, u_5\}$	2
$\{A, B, C, E\}$	$\{u_3\}$	1
$\{B, D, E\}$	$\{u_4\}$	1

5.2.3.3 For Finding $\boldsymbol{M(U, h = 2)}$

Running the proposed algorithm ***AH*₂*A*** on the transaction database U, there are the following results,

$$M(U, 2) = M(U, 2,1) \cup M(U, 2,2) \cup M(U, 2,3) \cup M(U, 2,4) \cup M(U, 2,5)$$

1. Finding $M(U, 2,1)$

$M(U, 2,1)$ can be obtained from $M(U, 1,1)$ in Table 5-2 by running step AH_2A1 of algorithm AH_2A. It is the same as $M(U, 1,1)$ since all support numbers in $M(U, 1,1)$ are more than 2.

2. Finding $M(U, 2,2)$

$M(U, 2,2)$, which is shown in Table 5-4, can be obtained from $M(U, 2,1)$ by running steps AH_2A2 and AH_2A3 of algorithm AH_2A.

Table 5-4. Patterns with Size 2 and *Min-support* 2 over *U*

2-size patterns w	Support Group w^U	Support Number $\|w^U\|$
$\{A,B\}$	$\{u_1, u_3, u_5\}$	3
$\{A,C\}$	$\{u_1, u_3, u_5\}$	3
$\{A,D\}$	$\{u_1, u_5\}$	2
$\{B,C\}$	$\{u_1, u_2, u_3, u_5\}$	4
$\{B,D\}$	$\{u_1, u_4, u_5\}$	3
$\{B,E\}$	$\{u_2, u_3, u_4\}$	3
$\{C,D\}$	$\{u_1, u_5\}$	2
$\{C,E\}$	$\{u_2, u_3\}$	2

3. Finding $M(U,2,3)$

$M(U,2,3)$ shown in Table 5-5 can be found by running step AH_2A3 of algorithm AH_2A at l = 3.

Table 5-5. Patterns with Size 3 and *Min-support* 2 over *U*

3-size patterns w	Support Group w^U	Support Number $\|w^U\|$
$\{A,B,C\}$	$\{u_1, u_3, u_5\}$	3
$\{A,B,D\}$	$\{u_1, u_5\}$	2
$\{A,C,D\}$	$\{u_1, u_5\}$	2
$\{B,C,D\}$	$\{u_1, u_5\}$	2
$\{B,C,E\}$	$\{u_2, u_3\}$	2

4. Finding $M(U,2,4)$

$M(U,2,4)$ shown in Table 5-6 by running step AH_2A3 of algorithm AH_2A at l = 4.

Table 5-6. Patterns with Size 4 and *Min-support* 2 over *U*

4-size patterns w	Support Group w^U	Support Number $\|w^U\|$
$\{A,B,C,D\}$	$\{u_1, u_5\}$	2

5. Finding $M(U,2,5)$

$M(U,2,5)$ can be found by running step AH_2A3 of algorithm AH_2A at l = 5. Since $C_5 = \{\{A,B,C,D,\emptyset\}\}, |C_5| = 1$, thus, $M(U,2,5) = \emptyset$.

6. Finding $M(U,2)$

$M(U,2)$ can be found by running step AH_2A4 of algorithm AH_2A. Namely,

$$M(U,2) = M(U,2,1) \cup M(U,2,2) \cup M(U,2,3) \cup M(U,2,4)$$

The result is shown in Table 5-7.

Table 5-7. Patterns with *Min-support* 2 over *U*.

Patterns w	Support Group w^U	Support Number $\|w^U\|$
$\{A\}$	$\{u_1, u_3, u_5\}$	3
$\{B\}$	$\{u_1, u_2, u_3, u_4, u_5\}$	5
$\{C\}$	$\{u_1, u_2, u_3, u_5\}$	4
$\{D\}$	$\{u_1, u_4, u_5\}$	3
$\{E\}$	$\{u_1, u_3, u_4\}$	3
$\{A,B\}$	$\{u_1, u_3, u_5\}$	3
$\{A,C\}$	$\{u_1, u_3, u_5\}$	3
$\{A,D\}$	$\{u_1, u_5\}$	2
$\{B,C\}$	$\{u_1, u_2, u_3, u_5\}$	4
$\{B,D\}$	$\{u_1, u_4, u_5\}$	3
$\{B,E\}$	$\{u_2, u_3, u_4\}$	3
$\{C,D\}$	$\{u_1, u_5\}$	2
$\{C,E\}$	$\{u_2, u_3\}$	2
$\{A,B,C\}$	$\{u_1, u_3, u_5\}$	3
$\{A,B,D\}$	$\{u_1, u_5\}$	2
$\{A,C,D\}$	$\{u_1, u_5\}$	2
$\{B,C,D\}$	$\{u_1, u_5\}$	2
$\{B,C,E\}$	$\{u_2, u_3\}$	2
$\{A,B,C,D\}$	$\{u_1, u_5\}$	2

5.2.3.4 For Finding $\mathbf{M}(\boldsymbol{U}, \boldsymbol{h} > \mathbf{2})$

By running algorithm *AHG*$_2$*A* on transaction database *U* shown in Table 5-1, $M(U, 3)$ shown in Table 5-8, $M(U, 4)$ shown in Table 5-9 and $M(U, 5)$ shown in Table 5-10 (until $M(U, 6) = \emptyset$) can be found.

Table 5-8. Patterns with *Min-support* 3 over *U*.

Patterns w	Support Group w^U	Support Number $\|w^U\|$
$\{A\}$	$\{u_1, u_3, u_5\}$	3
$\{B\}$	$\{u_1, u_2, u_3, u_4, u_5\}$	5
$\{C\}$	$\{u_1, u_2, u_3, u_5\}$	4
$\{D\}$	$\{u_1, u_4, u_5\}$	3
$\{E\}$	$\{u_1, u_3, u_4\}$	3
$\{A,B\}$	$\{u_1, u_3, u_5\}$	3
$\{A,C\}$	$\{u_1, u_3, u_5\}$	3
$\{B,C\}$	$\{u_1, u_2, u_3, u_5\}$	4
$\{B,D\}$	$\{u_1, u_4, u_5\}$	3
$\{B,E\}$	$\{u_2, u_3, u_4\}$	3
$\{A,B,C\}$	$\{u_1, u_3, u_5\}$	3

Table 5-9Patterns with *Min-support* 4 over *U*.

Patterns w	Support Group w^U	Support Number $\|w^U\|$
$\{B\}$	$\{u_1,, u_2, u_3, u_4, u_5\}$	5
$\{C\}$	$\{u_1,, u_2, u_3, u_5\}$	4
$\{B, C\}$	$\{u_1,, u_2, u_3, u_5\}$	4

Table 5-10. Patterns with *Min-support* 5 over *U*.

Patterns w	Support Group w^U	Support Number $\|w^U\|$
$\{B\}$	$\{u_1,, u_2, u_3, u_4, u_5\}$	5

5.2.3.5 For Finding $\mathbf{maxM(U, h \geq 2)}$

By running Algorithm *AMX* on $M(U,2)$, $M(U,3)$, $M(U,4)$ and $M(U,5)$, $maxM(U,2)$ shown in Table 5-11, $maxM(U,3)$ shown in Table 5-12, $maxM(U,4)$ shown in Table 5-13 and $maxM(U,5)$ shown in Table 5-14 can be found.

Table 5-11. Motifs *maxM*(*U*,2).

Patterns *w*	Support Group w^U	Support Number $\|w^U\|$
{B,C,E }	$\{u_2, u_3\}$	2
{A,B,C,D}	$\{u_1, u_5\}$	2

Table 5-12. Motifs *maxM*(*U*,3.)

Patterns *w*	Support Group w^U	Support Number $\|w^U\|$
{B,D}	$\{u_1, u_4, u_5\}$	3
{B,E}	$\{u_2, u_3, u_4\}$	3
{A,B,C}	$\{u_1,, u_3, u_5\}$	3

Table 5-13. Motifs *maxM*(*U*,4).

Patterns *w*	Support Group w^U	Support Number $\|w^U\|$
{B,C}	$\{u_1, u_2, u_3, u_5\}$	4

Table 5-14. Motifs *maxM*(*U*,5).

Patterns *w*	Support Group w^U	Support Number $\|w^U\|$
$\{B\}$	$\{u_1, u_2, u_3, u_4, u_5\}$	5

5.2.3.6 For Finding Association Rules from Motifs

Motif $\{B, C, E\}$ from $\max M(U, 2)$ can be taken as an example.

Table 5-15. Association Rules from Motifs *maxM*(*U*,2).

Patterns *x*	x^U	$\|x^U\|$	$x \rightarrow y$	$2/\|x^U\|$
$\{B,C\}$	$\{u_1, u_2, u_3, u_5\}$	4	$\{B,C\} \rightarrow \{E\}$	1/2
$\{B\}$	$\{u_1, u_2, u_3, u_4, u_5\}$	5	$\{B\} \rightarrow \{C, E\}$	2/5
$\{C\}$	$\{u_1, u_2, u_3, u_5\}$	4	$\{C\} \rightarrow \{B, E\}$	1/2
$\{B,E\}$	$\{u_2, u_3, u_4\}$	3	$\{B, E\} \rightarrow \{C\}$	2/3
$\{E\}$	$\{u_2, u_3, u_4\}$	3	$\{E\} \rightarrow \{B, C\}$	2/3
$\{C,E\}$	$\{u_2, u_3\}$	2	$\{C, E\} \rightarrow \{B\}$	1

Note: x^U -- Support Group; $|x^U|$--Support Number; $2/|x^U|$ -- Confidence.

An association rule is $x \dashrightarrow y$, where $\emptyset \subset x \subset \{B,C,E\}; y = \{B,C,E\} - x$, and with confidence $(x \rightarrow y) = \frac{|\{B,C,E\}^U|}{|x^U|} = \frac{2}{|x^U|}$.

Thus, there are $|\{x| \emptyset \subset x \subset \{B,C,E\}\}| = 2^{|\{B,C,E\}|}$-2=6 association rules for $\{B,C,E\}$ shown in Table 5-15.

5.3 Performance Analysis on the Proposed Association Rule Mining Algorithm

The performance analysis on the proposed association rule mining algorithm includes the correctness and the applicability. In order to analyse them, some experiments by running the *Apriori* algorithm and the proposed algorithm should be done.

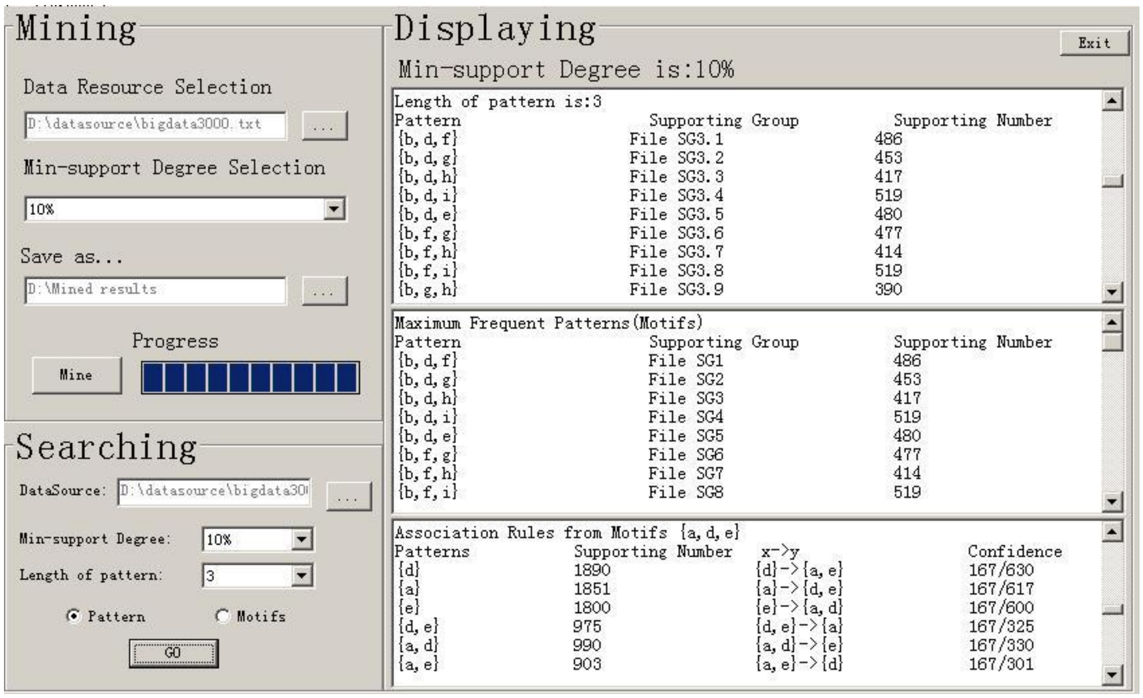
The work on collecting data resource (artificial and practical data) and setting up the experiment environment is similar to the work on the proposed sequence mining algorithm.

The experiment's results on the proposed association rule mining algorithm are better than the Apriori algorithm from the two aspects of runtime and memory. Therefore, the proposed association rule mining algorithm is available.

5.4 Supporting Tool: Association Rule Miner (ARM)

To support the method on applying data mining method to improve Component model and Service model, an Association Rule Miner (ARM) has been designed and implemented.

There are three functions in ARM, namely, mining function, searching function and displaying function. Figure 5-6 presents the main interface of ARM.



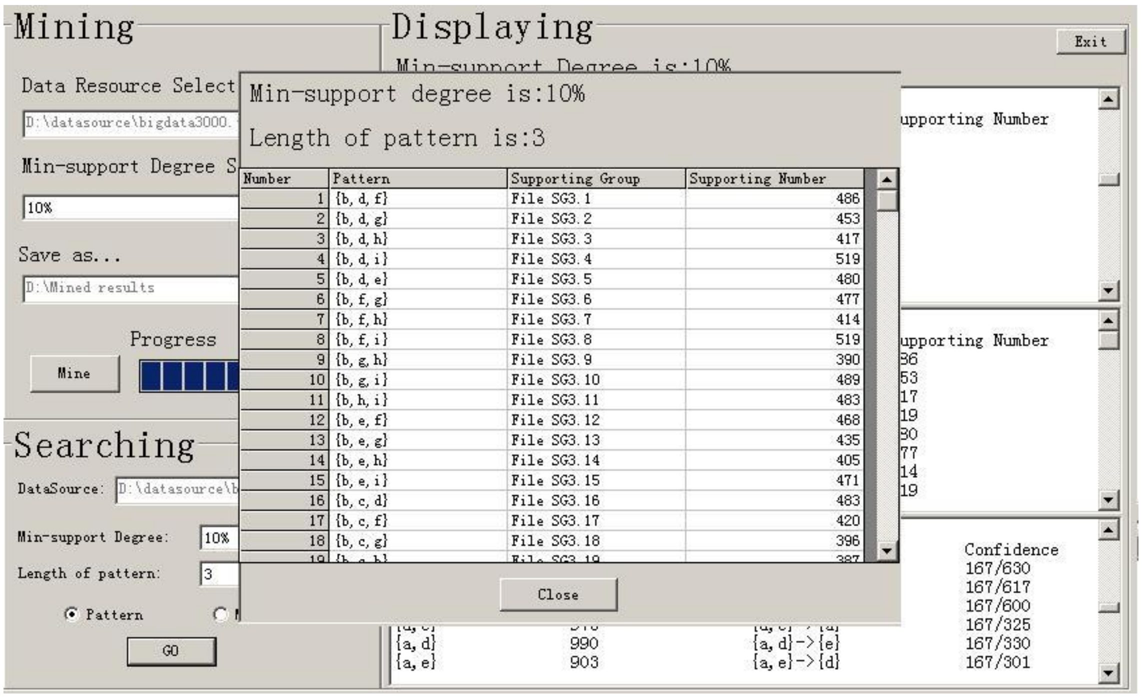
```
Data Resource Selection: D:\datasource\bigdata3000.txt
Min-support Degree Selection: 10%
Save as...: D:\Mined results

DataSource: D:\datasource\bigdata30
Min-support Degree: 10%
Length of pattern: 3
(•) Pattern   ( ) Motifs
```

```
Min-support Degree is:10%

Length of pattern is:3
Pattern     Supporting Group     Supporting Number
{b,d,f}     File SG3.1           486
{b,d,g}     File SG3.2           453
{b,d,h}     File SG3.3           417
{b,d,i}     File SG3.4           519
{b,d,e}     File SG3.5           480
{b,f,g}     File SG3.6           477
{b,f,h}     File SG3.7           414
{b,f,i}     File SG3.8           519
{b,g,h}     File SG3.9           390

Maximum Frequent Patterns(Motifs)
Pattern     Supporting Group     Supporting Number
{b,d,f}     File SG1             486
{b,d,g}     File SG2             453
{b,d,h}     File SG3             417
{b,d,i}     File SG4             519
{b,d,e}     File SG5             480
{b,f,g}     File SG6             477
{b,f,h}     File SG7             414
{b,f,i}     File SG8             519

Association Rules from Motifs {a,d,e}
Patterns    Supporting Number    x->y          Confidence
{d}         1890                 {d}->{a,e}    167/630
{a}         1851                 {a}->{d,e}    167/617
{e}         1800                 {e}->{a,d}    167/600
{d,e}       975                  {d,e}->{a}    167/325
{a,d}       990                  {a,d}->{e}    167/330
{a,e}       903                  {a,e}->{d}    167/301
```

Figure 5-6. The Main Interface of ARM.

Mining function is the key function, which reflects the performance of the proposed algorithm. The time complexity of this algorithm is subject to the size of each itemset, the number of items and the size of records in the itemset database. The "*saved as*" sub-function in the mining function is to save the useful data (some middle and final results) into the hard disk for utilising in the searching and displaying functions.

The searching function and displaying function are the auxiliary functions, which can contribute to the analysis and process of the mined patterns. The searching function can browse frequent patterns, supporting group and supporting number/degree at any level. For example, supposing the *min-support* number/degree is denoted as *min-support* and the element number of each pattern is denoted as *size*. Figure 5-7 shows the information on frequent patterns with *min-support* = 10% and *size* = 3. Figure 5-8 shows the information on supporting group.



Min-support degree is:10%

Length of pattern is:3

Number	Pattern	Supporting Group	Supporting Number
1	{b, d, f}	File SG3.1	486
2	{b, d, g}	File SG3.2	453
3	{b, d, h}	File SG3.3	417
4	{b, d, i}	File SG3.4	519
5	{b, d, e}	File SG3.5	480
6	{b, f, g}	File SG3.6	477
7	{b, f, h}	File SG3.7	414
8	{b, f, i}	File SG3.8	519
9	{b, g, h}	File SG3.9	390
10	{b, g, i}	File SG3.10	489
11	{b, h, i}	File SG3.11	483
12	{b, e, f}	File SG3.12	468
13	{b, e, g}	File SG3.13	435
14	{b, e, h}	File SG3.14	405
15	{b, e, i}	File SG3.15	471
16	{b, c, d}	File SG3.16	483
17	{b, c, f}	File SG3.17	420
18	{b, c, g}	File SG3.18	396

Data Resource Select: D:\datasource\bigdata3000.

Min-support Degree: 10%

Save as...: D:\Mined results

DataSource: D:\datasource\b

Min-support Degree: 10%

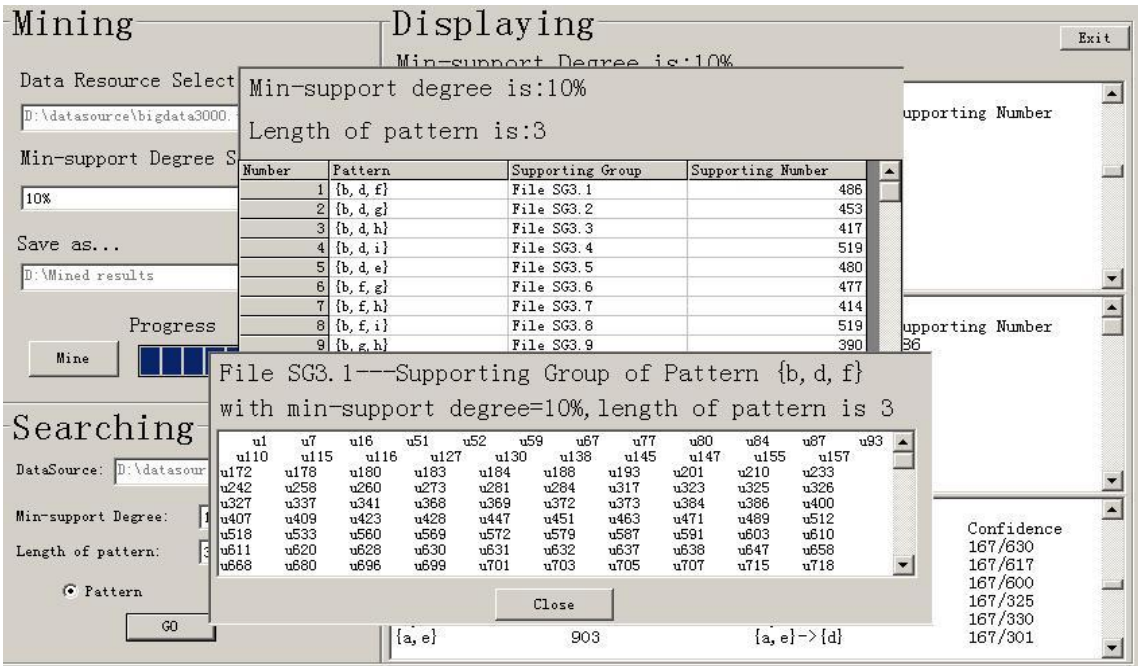
Length of pattern: 3

Min-support Degree is:10%

```
{a, d}    990    {a, d}->{e}    167/617
{a, e}    903    {a, e}->{d}    167/301
```

Confidence: 167/630, 167/617, 167/600, 167/325, 167/330, 167/301

Figure 5-7. Frequent Patterns with *Min-support*=10% and *size*=3.



Min-support degree is:10%

Length of pattern is:3

Number	Pattern	Supporting Group	Supporting Number
1	{b, d, f}	File SG3.1	486
2	{b, d, g}	File SG3.2	453
3	{b, d, h}	File SG3.3	417
4	{b, d, i}	File SG3.4	519
5	{b, d, e}	File SG3.5	480
6	{b, f, g}	File SG3.6	477
7	{b, f, h}	File SG3.7	414
8	{b, f, i}	File SG3.8	519
9	{b, g, h}	File SG3.9	390

File SG3.1---Supporting Group of Pattern {b, d, f}
with min-support degree=10%, length of pattern is 3

```
   u1    u7    u16   u51   u52   u59   u67   u77   u80   u84   u87   u93
  u110   u115   u116   u127   u130   u138   u145   u147   u155   u157
u172  u178  u180  u183  u184  u188  u193  u201  u210  u233
u242  u258  u260  u273  u281  u284  u317  u323  u325  u326
u327  u337  u341  u368  u369  u372  u373  u384  u386  u400
u407  u409  u423  u428  u447  u451  u463  u471  u489  u512
u518  u533  u560  u569  u572  u579  u587  u591  u603  u610
u611  u620  u628  u630  u631  u632  u637  u638  u647  u658
u668  u680  u696  u699  u701  u703  u705  u707  u715  u718
```

```
{a, e}    903    {a, e}->{d}
```

Confidence: 167/630, 167/617, 167/600, 167/325, 167/330, 167/301

Figure 5-8. The Information on Supporting Group.

5.5 Summary

The usage behaviour data can be mined to discover the potential concerns that can benefit SOA migration. The mined hidden information can be used for improving the Service model and Component model.

- The process of establishing a Component model is presented. The hierarchical directed acyclic graph is adopted to represent a Component model.
- An association rule mining algorithm is proposed for migrating e-learning legacy systems to an SOA environment. First, some concepts and notations are defined. Then, some theorems and algorithms are described. Finally, some examples are presented for the better understanding of the proposed theorems and algorithms.
- The performance of proposed algorithm is analysed. The experimental data shows that this association rule mining algorithm is applicable.
- An association rule miner has been developed based on the proposed association rule mining algorithm. It can be used to improve Service and Component model.

Chapter 6 Matching Strategies between Legacy Components and Domain Services with Text Similarity Measurement Techniques

Objectives

- To present matching strategy on keyword level.
- To present matching strategy on superficial semantic level.
- To propose a matching algorithm based on keyword level.
- To propose a matching algorithm based on superficial semantic level.

In related SOA migration literatures, little work has been done on how to measure (semi-) automatically the matching relationship between domain services and legacy components. No suitable method or algorithm can be adopted in this study. Thus, two matching algorithms are contributed in this Chapter. The text similarity measurement methods are used for similarity calculation between domain services and legacy components.

After passing the analysis stage, the Service model and Component model have been established. The process will go to the matching stage. In this Chapter, the matching strategies will be described. Concretely, they include weight assignment methods, matching algorithms and its' applications.

6.1 Matching Strategies Based on Text Similarity Measurement Method

In order to create SOA migration planning schemas, the similarity between any node(s) in legacy HDAG and leaf nodes in domain HDAG should be measured. Normally, the SOA migration planning schema includes which node(s) in Component model should be migrated to which node(s) in Service model and the related implementation means (Redevelopment, Modification and Wrapper). Two matching strategies are contributed in this Section.

6.1.1 Keywords-based Level Matching Strategy

The matching relationship between services and components can be measured by adopting text similarity measurement methods of text retrieval. A legacy HDAG is corresponding to a document set (corpus). A domain HDAG is corresponding to a query set. A node of a HDAG is corresponding to a document. Each item in a node of HDAG, such as $Item_{\text{data,}}$, $Item_{\text{son}}$, etc, is corresponding to a passage of a document. Therefore, the problem on the matching relationship measurements between services and components has been changed to the problem on document similarity calculation.

For calculating the documents' similarity, the representation of documents should be established. Normally, a document is represented as a set of keywords.

$$document = \{kw_1, kw_2, \dots, kw_i\}$$

For improving the quality of keyword representation, some domain thesaurus and domain knowledge base are needed for solving problems on synonym, near-synonym, polysemy (lexical ambiguity), etc. If a document is in Chinese language, word cutting is needed for choosing keywords. Clearly, some pre-process work should be done before applying this strategy. Figure 6-1 diagrams the corresponding relationships between SOA migration and text retrieval.

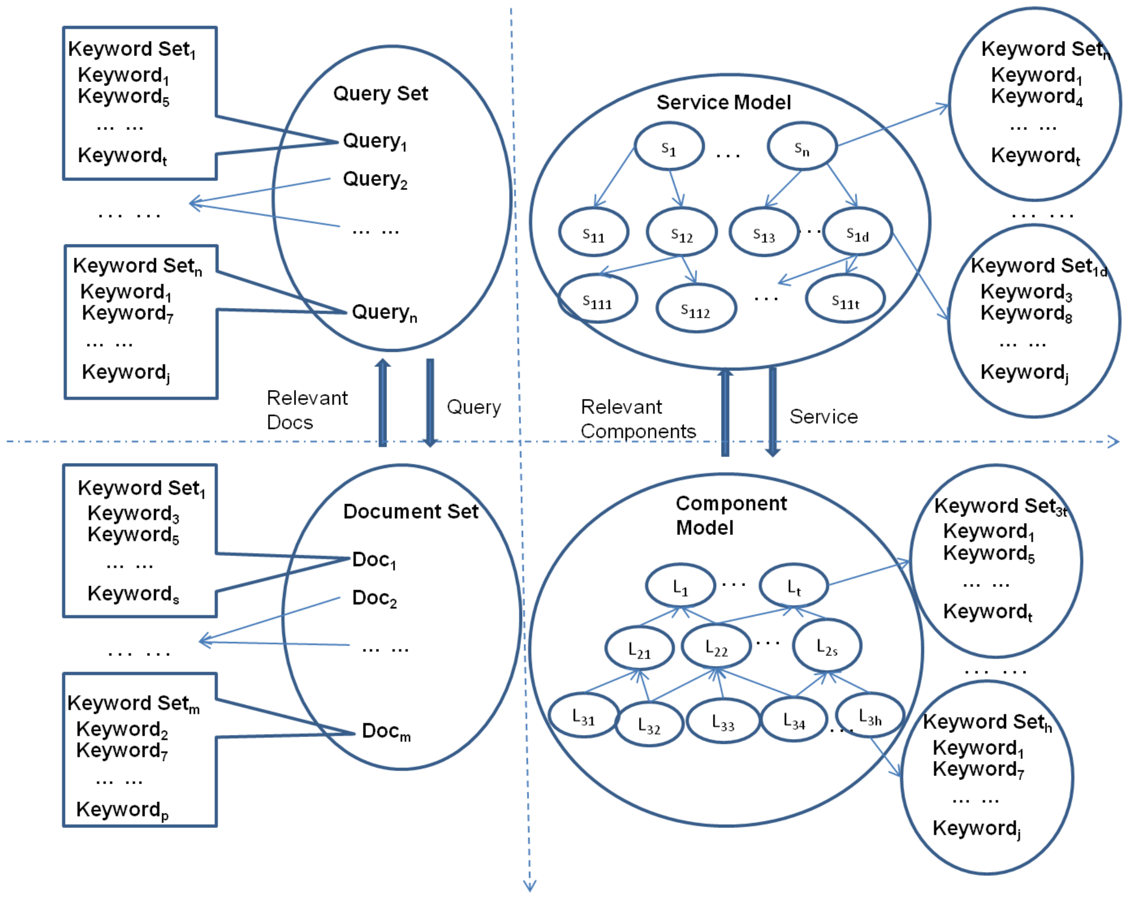


Figure 6-1. The Corresponding Relationships between SOA Migration and Text Retrieval.

6.1.1.1 Similarity Calculation Method Based on Set Operation

Since each node can be represented as a set of keywords, thus, set operation can be used to calculate similarity between nodes.

Set operation refers to any operation with sets, which includes *Union* operator, *Intersection* operator and *Complementation* operator.

Let $Node_{D_i}$ be the i^{th} node in a domain HDAG; let $Node_{L_j}$ be the j^{th} node in a legacy HDAG; Similarity calculation formula based on set operation is as follows,

$$\text{similarity}\left(Node_{D_\mathrm{i}}, Node_{L_\mathrm{j}}\right) = \frac{|Node_{D_\mathrm{i}} \cap Node_{L_\mathrm{j}}|}{|Node_{D_\mathrm{i}} \cup Node_{L_\mathrm{j}}|}$$

For example,

Suppose $Node_{D_1} = \{\mathrm{A, B, C, D}\}$, $Node_{L_3} = \{\mathrm{B, D, E}\}$, then,

$$\text{similarity}\left(Node_{D_1}, Node_{L_3}\right) = \frac{\left|Node_{D_1} \cap Node_{L_3}\right|}{\left|Node_{D_1} \cup Node_{L_{3j}}\right|}$$

$$= \frac{|\{\mathrm{A, B, C, D}\} \cap \{\mathrm{B, D, E}\}|}{|\{\mathrm{A, B, C, D}\} \cup \{\mathrm{B, D, E}\}|} = \frac{2}{5}$$

Usually, the similarity calculation method on set operation is suitable for the keyword number in each node of a HDAG is less than 20. If the keyword number in each node of a HDAG is more than 20, the similarity calculation method on vector space model will be adopted.

6.1.1.2 Similarity Calculation Method Based on Vector Space Model

There are several ways to calculate terms' weights. One of them is *tf*idf* method. Normally, index terms are keywords and phrases. In vector space model, in order to calculate matching relationships, the weight assignment method for each node (document) should be established. Assigning weights for each node (document) refers to assigning weights to keywords consisted in each node (document) since each node (document) is represented as the set of keywords.

In this study, a keyword-based matching algorithm by using set operation method and *tf*idf* method will be investigated. More details will be presented in Section 6.2.

6.1.2 Superficial Semantic-based Level Matching Strategy

Some problems exist in the keyword level, such as it ignores models' and nodes' syntactic information; the semantic information in nodes' items ($Item_{\text{data},}$, $Item_{\text{son}}$, etc.) cannot be used efficiently. In this case, some migration rules and experiences cannot provide directions during the course of SOA migration. Therefore, some syntactic and semantic information should be considered into matching strategies. A matching strategy on superficial semantic matching level will be presented in this study. More details will be presented in Section 6.3.

6.2 A Matching Algorithm Based on Keyword Level

The functionality or its subsets of legacy system can be exposed as services. For doing so, the matching algorithm between domain services and legacy components should be proposed. In this Section, the algebra theory will be taken as a theoretical support frame and the matching algorithm by using text similarity measurement method will be addressed.

6.2.1 The Weight Calculation Method of Nodes in a HDAG

The concrete weight method for nodes in representation of domain logics and legacy assets subjects to the theoretical support frame. In this study, the algebra theory will be taken as the theoretical support frame for describing the weight method of each node. Some information retrieval methods and techniques [111, 48] can be used to calculate the weight of each node in legacy and domain HDAG. A node$_i$ in a HDAG can be represented as a document$_i$, which is represented as a set of relevant keywords.

$$node_i = \{kw_{\mathrm{i}1}, kw_{i2}, \dots, kw_{in}\}$$

The set of nodes in atomic component level, business component level and application component level of a legacy HDAG and nodes in service or composite service level of a domain HDAG can be regarded as a document set. For this document set, keyword-based indexing method and *tf*idf* weight strategy can be adopted for the weight calculation of each node in a domain HDAG and a legacy HDAG. The weight matrix between keyword and node (document) can be obtained. Each node in a HDAG is a vector, which can be denoted as $\overrightarrow{N_s} = \left(\left(\overrightarrow{k_{s1}}, W_{s1}\right), \left(\overrightarrow{k_{s2}}, W_{s2}\right), \dots, \left(\overrightarrow{k_{sm}}, W_{sm}\right)\right)$.

In addition, if the frequencies of keywords in each node of a HDAG are very low, weight value 0 or 1 can be assigned to keywords, namely, if a keyword appears in a node , then assign value 1 as its weight, if not, assign 0 as its weight.

6.2.2 Notations Definition

In order to describe the proposed algorithm clearly, some notations are defined as follows:

- Application component level.
  - Let S be a set of all *nodes* of application component level in a legacy HDAG,

$$S = \{L_1, L_2, \ldots, L_t\}.$$

  - Let T be the element number of S_L, $T=|S|$.
  - For example, in Figure 6-2, $S = \{L_1, L_2, L_3\}$; $T=|S|= 3$.
- Business component level.
  - Let $S(L_i)$ be a set of all *son nodes* of *node* L_i,

$$S(L_i)=\{ L_{i1}, L_{i2}, \ldots, L_{il} \}.$$

  - Let $S_j(L_i)$ be the j^{th} element of the set $S(L_i)$.
  - Let $NPS(S(L_i))$ be the set of all *nonvoid proper subsets* of set $S(L_i)$.
  - Let $NPS_k(S(L_i))$ be the k^{th} *nonvoid proper subset* of set $S(L_i)$.
  - For example, in Figure 6-2, $S(L_1)=\{ L_{21}, L_{22}, L_{23}\}$.

$$S_1(L_1)= L_{21};\ S_2(L_1)= L_{22};\ S_3(L_1)= L_{23}.$$

$$NPS(S(L_1)) =\{\{L_{21}\},\{L_{22}\},\{L_{23}\},\{L_{21}, L_{22}\},\{L_{22}, L_{23}\},\{L_{21}, L_{23}\}\}.$$

$$NPS_2(S(L_1)) =\{L_{22}\}.$$

  - Let S_b be a set of all *nodes* of business component level in a legacy HDAG,

$$S_b=\{L_{b1}, L_{b2}, \ldots L_{bj}, L_{bk}\}.$$

  - Let K be the element number of S_b, $K =|S_b|$.
  - In Figure 6-2, $S_b= \{L_{21}, L_{22}, L_{23}, L_{24}, L_{25}\}$; $K =|S_b|= 5$.
- Atomic component level.
  - Let $A(S_j(L_i))$ be a set of all son nodes of set $S_j(L_i)$.
  - Let $A_k(S_j(L_i))$ be the k^{th} element of the set $A(S_j(L_i))$.
  - $NPS(A(S_j(L_i)))$ be the set of all *nonvoid proper subsets* of set $A(S_j(L_i))$.
  - $NPS_t(A(S_j(L_i)))$ be the t^{th} *nonvoid proper subset* of set $NPS(A(S_j(L_i)))$.
  - In Figure 6-2, $A(S_1(L_1)) = \{\emptyset\}$; $A(S_2(L_1)) = \{L_{31}, L_{32}\}$; $A(S_3(L_1)) = \{L_{32}, L_{34}\}$.

$$A_1(S_2(L_1))= L_{31};\ A_2\ (S_2(L_1))= L_{32.}$$

$$NPS(A(S_2(L_1)))=\{\{L_{31}\},\{L_{32}\}\}.$$

- Service or composite service level.

- Let S_D be a set of all leaf *nodes* in a domain HDAG. The nodes established by using data mining techniques are excluded.

$$S_D = \{S_1, S_2, \ldots, S_m\}$$

- Let M be the element number of S_D, $M = | S_D |$.
- In Figure 6-2, $S_D = \{ S_{31}, S_{32}, S_{33}, S_{25}\}$; $M = | S_D | = 4$.

● Output.

- Let *Ma* be a *similarity matrix*, *Ma*=*M***N*.
- Let *N* be the number of legacy nodes and some of their combination.

$$N = \left| \left(\bigcup_{i=1}^{T} L_i\right) \cup \left(\bigcup_{i=1}^{T} NPS\big(S(L_i)\big)\right) \cup \left(\bigcup_{j=1}^{K}\bigcup_{i=1}^{T} \mathrm{NPS}(\mathrm{A}\left(S_j(L_i)\right))\right) \right|$$

- In Figure 6-2, *Ma*=4*(3+8+10).

6.2.3 Algorithm *SMA-Keyword*

Similarity calculation is used to identify the matching relation between the existing components and domain services. This matching relation bridges the gap between domain and legacy through a set of the business processes aligned with legacy components.

According to the above analysis, a Similarity Matching Algorithm based on Keyword level (*SMA-Keyword*) by using *CosSim* method is addressed. There are five steps in *SMA-Keyword* algorithm [149],

Input: a domain HDAG, a legacy HDAG, threshold λ_{sim}

Output: Similarity Matrix (*M***N*)

Step1: Calculate similarity (for short, [1]$AppSim_{i\text{-}j}$) between a leaf node (S_i) in a Domain HDAG and a node (L_j) of application component level in a Legacy HDAG. // Calculate similarity between each target node and nodes in application component level.

Step2: if $AppSim_{i\text{-}j} \geq \lambda_{sim},$ then calculate similarity (for short, [2]$BusSim_{i\text{-}j}NPS_k$) between the target node (S_i) and each nonvoid proper subset ($NPS_k(S(L_j))$) of the set of son nodes of the node L_j. // Calculate similarity between target nodes and nodes in business component level.

Step3: if $BusSim_{i\text{-}j}NPS_k \geq \lambda_{sim}$, then calculate similarity (for short, $^3AtoSim_{i\text{-}j}NPS_tS_k$) between target nodes ($S_i$) and the t^{th} nonvoid proper subset (for short, $NPS_t(A(S_k(L_j)))$) of the set of son nodes of the k^{th} son node of the node L_j. in atomic component level. // Calculate similarity between target nodes and nodes in atomic component level

Step4: output $AppSim_{i\text{-}j}$, $BusSim_{i\text{-}j}NPS_k$, and $AtoSim_{i\text{-}j}NPS_t$.

Step5: go to Step1 until all target nodes have been processed.

Notes:

$^1AppSim_{i\text{-}j} = cosSim(S_i, L_j)$; // S_i is a set of keywords; L_j is a set of keywords.

$^2BusSim_{i\text{-}j}NPS_k = cosSim(S_i, NPS_k(S(L_j)))$.

$^3AtoSim_{i\text{-}j}NPS_t\,S_k = cosSim(S_i, NPS_t(A(S_k(L_j))))$.

In Algorithm *SMA-Keyword*, if the similarity between a target node and a node in application component level is less than λ_{sim}, the similarity between this target node and all son nodes of the node in business component level do not need be calculated, and assign value 0.0 to it directly. Some examples are shown in Sub-section 6.2.4.

Normally, this similarity matrix ($Ma = M * N$) is a sparse matrix. For further processing, this obtained similarity matrix should be reorganised to a matching relationship Table (for example, Table 6-1) by deleting some legacy parts whose similarities are smaller than the new threshold λ_{match}. Sub-section 7.2.2 shows more details.

6.2.4 An Example on Algorithm *SMA-Keyword*

By running the proposed algorithm on the Component model and Service model shown in Figure 6-2, the similarity matrix ($Ma\_Keyword$=4*21) is obtained.

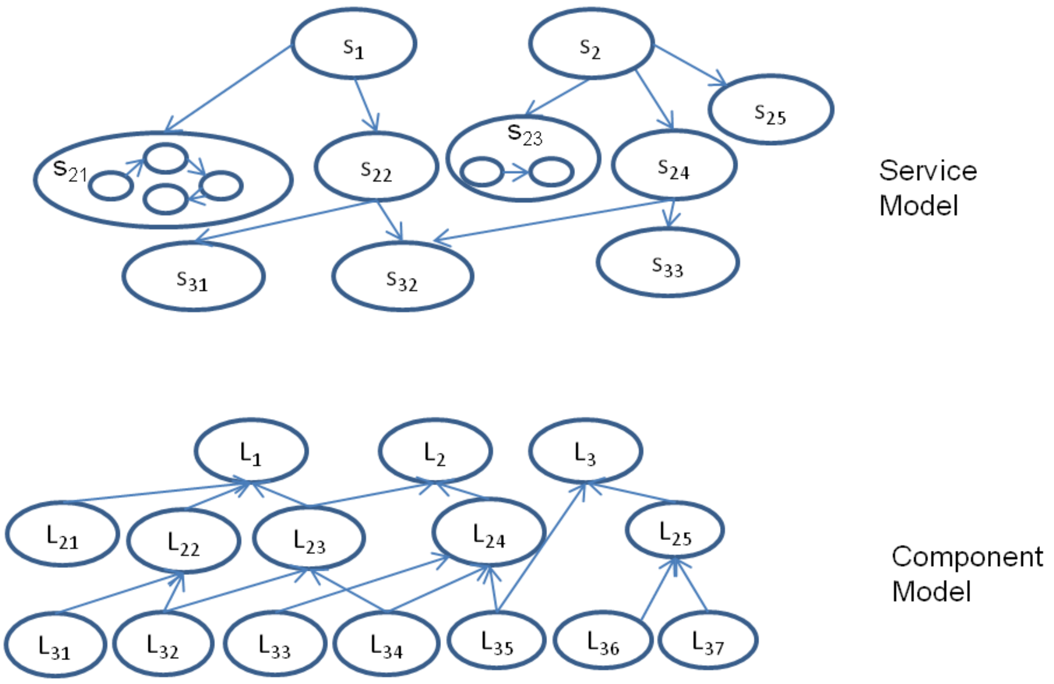


Figure 6-2. An Example of Component model and Service Model.

Similarity matrix (*Ma_Keyword*) on algorithm *SMA-Keyword*, suppose $\lambda_{\text{sim}} = 0.3$:

$$Ma\_Keyword =$$

$$\begin{array}{c|ccccccccccccccccccccc} & L_1 & L_2 & L_3 & L_{21} & L_{22} & L_{23} & L_{24} & L_{25} & L_{31} & L_{32} & L_{33} & L_{34} & L_{35} & L_{36} & L_{37} & L_4 & L_5 & L_6 & L_7 & L_8 & L_9 \\ \hline S_{31} & \mathbf{0.7} & 0.1 & 0.5 & 0.5 & \mathbf{0.6} & 0.2 & 0.0 & \mathbf{0.6} & 0.5 & 0.3 & 0.0 & 0.0 & 0.0 & 0.4 & 0.2 & \mathbf{0.9} & 0.3 & 0.4 & 0.0 & 0.0 & 0.0 \\ S_{32} & 0.2 & \mathbf{0.7} & 0.2 & 0.0 & 0.0 & 0.5 & \mathbf{0.8} & 0.0 & 0.0 & 0.0 & \mathbf{0.8} & \mathbf{0.7} & 0.3 & 0.0 & 0.0 & 0.0 & 0.2 & 0.2 & \mathbf{0.9} & 0.5 & 0.3 \\ S_{33} & 0.1 & \mathbf{0.3} & 0.2 & 0.0 & 0.0 & 0.2 & \mathbf{0.3} & 0.0 & 0.0 & 0.0 & 0.1 & 0.2 & 0.2 & 0.0 & 0.0 & 0.0 & 0.1 & 0.1 & 0.1 & \mathbf{0.3} & 0.1 \\ S_{25} & 0.1 & 0.4 & 0.6 & 0.0 & 0.0 & 0.2 & 0.3 & \mathbf{0.6} & 0.0 & 0.0 & 0.1 & 0.3 & 0.1 & \mathbf{0.5} & 0.4 & 0.0 & 0.1 & 0.1 & 0.2 & 0.2 & 0.1 \end{array}$$

Notes:

$L_4=L_{21}+L_{22}$; $L_5=L_{21}+L_{23}$; $L_6=L_{22}+L_{23}$;

$L_7=L_{33}+L_{34}$; $L_8=L_{34}+L_{35}$; $L_9=L_{33}+L_{35}$

Figure 6-3. Similarity Matrix (*Ma_Keyword*).

In this example,

- There exist combinations of legacy components. Then, how to calculate the similarity in this case? The solution is to combine the keywords in the corresponding item of each node together and generate "a suppositional node". For example, calculate *Similarity*(S_{31}, L_{23}+L_{24}):
    - First, combine L_{23}+L_{24} to a suppositional node L_{23_24} (shown in Figure 6-3),

which is also a set of keywords.

➢ And then, *cosSim*() method is still available for calculating *Similarity*(S_{31}, L_{23_24}).

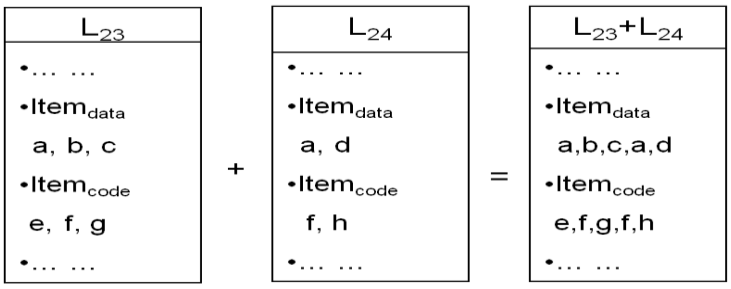


Figure 6-4. An Example of the Combination of Legacy Components.

Table 6-1. The Matching Relationships Based on Keyword Level*.

Domain	Legacy1	*Sim*_Keyword
S_{31}	L_{21}+L_{22}	0.9
	L_1	0.7
	L_{22}	0.6
	L_{25}	0.6
	L_3	0.5
	L_{21}	0.5
	L_{31}	0.5
S_{32}	L_{33}+L_{34}	0.9
	L_{24}	0.8
	L_{33}	0.8
	L_2	0.7
	L_{34}	0.7
	L_{23}	0.5
	L_{34}+L_{35}	0.5
S_{25}	L_3	0.6
	L_{25}	0.6
	L_{36}	0.5

*Note: suppose $\lambda_{match} = 0.5$.

- Table 6-1 presents the matching relationships between domain services and legacy components based on Keyword level.
- Since $BusSim_{31\text{-}1}NPS_2$ = *cosSim* (S_{31}, L_{22}) = 0.6, then, calculate

$AtoSim_{31\text{-}1}S_2NPS_2$ = $cosSim(S_{31}, L_{32})$ =0.3; since $BusSim_{31\text{-}1}NPS_3$ = $cosSim$ (S_{31}, L_{23}) = 0.2< λ_{sim}, then, $cosSim(S_{31}, L_{32})$=0.0, $cosSim(S_{31}, L_{34})$=0.0; in this case, two values (0.0, 0.3) are assigned to $cosSim(S_{31}, L_{32})$, the bigger one will be chosen.

- Since $AppSim_{31\text{-}2}$= $cosSim(S_{31}, L_2)$ =0.1< λ_{sim}, similarities between S_{31} and all L_2's son nodes and sub-son nodes are assigned to "0.0", i.e., L_{23}, L_{24}, L_{33}, L_{34} and L_{35}.
- For target $S_{33,}$ the highest similarity is just 0.3. It seems there is no component(s) matching with it.

6.3 A Matching Algorithm based on Superficial Semantic Level

For improving the measurement quality of matching relationship, a matching algorithm based on superficial semantic level will be discussed.

6.3.1 Analysis

Node information is consisted of information from *Data* item, *Function* item, *UML* item, *Data Mining* item, etc. Among these items, their roles for calculating matching relationship are different. Some of them are necessary conditions (items), which mean any two nodes of a HDAG are similar if and only if the necessary items in these two nodes are similar. For necessary items in nodes of a HDAG, they should be able to control the matching relationships. Thus, these factors should be separated from the others and processed in another ways. The heavy weights should be assigned to them. In keyword level, they are processed in the same ways.

In superficial semantic level, a $node_i$ of a HDAG is represented as a set of *items*,

$$Node_i = \{item_{i1}, item_{i2}, \dots, item_{il}\}$$

where, $item_{il}$ refers to the l^{th} item (passage) in i^{th} node (document)

An *item* is represented as a set of *keywords*,

$$item_i = \{kw_{i1}, kw_{i2}, \dots, kw_{it}\}$$

where, kw_{it} refers to the tth keyword in ith item (passage).

Thus, a *node$_i$* can be represented as a set of the sets of keywords.

$$Node_i = \{\{kw_{i11}, kw_{i12}, \dots, kw_{i1s}\}\{kw_{i21}, kw_{i22}, \dots, kw_{i2u}\}, \dots,$$

$$\{kw_{il1}, kw_{il2}, \dots, kw_{ilv}\}\}$$

where, kw_{ilv} refers to the vth keyword in lth item (passage) in ith node (document).

Each item in a node of HDAG, such as *Item*$_{\text{data},}$, *Item*$_{\text{son}}$, etc, is corresponding to a passage of a document. In a domain node and a legacy node, put the necessary items in the front positions. Figure 6-4 shows the corresponding relationship between a node and a document.

As for which items are necessary factors (items), there are no unified standards. Normally, *Data* item and *Function* item are necessary items for determining matching relationships. Thus, these two items can be regarded as necessary conditions.

In superficial semantic level, the weight of each passage is different. Especially, *Item$_{data}$* is very important. If the similarity between *Item$_{data}$* in Legacy and *Item$_{data}$* in Domain is very low, then, the similarity between the node containing *Item$_{data}$* in a legacy HDAG and the node containing *Item$_{data}$* in a domain HDAG will be zero. According to keyword matching level, in this case, the similarity between these two kinds of nodes may be high since the other items (such as *Item$_{doc}$*, *Item$_{function}$*, *etc.*) contain some identical keywords. In fact, the former is reasonable. Thus, the granularity of the similarity calculation should be a passage instead of a document. Furthermore, some semantic annotations should be attached for automatic semantic matching. For improving the quality of semantic annotations, some domain thesaurus and domain knowledge base are needed. Meanwhile, naming conventions in programming design are preferred to be obeyed during code development and domain requirement establishment. Especially, the names of data sources and main functions should be unified by manual work before adopting this strategy.

Semantic analysis is complex work, which can be applied in a superficial or a deep ways. In this study, a superficial semantic-based strategy will be investigated.

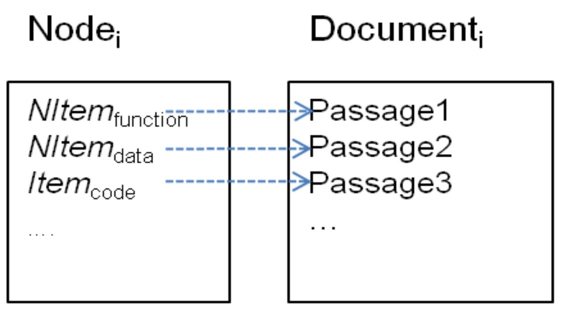


Figure 6-5. The Similar Relationship between a Node and a Document.

6.3.2 Notation Definition

In order to describe the proposed algorithm clearly, some notations are defined as follows.

- Similarity.
    - Let *SimItem*() be the similarity calculation function on necessary item(s) of the matched pairs between Domain and Legacy; let *SimItem* be the returned similarity value, namely, *SimItem* = *SimItem*().
    - Let *SimKeyword*() be the similarity calculation function by using document similarity approach (keyword level) on the necessary items between Domain and Legacy; let *SimKey* be the returned similarity value, namely, *SimKey* = *SimKeyword*().
    - Let *SimSet*() be the similarity calculation function by using set operation approach on the necessary items between Domain and Legacy; let *SimSet* be the returned similarity value, namely, *SimSet*= *SimSet*().
    - Let *SmaKeyword* be the returned similarity value by running algorithm *SMA-Keyword* described in section 6.3.3, namely, *SmaKeyword* = *SMA-Keyword*().
    - Let *SMA-Semantic*() be the similarity calculation function on superficial semantic level between Domain and Legacy; let *SmaSemantic* be the returned similarity value, namely, *SmaSemantic* = *SMA-Semantic*().
- Constant.
    - Let XML_{legacy} be a legacy HDAG; let XML_{domain} be a domain HDAG.
    - Let XML_{match} be a matching relationship Table returned by *SMA-Keyword*

algorithm.

- Let α be the weight coefficient of *SimItem*; Let β be the weight coefficient of *SmaKeyword*.
- Let λ_item be the threshold of *SimItem*.

● Node and Keyword.

- Let Num_D be the number of domain targets in a XML_{match}; Let Num_{Di} be the i^{th} domain target in a XML_{match}.
- Let Num_{Li} be the number of matched pairs between Num_{Di} and legacy component(s) in a XML_{match}; let Num_{Lij} be the j^{th} matched object between Num_{Di} and legacy component(s) in a XML_{match}.
- For example, in Table 6-2,

  Num_D=3; Num_{D1}=S_{31}; Num_{D2}=S_{32}; Num_{D3}=S_{25};

  Num_{L1}=7; Num_{L3}=3; Num_{L1_3}= L_{22}; Num_{L3_2}= L_{25}.

● Item and Keyword.

- Let $DNum_{item_i}$ be the number of necessary item of Num_{Di}; Let $LNum_{item_j}$ be the number of necessary item of Num_{Lij}.
- Let $DItem_i$ be the necessary items of Num_{Di}; let $LItem_j$ be the necessary items of Num_{Lij}.
- Let $DItem_{il}$ be the l^{th} necessary item of $DItem_i$; Let $LItem_{jl}$ be the l^{th} necessary item of $LItem_j$.
- Let Kw_t-$DItem_{il}$ be the t^{th} keyword in $DItem_{il}$; Let Kw_s-$LItem_{jl}$ be the s^{th} keyword in $LItem_{jl}$.
- Let *Set*-$DItem_{il}$ be the set of keywords in $DItem_{il}$; Let *Set*- $LItem_{jl}$ be the set of keywords in $LItem_{jl}$.

● Matrix.

Let *SM* be a similarity matrix between a XML_{domain} and a XML_{legacy}.

According to the above notation definitions, normally, there exist,

$SimItem = SimItem(DItem_i, LItem_j)$.

$SimKey = SimKeyword(DItem_{ij}, LItem_{lj})$.

$SimSet = SimSet(DItem_{ij}, LItem_{lj})$.

$SmaSemantic = \alpha * SimItem + \beta * SmaKeyword$.

$SM(i,j) = SMA\text{-}Semantic(\alpha, \beta, \lambda\_item, XML_{\text{legacy}}, XML_{\text{domain}}, XML_{\text{match}})$.

6.3.3 Algorithm *SMA-Semantic*

According to above analysis, in superficial semantic level, similarity calculation process can be divided into four steps:

Setp1: calculate similarity between each necessary item in domain services and the matched legacy components, namely, *SimItem*(*DItem$_i$*, *LItem$_j$*). Usually, the set operation approach (*SimSet*) can be utilised for the items that the number of their keywords is less than 20. A new keyword similarity approach (*SimKey*) is used for the others.

Step2: obtain the similarity returned by using keyword based level approach described in Sub-section 6.2.3, namely, SmaKeyword = sim_Keyword in Table 6-1.

Step3: combine these two similarities together, namely,

SmaSemantic = SMA-Semantic() =α* SimItem +β* SmaKeyword .

Tuning coefficients α and β can be determined by experiments.

Step4: output the similarity matrix Ma_Semantic.

In the following, a concrete Similarity Matching Algorithm based on superficial Semantic level (*SMA-Semantic*) is addressed.

```
Algorithm SMA-Semantic
Input: $\alpha, \beta, \lambda\_item, \lambda\_keyword, XML_{\text{legacy}}, XML_{\text{domain}}, XML_{\text{match}}$
Output: Similarity Matrix SM-Semantic
Begin
{    for (i=1 to $Num_D$) do
         for (j=1 to $Num_{Li}$) do
         {     $SM$(i,j)=0     //initial similarity matrix
               if ($DNum_{item_i} > 0$ and $LNum_{item_j} > 0$) then
```

```
                {    $SimItem = SimItem(DItem_i, LItem_j)$          //calculate item similarity
                     If ($SimItem > \lambda_{item}$) then
                      {
                        $SmaKeyword = Sim\_Keyword$      //keyword level
                         $SmaSemantic = \alpha * SimItem + \beta * SmaKeyword$    //combination
                        $SM(i,j) = SmaSemantic$       //output $SM$
                      }
                     Else
                      $SM(i,j)=0.0$
                }
            }
    Return $SM$
}
End
```

Sub-algorithm *SimItem*(*DItem$_i$*, *LItem$_j$*)

Input: the necessary items in the ith node in a XML_{domain} ($DItem_i$) and the jth node in a XML_{legacy} ($LItem_j$).

Output: the similarity on necessary items between $DItem_i$ and $LItem_j$

```
Begin
{
    $Sim=0$
    For (s=1 to $DNum_{item_i}$) do
    {    if ($|Set\text{-}DItem_{is}|<20$ or $|Set\text{-}LItem_{js}|<20$) then
               $SimItem = SimSet(Set\text{-}DItem_{is}, Set\text{-}LItem_{js})$ //set operation approach
         else
               $SimItem = SimKeyword(Set\text{-}DItem_{is}, Set\text{-}LItem_{js})$
         $Sim = Sim+SimItem$
    }
    $Sim = Sim/DNum_{item_i}$
    Return $Sim$
}
```

```
End

Sub-algorithm SimKeyword(Set-DItem_is, Set-LItem_js)
Input: Set-DItem_is; Set-LItem_js; the weight of each keyword appeared in Set-DItem_is is
        "1", otherwise , it is "0";  the weight of each keyword appeared in Set-LItem_js is
        "1", otherwise , it is "0".
Output: the similarity between DItem_is and LItem_js
Begin
{
    m=| Set-DItem_is ∪ Set-LItem_js |

    $SimKey = CosSim(DItem_{is}, LItem_{js}) = \frac{\sum_{t=1}^{m} Kw_t DItem_{is} * Kw_t LItem_{js}}{\sqrt{\sum_{t=1}^{m} Kw_t DItem_{is}^2 * \sum_{t=1}^{m} Kw_t LItem_{js}^2}}$

    Return SimKey
}
End

Sub-algorithm SimSet(Set-DItem_is, Set-LItem_js))
Input: Set-DItem_is; Set-LItem_js
Output: the similarity between DItem_is and LItem_js
Begin
{
    $SimSet = \frac{|Set-DItem_{is} \cap Set-LItem_{js}|}{|Set-DItem_{is} \cup Set-LItem_{js}|}$
    Return SimSet
}
End
```

For explaining the algorithm SMA-Semantic clearly, an example is depicted in the following Sub-section.

6.3.4 An Example on Algorithm *SMA-Semantic*

By running the proposed algorithm *SMA-Semantic* on the Component model and Service model shown in Figure 6-2, a matching relationship Table shown in Table 6-1, and suppose $\lambda_{item} = 0.7$, $\alpha = 0.7$, $\beta = 0.3$, the new matching relationship Table shown in Table 6-2 is obtained.

Table 6-2. The Matching Relationships Based on Keyword and Semantic level*.

Domain	Legacy1	*Sim_Keyword*	*Sim_Semantic*
S_{31}	L_{21}+L_{22}	0.9	0.96
	L_1	0.7	0.81
	L_{22}	0.6	0.73
	L_{25}	0.6	0.0
	L_3	0.5	0.0
	L_{21}	0.5	0.68
	L_{31}	0.5	0.0
S_{32}	L_{33}+L_{34}	0.9	0.8
	L_{24}	0.8	0.73
	L_{33}	0.8	0.87
	L_2	0.7	0.63
	L_{34}	0.7	0.0
	L_{23}	0.5	0.0
	L_{34}+L_{35}	0.5	0.0
S_{25}	L_3	0.6	0.6
	L_{25}	0.6	0.81
	L_{36}	0.5	0.64

*Note: $\lambda_{item} = 0.5$.

In this example,

- For target S_{31}, the *simItem* between S_{31} and its matched objects is shown in Table 6-3. The *simItems* of L_{25}, L_3 and L_{31} are less than $\lambda_{item} = 0.7$, their *Sim_Semantic* =0.0. Therefore, after passing SMA-Semantic algorithm, the size of matching relationship Table is reduced again. The rank in the order of *Sim_Semantic* is the same with the order of *Sim_Keyword.*

Table 6-3. The *SimItems* between S_{31} and its Matched Objects.

Domain	Legacy	*SimItem*
S_{31}	$L_{21}+L_{22}$	0.98
	L_1	0.85
	L_{22}	0.78
	L_{25}	0.2
	L_3	0.22
	L_{21}	0.75
	L_{31}	0.3

- For target S_{32}, *Sim_Semantic*(S_{32}, L_{33}) is more than *Sim_Semantic*(S_{32}, $L_{33}+L_{34}$) since *simItem*(S_{32}, L_{33}) is greater than *simItem*(S_{32}, $L_{33}+L_{34}$). The rank in the order of *Sim_Semantic* is different from the order of *Sim_Keyword*.
- Table 6-3 presents the matching relationships between domain services and legacy components based on Semantic level.

6.4 Supporting Tool

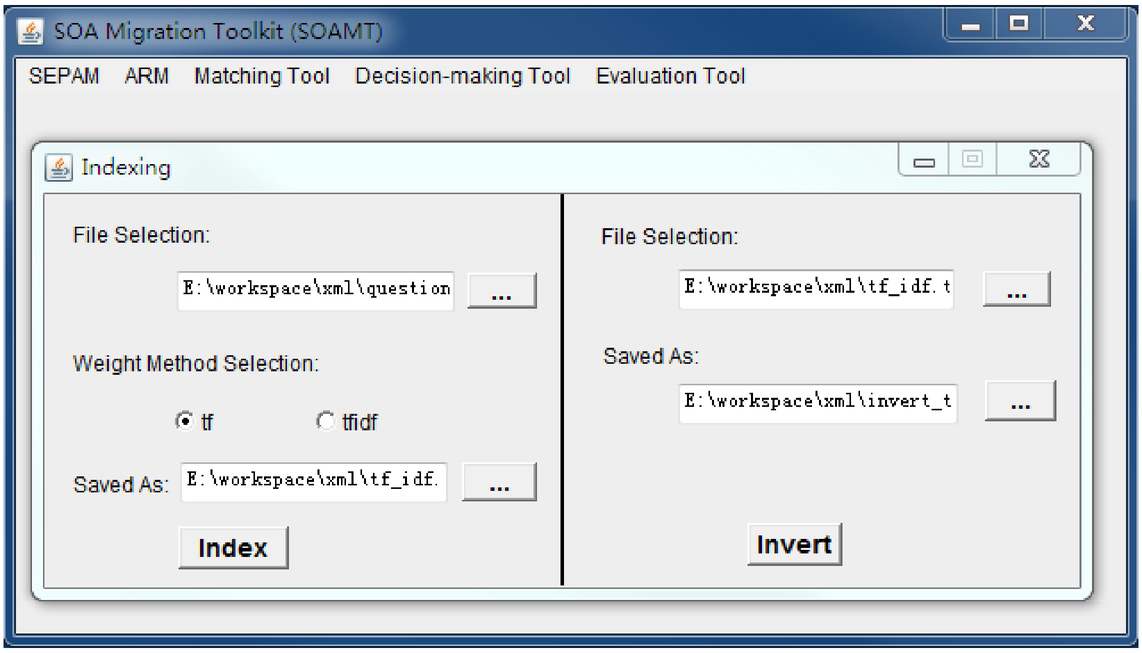
6.4.1 Matching Tool

A matching tool is developed based on the proposed matching strategies and algorithms. The matching tool is developed under the support of Lucene software package [88], which is a Java full-text search engine. However, it is not a complete application and just API can be used to add search capabilities to applications.

In matching tool, there are two functions available: indexing function and similarity calculation function.

6.4.2 Indexing Function

The main interface of this indexing function is shown in Figure 6-5. Users can choose weight methods of indexing terms and invert the indexing files into the inverted indexing files.



File Selection: E:\workspace\xml\question

Weight Method Selection: [✓ tf] tfidf

Saved As: E:\workspace\xml\tf_idf.

File Selection: E:\workspace\xml\tf_idf.t

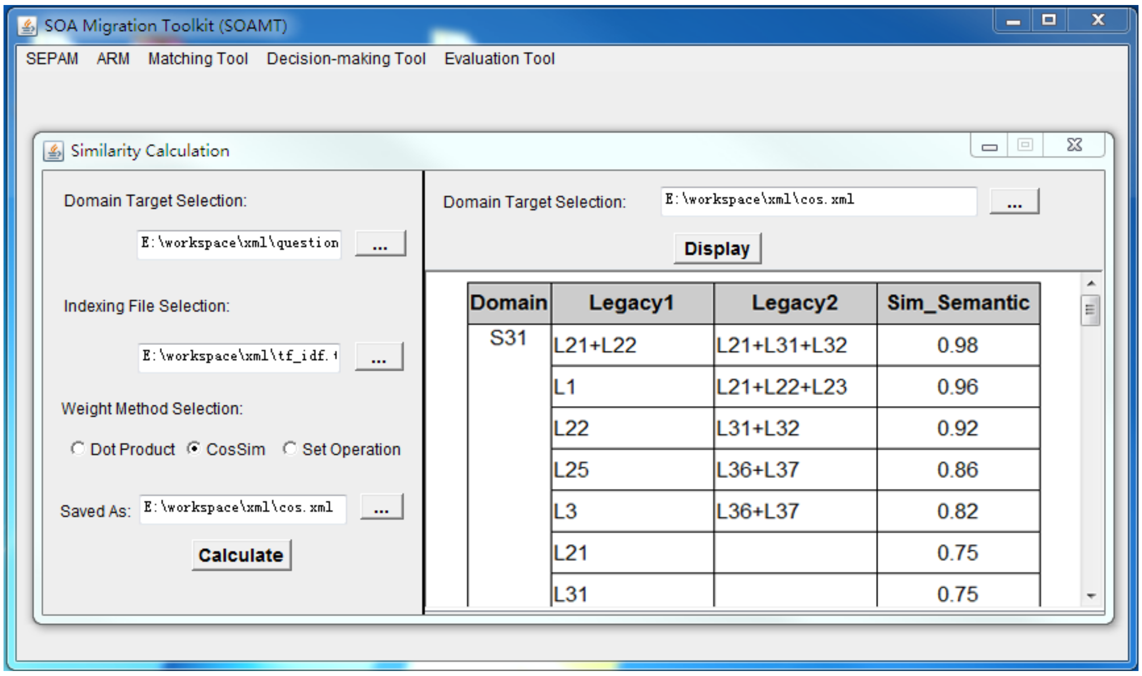
Saved As: E:\workspace\xml\invert_t

Figure 6-6. The Interface of the Indexing Function.

6.4.3 Similarity Calculation Function

Three similarity calculation methods (dot/inner product method, cosine similarity method and set operation method) and two weight assignment methods (*tf* method and *tf*idf* method) of indexing terms have been implemented in this function. The interface of similarity calculation function is shown in Figure 6-6.

The categories of saved indexing files include: indexing document with frequency number (for short, DocIndex), indexing file with *tf* weight method (for short, DocIndex (*tf*)), indexing file with *tf*idf* weight method (for short, DocIndex (*tfidf*)), inverted indexing file with *tf* weight method (for short, DocInvert (*tf*)) and inverted indexing file with *tf*idf* weight method (for short, DocInvert (*tfidf*)). The format of saved indexing files is shown in Table 6-4.



Domain Target Selection: E:\workspace\xml\question

Indexing File Selection: E:\workspace\xml\tf_idf.

Weight Method Selection: Dot Product [✓ CosSim] Set Operation

Saved As: E:\workspace\xml\cos.xml

Domain Target Selection: E:\workspace\xml\cos.xml

Domain	Legacy1	Legacy2	Sim_Semantic
S31	L21+L22	L21+L31+L32	0.98
	L1	L21+L22+L23	0.96
	L22	L31+L32	0.92
	L25	L36+L37	0.86
	L3	L36+L37	0.82
	L21		0.75
	L31		0.75

Figure 6-7. The Interface of Similarity Calculation Function.

Table 6-4. The Format of Indexing Files.

Files	Formats		
DocIndex	DocID	Frequency Number	Term/Word
DocIndex(tf)	DocID	Weight(tf)	Term/Word
DocIndex(tfidf)	DocID	Weight(tfidf)	Term/Word
DocInvert(tf)	Term/Word	Weight(tf)	DocID
DocInvert(tfidf)	Term/Word	Weight(tfidf)	DocID

6.5 Summary

Two matching strategies are described, which include weight assignment methods, matching algorithms and its' applications.

- Two matching strategies are described: one is keyword- based level, two similarity calculation methods (set operation method and vector space model method) are presented in this matching strategy; another one is superficial semantic-based level, necessary items are taken into account in this matching strategy.
- A matching algorithm based on keyword level is presented. Document similarity calculation method (*CosSim*) based on vector space model has been adopted to calculate the similarity between each node in a Legacy and in a domain HDAG.
- A matching algorithm based on superficial semantic level is presented. The corresponding relationships between a node in a domain or a legacy HDAG and a document in text retrieval are established. The concept of necessary item has been defined. The granulation of similarity calculation is cut to “passage”. The final similarity is the combination of similarity on necessary item and similarity on keywords.

Chapter 7 Creation and Evaluation of SOA Migration Planning Schemas

Objectives

- To define the SOA migration planning schema.
- To propose a decision-making method based on similarity matrix for SOA migration planning schemas.
- To propose a decision-making method based on hybrid information for SOA migration planning schemas.
- To propose evaluation methods on the SOA migration planning schemas.
- To represent an example for further described the proposed methods.

The returned results of the proposed SOA migration approach are SOA migration planning schemas, which can provide planning and predictable functions to persons who are facing problems such as whether or not to trigger an SOA migration project. In the above Chapters, a lot of preparation work has been done for creating SOA migration planning schemas. In this Chapter, the methods of creating and evaluating SOA migration planning schemas will be proposed.

7.1 SOA Migration Planning Schemas

7.1.1 Analysis of SOA Migration Planning Schema

In this Section, the questions, such as “what should be included in an SOA migration planning schema?” and “how can SOA migration planning schemas be created?”, will be answered.

For doing predictions, the SOA migration planning schemas should include a migration source object, a migration target object, a matching degree and implementation means. Migration source object refers to legacy components in the Component model; and migration target object refers to services in the Service model. Matching degree refers to the similarity that is similar to each other. Implementation means refers to how to implement this migration. Normally, there are three kinds of implementation means, namely, Wrapper means, Modification means and Redevelopment means. The choice of implementation means is subject to the user's directions and matching degree:

- Wrapper means

  It refers to the higher matching degree, normally, named "match". It requires developers to comprehend the components' interfaces instead of their internals.

- Modification means

  It refers to the medium matching degree, normally, named "part of match". Normally, it requires developers to reuse components by changing some part(s) of their internals.

- Redevelopment means

  It refers to the lower matching degree, normally, named "no match". This part needs to be developed from scratch.

The creation of SOA migration planning schemas is complex work. Decision-making amongst user's requirements, implementation cost, system performance, system functions, and so on, should be concerned. For the different users' requirements, there are different SOA migration planning schemas even though they are in the same project. The creation of SOA migration planning schemas is similar to service orchestration. In practice, most of them require user's direction.

7.1.2 User's Direction

From the user's perspective, the user's direction is a task assigned to users and it presents the users' requirements. The users can express what they wish to acquire through the user's direction. User's direction can be used for determining SOA migration implementation means. In an SOA migration situation, normally, users pay most attention on cost aspects, function aspects, performance aspects, etc.

In this study, user’s direction includes function-first, cost-first and performance-first. The function-first means users prefer more functions to the others. The cost-first means users prefer the cheapest cost to the others. The performance-first means users prefer the best performance to the others. SOA migration implementation means refer to Wrapper means, Modification means and Redevelopment means.

7.1.3 Methods on Creating SOA Migration Planning Schemas

In related SOA migration literature, little work has been done on how to draft the SOA migration planning schemas. No concrete method and algorithm is available for this research. Thus, three proposed methods for creating SOA migration planning schemas based on the analysis and descriptions in Chapters 4, 5 and 6 are presented in this Sub-section. One is a data mining method. The second one is the decision-making method based on similarity matrix. The third one is the decision-making method based on hybrid information. These methods are established from the practical, functional and reusable (non-functional) viewpoints respectively. The final SOA migration planning schemas are created by the combination of these three methods under the users’ directions. There are still some other viewpoints which are also important for SOA migration projects. However, due to time restrictions they are not considered in this study.

7.1.3.1 Data Mining Method

Data mining techniques can be used to discover the hidden information. The mined information can perfect the establishment of a Service model and a Component model. For those services and business processes that are established by using the mined information, they themselves are SOA migration planning schemas from the viewpoint of users’ practices. This kind of SOA migration planning schemas can be named practical SOA migration planning schemas.

For example, Nodes S_{21} and S_{23} in the Service model shown in Figure 4-4 have been established by the sequence pattern mining method. Thus, the SOA migration planning schemas for them can consist of sequence pattern mining results and wrapper means directly. In this way, the number of leaf nodes in a domain HDAG is reduced. In this example, just nodes S_{31}, $S_{32,}$ S_{33} and S_{25} (S_{21} and S_{23} are excluded) should be processed by using decision-making methods. Clearly, the efficiency for creating SOA migration planning schemas will be improved.

7.1.3.2 Similarity Matrix Method

Similarity matrix method is to use text retrieval techniques for determining functional matching relationships between domain services and legacy components. The SOA migration planning schemas are created from the viewpoint of the functional factors. This kind of SOA migration planning schemas can be named functional SOA migration planning schemas.

7.1.3.3 Hybrid Information Method

Hybrid information method is to create SOA migration planning schemas from the hybrid information, such as legacy software quality and so on. In this study, just the reusability attribute is considered. This kind of SOA migration planning schemas can be named hybrid SOA migration planning schemas.

In the following Sections, more descriptions on the last two methods will be presented.

7.1.4 Architecture of Creation and Evaluation of SOA Migration Planning Schemas

In order to validate the proposed methods, evaluation work on SOA migration planning schemas is also necessary. Figure 7-1 shows the architecture of creation and evaluation of SOA migration planning schemas.

The main work in this architecture includes:

- data mining method, which is described in Chapter 4 and Chapter 5;
- similarity matrix method, which is presented in Chapter 6 and Section 7.2;
- hybrid information method, which is presented in Section 7.3;
- user’s direction, which is introduced in Section 7.1.2;
- evaluation method, which is shown in Section 7.4.

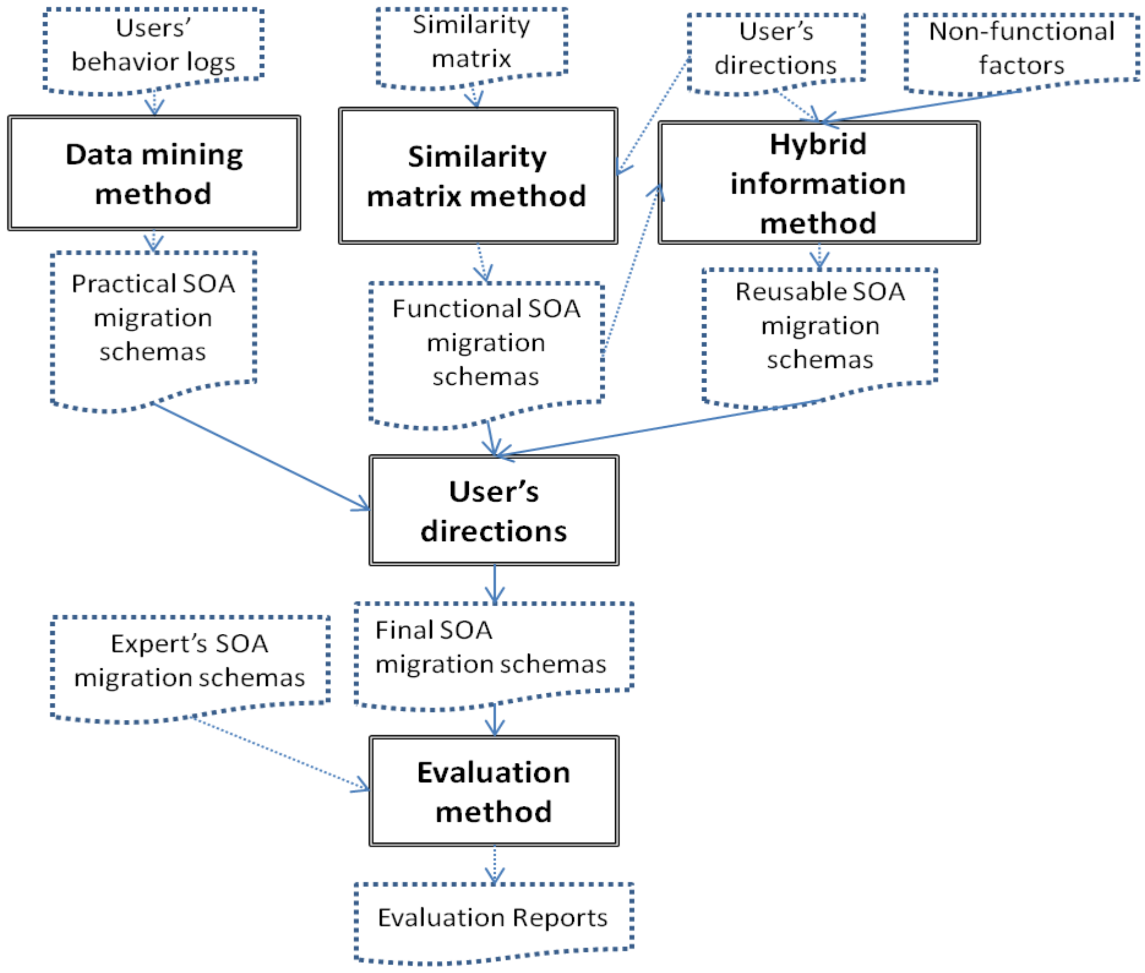

Users' behavior logs
Similarity matrix
User's directions
Non-functional factors
Data mining method
Similarity matrix method
Hybrid information method
Practical SOA migration schemas
Functional SOA migration schemas
Reusable SOA migration schemas
User's directions
Expert's SOA migration schemas
Final SOA migration schemas
Evaluation method
Evaluation Reports

Figure 7-1. The Architecture of Creation and Evaluation of SOA Migration Planning Schemas.

7.2 A Decision-making Method based on Similarity Matrix

After obtaining the matching relation between legacy components and domain services, the SOA migration planning schema will be created according to the user's directions. The decision-making process is an uncertainty reasoning process. Some uncertainty reasoning methods and heuristic search algorithms can be adopted in this stage.

Through running a matching algorithm, the similarity matrix *Ma* between components and domain services has been obtained. In order to determine migration schemas accurately, this similarity matrix *Ma* should be analysed further. *Ma* is a *M** *N* matrix where *M* is the set of services in a domain HDAG and *N* is the set of components and their kinds of combination in a legacy HDAG. In this Section, a Decision-Making (DM) method based on similarity matrix will be presented.

7.2.1 Symbol Definition

First, some symbols will be defined for describing the DM method.

- Let *Descend-rank* (E_i) represent the ranked elements in the i^{th} row of the matrix *Ma* in descendant order and output a *1*N* matrix (denoted as E_i).
- Let *Cut* (E_i, k) represent cutting *1*N* matrix E_i to a *1*k* matrix, E_i = *Cut* (E_i, k), where parameter k is subject to λ_{match}.
- Let $E_i(t)$ represent the returned value in the t^{th} column of the matrix E_i.

In order to explain these symbols clearly, an example is provided. Figure 7-2 illustrates the dimensionality reduction process (k=3). The left hand side of Figure 7-2 is the original similarity matrix; the right hand side of Figure 7-2 is the result of dimensionality reduction, which can be reorganised into a matching relationship Table (e.g., Table 7-1).

$$Ma = \begin{array}{c|cccc} & L_1 & L_2 & L_3 & L_4 \\ \hline D_1 & 0.2 & 0.1 & 0.4 & 0.3 \\ D_2 & 0.8 & 0.7 & 0.9 & 0.5 \\ D_3 & 0.3 & 0.8 & 0.6 & 0.4 \end{array} \Rightarrow \begin{cases} E_1: \begin{array}{c|ccc} & L_3 & L_4 & L_1 \\ D_1 & 0.4 & 0.3 & 0.2 \end{array} \\ E_2: \begin{array}{c|ccc} & L_3 & L_1 & L_2 \\ D_2 & 0.9 & 0.8 & 0.7 \end{array} \\ E_3: \begin{array}{c|ccc} & L_2 & L_3 & L_4 \\ D_3 & 0.8 & 0.6 & 0.4 \end{array} \end{cases}$$

Figure 7-2. An Example for the Dimensionality Reduction (*k*=3) for Similarity Matrix *Ma.*

Table 7-1. The Matching Relationship Table of Figure7-2.

Domain	Legacy	*Sim_Keyword*
E_1	L_3	0.4
	L_4	0.3
	L_1	0.2
E_2	L_3	0.9
	L_1	0.8
	L_2	0.7
E_3	L_2	0.8
	L_3	0.6
	L_4	0.4

7.2.2 A Proposed Method

The steps of a DM method based on similarity matrix are as follows [148]:

Step1: The dimensionality reduction will be done to matrix *Ma*. Just continue to have the first k matched legacy components that have the high similarities with domain services. The others will be neglected. Thus, matrix *Ma* (*M** *N*) can be changed to M matrices (1*k), in which each matrix (denoted as Ma_i) is a *1*k* matrix. For the *Ma_Keyword* matrix shown in Figure 6-3, there are,

- **Rank**

  $Ma_1 = Descend_{rank(Ma\_Keyword_{31})} =$

$$\begin{array}{c|cccccccccccc} & L_{21}+L_{22} & L_1 & L_{22} & L_{25} & L_3 & L_{21} & L_{23} & L_{36} & L_6 & L_{32} & \dots \\ S_{31} & (\ 0.9 & 0.7 & 0.6 & 0.6 & 0.5 & 0.5 & 0.5 & 0.4 & 0.4 & 0.3 & \dots) \end{array}$$

- **Dimensionality reduction**

  $Ma_1 = Cut(Ma_1, 7) = Descend_{rank(S_{31}, Ma\_Keyword)} =$

$$\begin{array}{c|ccccccc} & L_{21}+L_{22} & L_1 & L_{22} & L_{25} & L_3 & L_{21} & L_{23} \\ S_{31} & (\ 0.9 & 0.7 & 0.6 & 0.6 & 0.5 & 0.5 & 0.5\ ) \end{array}$$, where $\lambda_{match} = 0.5$, k=7.

- **Value**

  $Ma_1(2) = 0.7$.

Step2: Reorganising these matrices into a matching relationship table (see Table 6-2). Moreover, adding the equivalent parts for those remaining in legacy aspect into the matching relationship table. As for the equivalent part, in Figure 6-2, $AppSim_{31\text{-}1} = cosSim(S_{31}, L_1) = cosSim(S_{31}, L_{21}+L_{22}+L_{23})$; "$L_{21}+L_{22}+L_{23}$" is the equivalent part of "L_1". These two parts own the same similarity with the matched target in the keyword level. However, these two may be different in the hybrid information level. In creating migration schemas, these two parts are alternative in the keyword level. The one that has the higher value of hybrid information is the better one in the hybrid information level (see Section 7.3). An example shown in Table 6-2 is changed to Table 7-2 by adding Legacy2 Column.

Table 7-2. The Matching Relationships Based on Keyword and Semantic level*.

Domain	Legacy1	Legacy2	*Sim_Keyword*	*Sim_Semantic*
S_{31}	L_{21}+L_{22}	L_{21}+L_{31}+ L_{32}	0.9	0.96
	L_1	L_{21}+L_{22}+ L_{23}	0.7	0.81
	L_{22}	L_{31}+ L_{32}	0.6	0.73
	L_{25}	L_{36}+ L_{37}	0.6	0.0
	L_3	L_{36}+ L_{37}	0.5	0.0
	L_{21}		0.5	0.68
	L_{31}		0.5	0.0
S_{32}	L_{33}+L_{34}		0.9	0.8
	L_{24}	L_{33}+L_{34}+ L_{35}	0.8	0.73
	L_{33}		0.8	0.87
	L_2	L_{23}+ L_{24}	0.7	0.63
	L_{34}		0.7	0.0
	L_{23}	L_{32}+ L_{34}	0.5	0.0
	L_{34}+L_{35}		0.5	0.0
S_{25}	L_3	L_{25}+ L_{35}	0.6	0.6
	L_{25}	L_{36}+ L_{37}	0.6	0.81
	L_{36}		0.5	0.64

*Note: Column *legacy2* is the equivalent part of Column *legacy1*;

$\lambda_{item} = 0.5$.

Step3: Obtaining the available matching relationships. For each domain target, the remaining matched objects in Legacy1 or Legacy2 Column will be the candidate SOA migration sources. The matching relationship with the highest similarity (keyword or semantic) will be the most wanted from the view of comprehensive considerations.

Step4: Determining the implementation means. The implementation means of an SOA migration planning schema is classified into three categories: wrapper, which provides the new interface for the existing data, programs, applications and interfaces; modification, which moves the modified assets (with fewer changes) to a new architecture; and re-development, which develops requirements from scratch. The user's directions (cost-first, function-first and performance-first) include three thresholds ($\lambda_1 > \lambda_2 > \lambda_3$). At first, the matching relationship between the implementation means and the user's direction can be decided by experienced professionals. After obtaining the evaluation results on cost aspects and performance aspects of the proposed SOA migration planning schemas, this matching relationship can be revised again. Table 7-3 shows an example of the mapping relations between the implementation means and the user's directions.

Table 7-3. The Mapping Relation between the Implementation Means and the User's Direction.

User's direction	Similarity of a matching	Implementation means
Cost-first	$Sim \geq \lambda_2$	Wrapper
	$\lambda_2 > Sim \geq \lambda_3$	Modification
	Others	Re-development
Function-first	$Sim \geq \lambda_3$	Wrapper
	Others	Re-development
Performance-first	$Sim \geq \lambda_1$	Wrapper
	$\lambda_1 > Sim \geq \lambda_2$	Modification
	$Sim < \lambda_2$	Re-development

Step5: Integrating SOA migration planning schemas. The candidate SOA migration planning schemas should be created according to the user's direction.

7.2.3 An Example

According to the *Sim_Keyword*, suppose$\lambda_1 = 0.8, \lambda_2 = 0.6,$ and $\lambda_3 = 0.5,$ the SOA migration planning schemas for the matching relationships shown in Table 7-2 and the user's directions shown in Table 7-3 are shown in Table 7-4.

Table 7-4. An Example on Keyword Level.

Matching Relationships				[3]Implementation		
[1]Domain	[2]Legacy1	Legacy2	*Sim_Keyword*	User's		
				[4]CF	PF	FF
S_{31}	$L_{21}+L_{22}$	$L_{21}+L_{31}+$	0.9	W	W	W
	L_1	$L_{21}+L_{22}+$	0.7	W	M	W
	L_{22}	$L_{31}+ L_{32}$	0.6	W	M	W
	L_{25}	$L_{36}+ L_{37}$	0.6	W	M	W
	L_3	$L_{36}+ L_{37}$	0.5	M	R	W
S_{32}	$L_{33}+L_{34}$		0.9	W	W	W
	L_{24}	$L_{33}+L_{34}+$	0.8	W	W	W
	L_{33}		0.8	W	W	W
	L_2	$L_{23}+ L_{24}$	0.7	W	M	W
	L_{34}		0.7	W	M	W
S_{25}	L_3	$L_{25}+ L_{35}$	0.6	W	M	W
	L_{25}	$L_{36}+ L_{37}$	0.6	W	M	W
	L_{36}		0.5	M	R	W

Note: [1] for the targets that do not appear in the Domain Column, its implementation means are "Re-development", e.g., target S_{33} in Figure 6-2.

[2]for each target, just first five candidate SOA migration planning schemas are listed in this Table.

[3]W--Wrapper; R-- Re-development; M—Modification.

[4]CF--Cost-First; PF-- Performance-First; FF--Function-First.

According to the *Sim_Semantic*, suppose$\lambda_1 = 0.8, \lambda_2 = 0.7,$ and $\lambda_3 = 0.5,$ the SOA migration planning schemas for the matching relationships shown in Table 7-2 and the user's directions shown in Table 7-3 are shown in Table 7-5.

Table 7-5. An Example on Semantic Level.

Matching Relationships				[3]Migration		
[1]Domain	[2]Legacy1	Legacy2	*Sim_Semantic*	User's		
				[4]CF	PF	FF
S_{31}	$L_{21}+L_{22}$	$L_{21}+L_{31}+$	0.96	W	W	W
	L_1	$L_{21}+L_{22}+$	0.81	W	W	W
	L_{22}	$L_{31}+L_{32}$	0.73	W	M	W
	L_{25}	$L_{36}+L_{37}$	0.68	M	R	W
S_{32}	L_{33}		0.87	W	W	W
	$L_{33}+L_{34}$		0.8	W	W	W
	L_{24}	$L_{33}+L_{34}+$	0.73	W	M	W
	L_2	$L_{23}+L_{24}$	0.63	M	R	W
S_{25}	L_{25}	$L_{36}+L_{37}$	0.81	W	W	W
	L_{36}		0.64	M	R	W
	L_3	$L_{25}+L_{35}$	0.6	M	R	W

Note: [1] for the targets that do not appear in the Domain Column, its implementation means are "Re-development", e.g., target S_{33} in Figure 6-2.

[2]for each target, just first five candidate SOA migration planning schemas are listed in this Table.

[3]W--Wrapper; R-- Re-development; M—Modification.

[4]CF--Cost-First; PF-- Performance-First; FF--Function-First.

In this example,

- In the keyword and semantic level, the candidate migration schemas for some targets are different. The migration source objects of most targets are not changed, but, the similarities and the implementation means are different for some of them. For example, target S_{32}, its matched object is L_2; in keyword level, its similarity is 0.7, and implementation means in “Cost-first” is “Wrapper”; in semantic level, its similarity is 0.63, and implementation means in “Cost-first” is “Modification”.

- The number of candidate migration schemas in the semantic level is less than in the keyword level.

Therefore, it seems the constraints in a semantic level are stricter than in a keyword level.

7.3 An Optimal Decision-making Method Based on Hybrid Information

7.3.1 Analysis

Through analysing the candidate migration planning schemas created on the above proposed algorithm, some problems are exposed. It seems that the items (factors) contained in each node of a HDAG, which are from the angle of logical and relevant functional information, are not enough for ensuring the migration implementation. They are still the high-level descriptions of components. In fact, it is impossible to implement some of the proposed SOA migration planning schemas because of the poor quality of legacy software components. More attention should also be paid to the implementation of proposed migration planning schemas. More information from the angle of component quality should be considered because the high quality software component can benefit the SOA migration implementation. Therefore, the creation of final SOA migration planning schemas should consider both functional factors (such as functional similarity) and non-functional factors (such as quality of legacy software components).

In this study, the proposed approach is suited for the legacy software system that can be decomposed into components. Obviously, qualitative analysis and quantitative calculation on component quality should be concerned. The questions of “what indicators should be measured for the purpose of component reuse?”, “how to measure each one?”, “how to combine these indicators together for choosing the best target?”

and the like should be considered. An optimal method based on hybrid information will be described.

7.3.2 A Proposed Method

An optimal decision-making method based on hybrid information is as follows:

Step1: collecting the hybrid information for candidate SOA migration planning schemas

1.1 identify the evaluation indicators (factors specified in ISO 9126 can be candidates) for legacy software components from the point of view of reuse;

1.2 determine the methods to measure each indicator;

1.3 present a new evaluation method by combining these indicators together.

Step2: for each matched legacy component in the candidate SOA migration planning schemas, measure the hybrid information according to the established methods in Step1.

Step3: input the first *n* candidate SOA migration planning schemas ranked by descending order for a domain target; combine the hybrid information method and the original similarity matrix method together.

Step4: determine the implementation means according to the new ranking value; and output the re-ranked *n* optimal SOA migration planning schemas for the domain target. For Legacy1 and Legacy2 Column, the one that has the higher value of hybrid information (VHI) is the better one in the hybrid information level.

Step5: go to Step2 until all domain targets have been processed.

According to the proposed method, an example will be presented and each step (instantiation) is detailed.

7.3.3 An Example of the Proposed Method

The information on indicators of component quality measurement (such as reusability, adaptability, compose-ability, performance, maintainability, security, etc.) should be saved in each node of Component models (legacy HDAG). For the convenience of the proposed method's description and instantiation (illustration), the coupling degree, cohesion degree and adaptability will be chosen as indicators in this example. Generally, in SOA migration situation, the higher internal cohesion, the lower external coupling and high adaptability in the legacy software components are desired.

The details of the main steps in the proposed method will be presented in this part. The candidate SOA migration planning schemas shown in Table 7-5 will be taken as an example.

Step1: collecting the hybrid information for SOA migration planning schemas.

1.1 The component coupling degree measure (COUM), cohesion degree measure (COHM) and adaptability measure (ADAM) will be chosen as evaluation indicators because it is available to analyse source codes in the case study.

1.2 Little work has been done on cohesion, coupling and adaptability measurement methods since it is not the main research question in this study. In addition, it is time-consuming and costly to obtain some attributes of legacy components. Thus, a cost-effective problem-solving strategy is adopted for estimating these indicators. In this study, the estimation results in the case study are obtained from some developers of the legacy software system (some developers are still available). For simplifying processing, the estimation results of them are roughly divided into three levels: low, average and high; or, bad, average and good.

1.3 Table 7-6 shows the combination method of COUM results and COHM results (named CC-method). The combination method of CC-method results and the ADAM results is named CCA-method. For ranking the SOA migration planning schemas, the CCA-method results should be proposed as a numeric measurement. For doing this, the CC-method results and ADAM results should be assigned numeric values. They can be quantified in different ways, such as maximum and minimum values, or statistical values, etc. A simple method is adopted in this example: the range of measured degrees is from “0” to “2” with three degrees, namely, assign “2” to “good”; assign “1” to “average” and assign “0” to “bad”. Let CCQ be the CC-method quantitative results; let ADAMQ be quantitative ADAM results; let CCAQ be the CCA-method quantitative results; let *Max*CC be the maximum number in CC-method quantitative results; let *Max*ADAM be the maximum number in ADAM quantitative results. The calculation formula for

CCAQ is as follows,

$$\text{CCAQ} = \frac{\text{CCQ} + \text{ADAMQ}}{Max\text{CC} + MaxADAM}$$

Table 7-6. CC-method.

COUM Results	COHM Results	CC-method Results
High(Bad)	High(Good)	Average
High(Bad)	Average	Bad
High(Bad)	Low (Bad)	Bad
Average	High(Good)	Average
Average	Average	Average
Average	Low (Bad)	Bad
Low(Good)	High(Good)	Good
Low(Good)	Average	Good
Low(Good)	Low (Bad)	Average

According to this simple assignment scheme, Table 7-7 shows the CCAQ results. CCAQ is from “0” to “1” with totally five degrees.

Table 7-7. CCA-method.

CC-method Results	ADAM Results	CCAQ
Good	High (Good)	1
Good	Average	3/4
Good	Low (Bad)	1/2
Average	High (Good)	3/4
Average	Average	2/4
Average	Low (Bad)	1/4
Bad	High (Good)	2/4
Bad	Average	1/4
Bad	Low (Bad)	0

Step2: for each matched legacy component in the candidate SOA migration planning schemas, measure the hybrid information according to the established approaches in Step 1.

In this example, the "component cohesion", "component coupling", and "component adaptability" are chosen to be indicators; Table 7-8 presents the measured result on hybrid information (take target S_{31} as an example).

Table 7-8. CCAQ of the Example Shown in Table 7-5.

Domain	Number	Legacy	CCAQ	Note*
S_{31}	1	$L_{21}+L_{22}$	1/2	Legacy1
		$L_{21}+L_{31}+L_{32}$	3/4	Legacy2
	2	L_1	3/4	Legacy1
		$L_{21}+L_{22}+L_{23}$	1/2	Legacy2
	3	L_{22}	1/4	Legacy1
		$L_{31}+L_{32}$	1/2	Legacy2
	4	L_{25}	3/4	Legacy1
		$L_{36}+L_{37}$	1/2	Legacy2

Note*: legacy1 and legacy2 are alternative in an SOA migration planning schema.

Step3: input the first n candidate SOA migration planning schemas ranked by descending order for a domain target; and, combine CCA-method and the original similarity matrix method together.

In this example, choose n=4. The combination formula for ranking SOA migration planning schemas is,

$$\mathrm{VHI} = \alpha * similarity + \beta * \mathrm{CCAQ}$$

α and β are tuning coefficients, usually, $\alpha + \beta = 1$.

For target S_{31}, the re-ranked matching relationships according to their Value of Hybrid Information (VHI) are shown in Table 7-9, in which it is supposed $\alpha = 0.5$; β=0.5.

Table 7-9. The Re-ranked Matching Relationships of Target $S_{31.}$

Domain	Rank	Legacy	*Sim_semantic*	CCAQ	VHI	Re-rank
S_{31}	1	L_{21}+L_{31}+ L_{32}	0.96	1	0.98	1
	2	L_1	0.81	3/4	0.79	2
	3	L_{31}+ L_{32}	0.73	1/2	0.62	4
	4	L_{25}	0.68	3/4	0.72	3

Step4: determine the implementation means according to the new ranking value; output the re-ranked *n* optimal SOA migration planning schemas for the domain target. For Legacy1 and Legacy2 Column, the one that has the higher value of hybrid information is the better one in the hybrid information level. The final SOA migration planning schemas for target S_{31} are shown in Table 7-10, where user's direction is Performance-First.

Table 7-10. The Final SOA Migration Planning Schemas for Target $S_{31.}$

Domain	Rank	Legacy	VHI	Implementation Means
S_{31}	1	L_{21}+L_{31}+ L_{32}	0.98	Wrapper
	2	L_1	0.79	Modification
	3	L_{25}	0.72	Modification
	4	L_{31}+ L_{32}	0.62	Redevelopment

Step5: go to Step2 until all domain targets have been processed.

In this example, continue to do for target $S_{32,}$ and $S_{25.}$

Finally, the final SOA migration planning schemas based on hybrid information are created. Table 7-11 shows an example of final SOA migration planning schemas for Figure 6-2 (take Performance-First as an example).

Table 7-11. The Final SOA Migration Planning Schemas of Figure 6-2 (Performance-First).

DT	Proposed Migration Schemas (PMS)		
	Rank	LS	IM
S_{31}	1	L_{21}+L_{31}+L_{32}	Wrapper
	2	L_1	Modification
	3	L_{25}	Modification
	4	L_{31}+ L_{32}	Redevelopment
S_{32}	1	L_{33}+L_{34}	Modification
	2	L_{33}+L_{34}+ L_{35}	Modification
	3	L_2	Modification
	4	L_{33}	Modification
S_{33}	1		Redevelopment
S_{25}	1	L_{36}	Modification
	2	L_3	Modification
	3	L_{36}+ L_{37}	Modification

DT: Domain Target LS: Legacy Source
IM: Implementation Means MS: Migration Schema
W: Wrapper M: Modification R: Redevelopment

7.4 Evaluation of SOA Migration Planning Schemas

To show the detailed and comprehensive SOA migration planning schemas, the evaluation methods on the recommended SOA migration planning schemas should be investigated. One of the evaluation goals is to pursue the ideal SOA migration planning schemas according to the system's requirements and user's directions. The evaluation results can provide better predictions and planning on the SOA migration project for ensuring the success of this SOA migration project. In this study, evaluation work is done from the perspective of "function-first" (concerned with functional requirements), "performance-first" (concerned with non-functional requirements), and "cost-first" (concerned with non-functional requirements). The evaluation aspects include the cost aspect and the performance aspect of SOA migration planning schemas.

7.4.1 Implementation Cost Estimation on SOA Migration Planning Schemas

The migration source, migration target and concrete implementation means are described in the recommended SOA migration planning schemas. After doing more investigations, it is found that this information is not enough for users to predict the SOA migration project. Sometimes, prior implementation cost estimation should be investigated since some tradeoffs between cost and quality of components will occur in terms of the users' directions. The purpose of estimation is to provide a better comprehension on final SOA migration planning schemas in these aspects for facilitating the decision-making and implementation.

In the recommended SOA migration planning schemas, usually, there is no single legacy component matched with a domain migration target. In many cases, a migration schema is to integrate two or more components into a migration target. When combined, there are still some implementation efforts needed to be done, especially, for Modification implementation means and Redevelopment implementation means. Obviously, it is not enough that SOA migration planning schemas creation is just for the high level descriptions. Sometimes, the implementation costs for implementing the migration schemas should also be taken into account. If the similarities of two candidate migration schemas for the same migration target are approximate, the one having the lower implementation costs will be the ideal schema. For example, if the component implementation models, such as COMBA, COM, JavaBeans and so on, differ too much, the migration cost will be too expensive. The migration project may be cancelled due to budge overrun. In this way, the implementation costs for candidate SOA migration planning schemas should be estimated for obtaining the ideal schemas.

Nowadays, some models [107, 106, 134] exist for cost estimation, such as, neural network models, regression model, cost models (SLIM, COCOMO, etc.), probabilistic model, vector prediction models, etc. The cost estimation on a component based system is different from the conventional software system. Some concrete methods have been introduced in References [11, 125].

In this study, no new method for implementation cost estimation is addressed since it is not the key part of this study. A method using combination of empirical/ historical data (the measurement of past projects can be of great benefit to similar problem solving) and subjective expert estimations is adopted for this example. The cost estimation on SOA migration planning schemas shown in Table 7-11 is shown in Table 7-13.

7.4.2 Performance Evaluation of SOA Migration Planning Schemas

In this study, two decision-making methods on SOA migration planning schemas are addressed. For validating these two methods, the performance evaluation on created SOA migration planning schemas should be investigated.

In order to evaluate the created SOA Migration planning Schemas (for short, SOAMS), SOAMS created by human judges will be needed. Thus, totally, there are three kinds of SOAMS for a domain target, namely, SOAMS created by domain EXperts (in short, SOAMS-EX), SOAMS created by a Similarity Matrix method (SOAMS-SM) and SOAMS created by a Hybrid Information method (SOAMS-HI). The SOAMS-EX is taken as a baseline. The correlation between the SOAMS-EX and SOAMS-SM or SOAMS-HI will be measured. If the two schemas have a high positive similarity, the performance of proposed schemas will be regard as high performance, and vice versa.

In the similarity measurement, the key factors include the set of legacy components and implementation means. To quantify the similarity measurement, some set operation method will be involved.

A SOAMS-EX for Figure 6-2 is shown in Table 7-12. In the following, the similarity calculation between a SOAMS-EX (shown in Table 7-12) and a SOAMS-HI (shown in Table 7-11) will be exemplified.

Table 7-12. An SOAMS-EX and An SOAMS-HI.

DT	Expert's MS	
	LS	IM
S_{31}	L_{21}+L_{31}+L_{32}	Wrapper
S_{32}	L_{33}+L_{34}+ L_{35}	Modification
S_{33}	L_{24}	Modification
S_{25}	L_{36}+ L_{37}	Modification

DT: Domain Target LS: Legacy Source MS: Migration Schema IM: Implementation Means

In this example,

- the similarity calculation method is as follows:
  - calculate the similarity between each migration planning schema in SOAMS-HI and in SOAMS-EX for each target; then, take the biggest one as being the similarity for this target;
  - calculate the average similarity of all targets;
  - return this average similarity as the evaluation result.
- For target S_{31}, set operation method described in Sub-section 6.1.1.1 is used to calculate the similarity.

  $Sim(Ex, HI_1)=(L_{21}+L_{31}+L_{32}+\text{Wrapper})/(L_{21}+L_{31}+L_{32}+\text{Wrapper})=1$;

  $Sim(Ex, HI_2)= 0$;

  $Sim(Ex, HI_3)= 0$;

  $Sim(Ex, HI_4)=(L_{31}+L_{32})/(L_{21}+L_{31}+L_{32}+\text{Redevelopment}+\text{Wrapper})= 2/5$.

  Thus, the similarity for target S_{31} is 1 (the biggest one).
- In the same way for target S_{32}, the similarity is 1; for target S_{33}, the similarity is 0; for target S_{25}, the similarity is 1. The average similarity is 3/4.
- The evaluation result for SOAMS-HI is 3/4.

After passing the evaluation stage of SOA migration planning schemas, the final returned result will be an SOA migration evaluation report. The evaluation report can guide the migration process for avoiding undesirable results. An evaluation report on this example (shown in Figure6-2) is summarised in Table 7-13.

Information on cost estimation and performance evaluation can be of great benefit to user's direction. In the beginning, user's direction is experiential and subjective. After analysing this information, user's direction can be more reasonable and functional. The SOA migration process is a cyclic process. Usually, an iterative and incremental development method will be adopted.

Table 7-13. An Evaluation Report.

DT	Proposed Migration Schemas (PMS)					Expert's MS	Evaluation on PMS	
	Rank	LS	SSM	SHI	IM	LS	Cost	Performance
S_{31}	1	L_{21}+L_{31}+L_{32}	0.96	0.98	W	L_{21}+L_{31}+L_{32}	Average	1
	2	L_1	0.81	0.79	M		Average	
	3	L_{25}	0.68	0.72	M		Less	
	4	L_{31}+ L_{32}	0.73	0.62	M		Less	
S_{32}	1	L_{33}+L_{34}	0.8	0.78	M	L_{33}+L_{34}+ L_{35}	Average	1
	2	L_{33}+L_{34}+ L_{35}	0.73	0.75	M		Average	
	3	L_2	0.63	0.70	M		More	
	4	L_{33}	0.87	0.69	M		Less	
S_{33}					R	L_{24}		0
S_{25}	1	L_{36}	0.64	0.70	M	L_{36}+ L_{37}	Less	1
	2	L_3	0.6	0.68	M		Average	
	3	L_{36}+ L_{37}	0.81	0.66	M		Average	
Total							Average	3/4

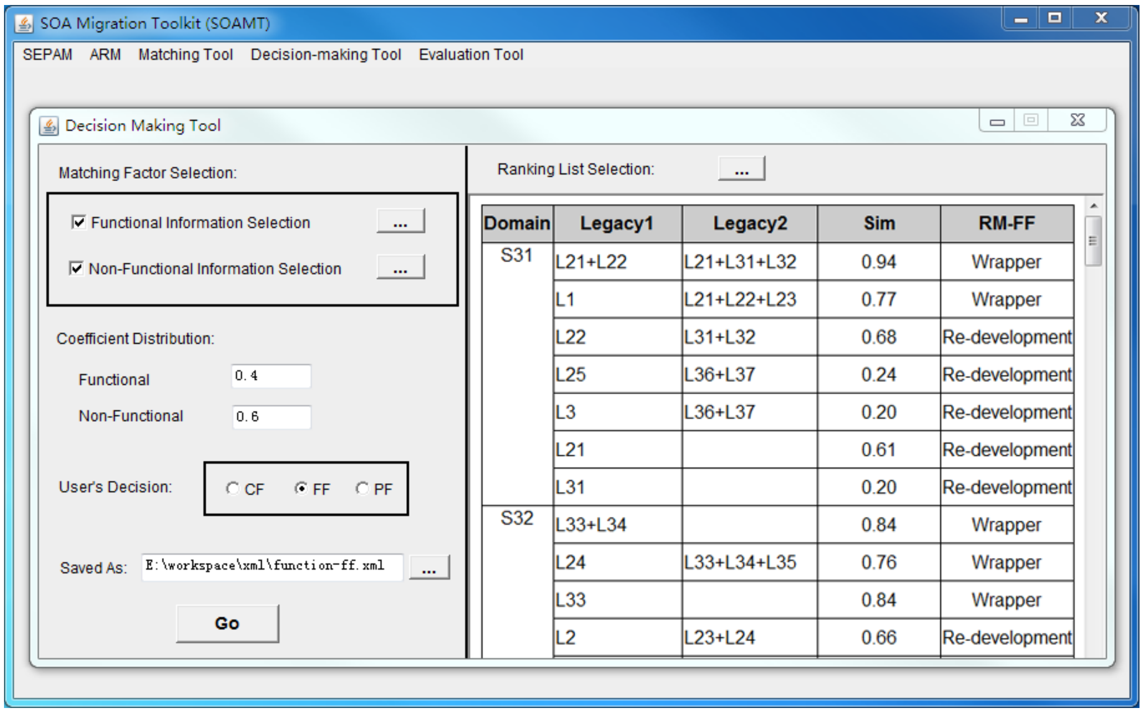
DT: Domain Target LS: Legacy Source SSM: Similarity on Similarity Matrix
VHI: Value on Hybrid Information IM: Implementation Means MS: Migration Schema
W: Wrapper M: Modification R: Redevelopment

7.5 Supporting Tools

Two kinds of tools are developed in the decision-making and evaluation stages, namely, a decision-making tool and an evaluation tool.

7.5.1 Decision-making Tool

A decision-making tool has been developed based on the proposed decision-making methods. In this tool, users can choose the decision-making factors and can tune the coefficient of each factor. Moreover, the user's direction can also be chosen in this tool. Figure 7-3 presents the interface of the decision-making tool.



Matching Factor Selection:

☑ Functional Information Selection

☑ Non-Functional Information Selection

Coefficient Distribution:

Functional 0.4

Non-Functional 0.6

User's Decision: CF [✓ FF] PF

Saved As: E:\workspace\xml\function-ff.xml

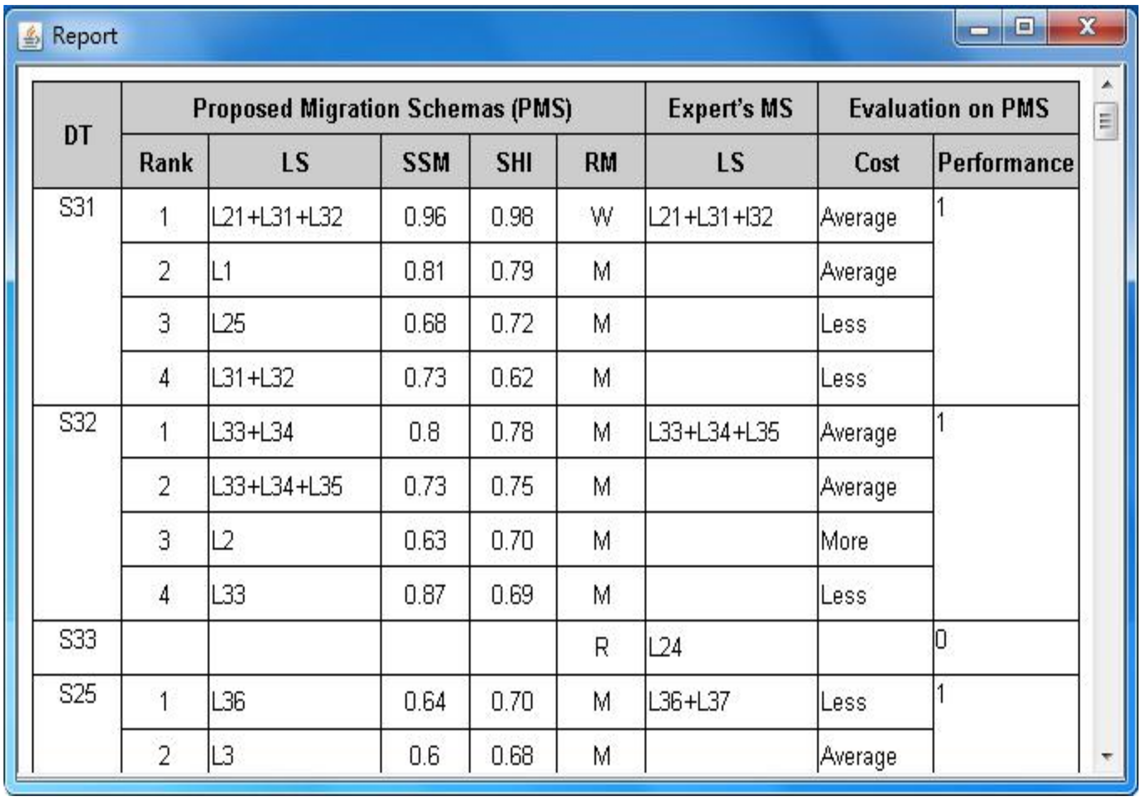
Ranking List Selection:

Domain	Legacy1	Legacy2	Sim	RM-FF
S31	L21+L22	L21+L31+L32	0.94	Wrapper
	L1	L21+L22+L23	0.77	Wrapper
	L22	L31+L32	0.68	Re-development
	L25	L36+L37	0.24	Re-development
	L3	L36+L37	0.20	Re-development
	L21		0.61	Re-development
	L31		0.20	Re-development
S32	L33+L34		0.84	Wrapper
	L24	L33+L34+L35	0.76	Wrapper
	L33		0.84	Wrapper
	L2	L23+L24	0.66	Re-development

Figure 7-3. The Interface of Decision-making Tool.

7.5.2 Evaluation Tool

In this thesis, a simple evaluation tool is adopted. First, cost assessment on SOA migration planning schemas has been done. Second, performance evaluation on SOA migration planning schemas has been done. Third, an evaluation report is summarised to show abundant information for users. Figure 7-4 presents an example of an evaluation report on SOA migration planning schemas.



DT	Proposed Migration Schemas (PMS)					Expert's MS	Evaluation on PMS	
	Rank	LS	SSM	SHI	RM	LS	Cost	Performance
S31	1	L21+L31+L32	0.96	0.98	W	L21+L31+I32	Average	1
	2	L1	0.81	0.79	M		Average	
	3	L25	0.68	0.72	M		Less	
	4	L31+L32	0.73	0.62	M		Less	
S32	1	L33+L34	0.8	0.78	M	L33+L34+L35	Average	1
	2	L33+L34+L35	0.73	0.75	M		Average	
	3	L2	0.63	0.70	M		More	
	4	L33	0.87	0.69	M		Less	
S33					R	L24		0
S25	1	L36	0.64	0.70	M	L36+L37	Less	1
	2	L3	0.6	0.68	M		Average	

Figure 7-4. An Evaluation Report on SOA Migration Planning Schemas.

7.6 Summary

Some work on creation and evaluation of SOA migration planning schemas has been done in this Chapter:

- The SOA migration planning schema is defined, which includes migration source object, migration target object, matching degree and implementation means.
- A decision-making method based on a similarity matrix for creating SOA migration planning schemas is proposed. The dimensionality reduction has been done to the matrix: just continue to have the first k matched legacy objects that have high similarities with domain targets; reorganising them into a matching relationship table, and, adding the equivalent parts for the remainder of the legacy aspect into the matching relationship table; obtaining the available matching relationships; determining the implementation means; integrating SOA migration planning schemas.

- A decision-making method based on hybrid information for creating SOA migration planning schemas is proposed. The hybrid information for candidate SOA migration planning schemas is collected. For each matched legacy object in the candidate SOA migration planning schemas, measure the hybrid information according to the proposed method; input the first n candidate SOA migration planning schemas ranked by descending order for a domain target; and, combine the hybrid information method and the original similarity matrix method (SIM-method) together; determine the implementation means according to the new ranking value; and output the re-ranked *n* optimal SOA migration planning schemas for the domain target.

- Evaluation methods on the SOA migration planning schemas are proposed. Implementation cost estimation and performance evaluation on SOA migration planning schemas have been done.

- An example has been presented for further describing the proposed methods. In this study, the same example runs through all proposed matching strategies and decision making methods, which is of great benefit to the reader.

- Two types of tools, namely, a decision-making tool and an evaluation tool, have been developed in SOAMT. In the decision-making tool, there are two functions available: the function based on similarity matrix and the function based on hybrid information. In the evaluation tool, there are three functions available: cost evaluation function, performance evaluation function and evaluation report function.

Chapter 8 Case Studies

Objectives

- To evaluate applying a data mining method to improve Service model and Component model in an e-learning field.
- To evaluate matching strategies between legacy components and domain targets in an e-learning field.
- To evaluate decision-making methods on SOA migration planning schemas in an e-learning field.
- To evaluate the SOA migration planning approach.

Two case studies for evaluating the main contributions of this thesis will be presented in this Chapter. The key contribution of this thesis is the proposed SOA migration planning approach. There are six core algorithms, strategies and methods in this proposed approach for the planning and prediction of SOA migration projects: an association rule mining algorithm for determining the relationships of legacy components [151]; a sequence pattern mining algorithm for service identification and composition [152, 154]; applying a data mining method to improve Service/Component model in SOA migration environment [147]; matching strategies between legacy components and domain targets [149]; decision-making methods on SOA migration planning schemas [148]; evaluation method on SOA migration planning schemas. Case studies will evaluate the reliability and availability of the proposed approach in an e-learning field. In case study I, the work on how to use the developed tools SEPAM and ARM to improve the Service model and Component model will be presented. In case study II, the work on creating SOA migration planning schemas of an e-learning legacy system by using a developed SOA migration toolkit will be presented.

8.1 Case Study I: Improving Service Model and Component Model with SEPAM and ARM

8.1.1 Overview

At present, most of middle and primary schools can access the Internet. This is the base for implementing e-education. Moreover, with the development of computer network technology, people are trying to make information technology become a kind of teaching and learning tool. Meanwhile, information technology is utilised during the process of teaching and learning. Thus, a great deal of education resources and software have been developed and applied in the education field.

With the rapid development of science and technology, some application systems have become legacy systems. They need to be reengineered. In this case study, a service composition method based on the sequence pattern mining technique for Migrating E-learning Legacy systems to a SOA (MELS) environment will be presented. This method will be applied to the practical data collected from an e-learning legacy system [66]. For evaluating this method, some investigation questionnaires are set up to collect satisfaction data. By adopting these two methods the returned results will be compared. If the two results have high positive similarity, the performance of the proposed method will be regarded as high performance, and vice versa.

8.1.2 A Service Composition Method for MELS by Using SEPAM

In this study, two data mining algorithms have been proposed. Then, how can these kinds of algorithms be used in an SOA migration situation? This question will be solved in this Sub-section. A service composition method for MELS by using SEPAM will be depicted. The mined sequence patterns are the supplementation of results of traditional domain analysis and legacy system analysis. They can be of great benefit to the establishment of a Service model (Section 4.1) and a Component model (Section 5.1). For example, the nodes S_{21} and S_{23} in Figure 4-4 are established by a sequence pattern mining technique (Section 4.2). Furthermore, they can benefit the creation of SOA migration planning schemas (Section 7.1.3). Figure 8-1 shows the framework of this proposed method.

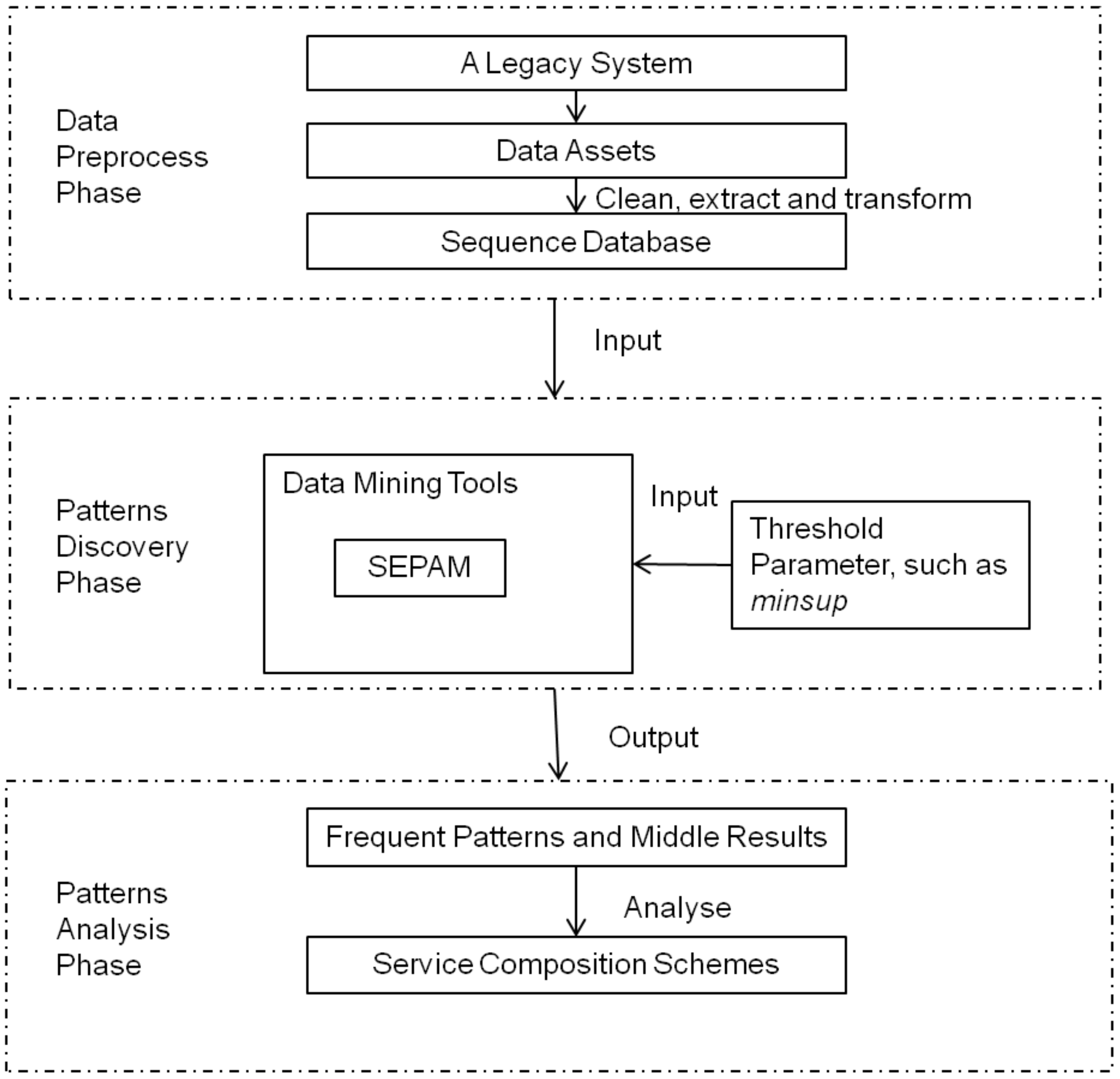


Figure 8-1. The Framework of a Service Composition Method.

There are three phases in this method:

- Data pre-process phase.

  The main work is to collect, clean, extract and transform the appropriate software engineering data for the pattern discovery.

- Pattern discovery phase.

  The main work is applying data mining tools to discover frequent patterns/motifs.

- Pattern analysis phase.

The main work is to analyse and conclude the results mined by data mining tools. The useful information will be used for service composition.

The key phase of this method is developing reasonable data mining tools to mine related legacy data. In the following, more details of this method will be presented.

8.1.3 Data Pre-process Phase

8.1.3.1 An E-learning Legacy System

An e–learning system for middle school students is shown in Figure 8-2 which has been developed by Ideal Information Technology Institute of Northeast Normal University in China [66, 100]. This e-learning system consists of well-designed object-oriented programs. However, the structure of this system cannot meet the user’s needs. This system will be migrated to SOA, shown in Figure 8-6 [31].

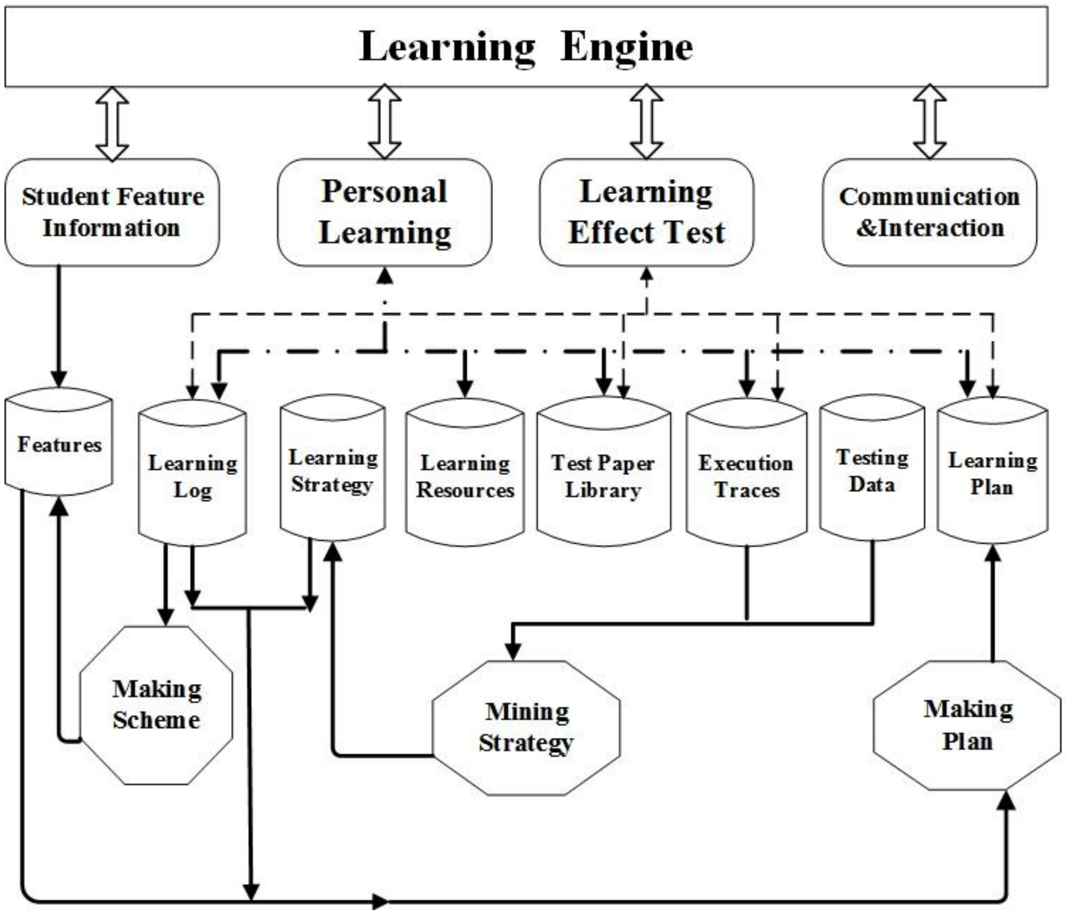


Figure 8-2. Architecture of an E-learning System.

An e-learning system must provide complete support for different teaching and learning modes. Normally, different teachers will provide different teaching strategies for the same learners. Different learners will adopt different learning schemes for the same content. In this case, during the design and development of an e-learning system, it is necessary to provide a learner with the appropriate learning support environment according to the learner’s preference. Therefore, the composition of learning schemes is a key task for MELS.

In MELS, service composition can be improved by mining execution trace data on usage behaviour of this e-learning legacy system. The usage behaviour data resource is the set of practical operation sequences requested by all users. The practical operation sequence represents a specific learning sequence which can be of great benefit to the service composition in MELS. Namely, these discovered patterns will be the candidate composite services (coarse-grained) in the SOA target system.

According to the theory of education, a learning process mainly includes the following learning modes: demonstration, read and analysis, interaction, exercise, learn by heart, review, test, feedback, etc. A learning process is a sequence composed of these learning modes. They will be different sequences for different learners for the same learning content. Therefore, the appropriate learning support environment for each learner means a specific learning sequence (coarse–grained service) composed of existing learning contents and learning modes (fine-grained service).

In this experiment, the composition of learning schemes in the SOA target system will be taken as examples. According to the suggestions of educational experts and excellent teachers, eight learning modes (items) are selected, such as demonstration (denoted by {1}), read and analysis (denoted by {2}), interaction (denoted by {3}), exercise (denoted by {4}), learn by heart (denoted by {5}), review (denoted by {6}), test (denoted by {7}) and feedback (denoted by {8}). An ordered list of these learning modes is named as a learning scheme, which can be regarded as a sequence. For example, the sequence of {read and analysis} {interaction}{exercise} (for short, {2}{3}{4}) is a learning scheme, which means a learner first reads and analyses related learning materials; and then communicates with the system to get more detailed explanations on his/her questions; and then does some exercises to strengthen his/her learning. The users can organise a learning scheme by themselves in this e-learning legacy system. Some learning schemes are collected from different learners to form a sequences database. In order to discover which learning patterns (supported by most learning schemes) are popular for most learners, the proposed algorithms are applied to mine this learning scheme sequences database. The mined frequent and maximal learning patterns (motifs) can be the candidates of coarse-grained services or business processes of the learning scheme aspect in the SOA target system.

8.1.3.2 Sequence Database

The preparation, collection and extraction of execution traces data are the basis for the application of the proposed method. Development and application of education software and resources [128] have made it possible for educational institutes to collect and store huge amounts of teaching and learning execution trace data [20].

Execution scenario refers to the functionality that the program gets executed. It is very important since it influences the results of the technique [14]. Thus, an execution scenario should be established for collecting trace data according to the purpose of application. For example, in order to discover popular learning patterns, the learning scheme module in the legacy system should be taken as an execution scenario to collect and extract the related trace data.

For ensuring the quality of mined results, the execution traces will be collected in a real operational environment and for a long time. In this case study, the related data from 20 middle schools that are using this e-learning system has been collected. In each school, 5 classes of 50 students were chosen as test objects. 48 tests were organised during four months. The students will adopt a learning scheme during one test, namely, each student can provide one record in one test. Totally, about 240,000 records can be obtained. The size of each record is about 0.1k. Thus, the data volume is about 24M, that is,

$$20 * 5 * 50 * 48 * 0.1k = 24000k = 24M$$

The log sub-system of this e-learning system collected these records and stored them in Table format of an SQL Server database. The collected original data with the main columns is shown in Table 8-1.

Table 8-1. The Collected Original Data.

ID	Learning Content	Starting Time	Knowledge Node Code	Learning Scheme	Learning Source Code	Score
… …	… …	… …	… …	… …	… …	… …
7021	Math in Grade 7	2007-05-04，14：03	2.2.3	1,2,3,4,7,8,37	172020,172118,172002,172010, 172102, ……	B
… …	… …	… …	… …	… …	… …	… …
7043	Math in Grade 7	2007-05-04，14：10	2.2.2	1,4,3,5, 7,8	172031,172118,172015,172011, 172112, ……	A
… …	… …	… …	… …	… …	… …	… …
7101	Math in Grade 7	2007-05-04，14：05	2.2.1	2,5,3,4,6,7,8	172020,172118,172002,172036, 172108, ……	C
… …	… …	… …	… …	… …	… …	… …

Notation: learning mode "*demonstration*" is denoted as 1, "*read and analysis*" is denoted as 2, "*interaction*" is denoted as 3, "*exercise*" is denoted as 4, "*learn by heart*" is denoted as 5, "*review*" is denoted as 6, "*test*" is denoted as 7 and "*feedback*" is denoted as 8.

In order to reduce the data volume, the related columns should be extracted from the original data for mining purposes. The related columns will be extracted and stored in a new structural file. For example, in order to discover popular learning patterns, the ID column and learning scheme column were extracted from Table 8-1. A sequence database on trace data of learning schemes can be obtained, which is shown in Table 8-2. This database can be expressed as a group of software trace sequences. The size of one record in this structural file is about 14bytes. Thus, the data volume of the mined sequence data is 3.2M, that is,

$$20 * 5 * 50 * 48 * 14/1024=3281k=3.2M$$

This sequence data of 3.2M is the input of the proposed sequence mining algorithm described in Section 4.2.

Table 8-2. Sequence Database.

ID	Sequences s_i	Length $\| s_i \|$
1	{1}{2}{3}{4}{7}{8}{3}{7}	8
2	{1}{4}{3}{5} {7}{8}	6
3	{5}{1}{2}{4}{3}{7}{5}{8}	8
4	{1}{2}{4}{3}{7}{8}{5}	7
5	{3}{2}{4}{3}{1}{6}{7}{8}	8
6	{7}{8}{3}{5}{4}{6}{7}{8}	8
7	{2}{1}{2}{4}{3}{7}{8}	7
8	{4}{3}{2}{7}{8}	5
9	{2}{1}{2}{4}{3}{7}{8}	7
10	{4}{3}{2}{3}{7}{8}	6
… …	… …	… …

8.1.4 Pattern Discovery Phase

Through analysing the existing sequence mining algorithms, it is found that they cannot meet the needs of the e-learning data. For example, these algorithms are used to find motifs on the fixed *min-support* number. In order to further analyse the mined patterns in the pattern analysis phase, it is not enough to just obtain motifs in a fixed *min-support* number. In addition, the data in an e-learning domain should consider the repeatable number associated with items to some extent. Moreover, the repeatable number of each item in e-learning data is small (normally, it is less than 5, for example, a learner can review some learning content 3 times or 4 times, but it is exceptional if he reviews them 10 times). Thus, if the existing quantitative sequence mining algorithms are applied, it will increase the costs of time and space greatly. Furthermore, it will affect the practicability of this method. In this case, according to the features of execution trace data on usage behaviour and the needs of further pattern analysis, a sequential pattern mining algorithm is proposed (more details are shown in Section 4.2) to mine execution traces of the e-learning legacy system.

In this experiment, the average length of all records is about 12 since the elements of each sequence can be repeated such as {2}{1}{4}{4}{2}{4}; the number of items is 8; the size of records in the structural file is 3.2M. In a system implementation aspect, a hash-tree data structure is adopted. In this case, the time cost of this algorithm is about one minute. Clearly, the performance of this algorithm is acceptable.

8.1.5 Pattern Analysis Phase

After obtaining the frequent patterns and middle results returned from the pattern discovery phase, the results should be analysed and refined. Normally, this kind of work will be done with the direction of project architects together with domain experts. The analysed results will form service composition schemes, which are the complementation of business logic analysis results. These complementary services can provide information for further improving this e-learning system, such as, some services can become the recommended e-learning patterns to learners; some services can direct the adding, deleting, updating or combining of some classes or functionality; etc.

For example, through analysing the results, it is found that 68% of the users like to adopt a learning scheme like {*demonstration*}{*analysis and read*}{*exercise*} {*interaction*} {*test*}{*feedback*}, namely, sequence {1}{2}{4}{3}{7}{8} which is a motif of the sequence database shown in Table 8-2. However, in the existing system, this pattern wasn’t included in the recommended learning patterns. In the new system, this pattern has been composed into a composite service and stored in the service library in the SOA system.

8.1.6 Evaluation

For this execution scenario, some investigation questionnaires were set up to collect satisfaction data. This method includes an online survey of 20 middle schools (the users of this e-learning system) and in-depth interviews with 30 education experts and excellent teachers. The investigation result is 90% the same as the result returned by applying this method. Clearly, this service composition method is available and applicable.

The similar experiments show that this method can also be used for component composition if sequential relationships exist among components. In addition, if association relationships exist among services or among components, similar experiments can be done to improve the Service model and the Component model by applying the association rule miner (ARM). The experiment results prove that the method on applying data mining techniques to improve the Service model and the Component model is available and applicable.

8.2 Case Study II: Creating SOA Migration Planning Schemas with SOAMT

8.2.1 Overview

An SOA migration planning approach is proposed in this study. There are five stages in this approach. The detailed descriptions of each stage are addressed. Especially, the concrete matching strategies and decision-making methods of creating SOA migration planning schemas are provided. The proposed approach can be of great benefit to the planning and deployment of SOA migration projects.

In this Section, a case study on creating SOA migration planning schemas is described to further evaluate the proposed approach. The migration resource is legacy components in an educational administration system for primary and secondary schools. The migration target is the new service oriented architecture for doing the work on student's management. Concretely,

- An e-learning legacy system and its existing problems will be described.
- For solving the existing problems, the new service oriented architecture will be designed and developed.
- The matching relationship between legacy components and domain targets will be established by adopting the proposed matching strategies. Some concrete work is presented. The algebra theory is regarded as the theoretical support frame. The text similarity measurement method in the vector space model will

be adopted to measure the similarity between legacy components and domain services.

- The SOA migration planning schemas will be created by adopting the proposed decision-making methods.
- Evaluation is done finally in this case study.

8.2.2 Legacy Assets

In China, Education Informatisation refers to the process that accelerates educational reformation and development, whose key parts include teaching and learning, teaching and scientific research, and education administration. The roles of education administration are related to communication, cooperation and sharing of information. The concrete tasks in the education administration aspect include human resource management, teaching and learning affairs management, official document management, general affairs management, home-school interaction management, etc. [128].

- Human resource management consists of personal information. The teacher’s information includes take or leave office, position change, assessment, etc.; the student’s information includes natural information, registration, awards, etc.
- Teaching and learning affairs management consists of examination management, curricula-variable management, social activity management, students’ comprehensive quality evaluation, etc.
- Official document management includes drafting, examining and approving, dispatching, receiving, classifying, etc. of official documents.
- General affairs management includes office supplies management, school building management, facilities management, etc.
- Home-school interaction management consists of the teacher’s functions, the student’s functions and the parents’ functions. The teacher’s functions include management of message, assignment, score, phase evaluation, etc. The student’s functions include management of assignment, message, score, school timetable, etc. the parents’ functions include management of message, assignment, notice, score, etc.

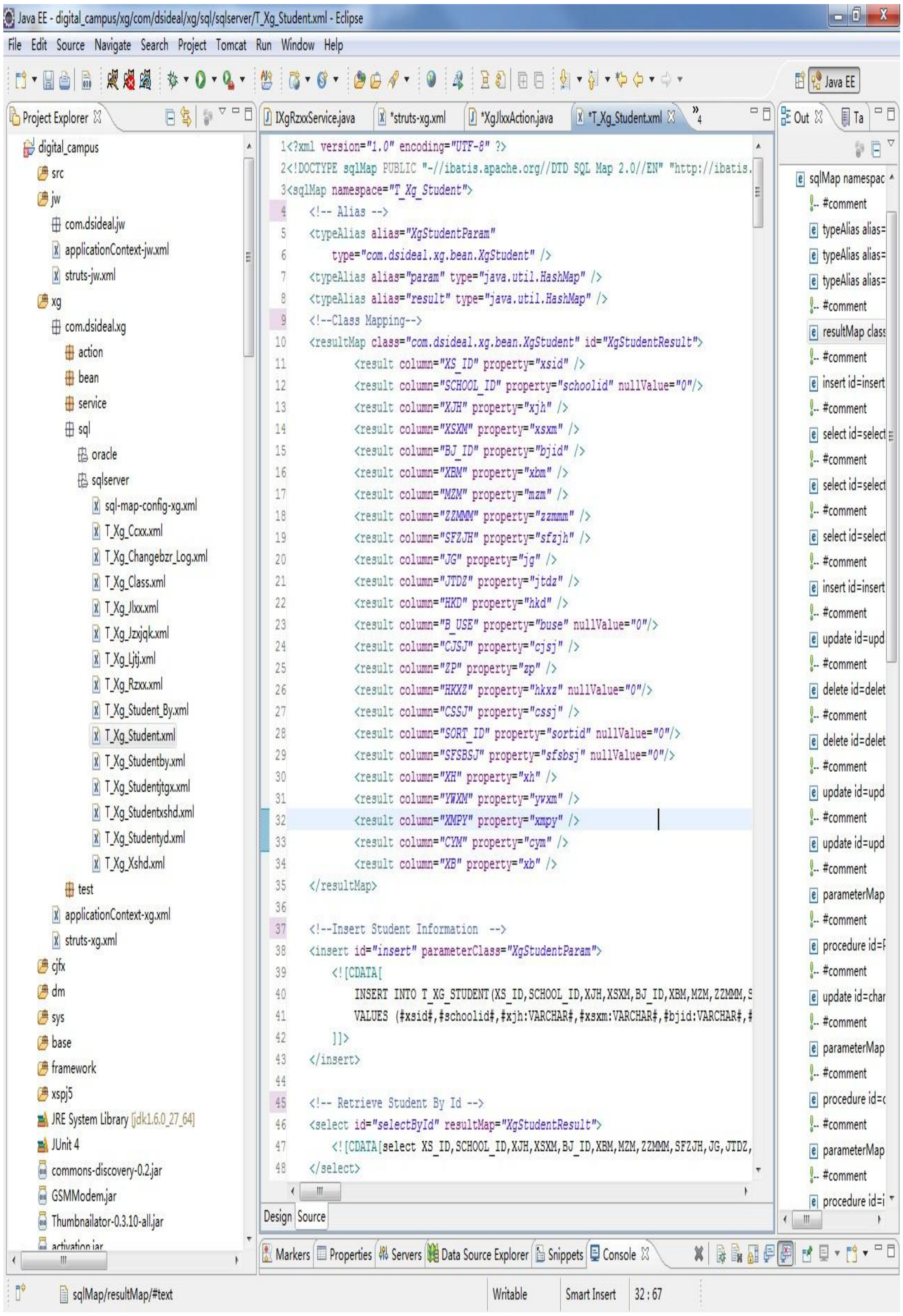
According to the above requirements, an education administration system has been developed under the environment of Java, Eclipse and Tomcat (middleware) [66]. In this system, there are a human resource management sub-system, a teaching and learning affairs management sub-system, an official document management sub-system, a general affairs management sub-system, and a home-school interaction management sub-system.

However, in this system, each function has been developed without thinking about the case that some business processes have to pass several sub-systems. The flexibility and adaptability of this legacy system are poor. It cannot meet the user's needs. For example, taking the work on students' management from the role of a teacher in charge of a class. For completing work on the students' management, the teacher involves logging in to three sub-systems and processing five function modules:

- Login human resource management sub-system for editing and retrieving student's basic information.
- Login teaching and learning affairs management sub-system for completing curriculum selection and score edition.
- Login home-school interaction management sub-system for completing attendance checking.
- Login teaching and learning affairs management sub-system for analysing and evaluating student's social activities.
- Login teaching and learning affairs management sub-system for doing students' comprehensive quality evaluation.

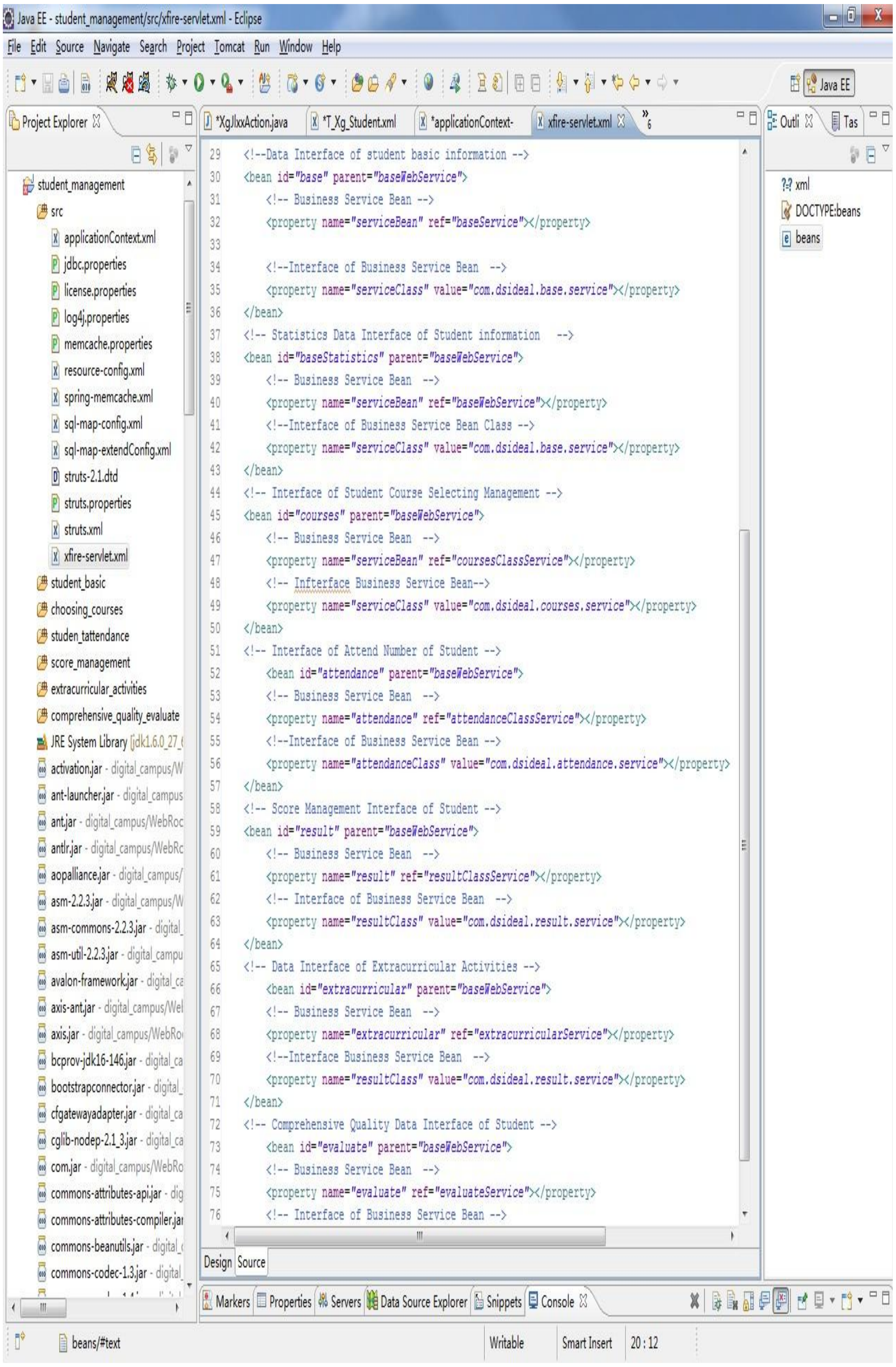
Clearly, this educational management system for primary and secondary schools has become a legacy system. It cannot meet the new requirements. The service-oriented architecture and web service technique can solve these kinds of problems.

For establishing a Component model, the e-learning legacy assets should be collected and processed. Figure 8-3 shows a screenshot of a legacy asset on object relationship mapping (ORM) configuration file. Figure 8-4 presents a screenshot of a legacy asset on component relationship.



```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE sqlMap PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN" "http://ibatis.
<sqlMap namespace="T_Xg_Student">
    <!-- Alias -->
    <typeAlias alias="XgStudentParam"
        type="com.dsideal.xg.bean.XgStudent" />
    <typeAlias alias="param" type="java.util.HashMap" />
    <typeAlias alias="result" type="java.util.HashMap" />
    <!--Class Mapping-->
    <resultMap class="com.dsideal.xg.bean.XgStudent" id="XgStudentResult">
            <result column="XS_ID" property="xsid" />
            <result column="SCHOOL_ID" property="schoolid" nullValue="0"/>
            <result column="XJH" property="xjh" />
            <result column="XSXM" property="xsxm" />
            <result column="BJ_ID" property="bjid" />
            <result column="XBM" property="xbm" />
            <result column="MZM" property="mzm" />
            <result column="ZZMMM" property="zzmmm" />
            <result column="SFZJH" property="sfzjh" />
            <result column="JG" property="jg" />
            <result column="JTDZ" property="jtdz" />
            <result column="HKD" property="hkd" />
            <result column="B_USE" property="buse" nullValue="0"/>
            <result column="CJSJ" property="cjsj" />
            <result column="ZP" property="zp" />
            <result column="HKXZ" property="hkxz" nullValue="0"/>
            <result column="CSSJ" property="cssj" />
            <result column="SORT_ID" property="sortid" nullValue="0"/>
            <result column="SFSBSJ" property="sfsbsj" nullValue="0"/>
            <result column="XH" property="xh" />
            <result column="YWXM" property="ywxm" />
            <result column="XMPY" property="xmpy" />
            <result column="CYM" property="cym" />
            <result column="XB" property="xb" />
    </resultMap>

    <!--Insert Student Information  -->
    <insert id="insert" parameterClass="XgStudentParam">
        <![CDATA[
            INSERT INTO T_XG_STUDENT(XS_ID,SCHOOL_ID,XJH,XSXM,BJ_ID,XBM,MZM,ZZMMM,S
            VALUES (#xsid#,#schoolid#,#xjh:VARCHAR#,#xsxm:VARCHAR#,#bjid:VARCHAR#,#
        ]]>
    </insert>

    <!-- Retrieve Student By Id -->
    <select id="selectById" resultMap="XgStudentResult">
        <![CDATA[select XS_ID,SCHOOL_ID,XJH,XSXM,BJ_ID,XBM,MZM,ZZMMM,SFZJH,JG,JTDZ,
    </select>
```

Figure 8-3. A Screenshot on Object Relationship Mapping (ORM) Configuration File.



Java EE - student_management/src/xfire-servlet.xml - Eclipse
File Edit Source Navigate Search Project Tomcat Run Window Help
Java EE
Project Explorer
*XgJlxxAction.java
*T_Xg_Student.xml
*applicationContext-
xfire-servlet.xml
Outli
Tas
student_management
src
applicationContext.xml
jdbc.properties
license.properties
log4j.properties
memcache.properties
resource-config.xml
spring-memcache.xml
sql-map-config.xml
sql-map-extendConfig.xml
struts-2.1.dtd
struts.properties
struts.xml
xfire-servlet.xml
student_basic
choosing_courses
studen_tattendance
score_management
extracurricular_activities
comprehensive_quality_evaluate
JRE System Library [jdk1.6.0_27
activation.jar - digital_campus/W
ant-launcher.jar - digital_campus
ant.jar - digital_campus/WebRoo
antlr.jar - digital_campus/WebRo
aopalliance.jar - digital_campus/
asm-2.2.3.jar - digital_campus/W
asm-commons-2.2.3.jar - digital_
asm-util-2.2.3.jar - digital_campu
avalon-framework.jar - digital_ca
axis-ant.jar - digital_campus/Wel
axis.jar - digital_campus/WebRo
bcprov-jdk16-146.jar - digital_ca
bootstrapconnector.jar - digital_
cfgatewayadapter.jar - digital_ca
cglib-nodep-2.1_3.jar - digital_ca
com.jar - digital_campus/WebRo
commons-attributes-api.jar - dig
commons-attributes-compiler.jar
commons-beanutils.jar - digital_
commons-codec-1.3.jar - digital_
xml
DOCTYPE:beans
beans
Design
Source
Markers
Properties
Servers
Data Source Explorer
Snippets
Console
beans/#text
Writable
Smart Insert
20 : 12

```
<!--Data Interface of student basic information -->
<bean id="base" parent="baseWebService">
    <!-- Business Service Bean -->
    <property name="serviceBean" ref="baseService"></property>

    <!--Interface of Business Service Bean  -->
    <property name="serviceClass" value="com.dsideal.base.service"></property>
</bean>
<!-- Statistics Data Interface of Student information   -->
<bean id="baseStatistics" parent="baseWebService">
    <!-- Business Service Bean  -->
    <property name="serviceBean" ref="baseWebService"></property>
    <!--Interface of Business Service Bean Class -->
    <property name="serviceClass" value="com.dsideal.base.service"></property>
</bean>
<!-- Interface of Student Course Selecting Management -->
<bean id="courses" parent="baseWebService">
    <!-- Business Service Bean  -->
    <property name="serviceBean" ref="coursesClassService"></property>
    <!-- Infterface Business Service Bean-->
    <property name="serviceClass" value="com.dsideal.courses.service"></property>
</bean>
<!-- Interface of Attend Number of Student -->
    <bean id="attendance" parent="baseWebService">
    <!-- Business Service Bean  -->
    <property name="attendance" ref="attendanceClassService"></property>
    <!--Interface of Business Service Bean -->
    <property name="attendanceClass" value="com.dsideal.attendance.service"></property>
</bean>
<!-- Score Management Interface of Student -->
<bean id="result" parent="baseWebService">
    <!-- Business Service Bean  -->
    <property name="result" ref="resultClassService"></property>
    <!-- Interface of Business Service Bean  -->
    <property name="resultClass" value="com.dsideal.result.service"></property>
</bean>
<!-- Data Interface of Extracurricular Activities -->
    <bean id="extracurricular" parent="baseWebService">
    <!-- Business Service Bean  -->
    <property name="extracurricular" ref="extracurricularService"></property>
    <!--Interface Business Service Bean  -->
    <property name="resultClass" value="com.dsideal.result.service"></property>
</bean>
<!-- Comprehensive Quality Data Interface of Student -->
    <bean id="evaluate" parent="baseWebService">
    <!-- Business Service Bean  -->
    <property name="evaluate" ref="evaluateService"></property>
    <!-- Interface of Business Service Bean -->
```

Figure 8-4. A Screenshot of Legacy Asset on Component Relationship.

After passing the legacy analysis stage, a Component model (migration sources) is established. An example of a Component model is shown in Figure 8-5.

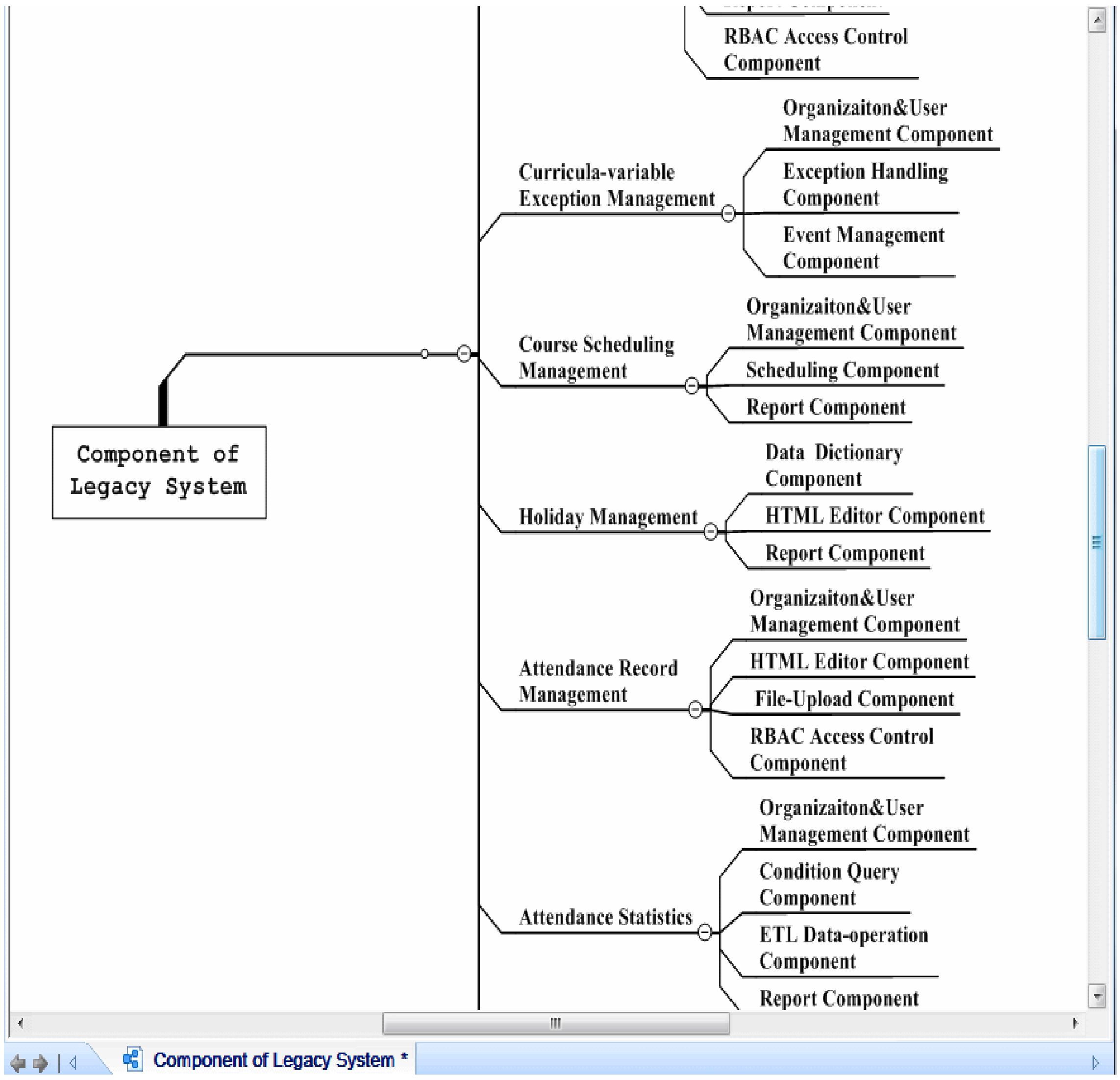

RBAC Access Control Component
Component of Legacy System
Curricula-variable Exception Management
Organizaiton&User Management Component
Exception Handling Component
Event Management Component
Course Scheduling Management
Organizaiton&User Management Component
Scheduling Component
Report Component
Holiday Management
Data Dictionary Component
HTML Editor Component
Report Component
Attendance Record Management
Organizaiton&User Management Component
HTML Editor Component
File-Upload Component
RBAC Access Control Component
Attendance Statistics
Organizaiton&User Management Component
Condition Query Component
ETL Data-operation Component
Report Component
Component of Legacy System *

Figure 8-5. A Screenshot of a Component Model.

8.2.3 A New Service Oriented Architecture

The new SOA [31] the legacy system that will be migrated to is shown in Figure 8-6. This architecture is a hierarchical and flexible reusable architecture based on domain-specific software architecture (DSA). By using this software architecture, the customised application software can be built in a visual studio just like a building block.

There are five layers in this new architecture, namely, the portal layer, the application layer, the business logical/service provider layer, the component layer and the platform supporting layer.

- The portal layer.

  This layer supplies a unified service access entrance and provides unified integrating and releasing information.

- The application layer.

  This layer supports the composure and choreography of customised services based on the user's specific role, such as students, teachers, parents and administrative staff, etc. Meanwhile, this application layer provides the functions of customising content and page style.

- The service providing layer.

  This layer provides some services to support various activities in a virtual digital campus, such as, management services, the teaching and learning services, teacher education and teaching research services.

  - The management services.

    This kind of service provides a full range of support services for the management of administrative, exam, academic affairs, personnel, and school property.

  - The teaching and learning services.

    It provide services to support the teaching process and teaching content, such as interaction services (interactions between the learner and other learners, between the learner and the instructor, and between the learner and experts), the digital resource sharing management services, and the test and evaluation services , etc.

  - The teacher education and teaching research services.

    It provides support services for improving teachers' ability to meet the needs of teaching and research. It mainly consists of a teacher remote training service, and the interactive teaching and research service, etc.

- The component layer.

  This layer is the foundation of the platform software architecture. It is a reusable collection of software units that is already validated by other projects. It mainly includes data management components, process control components, cache control components, report generation components, information retrieval components and other components. It provides common functions for some kinds of business systems. Moreover, it includes a domain component library that involves educational resources development, educational resources management, teaching & learning environment building, and educational information management, etc.

- The platform supporting layer.

  This layer is an application support system that consists of user authentication and authorisation, data exchange and operation management.

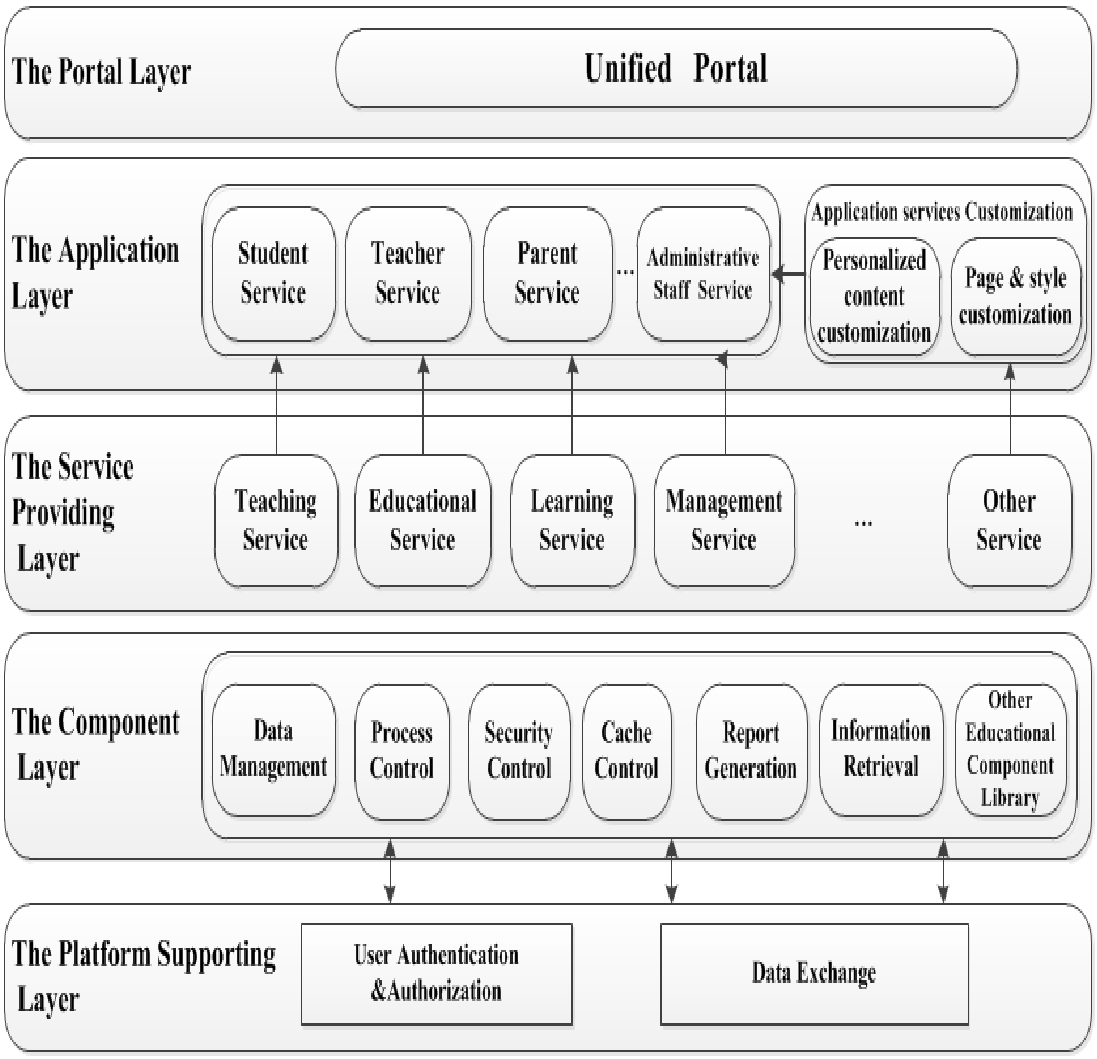


Figure 8-6. Migration Target -- The New Service Oriented Architecture.

After passing the domain analysis stage, a Service model (migration target) is established. It describes details of the Service Providing Layer shown in Figure 8-6. A screenshot of the Service model is shown in Figure 8-7.

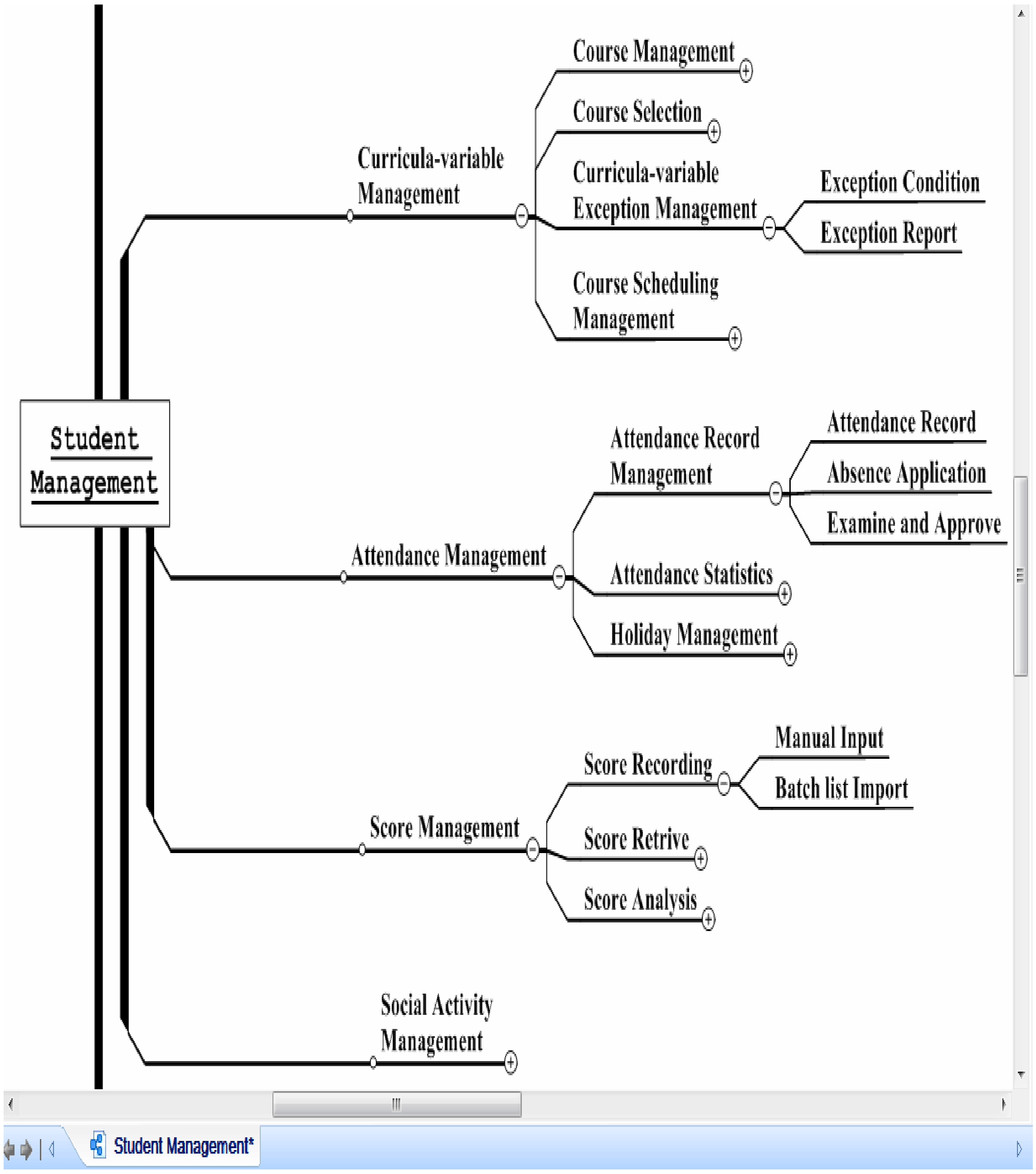


Figure 8-7. A Screenshot of the Service Model.

In this new architecture, the functions can be composed according to the user's needs. The management software of different versions (for primary school, high school, bureau of education, etc.) can be built quickly. The main feature of this architecture is to customise application software in a visual environment, such as, business process customisation, functional module customisation, login portal customisation, system message customisation, etc. The advantages of this new architecture can be embodied from the example on students' management described in Sub-section 8.2.2. The teacher can customise the work for completing student's management in a visual environment. It provides more convenience for the users.

In this new architecture, the modules and functions in the legacy system have been transformed into services that range from fine-grained to coarse-grained. The concrete tranformance means include Wrapper means, Modification means and Redevelopment means (Sub-section 7.1.1). Namely, some legacy components have been reused in whole; some legacy components have been reused in part; and, some components have been discarded.

Clearly, the new architecture is better than the former. Hence, it is necessary to migrate the legacy system to this new SOA architecture.

8.2.4 Creating SOA Migration Planning Schemas

After obtaining the Service model (migration targets, XML_{Domain}) and the Component model (migration sources, XML_{Legacy}), the processes on MELS based on TSM are as follows.

- First, establish indexing for the Service model and the Component model, and calculate the similarity between domain services and legacy components by using the matching tool.
- Second, according to the matching strategies described in Section 6.1, 6.2 and 6.3 to determine the matching relationships between migration sources and targets.
- Based on these matching relationships, according to proposed decision-making methods described in Section 7.1, 7.2 and 7.3 to create SOA migration planning schemas.
- According to the proposed evaluation method described in Section 7.4 to evaluate the created SOA migration planning schemas and output an evaluation report.

According to the above-mentioned processes, the flow chart of creating the SOA migrating planning schemas based on SOA Migration Toolkit is shown in Figure 8-8.

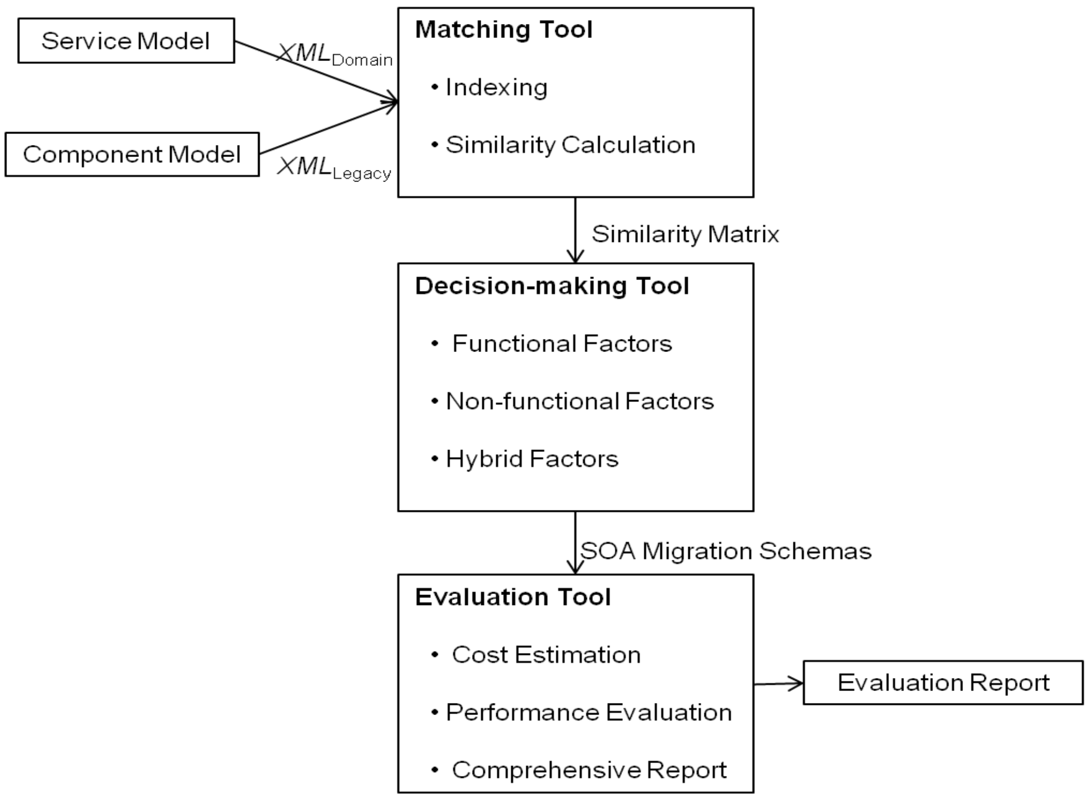


Figure 8-8. The Flow Chart of Creating SOA Migration Planning Schemas based on SOAMT.

The key part of SOA migration planning schemas refers to the corresponding relationships between domain services and legacy components. Figure 8-9 is a reduced (folded) screenshot that presents the key part of SOA migration planning schemas for migrating legacy components in an education administration legacy system to the new SOA architecture for completing the work on students' management. The detailed information will be presented in an evaluation report. The example of an evaluation report is shown in Figure 7-4.

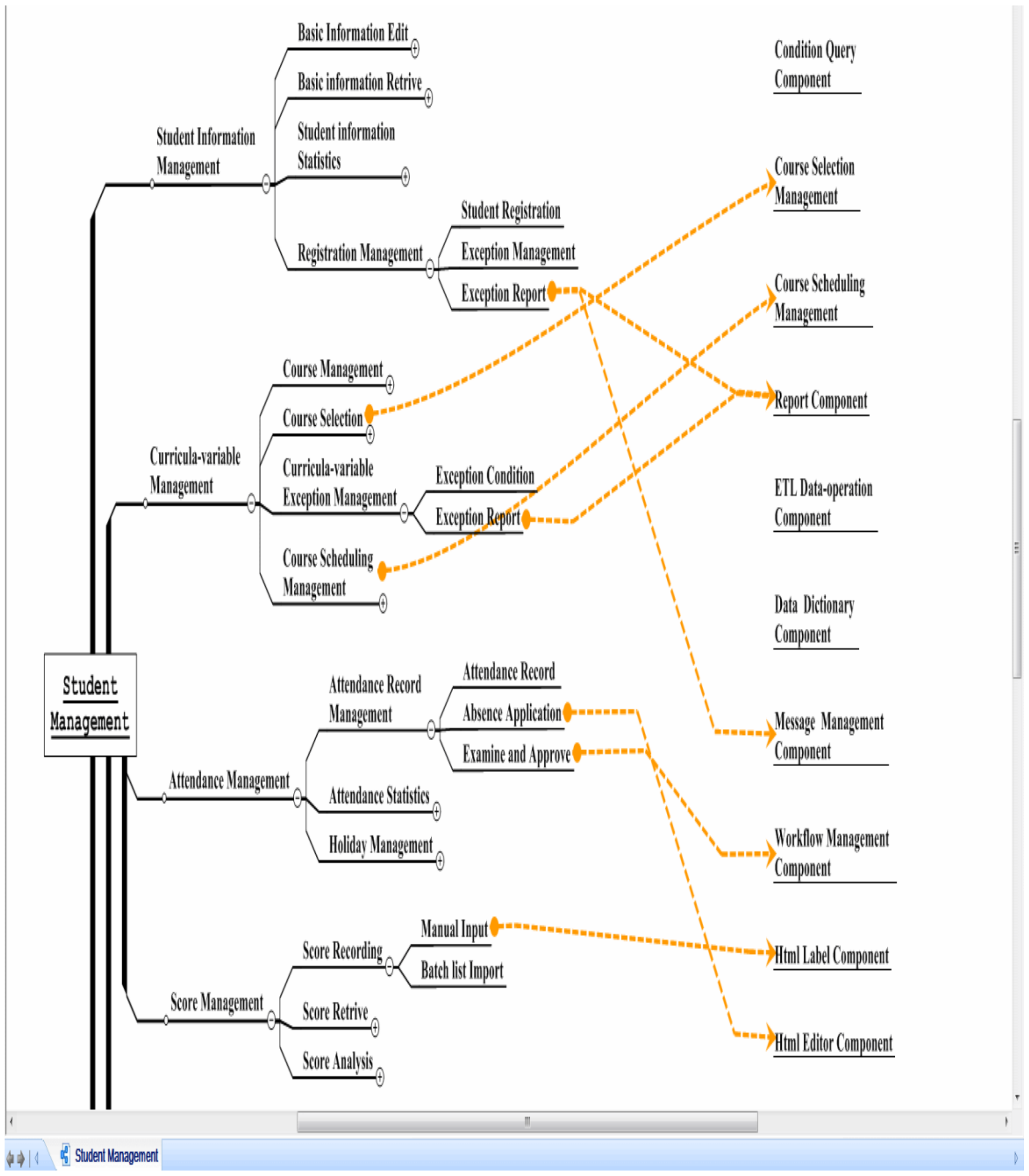
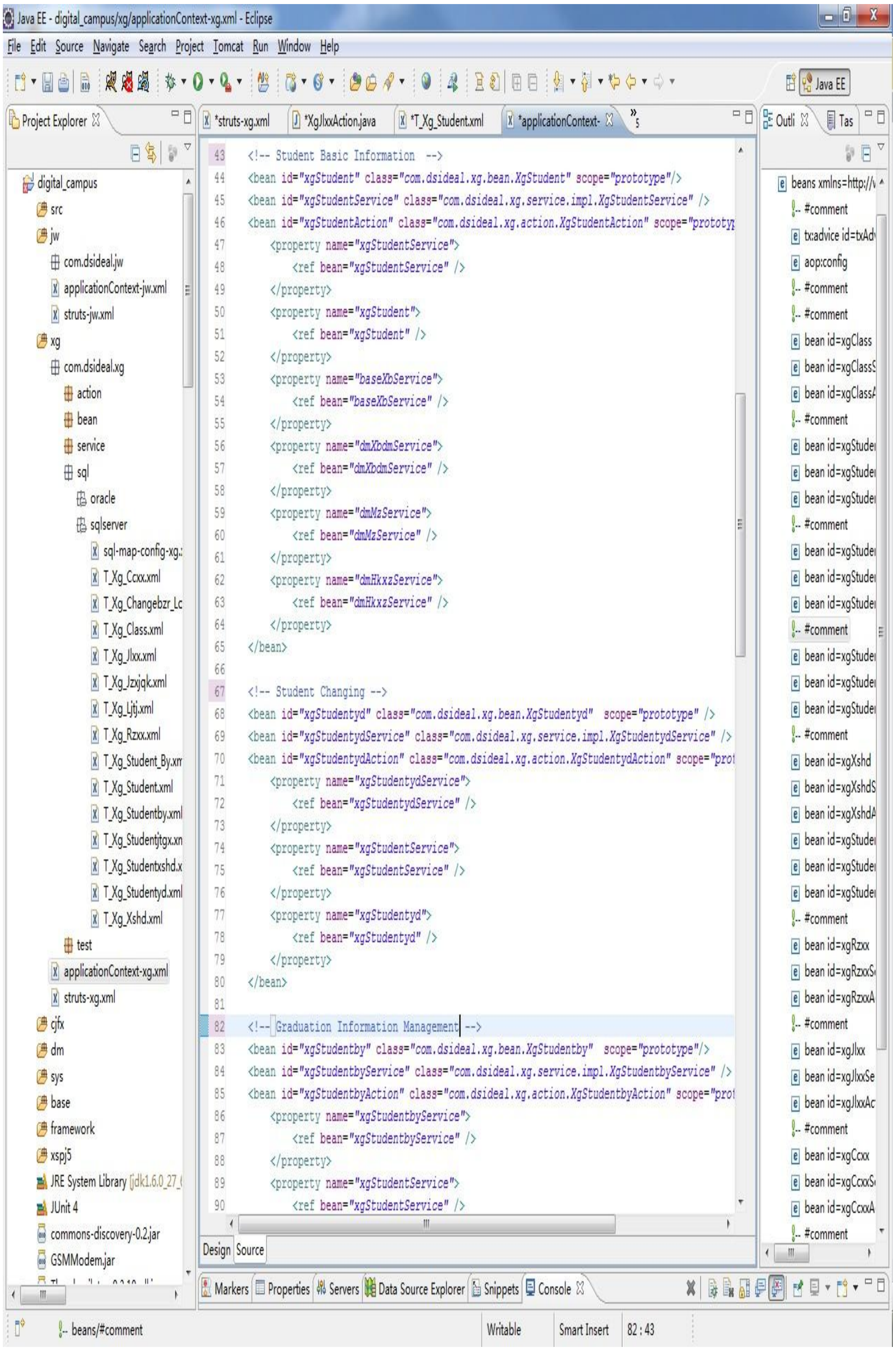


Figure 8-9. The Key Part of SOA Migration Planning Schemas in the Experiment.

A screenshot of service implementation is shown in Figure 8-10.



```xml
<!-- Student Basic Information  -->
<bean id="xgStudent" class="com.dsideal.xg.bean.XgStudent" scope="prototype"/>
<bean id="xgStudentService" class="com.dsideal.xg.service.impl.XgStudentService" />
<bean id="xgStudentAction" class="com.dsideal.xg.action.XgStudentAction" scope="prototyp
    <property name="xgStudentService">
        <ref bean="xgStudentService" />
    </property>
    <property name="xgStudent">
        <ref bean="xgStudent" />
    </property>
    <property name="baseXbService">
        <ref bean="baseXbService" />
    </property>
    <property name="dmXbdmService">
        <ref bean="dmXbdmService" />
    </property>
    <property name="dmMzService">
        <ref bean="dmMzService" />
    </property>
    <property name="dmHkxzService">
        <ref bean="dmHkxzService" />
    </property>
</bean>

<!-- Student Changing -->
<bean id="xgStudentyd" class="com.dsideal.xg.bean.XgStudentyd"  scope="prototype" />
<bean id="xgStudentydService" class="com.dsideal.xg.service.impl.XgStudentydService" />
<bean id="xgStudentydAction" class="com.dsideal.xg.action.XgStudentydAction" scope="prot
    <property name="xgStudentydService">
        <ref bean="xgStudentydService" />
    </property>
    <property name="xgStudentService">
        <ref bean="xgStudentService" />
    </property>
    <property name="xgStudentyd">
        <ref bean="xgStudentyd" />
    </property>
</bean>

<!-- Graduation Information Management -->
<bean id="xgStudentby" class="com.dsideal.xg.bean.XgStudentby"  scope="prototype"/>
<bean id="xgStudentbyService" class="com.dsideal.xg.service.impl.XgStudentbyService" />
<bean id="xgStudentbyAction" class="com.dsideal.xg.action.XgStudentbyAction" scope="prot
    <property name="xgStudentbyService">
        <ref bean="xgStudentbyService" />
    </property>
    <property name="xgStudentService">
        <ref bean="xgStudentService" />
```

Figure 8-10. A Screenshot of Service Implementation.

8.2.5 Evaluation

In this case study, the migration planning schemas based on the functional information (Similarity Matrix) produced by the proposed approach are about 80% the same as the schemas determined by experts. Through analysing the differences between these two results, it is found the main reasons for the differences are not in this approach itself. The main reasons are that the keyword-based similarity calculation lost some information between legacy components and domain targets. Just functional information is not enough for ensuring the quality of SOA migration planning schemas.

For improving the performance of this approach, some non-functional information should be taken into account. An improved decision-making method based on functional and non-functional information (named Hybrid Information) is also evaluated. The performance is 86% the same as the schemas determined by experts. Clearly, the performance is improved. The decision making method based on both functional and non-functional factors is better than the one based on functional factors only. Table 8-3 shows the performance comparison of these two methods.

Table 8-3. Performance Comparison.

	DM on Similarity Matrix	DM on Hybrid Information
Performance	80%	86%

8.3 Conclusion

Two case studies are described to support and evaluate the proposed approach. The application field is e-learning field, which is a large scale application field of information technology. It is known that IT can provide robust help in the education field. Educational software plays an important role in the software industry. The cases in the education field are complex enough. If some approaches and strategies can be supported and validated in the education field, they can be applied in most other fields. To some extent, the application scope of the proposed approach can be expanded. Hence, the proposed SOA migration planning approach is available and applicable.

8.4 Summary

Two case studies are presented in this Chapter to evaluate the proposed approach.

First, improving the Service model and the Component model with SEPAM and ARM.

- The proposed method is applied in an e-learning legacy system. Its architecture cannot satisfy the user’s requirements. Thus, this system will be migrated to a service-oriented architecture.
- In order to discover the coarse-grained services in MELS, a service composition method based on a sequence mining algorithm is proposed. The prototype system has been developed. Some data collection and pre-process methods are introduced.
- For evaluating this method, some questionnaire investigations were done. The questionnaire investigation result is 90% the same as the result obtained by adopting the proposed method. After doing many similar evaluations, it is concluded that the proposed method is promising.

Second, creating SOA migration planning schemas with SOAMT.

- Legacy assets and the new service oriented architecture are introduced. A legacy asset is an education administration system, which consists of a human resource management sub-system, teaching and learning affairs management sub-system, official document management sub-system, general affairs management sub-system and home-school interaction management sub-system. The legacy system has been developed under the environment of Java, Eclipse and Tomcat (middleware). However, in this system, each function has been developed without thinking about the case that some business processes have to pass several sub-systems. The flexibility and adaptability of this legacy system are poor. The new architecture is a hierarchical and flexible reusable architecture, based on domain-specific software architecture (DSA). By using this software architecture, the customised application software can be built in a visual environment.
- The SOA migration planning schemas based on functional factors are 80% the same as the planning schemas determined by experts. The SOA migration

planning schemas based on functional and non-functional factors are 86% the same as the planning schemas determined by experts. Clearly, the performance is improved. The decision making method based on both functional and non-functional factors is better than the one based on functional factors.

- The proposed approach has been applied in an e-learning field. The cases in the education field are complex. If some approaches and strategies can be supported and validated in the education field, they can be applied in most other fields. To some extent, the application scope of the proposed approach can be expanded. Hence, the proposed SOA migration planning approach is available and applicable.

Chapter 9 Conclusions

Objectives

- To summarise the thesis and draw conclusions
- To revisit original contributions
- To evaluate the research by answering the research questions, reviewing the research hypotheses and revisiting the success criteria
- To illustrate the limitations of the work
- To propose future work

9.1 Summary of Thesis

The principal research question in this thesis is: "How to obtain SOA migration planning schemas in order to determine if legacy software systems should be migrated to a SOA computation environment?". To answer this question, the following work has been done.

An SOA migration approach has been proposed, which includes 5 stages, namely, a preparation stage, analysis stage (domain analysis, legacy analysis and data mining method), matching stage, decision-making stage and an evaluation stage.

In the domain analysis sub-stage, a Service model has been established according to domain analysis results; in the legacy analysis sub-stage, a Component model has been established according to legacy analysis results. In the data mining sub-stage, an association rule mining algorithm is proposed for determining the association relationships among legacy components or among domain services. Moreover, a sequence pattern mining algorithm is proposed for service or component identification and composition. These proposed algorithms can perfect the Component model and the

Service model.

In the matching stage, two matching strategies based on text similarity measurement methods are proposed: one is a keyword-based matching strategy, another one is superficial semantic-based matching strategy. Concretely, a keyword-based matching algorithm and a superficial semantic-based matching algorithm are presented for calculating the matching degrees between legacy components and domain targets.

In the decision-making stage, two methods for creating SOA migration planning schemas are investigated. One is a similarity matrix-based method, in which just functional factors are taken into account. Another one is a method based on hybrid information, in which both functional factors and non-functional factors are taken into account.

In the evaluation stage, some simple methods on cost estimation and performance evaluation have been presented. Finally, an evaluation report is summarised according to the user's direction.

For validating the proposed approach, two case studies have been described. One is improving the Service model and the Component model with SEPAM and ARM. The proposed algorithms have been evaluated by using the real data of an e-learning legacy system and the performance is promising. Another one is creating SOA migration planning schemas with SOAMT. This performance is also encouraging. Therefore, the proposed SOA migration planning approach is available and applicable.

9.2 Revisiting Original Contributions

A general SOA migration planning approach has been proposed in Chapter 3. In this Section, the three original contributions described in Chapter 1 will be revisited.

C1. In Chapter 4, a sequence pattern mining algorithm for service/component identification and composition is described and evaluated. In Chapter 5, an association rule mining algorithm for determining the association relationships among legacy components or among domain services has been addressed and evaluated. These two algorithms can be of great benefit to the establishment of

Component model and Service model.

C2. In Chapter 6, the matching strategies based on a keyword level and a superficial semantic level have been investigated. They can be used for measuring the matching relationship between legacy components and domain services. The performance of the matching strategy based on superficial semantic level is better than the one based on keyword level.

C3. In Chapter 7, the decision-making methods based on a similarity matrix and hybrid information have been explored, in which not only the functional factors but also the non-functional factors are taken into account. The experimental results show that the method based on hybrid information is better than the method based on the similarity matrix.

9.3 Evaluation

9.3.1 Answering Research Questions

The evaluation of this study starts by answering the proposed research questions. The global research question presented in Chapter 1 was:

How to obtain SOA migration planning schemas (semi-)automatically instead of by traditional manual work in order to determine if legacy software systems should be migrated to a SOA computation environment?

This question has been answered by proposing an SOA migration planning approach. In this approach, the data mining techniques have been adopted to discover the hidden information from the application data of legacy systems; a text similarity measurement method has been deployed to quantify the matching relationships between legacy components and domain targets. Some heuristic decision making methods have been used to create SOA migration planning schemas. Moreover, the proposed approach has been deployed in two case studies for validation.

A set of research questions is defined follows to refine this global question in detail.

RQ1: why is there a need for migration planning schemas in SOA migration projects?

SOA migration is a complicated task. It is necessary to create SOA migration planning schemas before starting a migration project. Otherwise, once it fails, it will be an expensive mistake. Moreover, it may cause a catastrophic loss of money, time, and resources. Up to now, most of this kind of work is manual work. Therefore, a semi-automatic creating method on SOA migration planning schemas should be investigated. More details are shown in Section 1.1.

RQ2: What is a proposed SOA migration planning approach?

The definition and framework of an SOA migration planning approach are shown in Chapter 3. Moreover, the descriptions on each part of this approach are presented in Chapter 3.

- What are the key factors and their relationships in a proposed SOA migration planning approach?

  An SOA migration planning approach is defined as a 7-tuple : a service model, a Component model, theoretical support frame, matching strategy, decision-making method, user's direction, and SOA migration planning schema evaluation. Their relationships are shown in Figure 3.1.

- What kinds of legacy software system can be processed by the proposed approach?

  The proposed approach can deal with the legacy system that can be decomposed into components. Software components include procedures, modules, objects, files, etc.

- What is the final returned result by the proposed approach?

  The final returned result is an evaluation report, which is created according to the user's directions. More details are shown in Section 7.4.

RQ3: How to analyse legacy systems and domain logics in this approach?

The general methods for legacy analysis and domain analysis are available in this proposed approach. Data mining methods can be used to improve the analysis results returned by general methods.

- What techniques can be used to analyse legacy assets and domain requirements from the angle of application data of legacy systems?

  Data mining techniques can be used to analyse legacy assets and domain requirements from the angle of application data of legacy systems (Chapter 4 and Chapter 5).

- What are the results of domain analysis and legacy analysis in this approach?

  In this proposed approach, the result of domain analysis is a Service model (Section 4.1); the result of legacy analysis is a Component model (Section 5.1).

RQ4: How to measure the matching relationships between legacy components and domain services?

Some matching strategies and algorithms have been shown in Chapter 6.

- How to represent legacy components and domain services?

  A hierarchical directed acyclic graph (HDGA) is adopted to represent legacy components and domain services (Sub-section 4.1.2).

- What techniques can be used to calculate the matching degree between legacy components and domain services?

  Text similarity measurement methods can be used to calculate the matching degree between legacy components and domain services (Section 6.1).

- What are the matching strategies between legacy components and domain services?

  In this thesis, two matching strategies have been explored: one is based on a keyword level (Section 6.2); another one is based on a superficial semantic level (Section 6.3).

RQ5: How to create the SOA migration planning schemas?

The related work is described in Chapter 7.

- What is an SOA migration planning schema?

  An SOA migration planning schema consists of a migration source object, migration target object, matching degree and implementation means (Sub-section 7.1.1).

- How many methods can be used to create SOA migration planning schemas?

  Three methods have been developed to create SOA migration planning schemas: the first is a data mining method; the second is based on a functional factor (similarity matrix); the third is based on a non-functional factor (hybrid information, such as component quality). More details are presented in Sub-section 7.1.3, Section 7.2 and Section 7.3.

- How to evaluate the SOA migration planning schemas?

  A simple evaluation method is shown in Section 7.4.

RQ6: How to validate the proposed approach?

Two case studies have been designed and implemented for validating the proposed approach. More details can be found in Chapter 8.

9.3.2 Revisiting Research Hypotheses

After establishing these research questions, a series of research hypotheses based on them are developed. The underlying hypothesis of this thesis is:

Data mining techniques and text similarity measurement methods in vector space can be used to create migration planning schemas in SOA migration projects.

This hypothesis can be validated through an SOA migration planning approach, which mainly consists of an analysis stage, a matching stage, a decision-making stage and an evaluation stage. A set of hypotheses is derived from the underlying one:

RH1: Data mining techniques can be used to analyse legacy assets and domain requirements from the angle of application data in legacy systems.

A sequence pattern mining algorithm has been used to discover the hidden composite services and business processes from the user's usage behaviour data of legacy systems. It shows that this hypothesis is sound.

RH2: Text similarity measurement methods can be used to calculate the matching degree between legacy components and domain targets.

The corresponding relationships between SOA migration and text retrieval, as well as some developed matching strategies and algorithms, show that this hypothesis is sound.

RH3: User's direction is necessary during the process of creating SOA migration planning schemas. Moreover, the solution is obtained by iteration.

The creation process of the SOA migration planning schemas shown in case study II shows that this hypothesis is sound.

RH4: Not only functional factors but also non-functional factors should be taken into account during the process of creating SOA migration planning schemas.

The quality of SOA migration planning schemas based on both functional factors and non-functional factors is better than the one based on functional factors only. This comparison shows that this hypothesis is sound.

9.3.3 Revisiting the Measure of Success Criteria

In Chapter 1, five success criteria are defined to validate the success of the proposed research described in this thesis. This section will revisit the predefined measure of success.

- *What kind of legacy systems can the proposed approach deal with?*

  The proposed approach can deal with legacy systems that can be decomposed into components. Software components include procedures, modules, objects, files, etc. This approach cannot deal with non-decomposable legacy systems.

- *What types of data mining techniques can be used to analyse legacy assets and domain logics?*

  The association rule mining technique and sequence pattern mining technique can be used to analyse legacy assets and domain logics. An association rule mining algorithm and a sequence pattern mining algorithm have been developed and applied to improve the Service model, Component model and SOA migration planning schemas.

- *How can a text similarity measurement method be applied to establish the matching strategies between legacy components and domain services?*

  The corresponding relationships between SOA migration and text retrieval have been established. Some developed matching strategies and algorithms are also able to prove this.

- *How about the performance of this approach?*

  Two case studies have shown that the performance of the proposed approach is promising and encouraging.

- *How about the implementation of the proposed methods and strategies? For example, is it possible to build a practical tool based on a proposed method?*

  The answer is yes. An SOA Migration Toolkit (SOAMT) based on a Data Mining (DM) technique and a Text Similarity Measurement (TSM) method have been designed and implemented for creating SOA migration planning schemas in the case studies shown in Chapter 8.

9.4 Limitations

After discussing the original contributions and success criteria, the limitations of the proposed research described in this thesis are discussed as follows:

- *The pattern analysis stage is important in a service composition method based on a sequence pattern mining technique.*
  After passing the pattern discovery phase, a great number of frequent patterns will be discovered. Some of them are already known, some of them are ridiculous, and some of them are disguised. If some bad patterns are taken for good ones, then, the performance of the proposed approach will be worse.
- *The poor decomposition method of legacy system may reduce the effectiveness of the proposed approach.*

  The understanding and decomposition of a legacy system are the foundation for establishing a Component model. If the decomposition method is poor, the legacy analysis results (a Component model) will be inadequate and error prone. It may lead to create some mistaken SOA migration planning schemas. Moreover, it may cause loss of money, time, and resources. If the quality of the Component model is higher, the proposed approach will be more powerful.

9.5 Future Work

Based on the above discussions, it can be concluded that the proposed approach described in this thesis is a systematic and effective means for predicting and planning SOA migration projects. Meanwhile, the two case studies have further supported and verified the success of this study. In a SOA migration situation, this semi-automatic approach can instead of the traditional manual work. It will be of great benefit to decision makers and developers.

The research presented in this thesis is not the terminus. There is some future work that can be pursued based on the present work:

- After doing some practice by following this proposed approach, lot of data can be collected on SOA migration planning schemas. Some data mining techniques (such as, the mining method of classification rules based on Rough Set Theory) can be used with this data for discovering the hidden knowledge and rules for a SOA migration situation. The mined results can be used to direct similar cases instead of passing on the whole proposed SOA migration planning approach. Clearly, it can save time, money and resources. For example, the collected data of SOA migration planning schemas can be reorganised into an information table. Some condition attributes and a decision attribute are available in this information table. Some classification methods can be adopted to classify each new created SOA migration planning schema into two categories: realisable and unrealistic. In this situation, the proposed approach will be self-adaptive.
- More attention will be paid to the user’s direction part. It should be more perfect. Evaluation information of SOA migration planning schemas can be of great benefit to the user’s direction. After analysing the evaluation information, the user’s direction can be more reasonable and functional. More factors may be taken into

account. The threshold in User’ direction should be determined through hybrid information.

- If the numbers of legacy components and domain services are huge, the proposed approach may be time consuming or inefficient. In this case, the pre-process stage will be needed. Some classification or clustering methods can be utilised for dividing them into different categories or clusters. And then, re-establishing the corresponding relationships between legacy categories/clusters and domain categories/clusters. The proposed approach will be applied to these small-size matching pairs.

References

[1] A. Arsanjani, "Service-Oriented Modeling and Architecture: How to Identify, Specify, and Realize Services for your SOA",

http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/, November 2004.

[2] A. Brown, S. Johnston and K. Kelly, “Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications”. Cupertino, CA: Rational Software Corporation. *A Rational Software White Paper from IBM,* 2002, pp.11-15.

[3] A. Cimitile and G. Visaggio, "Software Salvaging and the Call Dominance Tree". *The Journal of Systems and Software*, Vol. 28(2), 1995, pp.117–127.

[4] A. Fuhr, T. Horn and V. Riediger, “Using Dynamic Analysis and Clustering for Implementing Services by Reusing Legacy Code”. *IEEE 18th Working Conference on Reverse Engineering*, 2011.

[5] A. Hamou-Lhadj and T. Lethbridge, “An Efficient Algorithm for Detecting Patterns in Traces of Procedure Calls,” *ICSE Workshop on Dynamic Analysis (WODA 2003)*, (Portland, OR, USA), May 2003.

[6] A. Lucia, A. Fasolino and E. Pompella, "A Decisional Framework for Legacy System Management". *Proceedings of the International Conference on Software Maintenance, IEEE CS Press*, 2001, pp. 642–651.

[7] A. Lucia, R. Francese, G. Scanniello and G. Tortora, "Developing Legacy System Migration Methods and Tools for Technology Transfer". *Software-Practice and Experience.* Vol. 38, 2008, pp. 1333–1364.

[8] A. Marchetto and F. Ricca, “From Objects to Services: toward a Stepwise Migration Approach for Java Applications,” *Int J Softw Tools Technol Transfer,* Vol.11(6), 2009, pp. 427–440.

[9] A. Mehta and G. Heineman, "Evolving Legacy System Features into Fine-Grained Components". *ICSE'02,* Buenos-Aires, Argentina, May 2002.

[10] A. Meng, "On Evaluating Self-Adaptive Software", *First International Workshop, IWSAS 2000*, Oxford, UK, Apr. 2000, pp.67-73.

[11] A. Mili, S. Chmiel and R. Gottumukkala et al., "Integrated Cost Model for Software Reuse", *ICSE 2000*.

[12] A. Papkov, "Develop a Migration Strategy from a Legacy Enterprise IT Infrastructure to an SOA-based Enterprise Architecture", *IBM Developer works*, http://www.ibm.com/developerworks/library/ws-migrate2soa/index.html?S_TACT =105AGX20&S_CMP=EDU, last visited in November, 2007.

[13] A. Umar and A. Zordan, "Reengineering for Service Oriented Architectures: A Strategic Decision Model for Integration versus Migration", *The Journal of Systems and Software*, Vol. 82, 2009, pp. 448–462.

[14] A. Zaidman, T. Calders and S. Demeyer et al., "Applying Webmining Techniques to Execution Traces to Support the Program Comprehension Process", *CSMR 2005*.

[15] B. Portier, "SOA terminology overview Part 1: Service, Architecture,Governance, and Business Terms", *IBM Developer Works*, http://www.ibm.com/developerworks/library/ws-soa-term1/index.html, May, 2007. Last visited in November, 2007.

[16] B. Portier, "SOA Terminology Overview Part 2: Development Processes, Models, and Assets", *IBM Developer works,* May, 2007. http://www.ibm.com/developerworks/webservices/library/ws-soa-term2/index.html, Last visited in November, 2007.

[17] B. Portier, "SOA Terminology Overview Part 3: Analysis and Design", *IBM Developer works,*

http://www.ibm.com/developerworks/webservices/library/ws-soa-term3/index.html, May, 2007. Last visited in November, 2007.

[18] B. Újházi, R. Ferenc and D. Poshyvanyk et al., "New Conceptual Coupling and Cohesion Metrics for Object Oriented System", *Working Conference on Source*

Code Analysis and Manipulation, 2010.

[19] B. Wu, D. Lawless, J. Bisbal and J. Grimson, "Legacy System Migration: A Legacy Data Migration Engine", *Proceeding of the 17th International Conference (DATASEM '97),* Brno, Czech Republic, Oct., 1997, pp 129-138.

[20] C. Dai, S. Yang and R. Knott, “Data Transfer Over the Internet for Real Time Applications”, *International Journal of Automation and Computing,* Vol. 3(4), 2006, pp. 414-424.

[21] C. Kim, J. Lim and R. Ng et al. "SQUIRE: Sequential Pattern Mining with Quantities", *Journal of system and software,* Vol.80(10), 2007, pp.1726-1745.

[22] C. Lindig and G. Snelting, "Assessing Modular Structure of Legacy Code based on Mathematical Concept Analysis". *Proceedings of the 19th International Conference on Software Engineering, ACM Press*. Boston, MA, 1997, pp. 349–359.

[23] C. Szyperski, "Component Software: Beyond Object-Oriented Programming", *Addison-Wesley Professional,* 2002.

[24] CBDI Forum, Enterprise Framework for SOA, http://www.cbdiforum.com/secure/interact/2005-03/eng_frame_soa.php.

[25] D. Champeaux, D. Lea and P. Faure, "Object Oriented System Development", *Addison-Wesley,* Reading Mass, 1993.

[26] D. Christopher, P. Manning and R. Schutze, "Introduction to Information Retrieval", *Cambridge University Press,* 2008.

[27] D. Grosso, D. Penta and D. Guzman et al., "An Approach for Mining Services in Database-oriented Applications ". *11TH European Conference on Software Maintenance and Reengineering (CSMR’07).*

[28] D. Linthicum, “Next Generation Application Integration: From Simple Information to Web Services”, *Addison Wesley Press,* 2003.

[29] D. Poshyvanyk and A. Marcus, "The Conceptual Coupling Metrics for Object-Oriented Systems, Software Maintenance", *ICSM’06,* pp. 469-478.

[30] D. Yakimovich, "Dissertation: A Comprehensive Reuse Model For COTS Software Products".

[31] D. Zhou, Z. Zhang and S. Zhong et al., “The Design of Software Architecture for e-Learning Platforms”, *3rd International Conference on E-Learning and Games*, Nanjing, China, Jun. 2008, pp.32-40.

[32] Domain Analysis, http://en.wikipedia.org/wiki/Domain_analysis. From Wikipedia, the free encyclopedia . Available online in May, 2012.

[33] E. Allen, T. Khoshgoftaar and Y. Chen, "Measuring Coupling and Cohesion of Software Modules: an Information-theory Approach", *in Proceedings of 7th International Software Metrics Symposium (METRICS'01),* Apr. 2001, pp. 124-134.

[34] E. Arisholm, L. Briand and A. Foyen, "Dynamic Coupling Measurement for Object-Oriented Software", *IEEE Transactions on Software Engineering,* Vol. 30(8), Aug. 2004, pp. 491-506.

[35] E. Chikofsty and J. Cross, "Reverse Engineering and Design Recovery: A Taxonomy", *IEEE Software,* Vol.7(1), Jan. 1990, pp. 13-17.

[36] E. Newcomer, "Understanding Web Services: XML, WSDL, SOAP, and UDDI, Independent Technology Guides". 2006.

[37] F. Chen., S. Li, and H. Yang et al., “Feature Analysis for Service-Oriented Reengineering”, *IEEE 12th ASIA-PACIFIC Software Engineering Conference (APSEC 2005),* Taipei, Taiwan.

[38] F. Lanubile and G. Visaggio, "Extracting Reusable Functions by Flow Graph-based Program Slicing". *IEEE Transactions on Software Engineering,* Vol.23(4), 1997, pp.246–259.

[39] F. Shull, J. Singer and D. Sjøberg, "Guide to Advanced Empirical Software Engineering". *Springer Verlag,* London. 2008.

[40] F. Zulkernine, "Service-Oriented Architecture (SOA)".

http://www.cs.queensu.ca/home/cords/soa.ppt

[41] G. Canfora, A. Cimitile and A. Lucia et al., "Decomposing Legacy Programs: A First Step towards Migrating to Client–server Platforms". *The Journal of Systems and Software,* Vol. 54, 2000, pp.99–110.

[42] G. Canfora, A. Cimitile and A. Lucia et al., "Decomposing Legacy Systems into

Objects: an Eclectic Approach", *Information & Software Technology*, 2001, pp.401-412.

[43] G. Canfora, A. Cimitile and M. Munro, "An Improved Algorithm for Identifying Reusable Objects in Code". *Software Practice and Experiences*, Vol. 26(1), 1996, pp. 24–48.

[44] G. Canfora, A. Fasolino and G. Frattolillo et al., "A Wrapping Approach for Migrating Legacy System Interactive Functionalities to Service Oriented Architectures", *The Journal of Systems and Software*, Vol. 81, 2008, pp. 463–480.

[45] G. Gui and P. Scott, "New Coupling and Cohesion Metrics for Evaluation of Software Component Reusability", *The 9th International Conference for Young Computer Scientists*, IEEE, 2008.

[46] G. Heineman and W. Councill, "Component-Based Software Engineering: Putting the Pieces Together", *Addison-Wesley Professional*, 2001.

[47] G. Lewis, E. Morris and D. Smith, “SMART: The Service-Oriented Migration and Reuse Technique” (CMU/SEI-05-TN-029). *Software Engineering Institute.* September 2005.

[48] G. Salton, Automatic Text Processing, *Adison-Wesley*, 1989.

[49] G. Sholom et al. “Application of Feature-Oriented Domain Analysis to the Army Movement Control Domain” (CMU/SEI-91-TR-28, ADA 256590). Pittsburgh, PA: *Software Engineering Institute*, Carnegie Mellon University, 1992.

[50] G. Visaggio, "Value-based Decision Model for Renewal Processes in Software Maintenance". *Annals of Software Engineering*. Vol. 9(1–2), 2000, pp.215–233.

[51] H. Gall, M. Jazayeri, and J. Krajewski, "CVS Release History Data for Detecting Logical Couplings", *6th International Workshop on Principles of Software Evolution (IWPSE'03)*. Sept. 2003, pp. 13 - 23.

[52] H. Kim and Y. Kwon, "Restructuring Programs through Program Slicing". *International Journal of Software Engineering and Knowledge Engineering*, Vol.4(3), 1994, pp. 349–368.

[53] H. Muller, M. Orgun and S. Tilley et al., "A Reverse-engineering Approach to

Subsystem Structure Identification". *Journal of Software Maintenance: Research and Practice*, Vol. 5, 1993, pp.181–204.

[54] H. Shirazi, N. Fareghzadeh and A. Seyyedi, “A Combinational Approach to Service Identification in SOA”, *Journal of Applied Sciences Research*, Vol. 5(10), 2009, pp. 1390-1397.

[55] H. Sneed and E. Nyary, "Downsizing Large Application Programs". *Journal of Software Maintenance: Research and Practice,* Vol. 6(5), 1994, pp.105–116.

[56] H. Sneed, "Planning the Reengineering of Legacy Systems". *IEEE Software*, Vol. 12(1), 1995, pp. 24–34.

[57] H. Sneed, “Program Interface Reengineering for Wrapping”, Proceeding of 4th WCRE, *IEEE Computer Society Press*, Amsterdam, Oct. 1997, pp. 206

[58] H. Sneed. "Integrating Legacy Software into a Service Oriented Architecture", *Proceedings of the Conference on Software Maintenance and Reengineering (CSMR'06)*. IEEE Computer Society, 2006.

[59] H. Washizaki, H. Yamamoto and Y. Fukazawa, "A Metrics Suite for Measuring Reusability of Software Components", *Proceedings of the Ninth International Software Metrics Symposium (METRICS’03)*.

[60] H. Yang and M. Ward, "Successful Evolution of Software Systems", *Artech House*, 2003.

[61] I. Ronen, N. Aizenbud and K. Kveler, “Service Identification in Legacy Code using Structured and Unstructured Analysis,” *IBM Programming Languages and Development Environments Seminar*, Haifa, 2007.

[62] IBM Quest Market-Basket Synthetic Data Generator, http://www.almaden.ibm.com/cs/quest/

[63] IBM SOA terms,

http://www.ibm.com/developerworks/library/ws-soa-term1/index.html

[64] IBM,http://download.boulder.ibm.com/ibmdl/pub/software/rational/web/

whitepapers/2003/legacy.pdf.

[65] IBM, Managing Legacy Integration with IBM Rational Software, Technical discussion, *Rational Software, IBM Software Group*, July, 2003.

[66] Ideal Information Technology Institute of Northeast Normal University in China. http://www.dsideal.net/.

[67] J. Bisbal and R. Richardson, "A Survey of Research into Legacy System Migration", *Technique report*.

https://www.scss.tcd.ie/publications/tech-reports/reports.97/TCD-CS-1997-01.pdf

[68] J. Bosch, "Superimposition: a Component Adaptation Technique", *Information and Software Technology*, Vol. 41, 1999, pp. 257–273.

[69] J. Guan, D. Bell and D. Liu, "Data Mining for Maximal Frequent Patterns, Intelligent Data Mining: Techniques and Applications", *Springer*, New York, 2005, pp. 137-162.

[70] J. Han and M. Kamber, "Data Mining: Concepts and Techniques", *China Machine Press*, 2006.

[71] J. Hanson, "Coarse-grained Interfaces Enable Service Composition in SOA", *JavaOne*, Aug. 2003.

[72] J. Heumann, "Requirements Discovery for Legacy Systems", Technical discussion, *Rational Software, IBM Software Group*, July, 2003.

[73] J. Lavery, B. Boldyreff and B. Ling et al., "Modelling the Evolution of Legacy Systems to Web-based Systems", *Journal of Software Maintenance and Evolution*, Vol.16(1), 2004, p.5

[74] J. Li, Z. Zhang and B.Qiao et al., "A Component Mining Approach to Incubate Grid Services in Object-Oriented Legacy Systems", *International Journal of Automation and Computing*, Vol.3(1), 2006, pp. 47-55.

[75] J. Pu, R. Millham and H. Yang, "Acquiring Domain Knowledge in Reverse Engineering the Web-based system into UML", *Conference of IASTED Software Engineering and Application*, 2003.

[76] J. Zhang, J. Chung and C. Chang, "Migration to Web Services Oriented Architecture – A Case Study". *Proceedings of the 2004 ACM Symposium of*

Applied Computing, ACM Press, Nicosia, Cyprus, Mar. 2004.

[77] J. Zhao, "Measuring Coupling in Aspect-Oriented Systems", *in Proceeding of 10th IEEE International Soft. Metrics Symposium (METRICS'04),* Chicago, USA, 2004.

[78] J. Ziemann, K. Leyking and T. Kahl et al., "Enterprise Model driven Migration from Legacy to SOA", *Software Reengineering and Services Workshop,* 2006.

[79] K. Bennett, “Legacy Systems: Copying with Success”, *IEEE Software,* Vol. 12(1), Jan. 1995, pp 19-23.

[80] K. Bennett, M. Ramage and M. Munro, "Decision Model for Legacy Systems". *IEEE Proceedings Software,* Vol. 146(3), 1999, pp. 153–159.

[81] K. Channabasavaiah, K. Holley and E. Tuggle, "Migrating to a Service-Oriented Architecture". *White paper,* G224-7298, IBM, 2004.

[82] K. Kang, J. Lee and K. Kim et al., “FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures”, *Annals of Software Engineering, Springer Netherlands*. Vol. 5, 1998, pp.143–168.

[83] K. Kang, S. Cohen and J.Hess et al., “Feature-Oriented Domain Analysis(FODA) Feasibility Study”, *Technical Report* (Approved for public release), CMU/SEI-90-TR-21, ESD-90-TR-222, Nov. 1990.

[84] L. Briand, P. Devanbu and W. Melo, "An Investigation into Coupling Measures for C++", in *Proceeding of International Conference on Software engineering (ICSE'97),* Boston, MA, May1997, pp. 412 - 421.

[85] L. Etzkorn, W. Hughes and C. Davis, "Automated Reusability Quality Analysis of OO Legacy Software". *Information and Software Technology,* Vol. 43, 2001, pp. 295-308.

[86] L. Markosian, P. Newcomb, and R.Brand et al. "Using an Enabling Technology to Reengineer Legacy Systems". *Communications of the ACM,* Vol. 37(5), 1994, pp. 58–70.

[87] L. Wilkes, "CBDI Report: SOA Reference Models. IT Strategic Office of the Danish Ministry of Science, Technology and Innovation". Version 1.0. October 2005.

[88] Lucene, http://lucene.apache.org.

[89] M. Brodie and M. Stonebraker, "Migrating Legacy Systems: Gateways, Interfaces & the Incremental Approach". *Morgan Kaufmann*, San Francisco, 1995.

[90] M. Christodorescu, S. Jha and C. Kruegel, “Mining Specifications of Malicious Behaviour”, *6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2007)*, Dubrovnik, Croatia, Sep. 2007, pp.5-14.

[91] M. DiBernardo, R. Pottinger and M. Wilkinson, "Semi-automatic Web Service Composition for the Life Sciences Using the BioMoby Semantic Web Framework", *Journal of Biomedical Informatics*, Vol. 41(5), Oct. 2008, pp. 837-847.

[92] M. El-Ramly, E. Stroulia and P. Sorenson, “From Run-time Behaviour to Usage Scenarios: an Interaction-Pattern Mining Approach”, *8th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM Press*, 2002, pp. 315–324.

[93] M. Hitz and B. Montazeri, "Measuring Coupling and Cohesion In Object-Oriented Systems". *Proceeding Int. Symposium on Applied Corporate Computing*, Oct. 1995, Monterrey, Mexico, pp. 78-84.

http://www.isys.uni-klu.ac.at/PDF/1995-0043-MHBM.pdf

[94] M. McGill, "An evaluation of factors affecting document ranking by information retrieval systems". *Project report, Syracuse University, School of Information Studies*, 1979.

[95] M. Mendonca and N. Sunderhaft, “Mining Software Engineering Data: A Survey”,1999. https://www.dacs.dtic.mil/techs/datamining/index.php, last visited in October, 2008.

[96] M. Pohlmann and M. Schonefeld, “An Evolutionary Integration Approach using Dynamic CORBA in a Typical Banking Environment”, presented at the *Case Studies Workshop of the Sixth European Conference on Software Maintenance and Reengineering*, Mar.2002. Budapest, Hungary.

[97] M. Shaw, "Architectural Issues in Software Reuse: It's Not Just The Functionality,

It's the Packaging", *Proceeding of IEEE Symposium on Software Reusability*, Apr. 1995.

[98] M. Zakaria, B. Djamal and T. Philippe et al., "Service Composition”, *in Data and Knowledge Engineering*, Vol. 62(2), Aug. 2007, pp. 327-351.

[99] Microsoft Corporation, “Enabling Real World SOA through the Microsoft Platform”, 2006.

[100] NorthEast Normal University (NENU) in China. http://www.nenu.edu.cn/

[101] O. Rotaru and M. Dobre, "Reusability Metrics for Software Components", 2005 *IEEE*.

[102] O. Zimmermann, P. Krogdahl and C. Gee, “Elements of Service-Oriented Analysis and Design”, *IBM Developer Work*, 2006. Last visited in Dec.2007. http://www.ibm.com/developerworks/webservices/library/ws-soad1/

[103] Oracle, "Oracle IT Modernization Series: The Types of Modernization", Oracle White Paper, 2006.

[104] P. Breuer, H. Haughton and K. Lano, "Reverse-engineering COBOL via Formal Methods". *Journal of Software Maintenance: Research and Practice*, Vol. 5, 1993, pp. 13–35.

[105] P. Chung, Y. Huang and S. Yajnik et al, " DCOM and CORBA Side by Side, Step By Step, and Layer by Layer," *C++ Report*, Vol. 10(1), Jan. 1998, pp. 18-29.

[106] P. Pendharkar and G.Subramanian, “Connectionist Models for Learning, Discovering, and Forecasting Software Effort: An Empirical Study”, *Computer Information Systems*, Vol. 43(1), 2002, pp. 7-14.

[107] P. Pendharkar, G. Subramanian and J. Rodger, "A Probabilistic Model for Predicting Software Development Effort" , *IEEE Transactions on Software Engineering*, Vol. 31(7), July 2005.

[108] R Agrawal and R. Srikant, “Fast Algorithms for Mining Association Rules”. *Proceedings of the 20th International Conference on Very Large Database (VLDB)*, Morgan Kaufmann, Santiago, Chile, Sep. 1994, pp.487-499.

[109] R. Gimnich, "SOA Migration – Approaches and Experience", *IBM Software Group*, SOA Advanced Technology / Enterprise Integration Solutions, Wikhelm-Fay-Str.30-34, D-65936 Frankfurt.

[110] R. Agrawal and R.Srikant, “IBM Research Report RJ9910: Mining Sequential Patterns”, *IBM research division*, Almaden research center, CA. 1994.

[111] R. Baeza-Yates and B. Ribeiro-Neto, "Modern Information Retrieval", *Addison-Wesley*, 1999.

[112] R. Seacord, D. Plakosh and G. Lewis, “Modernizing Legacy Systems: Software Technologies”, Engineering Processes and Business Practices, *Addison Wesley Press*, 2003.

[113] R. Srikant and R. Agrawal, “Mining Sequential Patterns: Generalizations and Performance Improvements”. *Proceedings of the Fifth International Conference on Extending Database Technology (EDBT), Springer-Verlag Avignon*, France, Mar. 1996, pp. 3-17.

[114] R. Srikant, “Fast Algorithms for Mining Association Rules and Sequential Patterns” [DB/OL], http://www.rsrikant.com/ Srikant’s Publications.htm.*Dissertations*, University of Wisconsin, Madison. 1996.

[115] S. Alahmari, E. Zaluska and D. De Roure, “A Service Identification Framework for Legacy System Migration into SOA,” *in IEEE Seventh International Conference on Services Computing. IEEE Computer Society*, 2010, pp. 614–617.

[116] S. Cetin, N. Altintas and H. Oguztuzun et al., “Legacy Migration to Service-Oriented Computing with Mashups”, *International Conference on Software Engineering Advances(ICSEA 2007).*

[117] S. Chidamber and C. F. Kemerer. "Towards a Metrics Suite for object-oriented Design". *In Proceeding of OOPSLA '91, ACM* ,1991, pp.197-211.

[118] S. Chidamber and C. Kemerer. "A Metrics Suite for Object-oriented Design". *IEEE Trans. Software Eng.*, Vol. 20(6), Jun.1994, pp.476-493.

[119] S. Chung, P. Young and J. Nelson, "Service-Oriented Software Reengineering: Bertie3 as Web Services". *Proceedings of the 2005 IEEE International Conference*

on Web Services (ICWS'05), IEEE Computer Society, Orlando, FL, USA, July 2005.

[120] S. Deng, J. Wu et al., "Automatic Web Service Composition Based on Backward Tree". *Journal of Software*, Vol.18(8), 2007, pp.1896—1910.

[121] S. Easterbrook, J. Singer, M. Storey and D. Damian, "Selecting empirical methods for software engineering research", Chapter 11 in Guide to Advanced Empirical Software Engineering. *Springer Verlag*: London, 2008, pp.282-311.

[122] S. Inaganti and G. Behara, "Service Identification: BPM and SOA Handshake", www.bptrends.com. Mar. 2007.

[123] S. Kramer and H. Kaindl, "Coupling and Cohesion Metrics for Knowledge-based Systems using Frames and Rules", *ACM Trans. on Soft. Engineering and Methodology (TOSEM),* Vol. 13(3), July 2004, pp. 332-358.

[124] S. Li, F. Chen, Z. Liang and H. Yang, "Using Feature-Oriented Analysis to Recover Legacy Software Design for Software Evolution", *International Conference on Software Engineering and Knowledge Engineering (SEKE'05)*, Taipei, Taiwan.

[125] S. Mahmood , R. Lai and Y. Kim et al., "A Survey of Component based System Quality Assurance and Assessment", *Information and Software Technology*, Vol. 47, 2005, pp. 693–707.

[126] S. Pressman, "Software Engineering - A Practitioner's Approach" - *Fourth Edition.* ISBN 0-07-052182-4, 1982.

[127] S. Thummalapenta and T. Xie, "PARSEWeb: A Programmer Assistant for Reusing Open Source Code on the Web", ASE'07, Atlanta, Georgia, USA. Nov. 2007, pp. 204-213.

[128] S. Zhong, J. Li and Z. Zhang et al., "Methods on Educational Resource Development and Application", *3rd International Conference on E-Learning and Games*, Nanjing, China, Jun. 2008, pp.290-301.

[129] SAP, "NetWeaver Open Integration Platform",

https://www.sdn.sap.com/irj/sdn/developerareas/netweaver.

[130] Software Engineering Institute, Camegie Mellon University, Pittsburgh, Pennsylvania 15213, USA. http://www.sei.cmu.edu.

[131] T. Biggerstaff, "Design Recovery for Maintenance and Reuse", *IEEE Software*, July 1989, pp. 36-49.

[132] T. Denmat, M. Ducassé and O. Ridoux, "Data Mining and Cross-checking of Execution Traces : A Re-interpretation of Jones, Harrold and Stasko Test Information Visualization", 2*0th IEEE/ACM International Conference on Automated Software Engineering,* Long Beach, California, USA, Nov. 2005, pp.396-399.

[133] T. Erl, "Service-Oriented Architecture (SOA): Concepts, Technology, and Design", *the Prentice Hall*, Service-Oriented Computing Series. 2007.

[134] T. Hastings and A. Sajeev, "A Vector-Based Approach to Software Size Measurement and Effort Estimation," *IEEE Trans. Software Eng.*, Vol. 27(4), 2001, pp. 337-350.

[135] T. Xie and A. Hassan, "Mining Software Engineering Data", http://ase.csc.ncsu.edu/dmse/dmse-icse07-tutorial.ppt#256,1,Mining Software Engineering Data, last visit August 30, 2008.

[136] T. Zimmermann, A. Zeller and P. Weissgerber et al., "Mining Version Histories to Guide Software Changes", *IEEE Transactions on Software Engineering*, Vol. 31(6), Jun. 2005, pp. 429-445.

[137] T. Xie, "Mining Software Engineering Data Bibliography", [Online],

available: http://ase.csc.ncsu.edu/dmse/setasks, Sep., 2010.

[138] V. Radha, V. Gulati and R. Thapar, "Evolution of Web Services Approach in SFMS – A Case Study". *Proceedings of the IEEE International Conference on Web Services (ICWS'04), IEEE Computer Society*, San Diego, CA,USA. Jul. 2004.

[139] Wikipedia, Component-based software engineering, Retrieved 2012-08-12, http://en.wikipedia.org/wiki/Component-based_software_engineering.

[140] Wikipedia, http://en.wikipedia.org/wiki/Cohesion_(computer_science), last visited on 01-07-2012.

[141] X.Liu and Q. Wang, "Study on Application of a Quantitative Evaluation Approach for Software Architecture Adaptability", *Proceedings of the Fifth International Conference on Quality Software (QSIC'05)*.

[142] Y. Chen and T. Huang. "A Novel Knowledge Discovering Model for Mining Fuzzy Multi-level Sequential Patterns in Sequence Databases". *Data & Knowledge Engineering*, Vol. 66(3), Sep. 2008, pp. 349-367.

[143] Y. Chen and T. Huang. A New Approach for Discovering Fuzzy Quantitative Sequential Patterns in Sequence Databases. *Fuzzy Sets and Systems*, Vol. 157(12), Jun. 2006, pp. 1641-1661.

[144] Y. Lee, B. Liang and S. Wu et al., "Measuring the Coupling and Cohesion of an Object-Oriented Program Based on Information Flow", *in Proceedings of International Conference on Software Quality*, Maribor, Slovenia, 1995.

[145] Z. Zhang and H. Yang, "One-Stone-Two-Birds: Legacy System Re-engineering and Web Services Development - - A Component-Based and Service-Oriented Approach" in *Proceedings of Postgraduate Research Conference in Electronics, Photonics, Communications & Networks, and Computing Science (PREP 2004)*, Hatfield, UK, 2004.

[146] Z. Zhang and H. Yang, "Incubating Services in Legacy Systems for Architectural Migration", IEEE Computer Society, *Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC'04)*.

[147] Z. Zhang, D. Zhou and H. Yang et al., "A Service Composition Approach Based on Sequence Mining for Migrating E-learning Legacy System to SOA", *International Journal of Automation and Computing*, Vol. 7(4), Nov. 2010, pp. 584-595.

[148] Z. Zhang, D. Zhou and S. Zhong et al., "Researches on the Decision-making Algorithm in an SOA migration Model"，*in the proceeding of the 9th international FLINS Conference on Foundations and Applications of Computational Intelligence (FLINS2010)*, Chengdu (EMei), China, Aug. 2010, pp.917-922.

[149] Z. Zhang, H. Yang and D. Zhou et al., "An SOA Based Approach to

User-Oriented System Migration"， in the proceeding of 10th IEEE international conference on computer and information technology(CIT2010), International symposium on advanced topics on information technologies and applications (ITA2010), in Bradford, UK, Jun. 2010.

[150] Z. Zhang, H. Yang and W. Chu, “Extracting Reusable Object-Oriented Legacy Code Segments with Combined Formal Concept Analysis and Slicing Techniques for Service Integration”, *Proceedings of the Sixth International Conference on Quality Software (QSIC'06).*

[151] Z. Zhang, L. Zhang and S. Zhong et al.,

“Improving Algorithm Apriori for Data Mining”, *8th International FLINs Conference on Computational Intelligence in Decision and Control*, Madrid , Spain, Sep. 2008, pp.17-22.

[152] Z. Zhang, L. Zhang and S. Zhong et al.,“A New Algorithm for Mining Sequential Patterns”, *5th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD’08)*, Jinan, China. Oct. 2008, pp.625-629.

[153] Z. Zhang, R. Liu and H. Yang, “Service Identification and Packaging in Service Oriented Reengineering,” *in Proceedings of the 7th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, 2005, pp. 241–249.

[154] Z. Zhang, S. Zhong and J. Guan, "Improving Algorithm AprioriAll/Some for Mining Sequential Patterns", *Journal of Northeast Normal University* (Natural Science Edition), Vol.39(4), Dec. 2007, pp.46-53.

Appendix A Templates and Examples of Related XML Files

This section presents some templates or examples of related XML files, which are defined in this study.

1. XML_{Domain}

```
<?xml version="1.0" encoding="UTF-8"?>
<dsideal>
  <domains>
        <application-level>
              <application-node id="appNode1">
                    <item-father></item-father>
                    <item-function></item-function>
                    <item-data></item-data>
                    <item-son></item-son>
              </application-node>
              <application-node id="appNode2">
                    <item-father></item-father>
                    <item-function></item-function>
                    <item-data></item-data>
                    <item-son></item-son>
              </application-node>
        </application-level>
        <business-process-level>
              <business-node id="busNode1">
                    <item-father></item-father>
                    <item-function></item-function>
                    <item-data></item-data>
                    <item-son></item-son>
              </business-node>
              <business-node id="busNode2">
                    <item-father></item-father>
                    <item-function></item-function>
                    <item-data></item-data>
                    <item-son></item-son>
              </business-node>
        </business-process-level>
        <service-or-composite-service-level>
              <service-node id="serNode1">
                    <item-father></item-father>
                    <item-function></item-function>
                    <item-data></item-data>
                    <item-son></item-son>
```

```
            </service-node>
            <service-node id="serNode2">
                <item-father></item-father>
                <item-function></item-function>
                <item-data></item-data>
                <item-son></item-son>
            </service-node>
        </service-or-composite-service-level>
    </domains>
    <domain-corrs>
        <application-business-corr>
            <corr>
                <application-node>appNode1</application-node>
                <business-nodes>
                    <business-node>busNode1</business-node>
                    <business-node>busNode2</business-node>
                </business-nodes>
            </corr>
            <corr>
                <application-node>appNode2</application-node>
                <business-nodes>
                    <business-node>busNode1</business-node>
                    <business-node>busNode2</business-node>
                </business-nodes>
            </corr>
        </application-business-corr>
        <business-service-corr>
            <corr>
                <business-node>busNode2</business-node>
                <service-nodes>
                    <service-node>serNode1</service-node>
                    <service-node>serNode2</service-node>
                </service-nodes>
            </corr>
            <corr>
                <business-node>busNode2</business-node>
                <service-nodes>
                    <service-node>serNode1</service-node>
                    <service-node>serNode2</service-node>
                </service-nodes>
            </corr>
        </business-service-corr>
    </domain-corrs>
</dsideal>
```

2. XML_{Legacy}

```
<?xml version="1.0" encoding="UTF-8"?>
<domains>
  <application-component-level>
        <application-node id="appNode1">
              <item-father></item-father>
              <item-function></item-function>
              <item-data></item-data>
              <item-code></item-code>
              <item-mining></item-mining>
              <item-uml></item-uml>
              <item-son></item-son>
        </application-node>
        <application-node id="appNode2">
              <item-father></item-father>
              <item-function></item-function>
              <item-data></item-data>
              <item-code></item-code>
              <item-mining></item-mining>
              <item-uml></item-uml>
              <item-son></item-son>
        </application-node>
  </application-component-level>
  <business-component-level>
        <business-node id="busNode1">
              <item-father></item-father>
              <item-function></item-function>
              <item-data></item-data>
              <item-code></item-code>
              <item-mining></item-mining>
              <item-uml></item-uml>
              <item-son></item-son>
        </business-node>
        <business-node id="busNode2">
              <item-father></item-father>
              <item-function></item-function>
              <item-data></item-data>
              <item-code></item-code>
              <item-mining></item-mining>
              <item-uml></item-uml>
              <item-son></item-son>
        </business-node>
  </business-component-level>
  <atomic-component-level>
        <service-node id="atoNode1">
              <item-father></item-father>
              <item-function></item-function>
              <item-data></item-data>
              <item-code></item-code>
              <item-mining></item-mining>
```

```
          <item-uml></item-uml>
          <item-son></item-son>
      </service-node>
      <service-node id="atoNode2">
          <item-father></item-father>
          <item-function></item-function>
          <item-data></item-data>
          <item-code></item-code>
          <item-mining></item-mining>
          <item-uml></item-uml>
          <item-son></item-son>
      </service-node>
  </atomic-component-level>
</domains>
<domain-corrs>
  <application-business-corr>
      <corr>
          <business-node>busNode2</business-node>
          <application-nodes>
              <application-node>appNode1</application-node>
              <application-node>appNode2</application-node>
          </application-nodes>
      </corr>
      <corr>
          <business-node>busNode1</business-node>
          <application-nodes>
              <application-node>appNode21</application-node>
              <application-node>appNode2</application-node>
          </application-nodes>
      </corr>
  </application-business-corr>
  <business-atomic-corr>
      <corr>
          <atomic-node>atoNode1</atomic-node>
          <business-nodes>
              <business-node>busNode1</business-node>
              <business-node>busNode2</business-node>
          </business-nodes>
      </corr>
      <corr>
          <atomic-node>atoNode2</atomic-node>
          <business-nodes>
              <business-node>busNode1</business-node>
              <business-node>busNode2</business-node>
          </business-nodes>
      </corr>
  </business-atomic-corr>
</domain-corrs>
```

3. XML$_{\text{Match}}$

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="7.2.xsl"?>
<!--
  The matching relationships between domain targets and legacy components based on Keyword and Semantic level
-->
<dsideal>
<domain-item domain="S31">
  <item>
        <Legacy1>L21+L22</Legacy1>
        <Legacy2>L21+L31+L32</Legacy2>
        <Sim_Keyword>0.9</Sim_Keyword>
        <Sim_Semantic>0.96</Sim_Semantic>
  </item>
  <item>
        <Legacy1>L1</Legacy1>
        <Legacy2>L21+L22+L23</Legacy2>
        <Sim_Keyword>0.7</Sim_Keyword>
        <Sim_Semantic>0.81</Sim_Semantic>
  </item>
  <item>
        <Legacy1>L22</Legacy1>
        <Legacy2>L31+L32</Legacy2>
        <Sim_Keyword>0.6</Sim_Keyword>
        <Sim_Semantic>0.73</Sim_Semantic>
  </item>
  <item>
        <Legacy1>L25</Legacy1>
        <Legacy2>L36+L37</Legacy2>
        <Sim_Keyword>0.6</Sim_Keyword>
        <Sim_Semantic>0.0</Sim_Semantic>
  </item>
  <item>
        <Legacy1>L3</Legacy1>
        <Legacy2>L36+L37</Legacy2>
        <Sim_Keyword>0.5</Sim_Keyword>
        <Sim_Semantic>0.0</Sim_Semantic>
  </item>
  <item>
        <Legacy1>L21</Legacy1>
        <Legacy2></Legacy2>
        <Sim_Keyword>0.5</Sim_Keyword>
        <Sim_Semantic>0.68</Sim_Semantic>
  </item>
  <item>
        <Legacy1>L31</Legacy1>
        <Legacy2></Legacy2>
        <Sim_Keyword>0.5</Sim_Keyword>
        <Sim_Semantic>0.0</Sim_Semantic>
  </item>
</domain-item>
<domain-item domain="S32">
  <item>
        <Legacy1>L33+L34</Legacy1>
        <Legacy2></Legacy2>
        <Sim_Keyword>0.9</Sim_Keyword>
        <Sim_Semantic>0.8</Sim_Semantic>
  </item>
  <item>
        <Legacy1>L24</Legacy1>
        <Legacy2>L33+L34+L35</Legacy2>
        <Sim_Keyword>0.8</Sim_Keyword>
        <Sim_Semantic>0.73</Sim_Semantic>
  </item>
  <item>
        <Legacy1>L33</Legacy1>
        <Legacy2></Legacy2>
        <Sim_Keyword>0.8</Sim_Keyword>
        <Sim_Semantic>0.87</Sim_Semantic>
  </item>
  <item>
        <Legacy1>L2</Legacy1>
        <Legacy2>L23+L24</Legacy2>
        <Sim_Keyword>0.7</Sim_Keyword>
        <Sim_Semantic>0.63</Sim_Semantic>
  </item>
  <item>
        <Legacy1>L34</Legacy1>
        <Legacy2></Legacy2>
        <Sim_Keyword>0.7</Sim_Keyword>
        <Sim_Semantic>0.0</Sim_Semantic>
  </item>
  <item>
        <Legacy1>L23</Legacy1>
        <Legacy2>L32+L34</Legacy2>
        <Sim_Keyword>0.5</Sim_Keyword>
        <Sim_Semantic>0.0</Sim_Semantic>
  </item>
  <item>
        <Legacy1>L34+L35</Legacy1>
        <Legacy2></Legacy2>
        <Sim_Keyword>0.5</Sim_Keyword>
        <Sim_Semantic>0.0</Sim_Semantic>
  </item>
</domain-item>
<domain-item domain="S25">
  <item>
        <Legacy1>L3</Legacy1>
        <Legacy2>L25+L35</Legacy2>
        <Sim_Keyword>0.6</Sim_Keyword>
```

```
            <Sim_Semantic>0.6</Sim_Semantic>
    </item>
    <item>
            <Legacy1>L25</Legacy1>
            <Legacy2>L36+L37</Legacy2>
            <Sim_Keyword>0.6</Sim_Keyword>
            <Sim_Semantic>0.81</Sim_Semantic>
    </item>
    <item>
            <Legacy1>L36</Legacy1>
            <Legacy2></Legacy2>
            <Sim_Keyword>0.5</Sim_Keyword>
            <Sim_Semantic>0.64</Sim_Semantic>
    </item>
</domain-item>
</dsideal>
```

4. XML$_{Direction}$

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="7.3.xsl"?>
<!--
  The mapping relation between the implementation means and the user’s direction
-->
<dsideal>
<pram S1="0.9" S2="0.8" S3="0.7"/>
<domain-item ud="cost-first(CF)">
    <item>
            <SIM>Sim≥λ 2</SIM>
            <RM>Wrapper</RM>
    </item>
    <item>
            <SIM>Others</SIM>
            <RM>Re-development</RM>
    </item>
</domain-item>
<domain-item ud="function-first(FF)">
    <item>
            <SIM>Sim≥λ 3</SIM>
            <RM>Wrapper</RM>
    </item>
    <item>
            <SIM>Others</SIM>
            <RM>Re-development</RM>
    </item>
</domain-item>
<domain-item ud="Performance-first(PF)">
    <item>
            <SIM>Sim≥λ 1</SIM>
            <RM>Wrapper</RM>
    </item>
    <item>
          <SIM>λ 1&gt;Sim≥λ 2</SIM>
          <RM>Modification</RM>
    </item>
    <item>
          <SIM>Sim&lt;λ2</SIM>
          <RM>Re-development</RM>
    </item>
  </domain-item>
  </dsideal>
```

5. XML$_{\text{CandidateSchema}}$

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="CandidateSchema7-2.xsl"?>
<!--
  A template of Candidate SOA migration planning schemas
-->
<dsideal>
  <domain-item domain="">
        <item>
              <Legacy1></Legacy1>
              <Legacy2></Legacy2>
              <Sim_Keyord></Sim_Keyord>
              <CF></CF>
              <PF></PF>
              <FF></FF>
        </item>
        <item>
              <Legacy1></Legacy1>
              <Legacy2></Legacy2>
              <Sim_Keyord></Sim_Keyord>
              <CF></CF>
              <PF></PF>
              <FF></FF>
        </item>
        <item>
              <Legacy1><PF></PF></Legacy1>
              <Legacy2></Legacy2>
              <Sim_Keyord></Sim_Keyord>
              <CF></CF>
              <PF></PF>
              <FF></FF>
        </item>
        <item>
              <Legacy1></Legacy1>
              <Legacy2></Legacy2>
              <Sim_Keyord></Sim_Keyord>
              <CF></CF>
              <PF></PF>
              <FF></FF>
          </item>
          <item>
              <Legacy1></Legacy1>
              <Legacy2></Legacy2>
              <Sim_Keyord></Sim_Keyord>
              <CF></CF>
              <PF></PF>
              <FF></FF>
          </item>
      </domain-item>
      <domain-item domain="">
          <item>
              <Legacy1></Legacy1>
              <Legacy2></Legacy2>
              <Sim_Keyord></Sim_Keyord>
              <CF></CF>
              <PF></PF>
              <FF></FF>
          </item>
          <item>
              <Legacy1></Legacy1>
              <Legacy2></Legacy2>
              <Sim_Keyord></Sim_Keyord>
              <CF></CF>
              <PF></PF>
              <FF></FF>
          </item>
          <item>
              <Legacy1></Legacy1>
              <Legacy2></Legacy2>
              <Sim_Keyord></Sim_Keyord>
              <CF></CF>
              <PF></PF>
              <FF></FF>
          </item>
          <item>
              <Legacy1></Legacy1>
              <Legacy2></Legacy2>
              <Sim_Keyord></Sim_Keyord>
              <CF></CF>
              <PF></PF>
              <FF></FF>
          </item>
          <item>
              <Legacy1></Legacy1>
              <Legacy2></Legacy2>
              <Sim_Keyord></Sim_Keyord>
              <CF></CF>
              <PF></PF>
              <FF></FF>
          </item>
```

```
  </domain-item>
  <domain-item domain="">
        <item>
              <Legacy1></Legacy1>
              <Legacy2></Legacy2>
              <Sim_Keyord></Sim_Keyord>
              <CF></CF>
              <PF></PF>
              <FF></FF>
        </item>
        <item>
              <Legacy1></Legacy1>
              <Legacy2></Legacy2>
              <Sim_Keyord></Sim_Keyord>
              <CF></CF>
              <PF></PF>
              <FF></FF>
        </item>
        <item>
              <Legacy1></Legacy1>
              <Legacy2></Legacy2>
              <Sim_Keyord></Sim_Keyord>
              <CF></CF>
              <PF></PF>
              <FF></FF>
        </item>
  </domain-item>
</dsideal>
```

6. $\text{XML}_{\text{FinalSchema}}$

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="7.7.xsl"?>
<!--
  A template of final SOA migration schemas
-->
<dsideal>
  <domain-item>
        <item>
              <Num></Num>
              <DT></DT>
              <LS></LS>
              <SSM></SSM>
              <VHI></VHI>
              <CF></CF>
              <PF></PF>
              <FF></FF>
        </item>
        <item>
              <Num></Num>
              <DT></DT>
```

```
                <LS></LS>
                <SSM></SSM>
                <VHI></VHI>
                <CF></CF>
                <PF></PF>
                <FF></FF>
            </item>
            <item>
                <Num></Num>
                <DT></DT>
                <LS></LS>
                <SSM></SSM>
                <VHI></VHI>
                <CF></CF>
                <PF></PF>
                <FF></FF>
            </item>
            <item>
                <Num></Num>
                <DT></DT>
                <LS></LS>
                <SSM></SSM>
                <VHI></VHI>
                <CF></CF>
                <PF></PF>
                <FF></FF>
            </item>
      </domain-item>
    </dsideal>
```

7. XML$_{\text{Expert Schemas}}$

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="7.8.xsl"?>
<!--
Expert Migration Schema
-->
<dsideal>
<Expert-item>
  <item>
        <Num></Num>
        <DT></DT>
        <LS></LS>
  </item>
  <item>
        <Num></Num>
        <DT></DT>
        <LS></LS>
  </item>
  <item>
```

```
            <Num></Num>
            <DT></DT>
            <LS></LS>
     </item>
   </Expert-item>
   </dsideal>
```

8. XML$_{\text{Performance}}$

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet    href="performance.xsl"      type="text/xsl" ?>
<dsideal>
   <performance-item dt="S31">
      <item>
         <PLS>L21+L31+L32</PLS>
         <ELS>L21+L31+L32</ELS>
         <Performance>1</Performance>
      </item>
      <item>
         <PLS>L1</PLS>
         <ELS></ELS>
         <Performance></Performance>
      </item>
      <item>
         <PLS>L25</PLS>
         <ELS></ELS>
         <Performance></Performance>
      </item>
      <item>
         <PLS>L31+L32</PLS>
         <ELS></ELS>
         <Performance></Performance>
      </item>
   </performance-item>
   <performance-item dt="S32">
      <item>
         <PLS>L33+l34</PLS>
         <ELS>L33+L34+L35</ELS>
         <Performance>1</Performance>
      </item>
      <item>
         <PLS>L33+L34+L35</PLS>
         <ELS></ELS>
         <Performance></Performance>
      </item>
      <item>
         <PLS>L2</PLS>
         <ELS></ELS>
         <Performance></Performance>
      </item>
```

```
            <item>
                <PLS>L33</PLS>
                <ELS></ELS>
                <Performance></Performance>
            </item>
        </performance-item>
        <performance-item dt="S33">
            <item>
                <PLS></PLS>
                <ELS>L24</ELS>
                <Performance>0</Performance>
            </item>
        </performance-item>
        <performance-item dt="S25">
            <item>
                <PLS>L36</PLS>
                <ELS>L36+L37</ELS>
                <Performance>1</Performance>
            </item>
            <item>
                <PLS>L3</PLS>
                <ELS></ELS>
                <Performance></Performance>
            </item>
            <item>
                <PLS>L36+L37</PLS>
                <ELS></ELS>
                <Performance></Performance>
            </item>
        </performance-item>
</dsideal>
```

9. $\mathrm{XML}_{\mathrm{EvaluationReport}}$

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="7.13.xsl"?>
<!--
  An evaluation report
-->
<dsideal>
<domains>
  <domain-item domain="S31" performance="1">
        <item>
                <pms-rank>1</pms-rank>
                <pms-ls>L21+L31+L32</pms-ls>
                <pms-ssm>0.96</pms-ssm>
                <pms-shi>0.98</pms-shi>
                <pms-rm>W</pms-rm>
                <ls>L21+L31+l32</ls>
```

```
            <cost>Average</cost>
        </item>
        <item>
            <pms-rank>2</pms-rank>
            <pms-ls>L1</pms-ls>
            <pms-ssm>0.81</pms-ssm>
            <pms-shi>0.79</pms-shi>
            <pms-rm>M</pms-rm>
            <ls></ls>
            <cost>Average</cost>
        </item>
        <item>
            <pms-rank>3</pms-rank>
            <pms-ls>L25</pms-ls>
            <pms-ssm>0.68</pms-ssm>
            <pms-shi>0.72</pms-shi>
            <pms-rm>M</pms-rm>
            <ls></ls>
            <cost>Less</cost>
        </item>
        <item>
            <pms-rank>4</pms-rank>
            <pms-ls>L31+L32</pms-ls>
            <pms-ssm>0.73</pms-ssm>
            <pms-shi>0.62</pms-shi>
            <pms-rm>M</pms-rm>
            <ls></ls>
            <cost>Less</cost>
        </item>
    </domain-item>
    <domain-item domain="S32" performance="1">
        <item>
            <pms-rank>1</pms-rank>
            <pms-ls>L33+L34</pms-ls>
            <pms-ssm>0.8</pms-ssm>
            <pms-shi>0.78</pms-shi>
            <pms-rm>M</pms-rm>
            <ls>L33+L34+L35</ls>
            <cost>Average</cost>
        </item>
        <item>
            <pms-rank>2</pms-rank>
            <pms-ls>L33+L34+L35</pms-ls>
            <pms-ssm>0.73</pms-ssm>
            <pms-shi>0.75</pms-shi>
            <pms-rm>M</pms-rm>
            <ls></ls>
            <cost>Average</cost>
        </item>
        <item>
            <pms-rank>3</pms-rank>
            <pms-ls>L2</pms-ls>
            <pms-ssm>0.63</pms-ssm>
            <pms-shi>0.70</pms-shi>
            <pms-rm>M</pms-rm>
            <ls></ls>
            <cost>More</cost>
        </item>
        <item>
            <pms-rank>4</pms-rank>
            <pms-ls>L33</pms-ls>
            <pms-ssm>0.87</pms-ssm>
            <pms-shi>0.69</pms-shi>
            <pms-rm>M</pms-rm>
            <ls></ls>
            <cost>Less</cost>
        </item>
    </domain-item>
    <domain-item domain="S33" performance="0">
        <item>
            <pms-rank></pms-rank>
            <pms-ls></pms-ls>
            <pms-ssm></pms-ssm>
            <pms-shi></pms-shi>
            <pms-rm>R</pms-rm>
            <ls>L24</ls>
            <cost></cost>
        </item>
    </domain-item>
    <domain-item domain="S25" performance="1">
        <item>
            <pms-rank>1</pms-rank>
            <pms-ls>L36</pms-ls>
            <pms-ssm>0.64</pms-ssm>
            <pms-shi>0.70</pms-shi>
            <pms-rm>M</pms-rm>
            <ls>L36+L37</ls>
            <cost>Less</cost>
        </item>
        <item>
            <pms-rank>2</pms-rank>
            <pms-ls>L3</pms-ls>
            <pms-ssm>0.6</pms-ssm>
            <pms-shi>0.68</pms-shi>
            <pms-rm>M</pms-rm>
            <ls></ls>
            <cost>Average</cost>
        </item>
        <item>
            <pms-rank>3</pms-rank>
            <pms-ls>L36+L37</pms-ls>
            <pms-ssm>0.81</pms-ssm>
```

```
                <pms-shi>0.66</pms-shi>
                <pms-rm>M</pms-rm>
                <ls></ls>
                <cost>Average</cost>
            </item>
  </domain-item>
</domains>
<total>
  <cost>Average</cost>
  <performance>3/4</performance>
</total>
</dsideal>
```

10. XML$_{Cost}$

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="7.2.xsl"?>
<dsideal>
   <cost-item dt="S31">
      <item>
         <ProposeLs>L21+L31+L32</ProposeLs>
         <Cost>Average</Cost>
      </item>
      <item>
         <ProposeLs>L1</ProposeLs>
         <Cost>Average</Cost>
      </item>
      <item>
         <ProposeLs>L25</ProposeLs>
         <Cost>Less&gt;</Cost>
      </item>
      <item>
         <ProposeLs>L31+L32</ProposeLs>
         <Cost>Less</Cost>
      </item>
   </cost-item>
   <cost-item dt="S32">
      <item>
         <ProposeLs>L33+L34</ProposeLs>
         <Cost>Average</Cost>
      </item>
      <item>
         <ProposeLs>L33+L34+L35</ProposeLs>
         <Cost>Average</Cost>
      </item>
      <item>
         <ProposeLs>L2</ProposeLs>
         <Cost>More</Cost>
      </item>
      <item>
         <ProposeLs>L33</ProposeLs>
```

```
                <Cost>Less</Cost>
            </item>
        </cost-item>
        <cost-item dt="S33">
            <item>
                <ProposeLs/>
                <Cost/>
            </item>
        </cost-item>
        <cost-item dt="S25">
            <item>
                <ProposeLs>L36</ProposeLs>
                <Cost>Less</Cost>
            </item>
            <item>
                <ProposeLs>L3</ProposeLs>
                <Cost>Average</Cost>
            </item>
            <item>
                <ProposeLs>L36+L37</ProposeLs>
                <Cost>Average</Cost>
            </item>
        </cost-item>
</dsideal>
```

Appendix B List of Publications

[1] **Z. Zhang**, D. Zhou, H. Yang and S. Zhong, “A Service Composition Approach Based on Sequence Mining for Migrating E-learning Legacy System to SOA”, *International Journal of Automation and Computing,* Vol.7(4), Nov. 2010, pp. 584-595.

[2] **Z. Zhang**, H. Yang, D. Zhou and S. Zhong, “An SOA Based Approach to User-Oriented System Migration”, *in the proceeding of International symposium on advanced topics on information technologies and applications (ITA2010),* in Bradford, UK, Jun. 2010.

[3] **Z. Zhang**, D. Zhou, S. Zhong and H. Yang, “Researches on the Decision-making Algorithm in an SOA migration Model” *, in the proceeding of the 9th international FLINS Conference on Foundations and Applications of Computational Intelligence (FLINS2010),* Chengdu (EMei), China, Aug. 2010, pp. 917-922.

[4] **Z. Zhang**, W. Wang, Z. Zhou and Y. Chen, “Reasearch on the Establishment of Structural E-learning Resources”, *Edutainment2010,* Changchun, China, Aug. 2010, pp. 92-99.

[5] **Z. Zhang**, L. Zhang, S. Zhong and J. Guan, “A New Algorithm for Mining Sequential Patterns”, *5th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD’08),* Jinan, China. Oct. 2008, pp. 625-629.

[6] **Z. Zhang**, L. Zhang, S. Zhong, and J. Guan, “Improving Algorithm Apriori for Data Mining”, *8th International FLINs*

Conference on Computational Intelligence in Decision and Control, Madrid , Spain, Sep. 2008, pp.17-22.

[7] **Z. Zhang**, S. Zhong and J. Guan, "Improving Algorithm AprioriAll/Some for Mining Sequential Patterns, *Journal of Northeast Normal University* (Natural Science Edition), Vol.39(4), Dec. 2007, pp.46-53.

[8] D. Zhou, **Z. Zhang**, S. Zhong and P. Xie, "The Design of Software Architecture for E-learning Platforms", *3rd International Conference on E-Learning and Games*, Nanjing, China, Jun. 2008.

[9] S. Zhong and **Z. Zhang**, "Researches on the Integration of Information and Curriculum", *Journal of China Educational Technology*. Oct. 2007.

[10] Y. Chen, **Z. Zhang** and T. Zhang, "A Searching Strategy in Topic Crawler by Using Ant Colony Algorithm". *Microcomputer &Its Application*, Vol.30(321), Jan. 2011, pp. 53-56.

[11] Y. Chen and **Z. Zhang**, "An Application of Intelligent Single Particle Optimizer in Cluster Analysis". *Journal of Nanjing University*, Vol.47(5), Sep. 2011, pp. 578-584.

[12] Y. Chen, **Z. Zhang** and L. Xu, "Analysis and Design of Recommender Model for Learning Resources Based on Data Warehouse", *2010 Third International Conference on Education Technology and Training*, pp. 548-552.

[13] S. Zhong, J. Li, **Z. Zhang**, Y. Zhong and J. Shang, "Methods on Educational Resource Development and Application", *3rd International Conference on E-Learning and Games,* Nanjing, China, Jun. 2008.

[14] W. Wang, S. Zhong, **Z. Zhang**, S. Lv and L. Wang, "Empirical Research and

Design of M-learning System for College English", *4rd International Conference on E-Learning and Games,* Banff, Canada, Aug. 2009.

[15] P. Xie, D. Zhou, S. Zhong, **Z. Zhang** and S. Li, "An Ontology-based Service Composition Approach for Integrating E-Government Systems". *2009 International Conference on Web Information Systems and Mining* (WISM2009), pp. 570-574.

[16] D. Zhou, L. Qin, P. Xie, **Z. Zhang** and H. Tao, "SOA-Based Education Information System Interoperability Model", *Journal of Information & Computational Science,* Vol.7(5), May 2010, pp. 1165-1174.