UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

Institut für Softwaretechnik

Identifying Legacy Code for Service Implementation using Dynamic Analysis and Data Mining

Master Thesis

in fulfillment of the requirements for the degree of
Master of Sience
in Computer Science (Informatik)

submitted by

Andreas Fuhr
afuhr@uni-koblenz.de

Supervisors: Prof. Dr. Jürgen Ebert, Tassilo Horn, Dr. Volker Riediger, Institute for Software Technology, University of Koblenz-Landau, Germany

First expert: Prof. Dr. Jürgen Ebert, Institute for Software Technology, University of Koblenz-Landau, Germany

Second expert: Dr. Volker Riediger, Institute for Software Technology, University of Koblenz-Landau, Germany

Koblenz, March 2011

Abstract

Identifying reusable legacy code able to implement SOA services is still an open research issue. This master thesis presents an approach to identify legacy code for service implementation based on dynamic analysis and the application of data mining techniques.

As part of the SOAMIG project, code execution traces were mapped to business processes. Due to the high amount of traces generated by dynamic analyses, the traces must be post-processed in order to provide useful information.

For this master thesis, two data mining techniques – cluster analysis and link analysis – were applied to the traces. First tests on a Java/Swing legacy system provided good results, compared to an expert's allocation of legacy code.

Zusammenfassung

Die Identifizierung von wiederverwendbarem Source-Code für die Implementierung von SOA Services ist noch immer ein ungelöstes Problem. Diese Masterarbeit beschreibt einen Ansatz zur Identifizierung von Legacy-Code, der für eine Service-Implementierung geeignet ist.

Der Ansatz basiert auf dynamischer Analyse und dem Einsatz von Data Mining Techniken. Im Rahmen des SOAMIG Projekts wurden durch dynamische Analyse Geschäftsprozesse auf Source-Code abgebildet. Der große Umfang der daraus resultierenden Traces macht eine Nachbearbeitung der Ergebnisse notwendig.

In dieser Masterarbeit wurde die Anwendbarkeit von Data Mining Techniken zur Nachbearbeitng der dynamischen Traces untersucht. Zwei Data Mining Verfahren, Cluster-Analyse und Link-Analyse, wurden auf die dynamischen Traces einer Java/Swing Beispielsoftware angewendet.

Die Ergebnisse deuten auf eine gute Verwendbarkeit der beiden Data Mining Techniken zur Identifizierung von Legacy-Code für die Service-Implementierung hin.

Contents

1 Introduction **1**
1.1 Structure of the Thesis . . . 2

2 Research Context **3**
2.1 Problem Definition . . . 3
2.2 Current State of Research . . . 3
2.3 Contribution of the Thesis . . . 4

I Service-Oriented Architectures **5**

3 Principles of Service-Oriented Architectures **7**
3.1 A Generic Architecture for Service-Oriented Architectures . . . 8
3.2 Consumers Layer . . . 8
3.3 Business Processes Layer . . . 10
3.4 Services Layer . . . 11
3.4.1 Composite Services . . . 12
3.5 Components Layer . . . 12
3.6 Operational Systems Layer . . . 13

4 SOA Migration **15**
4.1 The SOAMIG Process . . . 15
4.1.1 Phases . . . 16
4.1.2 Core Disciplines . . . 17
4.1.3 Supporting the Migration Processes . . . 20

II Data Mining **21**

5 Data Mining: Introduction **23**
5.1 CRISP: Data Mining Reference Process . . . 23
5.1.1 Business Understanding . . . 25
5.1.2 Data Understanding . . . 25
5.1.3 Data Preparation . . . 25
5.1.4 Modeling . . . 25
5.1.5 Evaluation . . . 25
5.1.6 Deployment . . . 26
5.2 Categories of Data Mining Approaches . . . 26
5.2.1 Segmentation Approaches . . . 26
5.2.2 Classification Approaches . . . 27

5.2.3 Link Analysis Approaches 27

6 Cluster Analysis **29**
6.1 Data Types and Levels of Measurement 29
6.2 Similarity and Distance Measures for Homogeneous-Scaled Attributes 31
6.2.1 Measures for Nominal-Scaled Attributes 33
6.2.2 Measures for Binary-Scaled Attributes 33
6.2.3 Measures for Ordinal-Scaled Attributes 34
6.2.4 Measures for Metric-Scaled Attributes 35
6.3 Measures for Mixed-Scaled Attributes 36
6.3.1 General Similarity Coefficient 37
6.4 Measures for Cluster Distances 38
6.4.1 Log-Likelihood Distance 39
6.5 Clustering Algorithms 40
6.5.1 Hierarchical Clustering Approaches 40
6.5.2 Partitioning Clustering Approaches 43
6.5.3 Fuzzy Clustering Approaches 44
6.6 Interpreting Clustering Solutions 46
6.6.1 Rating on Data With Known Allocation 46
6.6.2 Rating on Data With Unknown Allocation 47

7 Link Analysis With Association Rules **51**
7.1 Quality Measures for Association Rules 52
7.1.1 Support, Confidence and Lift 52
7.2 The Apriori Approach 53
7.2.1 The Apriori Algorithm 54

8 Data Mining Tool: PASW **57**

III Service Realization by Legacy Code **61**

9 Running Example **63**
9.1 Scenario 63
9.2 Business Modeling 64

10 Dynamic Analysis **67**
10.1 Using Business Processes for Dynamic Analysis 67
10.1.1 Workflow Models: An Implementation-Centric View on Business Processes 68
10.2 Workflow Model for Extraction Business Process 68
10.3 Set-Up for the Dynamic Analysis 71

11 Using Cluster Analysis to Identify Legacy Code for Service Implementation **75**
11.1 First Approach: Clustering Classes According to Usage in Activities 75
11.1.1 Data Preparation 76
11.1.2 Modeling 77
11.1.3 Evaluation 79
11.1.4 Summary of the First Approach 87

11.2 Second Approach: Clustering Classes According to Usage in Activities Extended by Knowledge about Packages . . . 88
11.2.1 Data Preparation . . . 89
11.2.2 Modeling . . . 89
11.2.3 Evaluation . . . 90
11.2.4 Summary of the Second Approach . . . 95
11.3 Discussing the Suitability of Clustering Approaches for Identifying Legacy Code able to Support Services . . . 95

12 Using Association Rules to Identify Legacy Code for Service Implementation 97
12.1 First Approach: Identify Legacy Code Supporting Activities . . . 97
12.1.1 Data Preparation . . . 97
12.1.2 Modeling . . . 98
12.1.3 Evaluation . . . 98
12.2 Second Approach: Identify Joint Legacy Code . . . 99
12.2.1 Data Preparation . . . 99
12.2.2 Modeling . . . 100
12.2.3 Evaluation . . . 100
12.2.4 Discussing the Suitability of Association Rules for Identifying Legacy Code able to Implement Services . . . 102

13 Conclusion and Outlook 103

Appendix 105

A Common Usage of placeholders 105
A.1 Distance Functions . . . 105

B Examples 107
B.1 Example for Fuzzy k-means Algorithm . . . 107

Bibliography 111

List of Tables

6.1 Levels of Measurement [Möhring, 2009] 31
6.2 Contingency matrix between two items p_i and p_j 33
6.3 Measures for binary-scaled attributes [Gan et al., 2007] 34
6.4 Proposed interpretation of silhouette coefficient [Kaufman and Rousseeuw, 2009] 48

11.1 Evaluation coefficients for the five clustering solutions that have been computed by PASW 79
11.2 Quality measures for both clustering approaches 90

B.1 Example data set containing six items with two ratio-scaled attributes each. . . . 107
B.2 Initial membership matrix U_1 108
B.3 Updated membership matrix U_2 109
B.4 Checking the termination criterion $U_1 - U_2$ 109
B.5 Updated membership matrix U_3 110
B.6 Checking the termination criterion $U_2 - U_3$ 110

List of Figures

3.1 SOA layers according to Arsanjani et al. [2008] 9
3.2 Assembly of a business process 10
3.3 Separation of concerns in services 11
3.4 Structure of composite service 12
3.5 Facade pattern used to integrate various service components 13

4.1 The four phases of a SOA migration project and their disciplines 16
4.2 The seven SOAMIG process core disciplines 18

5.1 CRISP-DM phases according to Chapman et al. [1999] 24

6.1 Hierarchy of data types 30
6.2 Hierarchy of levels of measurement 30
6.3 Various distance measures for cluster distances 38

8.1 PASW workbench 58

9.1 Exemplary business process: Extraction of legacy code for service implementation (BPMN Notation) 65

10.1 Relation of business model to workflow model 68
10.2 Exemplary business process: Extraction of legacy code for service implementation 69
10.3 Details of `load graph` functionality 69
10.4 Details for activity `Select element` 70
10.5 Details for activity `Refactor element` 70
10.6 Set-up of the dynamic analysis 71
10.7 The Business Process Tracer tool 72

11.1 Visual clustering according to usage. The blue box is a business service, the red box is a composite service and the green box is a helper service 76
11.2 Modeling of the clustering in PASW 78
11.3 First approach: Relative occurrence view on clustering solution computed by TwoStep algorithm 81
11.4 First approach: Detailed relative occurrence view on attribute PER_Refactor element in Cluster-3 82
11.5 First approach: Detailed absolute occurrence view on attribute PER_Refactor element in Cluster-3 82
11.6 First approach: Comparing Cluster-2 and Cluster-3 of the TwoStep solution . . . 84
11.7 First approach: Comparing the TwoStep 3-clusters solution to the true allocation . 85
11.8 Relative occurrence view on k-Means 6-clusters solution 85

11.9 First approach: Comparing the k-Means 6-clusters solution to the true allocation defined by the expert 86
11.10 First approach: Relative occurrence view on k-Means 7-clusters solution 87
11.11 First approach: Comparing the k-Means 7-clusters solution to the true allocation defined by the expert 88
11.12 Second approach: Modeling 89
11.13 Clustering solution for the TwoStep algorithm including packages 92
11.14 Absolute distribution of classes in Cluster-3 93
11.15 Comparing the TwoStep solution of the second approach to the true allocation . . 94
11.16 Comparing the k-Means 6 clusters solution of the first approach (without packages) and the second approach (with packages) 94
11.17 Comparing the k-Means 7 clusters solution of the first approach (without packages) and the second approach (with packages) 94

12.1 Result of first association rules approach: Mapping classes to activities 98
12.2 Result of second association rules approach: Identifying associated functionality 100
12.3 Transitive processing of association rules 101

Chapter 1

Introduction

For the last decade, Service-Oriented Architectures (SOA) have been one of the most promising approaches to handle business needs on software: faster time-to-market, lower costs, high reusability and business integration are buzzwords that drove most companies to implementing SOAs [Martin, 2010]. Although SOA has been called dead some time ago [Burton Group, 2009], SOAs are still interesting to industry. Many companies are implementing SOAs today or are planning to do so in future [Martin, 2010]. While the idea of SOA is older than a decade now, no consensus has been found about how to implement SOAs right, so far. Various problems are still unsolved, leaving open a broad field for researchers.

For this master thesis it has been explored how to solve one key problem of SOA development: how to implement services by reusing legacy code. A *service* is viewed as an encapsulated, reusable and business-aligned capability coming with a well-defined *service specification* that provides an interface description of the functionality. The service specification is implemented by a *service component* which is supplied by a *service provider*. Its functionality is used by *service consumers* [Arsanjani et al., 2008].

Having the goal of reengineering an existing system towards a SOA, services may be implemented by reusing existing code. However, this is not a trivial task and still an open research issue [Kontogiannis et al., 2007].

As part of the SOAMIG project (a cooperation project between university and industry aiming at migrating a legacy system towards SOA), dynamic analysis has been used to trace legacy code that is executed during business processes. This information was thought to be useful for identifying legacy code able to implement services. However, the dynamic analysis produced a large amount of traces which must be post-processed in order to extract useful information.

For this master thesis, it has been evaluated how data mining techniques can be used to analyze mappings of business processes to code which have been extracted by static and dynamic analysis. Two data mining techniques have been chosen from the broad set of data mining techniques available: cluster analysis and link analysis with association rules. Both techniques were applied to a Java/Swing tool developed during the SOAMIG project to support the extraction of code from legacy systems.

Both techniques produced useful results. Two variations of cluster analysis approaches were applied to the identification of legacy code. Both were able to compute useful results. Comparisons to a manual clustering solution created by an expert showed, that both approaches lead to clusters of legacy code that would be able to support the implementation of services. Link analysis with association rules created useful results, too. However, the association rules were not as expressive as the clustering results.

In the following, the structure of the thesis is presented.

1.1 Structure of the Thesis

The remaining thesis is structured as follows.

Chapter 2 introduces the academic context of this thesis.

Part I gives an overview about the context of this thesis – migrating legacy software towards Service-Oriented Architectures. Chapter 3 introduces the basic concepts of Service-Oriented Architectures. In Chapter 4, a migration process towards Service-Oriented Architectures is introduced.

Part II provides theoretical foundations about data mining techniques. Chapter 5 introduces data mining and a process for data mining projects. In addition, three categories of techniques are briefly described. It is motivated what techniques have been chosen for the identification of legacy code. Chapter 6 introduces cluster analysis. Similarity and dissimilarity measures are defined, three kinds of clustering algorithms are described and evaluation criteria for clustering solutions are introduced. Chapter 7 introduces a link analysis approach based on the identification of association rules. Chapter 8 shortly introduces PASW, the data mining tool used in this thesis.

In Part III, the application of data mining techniques on the identification of legacy code able to implement services is described. Chapter 9 introduces the Java/Swing legacy tool that has been used as guinea pig in this thesis. Chapter 10 describes how dynamic analysis has been used to trace the execution of legacy code during a business process. Chapter 11 describes how to apply cluster analysis on the traces that have been generated during dynamic analysis to identify legacy code able to implement services. Chapter 12 describes how to use link analysis to accomplish the same task.

Chapter 13 summarizes the results of the thesis and gives a short outlook on future research needed to implement the results of the thesis in real SOA migration projects.

Chapter 2

Research Context

In this chapter, the problem of identifying legacy code able to implement services is introduced. The current state of research is presented and the contributions of this thesis are described.

2.1 Problem Definition

Service-oriented architectures try to integrate business knowledge into software development such that software reflects the structure of the business processes. This is often called *Business-IT-Alignment*. For this reason, the core elements of a SOA – the *services* – are tightly related to the business processes they support.

As one main objective of migration projects is the reuse of existing code, legacy code must be identified that is able to implement the services. However, legacy systems are often not structured according to business concerns. Therefore, legacy code to support one business process may be scattered across wide parts of the legacy system. This makes it difficult to identify legacy code that is able to implement services.

In the following, the current state of research on identifying legacy code for service implementation is described.

2.2 Current State of Research

Identification of legacy code able to implement services is still an open research issue and not yet explored very well [Kontogiannis et al., 2007]. However, some work has been done in static analysis.

IBM seems[1] to analyze names of operations and comment lines in order to find services [Ronen et al., 2007]. They use information retrieval techniques to extract information from names and comments. Services are defined first and code is analyzed to search for possible implementations. This technique reflects common state of the art in trying to integrate semantics in analysis. However the approach fails as soon as name conventions are not met or code is not documented well. In fact,

[1]The only source of information is a Powerpoint presentation of the IBM Haifa, Israel Research Lab. More information is not available as IBM strictly hides its techniques due to security strategies.

most real legacy code is messy and does not provide much information in names and comments. So IBM's approach relies on conventions too much.

Marchetto and Ricca [2008] manually walk through legacy code first and tag classes and methods by hand. In a second step, an automated routine identifies services from the tagged legacy code. This approach requires a good understanding of the legacy system to be able to tag the parts correctly. In addition, the manual tagging is tedious for real, big systems.

2.3 Contribution of the Thesis

In contrast to static analysis of source code, this thesis is based on a dynamic approach. As part of the SOAMIG project, legacy source code used during the execution of business processes has been traced. However, this analysis resulted in a large amount of data which require further analysis to extract information useful for identifying legacy code able to implement services.

In this thesis, the suitability of data mining techniques for analyzing these traces has been evaluated. Two data mining techniques – cluster analysis and link analysis with association rules – have been applied successfully on the traces of a guinea pig legacy system. Results indicate that these two techniques are suited adequate to identify legacy code service implementation.

Part I

Service-Oriented Architectures

Chapter 3

Principles of Service-Oriented Architectures

In this chapter, basics of Service-Oriented Architectures (SOAs) are introduced. As SOAs form the context of the thesis, the main concepts are defined and their importance for identifying legacy code able to implement services is motivated.

Service-Oriented Architecture (SOA) is a software engineering paradigm first mentioned 1996 by the Gartner Group [Gartner Group, 1996a,b]. Until today, there is no consent about a clear definition of SOA. According to OASIS, SOA is defined as

> " *a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.* " [MacKenzie, 2006, p. 29]

With SOA, software is structured into loosely-coupled and distributed capabilities, called *services*. Services provide business functionalities via well-defined interfaces, are bound to consumer applications dynamically and exchange information via messages. Although encapsulating software into well-separated capabilities was no new invention of SOA[1], SOA firstly introduced a strong focus on business processes. Based on this business-IT-alignment combined with the dynamic binding of services, SOA promises

a) improved flexibility in customer applications,
b) higher innovation due to this flexibility,
c) development of new software by assembling existing services,
d) optimization of processes and
e) faster time-to-market.

In recent years, SOA first became a hype (SOA was sold as a "product" and everybody wanted SOA although he didn't know what it was). Today, the SOA hype has settled. The idea of SOA is much clearer today. Many processes have been invented how to develop SOAs[2]. However, SOA is still attractive to customers: According to a study in 2010 observing the history of SOA [Martin,

[1]Component-Based Development (CBD) and even Object-Oriented Programming (OOP) already followed that idea of separating functionality of a software into more or less coarse-grained "blocks", called components in CBD and classes in OOP.

[2]For an overview about SOA development processes see [Thomas et al., 2010]

2010], 63% of the companies already use SOA in their company. 31% plan to use a SOA in future and only 6% do not want to use SOA. In addition, only 9% of the companies that use SOA or plan to do so, already finished the implementation of the SOA. These numbers indicate that, still today, there is a broad need for SOA approaches.

In the remaining chapter, a generic architecture for SOAs is introduced. Following the layering of this generic architecture, the main aspects of SOAs are described. This view on SOAs is following IBM's interpretation of Service-Oriented Architectures [Arsanjani et al., 2008; IBM Corporation, 2007].

3.1 A Generic Architecture for Service-Oriented Architectures

As already mentioned, SOA is not one fixed definition of an architecture. However, SOAs share the strong focus on business processes and the separation of interfaces (describing what functionality is provided) and implementations (implementing the functionality). Therefore, the following generic architecture can be found in most SOAs (at least in a similar way).

The generic SOA architecture is structured into five layers as shown in Figure 3.1 on the facing page.

The *Consumers* layer represents applications that are used to execute business processes, e.g., user interfaces. The *Business processes* layer models the business process workflows and maps processes to services. On this layer, the orchestration of services – i.e., the assembly of separate services into a complete system – is implemented. The *Services* layer contains the services. Services can be atomic (all functionality of the service is implemented by the corresponding service component) or composed of other services (all or some functionality of the service is provided by other services). The *Service components* layer contains the service components responsible for implementing the services. Each component implements a service or parts of the service (therefore, one service can be implemented by many components). Each component can be implemented in a different way. As examples, components can be implemented by reusing legacy code (e.g., by wrapping or transforming code), by using packaged applications or by implementing functionality from scratch. The *Operational systems* layer contains the entities used to implement the service components. Legacy applications, packaged applications or new code are located here.

In the following sections, the layers are described in more detail. For each layer, the basic concepts are introduced. In addition, design patterns that will be used in the remaining thesis are motivated.

3.2 Consumers Layer

The Consumers layer represents the interface between SOA and *service consumers* – i.e., applications that are using services. Service consumers are for example other software systems using services or graphical user interfaces (GUIs) accessing the service directly.

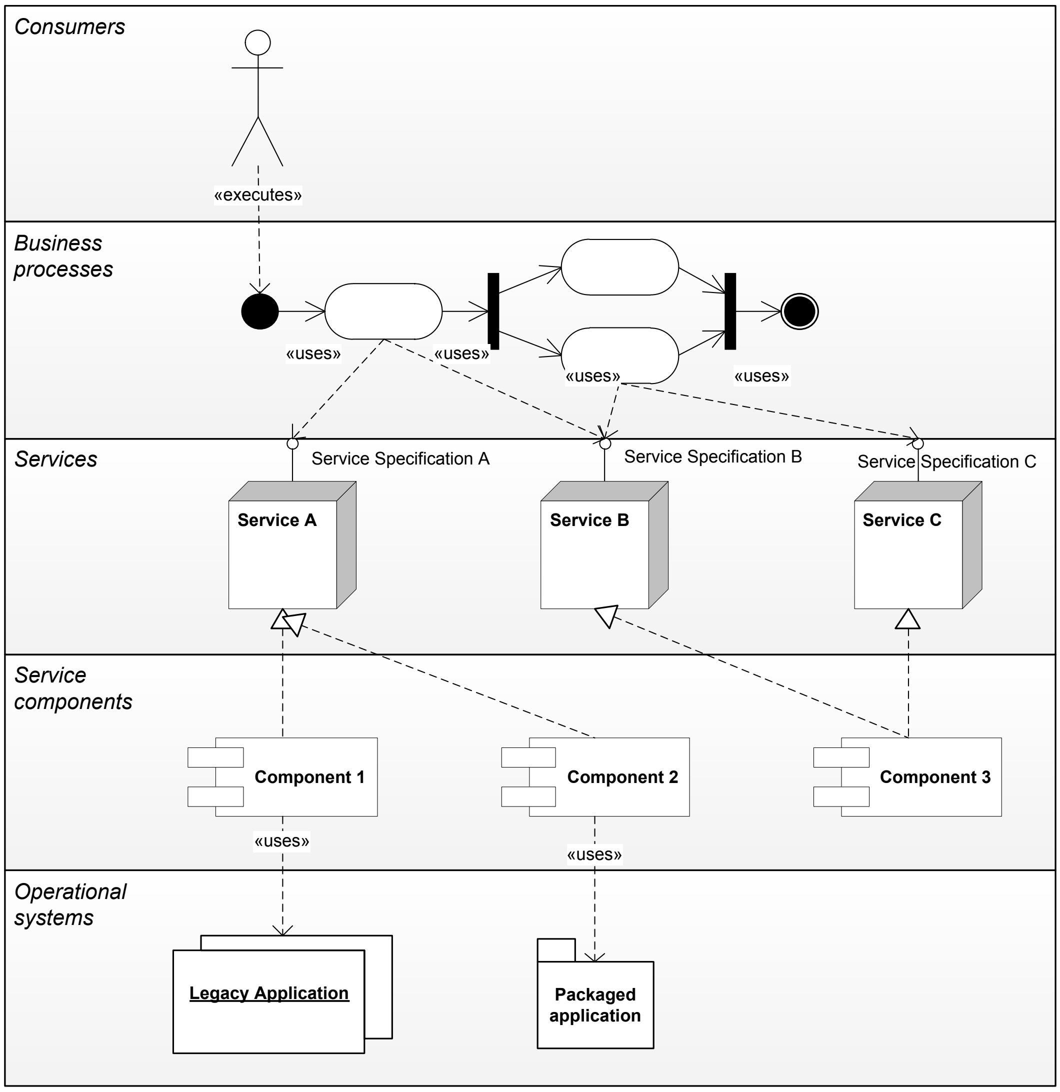


Figure 3.1: SOA layers according to Arsanjani et al. [2008]

3.3 Business Processes Layer

The Business processes layer models the workflow of the business processes.

Definition 3.3.1 (*Business process*)

A business process is

> *" a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer. " [Hammer and Champy, 2006, p. 94]*

Therefore, a business process defines what inputs are required and what work is to be done in order to create some output that is useful to a customer (Figure 3.2). *Activities* are the detailed descriptions of steps during a business process. Activities can be composed of other activities. As business processes do not only describe activities that are performed on software systems, it is useful to mark which activities are to be supported by software services (`Service Activity`).

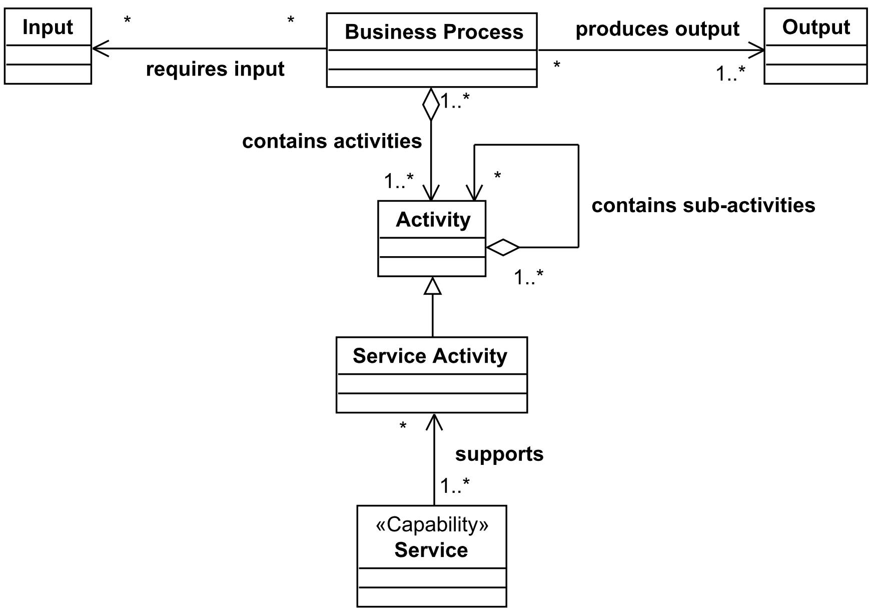


Figure 3.2: Assembly of a business process

To form a complete software system, services are assembled. In SOA this is called *service orchestration*. On the business processes layer it has already been modeled how business processes are assembled. To orchestrate services to support a business process, services are mapped to activities of the processes. This mapping explicitly defines what functionality of a service is needed and what information is exchanged. In an ideal situation, the software system can be assembled by using the business process model and by using the mapping of services to processes. However, business models often differ from how tasks are executed in reality. Therefore, they might not be used to derive a service orchestration.

3.4 Services Layer

In SOA, functionality is encapsulated in *services*.

Definition 3.4.1 (*Service*)

Arsanjani et al. [2008] define a service as

> *" a coarse-grained, encapsulated and reusable business-aligned capability that is exposed via well defined interfaces to its environment. "*

Services are coarse-grained, reusable entities that support one or more business processes. Services strictly follow the "separation of concerns" paradigm. Figure 3.3 shows the separation of concerns in services.

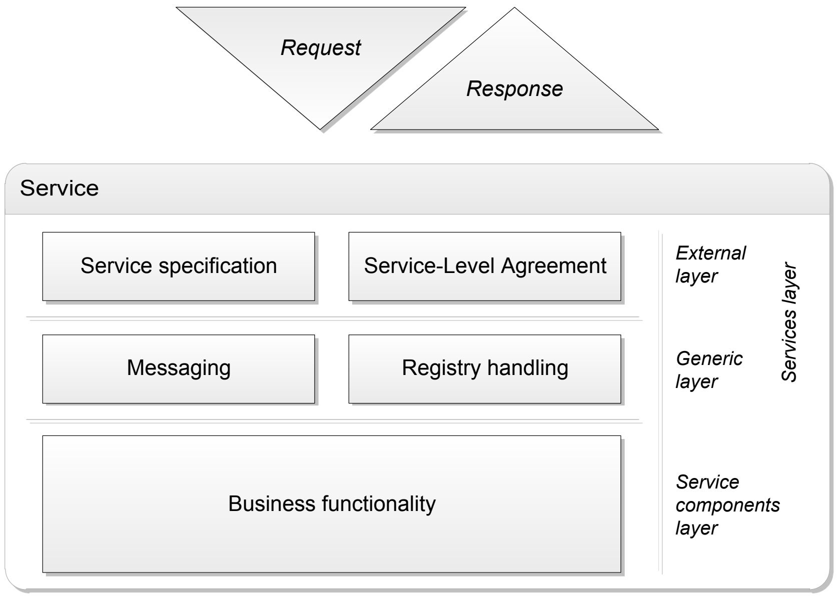


Figure 3.3: Separation of concerns in services

Only the *External layer* of the services structure is exposed to the public. On this layer, the *service specification* and the *Service-Level Agreement (SLA)* are accessible to service consumers. The service specification is an interface description defining what functionality the service provides and how to access it. The SLA is a contract about what quality a service ensures. E.g., SLAs include definitions of maximum response times.

The *Generic layer* provides functionality that all services have in common. On this layer, functionality to handle messaging operations (e.g., sending and retrieving) is located. In addition, functionality to deal with service registries is implemented. *Service registries* (also called service broker) are central registries that manage what services are available. Service registries inform service consumers about what service to use for a given use case.

The *Service components layer* is separated from this general service functionality. On this layer, the business functionality of a service is implemented by *service components* which are described in Section 3.5.

Services can be atomic or composed of other services (they are then called *composite service*). While atomic services implement all functionality on their own, composite services use other services to implement all or parts of the provided functionality.

3.4.1 Composite Services

Composite services use other services to implement all or parts of their functionality. A composite service exposes one composite service specification and requires multiple services to implement this composite specification (Figure 3.4).

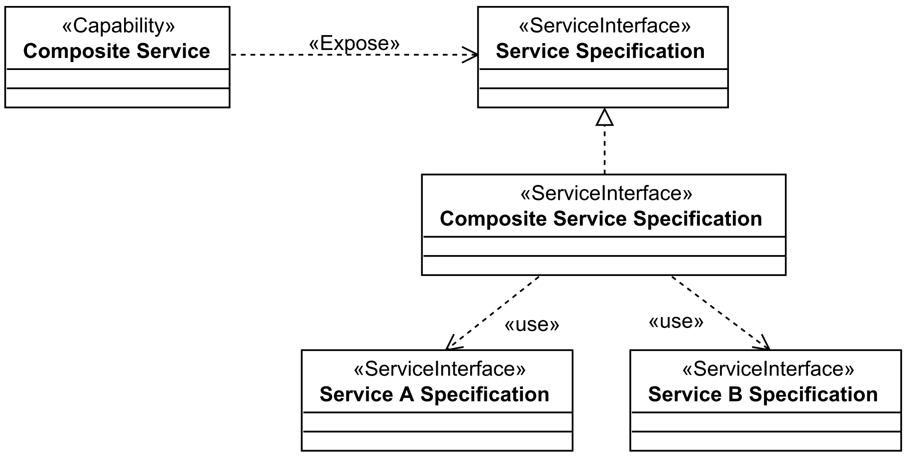


Figure 3.4: Structure of composite service

To be able to specify a composite service sufficiently, a composite service specification models which other services are required to implement the service. Therefore, it realizes one service specification that is exposed and requires multiple service specifications that are used to implement the provided functionality [Wahli, 2007].

3.5 Components Layer

The functionality of services is implemented by *service components*. Service components provide business functionality. They realize the service specification of the service. Components can be developed by using traditional object-oriented programming approaches or can be implemented by reusing parts of legacy systems (see Chapter 4). One service component can implement several services or only parts of one service.

As services can use multiple components to implement their functionality, the *facade pattern* as shown in Figure 3.5 on the facing page is often used to integrate all components [Wahli, 2007]. The

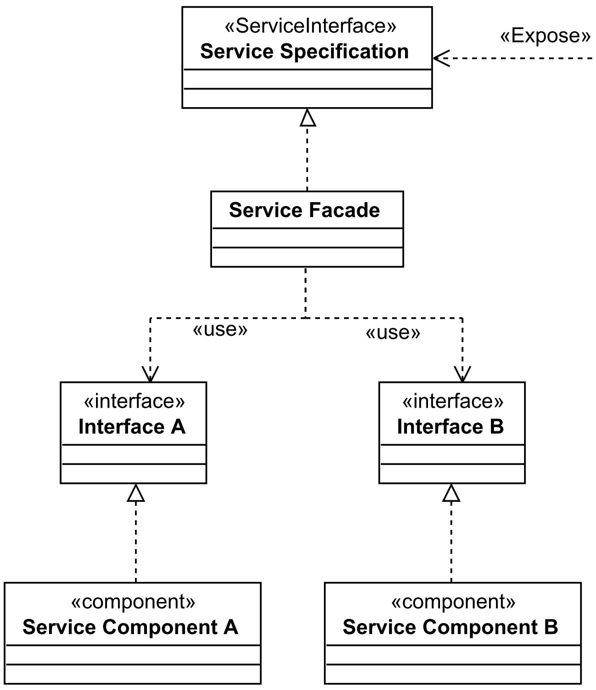


Figure 3.5: Facade pattern used to integrate various service components

task of the facade class is to delegate requests for functionality to the right component. In addition, sub-results are assembled in the facade class before they are returned to the service consumer.

3.6 Operational Systems Layer

The *operational systems layer* finally describes how a service component is implemented. For each component the implementation strategy may be selected separately. Amongst others, the following choices are available:

- Implement from scratch
- Buy services from third-party providers
- Wrap third-party packaged application (Commercial of the shelf, COTS)
- Wrap legacy code
- Transform legacy code

Implementation from scratch is used if the functionality does not yet exist or other approaches to reuse existing technology have failed. Third-party providers may provide ready-to-use services that can be used. In addition, third-party non-SOA applications providing the required functionality may be wrapped in order to access this existing functionality.

The two remaining choices can be used if legacy applications providing the required functionality exist. Legacy code could then be wrapped to be accessed by the service component. Alternatively, legacy code may be transformed into a newer technology to become directly accessible to the service component.

Chapter 4 will describe approaches how to reuse legacy code for SOA migrations in more detail.

Chapter 4

SOA Migration

In this chapter, a process for migrating legacy system towards SOAs is introduced. The process shows what activities are needed in SOA migration projects and where the identification of legacy code able to implement services can support such migration projects.

Today, almost every company runs systems that have been implemented a long time ago. These systems, and even those that have been developed in the last years, are still under adaptation and maintenance to address current needs. Adapting legacy software systems to new requirements often needs to make use of new technological advances. Business value of existing systems can only be preserved by transferring legacy systems into new technological surroundings. Migrating legacy systems, i.e., transferring software systems to a new environment without changing the functionality [Sneed et al., 2010], enables already proven applications to stay on stream instead of passing away after some suspensive servicing [Rajlich and Bennett, 2000].

Migrating legacy systems to services enables both, the reuse of already established and proven software components and the integration with new services, including their orchestration to support changing business needs. In order to gain most benefit from a migration, a comprehensive approach supporting the migration process and enabling the reuse of legacy code is required.

In the following, a software migration process towards Service-Oriented Architectures is presented.

4.1 The SOAMIG Process

This section introduces the SOAMIG process that has been established during the SOAMIG project. The SOAMIG process describes activities to migrate legacy systems into a Service-Oriented Architecture [Erdmenger et al., 2011b].

During the life-cycle of a SOA migration project, four different *phases* are distinguished (Figure 4.1 on the next page): Preparation, Conceptualization, Migration and Transition. During each phase, various *disciplines* describe what has to be done during this phase. Each discipline consists of *activities* ordered in a *workflow* to describe the tasks in detail.

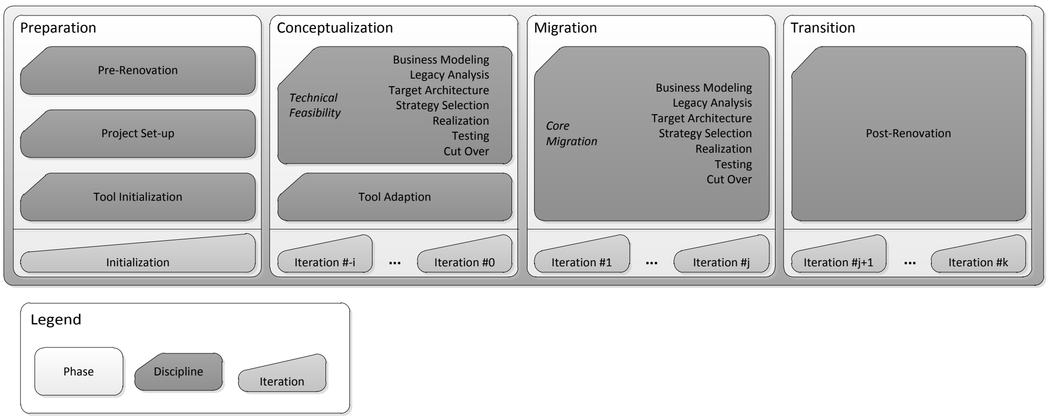


Figure 4.1: The four phases of a SOA migration project and their disciplines

4.1.1 Phases

In the following, the four phases and their disciplines are described.

Preparation Phase

The *Preparation* phase deals with setting up the project and with enabling a legacy system for migration.

The discipline *Pre-Renovation* deals with enabling a legacy system for migration. Legacy system technology or design may prohibit migration efforts. Therefore, the legacy code might have to be reengineered to allow a migration.

The discipline *Project Set-up* deals with organizational aspects of the project management. This discipline includes activities like defining project goals, work packages, schedules or managing resources.

During *Tool Initialization*, tools are adapted to the project needs and installed or developed especially for that project.

Conceptualization Phase

The *Conceptualization* phase deals with checking the feasibility of the migration by migrating an exemplary part of the system (technical cut-through).

During *Technical Feasibility*, the core disciplines of the SOAMIG process are performed to migrate a small but representative part of the system to check for feasibility of the migration effort. These seven core disciplines are the same disciplines as during the Migration phase. They are introduced in Section 4.1.2 separately.

In addition, results and lessons-learned of the first cut-through are used to adapt the tools used for the migration (*Tool Adaption* discipline).

Migration Phase

The *Migration* phase is the core phase of the migration project where the complete system is migrated (maybe in multiple iterations). The seven core disciplines are described in Section 4.1.2 separately.

Transition Phase

Finally, during the *Transition* phase, a renovation of the migrated system is performed to clean-up the migrated code. A *Post-Renovation* might be necessary as some migration techniques may reduce the maintainability of the system. Therefore, it might be necessary to reengineer the migrated system to enhance quality again.

4.1.2 Core Disciplines

During Conceptualization and Migration, the seven core SOAMIG disciplines are performed to migrate (parts of) the legacy system.

Figure 4.2 on the following page shows the seven core disciplines of the SOAMIG process. All disciplines exchange artifacts with an integrated repository that stores all artifacts as one model. The disciplines can be brought into a loose, logical order by reading the figure clockwise starting from the top (Business Modeling). However, in a real project, the order may change and activities may overlap to reflect dynamic project needs.

In the following sections, the seven core disciplines are shortly introduced.

Business Modeling

During Business Modeling, the business processes of a customer are gathered. As SOA is closely related to business processes, it is important to define a complete model of the customer's processes. The business model is created using a standardized modeling language (e.g. UML 2.0 activity diagrams or BPMN) and stored in the SOAMIG repository for later analysis.

In addition to business processes, a business vision, business goals and goals for the migration project are defined in this phase.

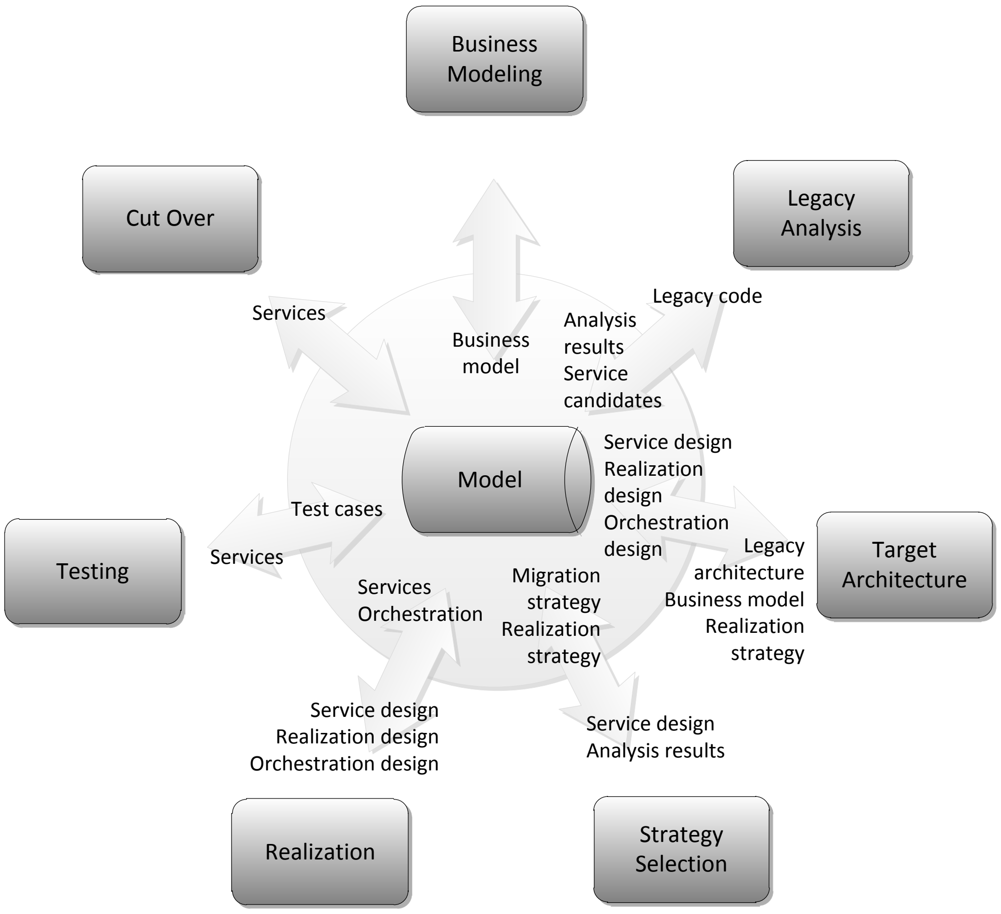


Figure 4.2: The seven SOAMIG process core disciplines. The figure can be read clockwise by starting from the discipline on the top (Business Modeling). Text over the arrows stands for artifacts that are read from or written to the integrated model.

Legacy Analysis

The goal of the Legacy Analysis phase is to explore the legacy system and to understand what functionality it provides. Amongst legacy source code, various artifacts like legacy architecture, test cases or user documentation are analyzed to get a complete overview about the system. Legacy source code is transformed into a model and stored in the SOAMIG repository. Static and dynamic analysis techniques are used to get a more abstract view on the system.

One main goal of this phase is to identify initial service candidates that support all business processes of the customer. Various artifacts like legacy code or business processes are analyzed to identify a complete set of service candidates.

Target Design

During Target Design, the target architecture of the to-be migrated system is defined. Service interfaces are specified and services are composed.

The target architecture consists of three designs, the orchestration design, the service design and the realization design. The orchestration design depicts how to orchestrate the services to support the business processes. The service design describes the service interfaces of the required services. The realization design describes how to implement the services.

Strategy Selection

Strategy Selection deals with how to implement the services. Based on the project requirements, each service can be implemented by

- reimplementing functionality,
- wrapping or
- transforming legacy code.

During this phase it is defined how each service should be implemented. This may include identifying legacy code that is able to be used for wrapping or migration.

Implementation

During this phase, the service are implemented. Wrappers are written, legacy code is transformed and services are reimplemented. In addition, services are composed in this phase.

Testing

Testing is performed throughout the SOAMIG process. However, after implementing the services, the focus lies on ensuring that everything works as expected. In addition, testing ensures that the new systems behaves like the legacy system. Therefore, regression tests are performed in this phase.

Cut Over

The Cut Over phase deals with rolling out the new system for the customer. Services are published and service consumer software and the Enterprise Service Bus are installed in the customer's system environment. In addition, the system is monitored to ensure that it performs in this environment as expected.

This phase concludes the SOAMIG process. The following section describes where a SOA migration process like the SOAMIG process can be supported by the results of this thesis.

4.1.3 Supporting the Migration Processes

In the SOAMIG process, the identification of legacy code able to implement services is located in the three phases

1. Target design
2. Strategy selection
3. Implementation

Being able to identify which code can be reused to implement services supports the decision on how to implement the service, as well as the design of the service realization. Of course, knowing what code to use for implementation, supports the service implementation, too. Therefore, these three activities of the process can profit from the results of this thesis.

This concludes the introduction of Service-Oriented Architectures. In the following, theoretical principles of data mining are explained.

Part II

Data Mining

Chapter 5

Data Mining: Introduction

In this chapter, data mining will be shortly introduced. A process for data mining will be provided and three categories of data mining techniques will be described.

In recent years, growing disk capacities increased the ability to store all kinds of information. Today, most actions concerning computers are logged: Bank transactions, online shopping protocols, internet communication or web site usage statistics lead to an information flood stored on disk.

The goal for *data mining* is to analyze such data, to extract useful information and to support future decisions by those information. Data mining applies mathematical-statistical-based techniques to these data to extract patterns or dependencies and to forecast future trends.

Definition 5.0.1 (*Data Mining*)

Berry and Linoff [2004] define data mining as

> *" the exploration and analysis, by automatic or semiautomatic means, of large quantities of data in order to discover meaningful patterns and rules "*

As data mining is a complex process due to huge amounts of data, a process is needed to structure data mining activities. Section 5.1 introduces such a process. Section 5.2 introduces three groups of data mining techniques.

5.1 CRISP: Data Mining Reference Process

As data mining activities van get very complex, a process to structure data mining activities has been invented in late 1996 [Chapman et al., 1999]. The *Cross Industry Standard Process for Data Mining (CRISP-DM)* defines six phases for data mining projects as shown in Figure 5.1 on the next page.

The six phases are described in the following.

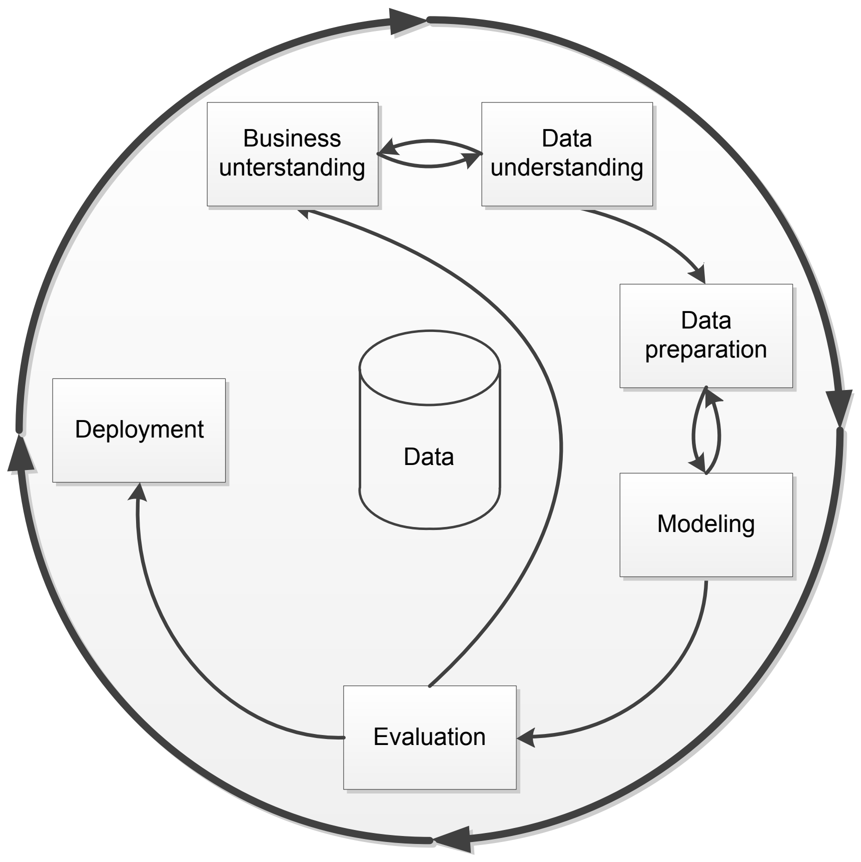


Figure 5.1: CRISP-DM phases according to Chapman et al. [1999]

5.1.1 Business Understanding

Business understanding focuses on learning what data is gathered in the business domain of the customer and what this data means. As data mining strongly relies on finding meaningful information in data, the data analyst must know how to interpret solutions and how the results can be useful to the customer.

5.1.2 Data Understanding

Data understanding is closely related to business understanding. Only the focus of this phase lies on the data that should be explored. A clear understanding about what the attributes mean, what values are stored in them is necessary. In addition, first descriptive statistics about the data (e.g. means, distributions, attribute types or missing data) are gathered.

5.1.3 Data Preparation

Data preparation is the task that takes most of the effort in data mining projects. During data preparation, the data is prepared for analysis. It is selected, which data is relevant for the data mining goal. New data is created (e.g., derived from existing data), different data sources are integrated and the data is cleaned.

A core task of this phase is to deal with *missing values*. Missing values are entries in the data set that are missing or that are encoded by a special value indicating that they are missing. Missing values are often expressed by `NULL`, `-` or `$NULL$`. However, sometimes missing values are encoded by more special values, e.g., `666` might indicate a missing value.

As missing values distort many data mining techniques, they must be replaced by more meaningful values (e.g., they are replaced by the mean of the attribute).

5.1.4 Modeling

During modeling, the data mining techniques are selected, implemented and applied to the data. The selection and application of data mining techniques introduced in this thesis is located in this phase.

5.1.5 Evaluation

During evaluation, the data mining techniques are evaluated. It is tested how good they solve the data mining problems and how good they can answer questions of the customer. If the evaluation indicates bad results, the techniques are adapted.

5.1.6 Deployment

Finally, the data mining techniques that passed the evaluation are deployed to the customer. This phase includes the application of models that have been trained on test-data to real data of the customer.

This concludes the CRISP-DM process. The following sections introduce data mining techniques that will be applied to the identification of legacy code to implement services in Part III.

5.2 Categories of Data Mining Approaches

Data mining techniques can be grouped into three categories [Möhring, 2009]:

- *Segmentation approaches* segment heterogeneous data into homogeneous subgroups
- *Classification approaches* classify new data according to a model learned from existing data
- *Link analysis approaches* extract rules and dependencies between data

In the following, for each category techniques will be introduced. Each technique is shortly evaluated if it is suited for identifying legacy code able to implement services.

5.2.1 Segmentation Approaches

Segmentation approaches deal with dividing heterogeneous data into more homogeneous subgroups. In contrast to classification techniques introduced in Section 5.2.2, these groups are not known before segmentation. They are identified during execution of the algorithms.

Clustering Approaches

Clustering approaches compute the dissimilarity or similarity between items (tuples of data). Based on these values, a heterogeneous data set is divided into more homogeneous subgroups, called clusters.

Suitability : Clustering approaches can be used to separate legacy code into homogeneous subgroups. The subgroups may represent code belonging together and therefore being able to implement services.

Chapter 6 will introduce clustering techniques in detail. Chapter 11 will describe how clustering techniques can be used to identify legacy code able to implement services.

Self-Organizing Maps

Self-Organizing Maps (SOM) are a type of artificial neural network that is trained by unsupervised learning, i.e., without knowing correct outcome of the learning process. SOMs create a map representing the data set. They order similar items near to each other on this map. SOMs can be used to visually segment data.

Suitability: Although, SOMs can be used to identify groups in data, SOM solutions are often hard to interpret. They are therefore not applied to the identification if legacy code able to implement services in this thesis.

5.2.2 Classification Approaches

Classification approaches build a model trained by analyzing data sets with known outcome (supervised learning). The training data set is analyzed to identify dependencies between *input variables* (variables influencing the outcome) and one *class variable* (variable encoding the outcome). The model can the be applied to unknown data of the same structure to predict the outcome.

Techniques of this category are for example:

- Decision trees
- Support Vector Machines
- Discriminant Analysis
- Logistic Regression
- Bayes Classification
- Feedforward Backpropagation Networks

As classification approaches in general are not suited (see below), they are not described here.

Suitability: Classification approaches require a training data set with known outcome. This training set must represent a representative set containing all possible data values. In order to be able to identify legacy code for service implementation, a user would first have to classify a large amount of example code for *all* services. Classification approaches could then forecast for the remaining code to which service it might belong.

However, the classification of exemplary code for all services is time consuming and in industrial projects not realizable. Therefore, all techniques of this category are not applied in this thesis.

5.2.3 Link Analysis Approaches

Link analysis approaches try to identify rules in data about which items are related. The approaches analyze if items often appear together in *transactions*. A transaction is a closed action, e.g., a complete and finished purchase.

Association Rules

Association rules identify rules of the kind *Precondition* → *Consequences*. Such rules are often used in recommender systems (customers who bought X also bought Y and Z).

Suitability : In identifying legacy code for service implementation, such rules could be used to identify "core functionality" of services: If activity A is executed, classes C, D and E are always called.

Chapter 7 will introduce the foundations of link analysis with association rules in detail. Chapter 12 will describe the results of applying association rules on the example dataset.

Sequence Detection

Sequence detection is similar to association rules except that the sequence of the actions is considered.

Suitability : For identifying legacy code able to implement services, sequences do not play a role (it is not important *when* the code of a service is executed). However, sequence detection could be interesting for service orchestration. Often occurring sequences of activities can give a hint how to orchestrate the services supporting these activities. However, this is not part of the topic of this theses and will therefore not be evaluated here.

Summary of Data Mining Techniques

Two techniques have been identified that seem to be suited for identifying legacy code able to implement services. The theory behind these techniques will be described in the remaining chapters of this part. Part III will describe how to apply the techniques on the identification of legacy code for service implementation.

Chapter 6

Cluster Analysis

In this chapter, cluster analysis approaches are introduced. General definitions for levels of measurement and similarity and dissimilarity measures are provided. In addition, techniques to evaluate clustering solutions are described.

Cluster analysis (also called clustering) approaches use similarity and dissimilarity measures to compute how closely related items (tuples of data) are.

Definition 6.0.1 (*Item*)

An item p_i is one entry in an itemset $P = \{p_1, \dots, p_n\}$ describing one single piece of data. An item consists of attributes *a describing properties of the item. The number of attributes of an item is called the* dimension *d of the item.*

Each attribute has a *data type* and a level of measurement (also called scale).

6.1 Data Types and Levels of Measurement

Definition 6.1.1 (*Data type*)

The data type of an attribute denotes the degree of quantization in the data [Gan et al., 2007]. That is, the data type defines what kind of values an attribute can have.

An attribute can be typed into *discrete* or *continuous* (Figure 6.1). Discrete attributes have a finite number of values, called *categories*.

Definition 6.1.2 (*Category*)

The possible values of a discrete attribute a are called categories. *The set of categories for an attribute a is called the* domain *L of the attribute a.*

If exactly two categories exist ($L = 2$), an attribute is of binary type, otherwise ($L \neq 2$) of nominal type. Therefore, the nominal type is a generalization of the binary type. In contrast to discrete attributes, continuous attributes can have an infinite number of values.

In addition to the data type, attributes also have a level of measurement.

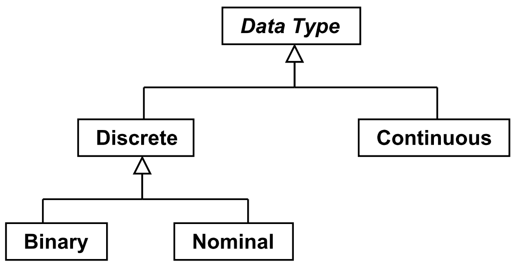


Figure 6.1: Hierarchy of data types

Definition 6.1.3 (*Level of measurement*)

The level of measurement (also called scale*) describes the relative significance of attribute values [Gan et al., 2007].*

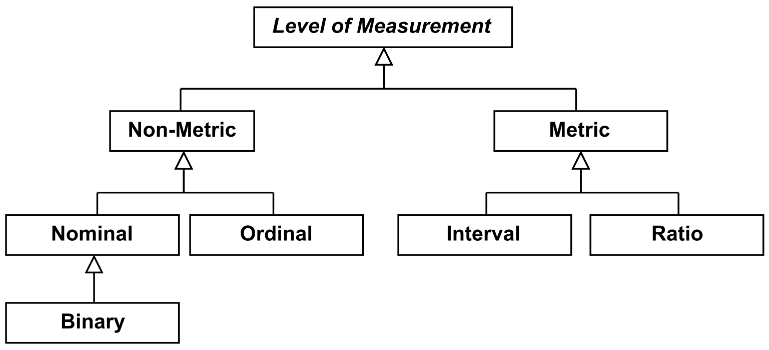


Figure 6.2: Hierarchy of levels of measurement

In general, *non-metric* and *metric* scales are differentiated (Figure 6.2). Non-metric (qualitative) attributes are represented by a discrete set of categories. Metric (quantitative) attributes are represented by numeric measurements. Table 6.1 on the next page summarizes the four levels of measurement proposed by Stevens [1946]. *Nominal-scaled* attributes express distinctions between their values, but do not impose an order between the values. Examples for nominal-scaled attributes are color or religious affiliation. A special case are *binary-scaled* attributes, which are nominal-scaled and only have two values like gender (male/female). *Ordinal-scaled* attributes impose an order on their values. However, the distance between values is not quantifiable. An example for ordinal-scaled attributes are marks in school. In contrast, *interval-scaled* attributes have fixed distances between their values. An example is temperature measured in degrees Celsius. In addition, *ratio-scaled attributes* define a natural, non-arbitrary zero point. Examples are height, weight or temperature measured in degrees Kelvin.

According to the similarity values computed during clustering, items that are "similar" are put together into one *cluster*.

Definition 6.1.4 (*Cluster*)

A cluster $C_x = \{p_1, \ldots, p_m\}$ is a set of items belonging together according to given clustering

	Non-Metric (Qualitative)		Metric (Quantitative)	
	Nominal	Ordinal	Interval	Ratio
Description	Values stand for labels or names (binary: only two values). Only equality can be compared.	Values stand for labels or names and impose an order.	Values are ordered and measured in fixed or equal distances.	Values for which a natural zero point is defined.
Operations	`equals()`, `!equals()`	$\ldots <, \leq, > \geq$	$\ldots +, -$	$\ldots *, \div$
Examples	religious affiliation, gender	marks in school	temperature in degrees Celsius	Height, weight

Table 6.1: Levels of Measurement [Möhring, 2009]

criteria (e.g., distance or similarity values). The set $C = \{C_1, \ldots, C_k\}$ is the set of all disjoint clusters for the given dataset, called clustering solution*. One dataset can have many different clustering solutions.*

At the end of the clustering process, items in one cluster are similar to each other and dissimilar to items of other clusters, ideally. Therefore, a cluster builds a homogeneous subgroup of items. A user has to interpret the meaning of the clusters then [Schulze, 2007]. During the clustering process, four core decisions are to be made:

1. Select a similarity or distance measure (Section 6.2)
2. Select a cluster distance measure (Section 6.4)
3. Select a clustering algorithm (Section 6.5)
4. Decide how to interpret clustering solution (Section 6.6)

These steps are described in the remaining chapter in more detail.

6.2 Similarity and Distance Measures for Homogeneous-Scaled Attributes

Similarity and distance (dissimilarity) measures compute a numeric value describing how similar or dissimilar two items are. They are defined as follows [Schmitt, 2004].

Definition 6.2.1 (*Similarity measure s*)

Let $P = \{p_1, \ldots, p_n\}$ be a set of items. A function

$$s : P \times P \to [0,1]$$

is called similarity function *iff*

$$\begin{aligned} &\forall p_i, p_j \in P \\ &s(p_i, p_i) = 1 \wedge && (6.1) \\ &s(p_i, p_j) = s(p_j, p_i) && (6.2) \end{aligned}$$

Definition 6.2.2 (*Distance measure* d)

Let $P = \{p_1, \ldots, p_n\}$ be a set of items. A function

$$d : P \times P \to \mathbb{R}_0^+$$

is called distance function *iff*

$$\begin{aligned} &\forall p_i, p_j \in P \text{ be:} \\ &d(p_i, p_i) = 0 \wedge && (6.3) \\ &d(p_i, p_j) = d(p_j, p_i) && (6.4) \end{aligned}$$

Similarity and distance measures are related and can be transformed. A similarity value can be calculated from a distance and vice versa.

$$s(p_i, p_j) = 1 - \frac{d(p_i, p_j)}{\max\limits_{p_i, p_j \in P} (d(p_i, p_j))} \tag{6.5}$$

$$d(p_i, p_j) = 1 - s(p_i, p_j) \tag{6.6}$$

Depending on the level of measurement of the attributes, different similarity and distance measures are used. In the remaining section, various measures are introduced for each level of measurement. In this section, items are assumed to be *scale-homogeneous*.

Definition 6.2.3 (*Scale-homogeneous items*)

Two items are scale-homogeneous *if both items have the same attributes and all attributes of both items have the same data type and the same level of measurement.*

Section 6.3 will introduce measures for mixed-scaled items.

Definition 6.2.4 (*Mixed-scaled items*)

Two items are mixed-scaled *if both items have pairwise the same attributes but the attributes of an item do not need to have the same level of measurement.*

6.2.1 Measures for Nominal-Scaled Attributes

Nominal-scaled attributes provide a finite set of categories. Categories of nominal-scaled attributes are not ordered and therefore can only be compared for equality. The similarity between items with nominal, non-binary attributes can be computed by using the *Generalized Simple Matching Coefficient* $s_{GSMC}(p_i, p_j)$. It is defined as

$$s_{GSMC}(p_i, p_j) = \frac{N_s}{d} \tag{6.7}$$

with N_s being the number of attributes where both attributes have the same value and d the dimension of the two items. A corresponding distance measure is provided by:

$$d_{GSMC}(p_i, p_j) = 1 - \frac{N_s}{d} \tag{6.8}$$

Alternatively, nominal-scaled attributes may be transformed into several binary-scaled attributes. Each category is transformed into a new binary attribute representing this category. The state of the original attribute is represented by setting the binary attribute corresponding to this state to 1 and all other binary attributes to 0. Similarity and distance can then be measured as presented in Section 6.2.2. However, depending on the number of categories, this approach might generate many new attributes [Möhring, 2009].

6.2.2 Measures for Binary-Scaled Attributes

Binary-scaled attributes are nominal-scaled attributes with only two categories. They can be compared by using the General Simple Matching Coefficient that had already been presented in Section 6.2.1. In addition, various other measures weighting the different states of the items exist. These measures use a contingency matrix between the items p_i and p_j (Table 6.2).

p_i / p_j	1	0
1	N_{11}	N_{10}
0	N_{01}	N_{00}

Table 6.2: Contingency matrix between two items p_i and p_j

N_{11} is the number of attributes, both items have set to 1, N_{00} the number of attributes, both items have set to 0. N_{10} and N_{01} are the number of attributes, one of the items has set to 1 and the other item has set to 0 respectively. The sum of all four values is the dimension of the attribute $d = N_{11} + N_{10} + N_{01} + N_{00}$. Using this contingency matrix, similarity and dissimilarity of two binary-scaled items may be computed.

There are two types of measures for binary-scaled attributes: symmetric and asymmetric measures [Gan et al., 2007]. Symmetric measures take the value of N_{00} into account while asymmetric measures do not. Often, the number of attributes two items have *not* in common is much greater than they do have in common. Considering the N_{00} values may then distort a comparison of both items. Therefore, the N_{00} are ignored in measures for asymmetric attributes. Table 6.3 on the next page shows various similarity and distance measures for both, symmetric and asymmetric binary-scaled attributes.

Measure	$s(p_i, p_j)$	$d(p_i, p_j)$	Range
	Symmetric measures		
Simple Matching	$\frac{N_{11}+N_{00}}{d}$	$\frac{N_{10}+N_{01}}{d}$	$[0,1]$
Tanimoto/Rogers	$\frac{N_{11}+N_{00}}{d+N_{10}+N_{01}}$	$\frac{2(N_{10}+N_{01})}{d+N_{10}+N_{01}}$	$[0,1]$
	Asymmetric measures		
Jaccard	$\frac{N_{11}}{N_{11}+N_{10}+N_{01}}$	$\frac{N_{10}+N_{01}}{N_{11}+N_{10}+N_{01}}$	$[0,1]$
Russel-Rao	$\frac{N_{11}}{d}$	$1-\frac{N_{11}}{d}$	$[0,1]$

Table 6.3: Measures for binary-scaled attributes [Gan et al., 2007]

6.2.3 Measures for Ordinal-Scaled Attributes

As with nominal-scaled attributes, values of ordinal-scaled attributes represent categories, too. Therefore, measures for nominal-scaled attributes may be used for ordinal-scaled ones, too (see Section 6.2.1).

However, for ordinal-scaled attributes, categories are ordered. This information about ordering can be used to calculate more specific distances and similarities for ordinal-scaled attributes than for nominal-scaled ones.

Let $L(a) = \{l_1, \dots, l_o\}$ be the domain of an attribute a. The "distance" between two categories l_x and l_y can be defined as how many categories l_z are between the two categories:

$$d(l_x, l_y) = \#\{l_z \mid l_x \leq l_z < l_y\} \tag{6.9}$$

The greater the number of "intra-categories", the greater is the difference between the two categories.

Based on this observation, ordinal-scaled attributes are often transformed into interval-scaled attributes. Here, the problem is to introduce an appropriate numeric value for each category that preserves the order of the categories and provides an appropriate spacing between the values.

One approach commonly used is to recode categories into their *rank*, that is their index in the ordered set L: $l_1 = 1, l_2 = 2, \dots, l_o = o$. For each attribute a of item p_i, the category of the attribute is replaced by its rank $r_i(a)$.[1]

The similarity or distance between items with ordinal-scaled attributes can then be calculated using similarity or distance measures for metric-scaled attributes (which will be introduced in the following section) using the rank instead of the category.

6.2.4 Measures for Metric-Scaled Attributes

Metric-scaled attributes (interval-scaled and ratio-scaled) are defined by numeric measurements following a linear scale (i.e., the space between measurements is fixed). In the following, various distance measures are presented[2].

One of the distance functions for metric-scaled attributes most commonly used is the *Minkowski Distance*. It is defined as follows.

Definition 6.2.5 (*Minkowski Distance* d_{L_m})

Let $P = \{p_1, \dots, p_n\} \in \mathbb{R}$ be a set of items. If each item p_i has d dimensions, the Minkowski Distance is a function

$$d_{L_m} : P \times P \to \mathbb{R}_0^+$$

computing the distance between two items p_i and p_j:

$$d_{L_m}(p_i, p_j) = \left(\sum_{a=1}^{d} \mid p_i[a] - p_j[a] \mid^m \right)^{\frac{1}{m}} \tag{6.10}$$

Depending on the choice of the parameter m, the distance is computed in different ways. The most common values of m lead to functions having their own names.

The City Block Distance ($m = 1$) or *Manhattan Distance* expresses the length of the shortest geometrical path between two items p_i and p_j in a d-dimensional space with the restriction that all segments of the path must be parallel to one of the axes:

$$d_{L_1}(p_i, p_j) = \sum_{a=1}^{d} \mid p_i[a] - p_j[a] \mid \tag{6.11}$$

The Euclidean Distance ($m = 2$) expresses the shortest, geometrical distance between two items p_i and p_j in an d-dimensional space:

$$d_{L_2}(p_i, p_j) = \sqrt{\sum_{a=1}^{d} (p_i[a] - p_j[a])^2} \tag{6.12}$$

[1]If different attributes shall be compared – i.e., they have a different number of categories – ranks are normalized by the number of categories. The normalized value is then $z_i(a) = \frac{r_i(a)}{|L(a)|} \in [0, 1]$

[2]Usually, similarity is not computed for metric-scaled attributes. However it can be calculated by transforming the distance as introduced in Equation 6.5 on page 32

The Chebyshev Distance ($m = \infty$) or *maximum distance* is defined as the maximum value of the distances of the attributes [Gan et al., 2007]:

$$d_{L_\infty}(p_i, p_j) = \max_{a=1}^{d} \mid p_i[a] - p_j[a] \mid \tag{6.13}$$

Normalization of Metric-scaled Measurement Units

Assuming scale-homogeneity, it is not ensured that an attribute in two items has the same measurement unit. E.g., a length might be measured in meters in the first item and in centimeters in the second item (both attributes have the same data type, the same level of measurement and describe the same property).

Unfortunately, the choice of different measurement units for metric-scaled attributes has a high impact on distance measures. The choice of measurement units may act as weighting of an attribute that might lead to unintended distance values (e.g., expressing one attribute containing lengths (a_1) in centimeters and expressing another attribute containing other lengths (a_2) in meters, weights a_1 by a factor of 100). To solve this problem, values are *normalized*, e.g., by using the *z-transformation*. The normalized value $z(p_i[a])$ for an attribute a of item p_i is computed as follows:

$$z(p_i[a]) = \frac{p_i[a] - \text{mean}(a)}{\text{stdDev}(a)} \tag{6.14}$$

with the mean mean and the standard deviation stdDev of a defined as (n being the size of the set of items):

$$\text{mean}(a) = \frac{1}{n} \sum_{p_i \in P} p_i[a] \tag{6.15}$$

$$\text{stdDev}(a) = \sqrt{\frac{1}{n-1} \sum_{p_i \in P} p_i[a]^2} \tag{6.16}$$

Using this z-transformed value instead of the original value reduces the impact of measurement units significantly.

6.3 Measures for Mixed-Scaled Attributes

Up to this point, items have been assumed to be scale-homogeneous. However, items are described by attributes with different levels of measurement, in most cases. In order to unify measurement, all attributes could be transformed to the same level of measurement. For example, all attributes could be transformed into binary-scaled attributes. However, this approach leads to a loss of information. Alternatively, all attributes could be transformed into interval-scaled ones. As a drawback, additional unknown information must be added to lower-scaled attributes (e.g., for nominal attributes, some ordering must be defined, although there is no ordering imposed).

So both transformation approaches lead to some sort of wrong information brought into the data. Therefore, a measurement must be found that is able to compute the distance taking all levels of measurement into account.

The general idea is to measure the similarity or distance for each level of measurement separately [Gan et al., 2007]. For all nominal attributes of the two items, one similarity or distance value is computed using a measure for nominal attributes; similarly, for all binary, ordinal or metric attributes of the two items, an appropriate measure is used. Finally, all similarity or distance values are summed up to get an overall value. Using distance measures, e.g., the following single values could be calculated. *For nominal-scaled attributes*, the Generalized Simple Matching Coefficient (Equation 6.7 on page 33) can be used: $d_{GSMC}(p_i, p_j) = \frac{1-N_{s,s}}{u}$. *Asymmetric binary-scaled attributes* can be compared by using the Jaccard Coefficient (Table 6.3 on page 34) $d_{Jaccard} = \frac{N_{10}+N_{01}}{N_{11}+N_{10}+N_{01}}$. *Metric-scaled and ordinal-scaled attributes* can be compared by transforming ordinal-scaled values to metric-scaled values; then, the Euclidean distance (Equation 6.12 on page 35) can be used for metric-scaled and transformed ordinal-scaled attributes: $d_{L_2}(p_i, p_j) = \sqrt{\sum_{a=1}^{d} (p_i[a] - p_j[a])^2}$. Finally, all distances are summed up to a single value: $d_{Overall} = \omega_1 d_{GSMC} + \omega_2 d_{Jaccard} + \omega_3 d_{L_2}$ where $\omega_1, \ldots, \omega_3$ are arbitrary weightings of the single values.

However, this approach does not lead to normalized values. Especially for similarity measures, the overall value must be in the range $[0, 1]$. Therefore, an adapted approach must be used, normalizing the result to the range of $[0, 1]$.

6.3.1 General Similarity Coefficient

The following, widely used approach has been developed for this purpose [Gan et al., 2007]. For two d-dimensional points p_i and p_j with attributes a, the *General Similarity Coefficient s_{GSC}* is defined as follows:

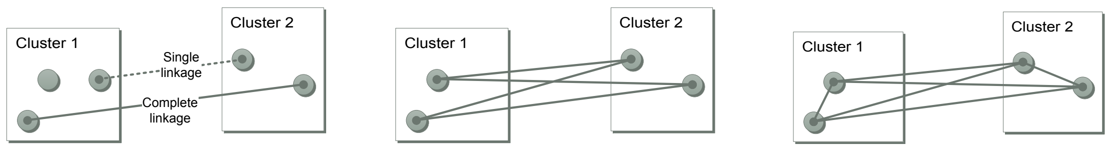
$$s_{GSC}(p_i, p_j) = \frac{\sum\limits_{a=1}^{d} w(p_i[a], p_j[a]) * s(p_i[a], p_j[a])}{\sum\limits_{a=1}^{d} w(p_i[a], p_j[a])} \tag{6.17}$$

$w(p_i[a], p_j[a])$ and $s(p_i[a], p_j[a])$ are defined differently for each level of measurement.[3] For asymmetric-binary-scaled attributes they are defined as:

$$s(p_i[a], p_j[a]) = \begin{cases} 1 & \text{both attributes have the value } \texttt{true} \\ 0 & \text{otherwise} \end{cases}$$

$$w(p_i[a], p_j[a]) = \begin{cases} 0 & \text{both attributes have the value } \texttt{false} \\ 1 & \text{otherwise} \end{cases}$$

[3]This version of the S_{GSC} is adapted to ignore missing values as there are no missing values in the data used for this thesis. For the original version including missing values see [Gan et al., 2007].



(a) Single linkage and complete linkage **(b)** Between-groups linkage **(c)** Within-groups linkage

Figure 6.3: Various distance measures for cluster distances

For nominal- or ordinal-scaled attributes they are defined as:

$$s(p_i[a], p_j[a]) = \begin{cases} 1 & p_i[a] = p_j[a] \\ 0 & \text{otherwise} \end{cases}$$
$$w(p_i[a], p_j[a]) = 1$$

For metric-scaled attributes they are defined as:

$$s(p_i[a], p_j[a]) = 1 - \frac{|\ p_i[a] - p_j[a]\ |}{R_a} \qquad \text{with } R_a \text{ the size of the } \textit{range} \text{ of the attribute } a.$$
$$w(p_i[a], p_j[a]) = 1$$

The General Similarity Coefficient results in values between 1 (items are identical) and 0 (items are extremely different).

6.4 Measures for Cluster Distances

The similarity and distance measures presented in the previous sections all compute the similarity or distance between two *items*. In addition, measures to compute the similarity or distance between two *clusters* are needed.

In the following, various distance measures between two disjoint clusters $C_x = \{p_1, \ldots, p_n\}$ with n items and $C_y = \{q_1, \ldots, q_m\}$ with m items are described (Figure 6.3).

The *single-linkage* (nearest-neighbor) approach defines the distance between two clusters C_x and C_y as the *shortest* distance (according to a given distance measure) between their items (Figure 6.3a):

$$d_{NearestNeighbor}(C_x, C_y) = \min_{p_i \in C_x, p_j \in C_y} d(p_i, q_j) \tag{6.18}$$

The *complete-linkage* (furthest neighbor) approach defines the distance between two clusters as the *largest* distance between their items (Figure 6.3a):

$$d_{FurthestNeighbor}(C_x, C_y) = \max_{p_i \in C_x, p_j \in C_y} d(p_i, q_j) \tag{6.19}$$

The *between-groups linkage* (also called between-groups average, BAverage) approach defines the distance between two clusters as the mean of all distances between each item in the first cluster and each item in the second cluster (Figure 6.3b on the preceding page):

$$d_{BAverage}(C_x, C_y) = \underset{p_i \in C_x, p_j \in C_y}{\text{mean}} d(p_i, q_j) \tag{6.20}$$

The *within-groups linkage* (also called within-groups average, WAverage) approach defines the distance between two clusters as the mean of all distances between all items in the two clusters (Figure 6.3c on the facing page). With $C_{x,y} = C_x \cup C_y$, WAverage is defined as:

$$d_{WAverage}(C_x, C_y) = \underset{p_i, p_j \in C_{x,y}}{\text{mean}} d(p_i, p_j) \tag{6.21}$$

A more complex distance measure is the log-likelihood distance which is described in the following.

6.4.1 Log-Likelihood Distance

The log-likelihood distance between two clusters C_x and C_y is defined as [SPSS, 2006]:

$$d_{log}(C_x, C_y) = \xi_{C_x} + \xi_{C_y} - \xi_{\langle C_x, C_y \rangle} \tag{6.22}$$

where ξ_{C_x}, ξ_{C_y} are the log-likelihoods for clusters C_x and C_y and $\xi_{\langle C_x, C_y \rangle}$ is the log-likelihood for the union of clusters C_x and C_y.

The log-likelihood ξ_c for one cluster c[4] is:

$$\xi_c = -N_c \left(\sum_{a \in A^{met}} \frac{1}{2} \log(\hat{\sigma}_a^2 + \hat{\sigma}_{c,a}^2) + \sum_{a \in A^{nmet}} \hat{E}_{c,a} \right) \text{ with} \tag{6.23}$$

$$\hat{E}_{c,a} = -\sum_{l \in L_a} \frac{N_{c,a,l}}{N_c} \log \frac{N_{c,a,l}}{N_c} \tag{6.24}$$

A^{met} is the total number of metric-scaled attributes, A^{nmet} is the total number of non-metric-scaled attributes. N_c is the number of items in cluster c. $N_{c,a,l}$ is the number of items in cluster c belonging to the l-th category of the a-th attribute. $\hat{\sigma}_a^2$ is the estimated variance for the a-th metric-scaled attribute for all items. $\hat{\sigma}_{c,a}^2$ is the estimated variance for the a-th metric-scaled attribute for items in cluster c.

The log-likelihood ξ_c for one cluster c measures a kind of variance within this cluster. The first part of the formula $-N_c \sum_{a \in A^{met}} \frac{1}{2} \log(\hat{\sigma}_a^2 + \hat{\sigma}_{c,a}^2)$ measures the variance of metric-scaled attributes. The second part $\sum_{a \in A^{nmet}} \hat{E}_{c,a}$ measures the variance for non-metric-scaled attributes [Bacher et al., 2004].

The log-likelihood *distance* $d_{log}(C_x, C_y) = \xi_{C_x} + \xi_{C_y} - \xi_{\langle C_x, C_y \rangle}$ measures if the sum of variances of both clusters is bigger than the variance would be if the clusters would have been joined.

In the following, various clustering algorithms are introduced.

[4]For better readability of formulas, a small c has been used to describe an arbitrary cluster instead of C_x

6.5 Clustering Algorithms

The clustering algorithm defines how items are grouped together. There are several types of approaches which are described briefly in the following. For a detailed discussion of clustering algorithms, see [Gan et al., 2007].

6.5.1 Hierarchical Clustering Approaches

Hierarchical clustering approaches build clusters iteratively. There are two ways to perform the clustering process:

1. Agglomerative hierarchical clustering
2. Divisive hierarchical clustering

Both approaches are described in the remaining section.

Agglomerative hierarchical clustering approaches can be seen as bottom-up approaches. Algorithm 1 describes the agglomerative hierarchical clustering algorithm.

Algorithm 1 Agglomerative hierarchical clustering algorithm

```
1: Place each item $p_i \in P$ in an own cluster: $C_1 = \{p_1\}, \ldots, C_n = \{p_n\}$
2: while # clusters > 1 do
3:     Compute distances $d(C_x, C_y)$ between all clusters
4:     Merge two clusters with:
           $\min_{C_x, C_y \in C}(d(C_x, C_y))$
5: end while
```

At the beginning, each item is associated to its own cluster. The distances between all clusters are computed using one of the measures for cluster distances (see Section 6.4). Then, the two most similar clusters are merged iteratively. The iteration ends when all items are in one cluster. Alternatively, a user can stop the clustering process at any time if he decides that the current clustering solution is the intended one.

In *divisive hierarchical clustering approaches*, all items are in one cluster at the beginning. In each step, one cluster is split into two smaller clusters, according to given criteria. The division process ends when all items are in their own cluster (containing only one entry) or a user stops the clustering process. A splitting criterion might be the average dissimilarity to all other items, as used by DIANA (Divisive Analysis, Algorithm 2 on the facing page).

DIANA iteratively removes the most dissimilar item from the largest cluster and creates a new cluster with this item. It then moves items of the largest cluster that are more similar to the new cluster, there. The iteration stops when all clusters contain only one item or when a user stops the clustering process.

Algorithm 2 DIANA algorithm

```
1: Associate all items $p_i \in P$ to one single initial cluster $C_1 = \{p_1, \dots, p_n\}$
2: repeat
3:     Select cluster with the largest diameter:    ▷ The diameter is the largest dissimilarity of two items.
           $$\text{diameter}(C_x) = \max_{C_x \in C}(\max_{p_i, p_j \in C_x}(d(p_i, p_j)))$$
4:     repeat
5:         Find item with the biggest average dissimilarity to all other items:
               $$p_{max} = \max_{p_i \in C_x}(\text{average}_{p_j \in C_x}(d(p_i, p_j)))$$
6:         Initialize a new cluster $S$ (called splinter group) with $p_{max}$:
               $$S = \{p_{max}\}$$
7:         For all items $p_i \notin S$, compute:
               $$D(p_i) = \text{average}_{p_j \notin S}(d(p_i, p_j)) - \text{average}_{p_j \in S}(d(p_i, p_j))$$
8:         if $D(p_i) \geq 0$ then
9:             $S = \{p_i\} \cup S$    ▷ $p_i$ is more similar to $S$ than to $C_x$
10:        end if
11:    until $D_i < 0$
12: until all clusters contain only 1 item
```

A major drawback of both hierarchical clustering approaches is the inability to change decisions made earlier in the clustering process. Once two clusters are merged (or divided), they can not be split up (or merged) anymore. In addition, hierarchical approaches are slow on large datasets as they must pre-compute a complete distance matrix in advance.

A major advantage of hierarchical approaches is the open number of clusters as the user can interrupt the clustering process at any time.

A more complex, hierarchical clustering approach facing the performance issue, is the TwoStep algorithm which is described in the following. Section 6.5.2 will introduce a different approach – the partitioning clustering approach – facing the inability to change decisions made earlier in the clustering process.

TwoStep Hierarchical Clustering Approach

TwoStep, developed by Chiu et al. [2001] is a hierarchical agglomerative clustering approach using two stages of clustering [SPSS, 2001]:

1. Sequential pre-clustering of the dataset to reduce the number of items the hierarchical algorithm must process
2. Hierarchical agglomerative clustering of the pre-clusters into the final solution

The first stage scans over the dataset in sequential order and decides if an items should be allocated to a pre-cluster or should start a new pre-cluster based on a distance criterion. The pre-clustering is implemented by using *cluster feature (CF) trees*, a special form of trees, introduced with the BIRCH clustering algorithm [Zhang et al., 1996].

Definition 6.5.1 (*Cluster Feature (CF)*)

A cluster feature is a summarized representation of a cluster. A cluster feature stores the following information about a cluster: the number of items in the cluster, the linear sum of the items and the squared sum of the items.

Cluster feature trees are hight-balanced trees that have cluster features as nodes. Non-leaf nodes have one cluster feature summarizing the cluster feature values of their child nodes. Leaf nodes represent clusters.

The standard settings of the implementation result in a maximum number of 512 pre-clusters.

In the second stage, the pre-clustered dataset is processed by an hierarchical agglomerative algorithm using the log-likelihood distance as distance measure (see Section 6.4.1). Because the algorithm has to process 512 pre-clusters at most, performance issues do not influence the algorithm.

Determining the Best Number of Clusters Automatically

As an additional feature, the TwoStep algorithm provides a possibility to compute the "best" number of clusters automatically. This number is determined in two steps. First, the *maximum* number of clusters is estimated using *Akaike's Information Criterion (AIC)* or *Bayes' Information Criterion (BIC)*. They are defined as follows [Bacher et al., 2004; SPSS, 2006].

Definition 6.5.2 (*Akaike's Information Criterion (AIC)*)

Let k be the number of clusters of the current clustering solution. Akaike's Information Criterion (AIC) for this solution is defined as:

$$AIC_k = -2\sum_{i=1}^{k} \xi_{C_i} + r_k \tag{6.25}$$

where

$$r_k = k\left\{2A^{met} + \sum_{a=1}^{A^{nmet}} L_a - 1\right\} \tag{6.26}$$

L_a is the number of categories for the a-th attribute and the other parameters are defined as in Section 6.4.1. r_k is the sum of all metric-scaled attributes and all categories of non-metric-scaled attributes in all k clusters.

Definition 6.5.3 (*Bayes' Information Criterion (BIC)*)

Bayes' Information Criterion (BIC) is defined as:

$$BIC_k = -2\sum_{i=1}^{k} \xi_{C_i} + r_k \log n \tag{6.27}$$

where n is the number of all items in the dataset and the other parameters are defined as in Definition 6.5.2.

With ε being an arbitrary threshold value[5], the maximum number of clusters is the number of clusters of a solution where

$$\frac{AIC_k}{AIC_{k-1}} < \varepsilon \text{ or} \tag{6.28}$$

$$\frac{BIC_k}{BIC_{k-1}} < \varepsilon \tag{6.29}$$

still holds.

Second, the "best" number of clusters is determined by looking at the "ratio change" of distances between two clusters when they are merged:

$$R(k) = \frac{d_{k-1}}{d_k} \tag{6.30}$$

d_{k-1} is the distance when k clusters are merged to $k-1$ clusters. The "best" number of clusters is obtained where a big jump in the ratio $R(k)$ occurs. A big jump in $R(k)$ indicates, that two clusters have been merged although their distance was much bigger than in merges before. Therefore, this merge seems to be wrong.

Summarizing, the TwoStep algorithm is able to handle large datasets with mixed-scaled items and to determine the right number of clusters automatically. However, a decision made during the clustering process is still not revertible. In the following, a different approach able to change decisions during the clustering process is introduced.

6.5.2 Partitioning Clustering Approaches

Partitioning clustering approaches use an *error function* (or objective function) for computing how good a clustering solution is. Starting from an initial clustering solution (with a fixed number of clusters), the approaches optimize the solution against this error function. The approaches terminate as the clustering solution does not change anymore.

One of the most popular partitioning approaches is the *k-means algorithm* using the following error function E.

Let $C = \{C_1, \dots, C_k\}$ be the k clusters of the dataset. Then E is defined as:

$$E = \sum_{C_x \in C} \sum_{p_i \in C_x} d(p_i, \mu(C_x)) \tag{6.31}$$

[5]In the original implementation, ε is 0.04

with $\mu(C_x)$ being the mean of the cluster C_x.

The *k*-means algorithm tries to minimize this error function iteratively (Algorithm 3).

Algorithm 3 *k*-means algorithm

```
1: Set k initial cluster means (seeds): $M = \{\mu(C_1), \dots, \mu(C_k)\}$
                                                  ▷ Initialization step
2: repeat
3:     Associate all items to nearest cluster (i.e., with the minimal distance to the cluster):
           $d_{minimal} = \min_{p_i \in P, C_x \in C} (d(p_i, C_x))$
                                                  ▷ Assignment step
4:     Re-compute cluster means $\hat{M} = \{\hat{\mu}(C_1), \dots, \hat{\mu}(C_k)\}$ with:
           $\hat{\mu}(C_x) = \frac{1}{|C_x|} \sum_{p_i \in C_x} (p_i),\ C_x \in C$
                                                  ▷ Update step
5: until clusters do not change anymore
```

First, *k* initial cluster means (*seeds*) are set (initialization). They can be selected randomly, computed by some heuristic (e.g., *k* seeds are set in a way, that their distances are maximal) or given by a user. The remaining algorithm is an iteration of *assignments* (items are associated to the nearest mean, or seed in the first iteration) and *updates* (the means of all clusters are recomputed). As distance measure, any of the measures introduced in Section 6.2.4 can be used. The original implementation of the *k*-means algorithm used the squared Euclidean distance [Hartigan and Wong, 1979]. The iteration terminates when the clusters do not change anymore.

Partitioning approaches have one big advantage over hierarchical approaches: they are much faster. However, the number of clusters is fixed and the clustering result highly depends on the initial cluster seeds.

In addition, *k*-means requires pure metric-scaled variables in order to be able to compute the mean. Non-metric variables must be transformed into metric ones before using *k*-means.

6.5.3 Fuzzy Clustering Approaches

The clustering approaches presented in the previous sections assigned an item to exactly one cluster. This is called a hard or crisp assignment. In practice, it is often unintuitive to choose such a strict association.

Fuzzy clustering approaches provide a more natural assignment to clusters in such situations. Here, one item is mapped to all clusters by a membership function. The membership function $u(p_i, C_x)$ describes the *degree of membership* of each item p_i to each cluster C_x:

$$u : P \times C \to [0, 1] \tag{6.32}$$

Calculating the membership for all items and clusters results in a matrix U, called the *fuzzy k-partition* of the dataset:

$$U = \begin{bmatrix} u(p_1,C_1) & \cdots & u(p_1,C_x) & \cdots & u(p_1,C_k) \\ \vdots & \ddots & & & \vdots \\ u(p_i,C_1) & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ u(p_n,C_1) & \cdots & \cdots & \cdots & u(p_n,C_k) \end{bmatrix} \tag{6.33}$$

One of the well known fuzzy clustering algorithms is the *fuzzy k-means algorithm*. The fuzzy k-means algorithm tries to minimize the following error function iteratively:

$$E_q = \sum_{i=1}^{n} \sum_{j=1}^{k} u_q(p_i,C_x) d^2(p_i,\mu(C_x)) \tag{6.34}$$

where q is a parameter controlling the "fuzziness" of the resulting clusters ($q > 1$). A value of q near to 1 leads to crisper assignments of items to clusters, a high value leads to more fuzzy assignments

The fuzzy k-means algorithm works as described in Algorithm 4.

Algorithm 4 Fuzzy k-means algorithm

```
1: Set initial cluster means (seeds): $M = \{\mu(C_1),\ldots,\mu(C_k)\}$
2: Compute initial membership Matrix $U$ with:
       $u_q(p_i,C_x) = \frac{\left(d^2(p_i,C_x)\right)^{-\frac{1}{q-1}}}{\sum_{l=1}^{k}\left(d^2(p_i,C_l)\right)^{-\frac{1}{q-1}}}, \; p_i \in P, C_x \in C$
                                        ▷ Initialization steps
3: repeat
4:     Re-compute cluster means $\hat{M} = \{\hat{\mu}(C_1),\ldots,\hat{\mu}(C_k)\}$ with:
           $\hat{\mu}(C_x) = \frac{\sum_{i=1}^{n} u_q(p_i,C_x)p_i}{\sum_{i=1}^{n} u_q(p_i,C_x)}, \; C_x \in C$
                                        ▷ Update step
5:     Update membership matrix $U$ to $\hat{U}$ according to step 2.
                                        ▷ Assignment step
6: until $\max_{p_i \in P, C_x \in C} \mid u_q(p_i,C_x) - \hat{u}_q(p_i,C_x) \mid < \varepsilon$    ▷ $\varepsilon$ is a termination criterion with $0 \le \varepsilon \le 1$
```

Similar to the k-means algorithm, the clustering process can be grouped into the steps initialization, assignment and update. During the two initialization steps, the cluster seeds are set (randomly, by a heuristic or by a user). In addition, the initial membership matrix is computed. During the following iterations, the cluster means are recomputed (update step) and the membership matrix is updated (assignment step). The iteration stops when the change of the assignments is smaller than a given termination criterion ε. In Section B.1, a detailed example of the fuzzy k-means algorithm is provided.

After having executed one of the clustering approaches presented in this section, the clustering solution must be interpreted.

6.6 Interpreting Clustering Solutions

After the clustering algorithms have finished, it is a user's task to interpret the clustering solution. There are no "right" or "wrong" clustering solutions. Based on the initial research question, there are only solutions that help answering this question, or not. Therefore, a clustering solution can be called "suitable" if it is interpretable and gives answers to the initial research question.

If a clustering solution is not suitable, other algorithms could be used and parameters could be adjusted to get different clustering results that might be more suitable. Therefore, clustering analyses are heavily influenced by subjective decisions and interpretations.

However, it is often tried to rate clustering solutions more objectively. To rate a clustering solution, two main approaches are possible:

1. rating on data with known allocation (supervised rating)
2. rating on data with unknown allocation (unsupervised rating)

They are introduced in the following.

6.6.1 Rating on Data With Known Allocation

If the true allocation of items is known, this knowledge can be used to evaluate how exact a clustering technique matches this true allocation.

Let A be the true allocation of the data and C be the solution computed by a clustering approach. To rate how good the clustering solution matches the true allocation, four values are defined [Gan et al., 2007]:

1. N_{11} is the number of pairs of items which are in the same cluster in C and in the same cluster in A
2. N_{10} is the number of pairs of items which are in the same cluster in C but in different clusters in A
3. N_{01} is the number of pairs of items which are in different clusters in C but in the same cluster in A
4. N_{00} is the number of pairs of items which are in different clusters in C and in different clusters in A

N is then $N = N_{11} + N_{10} + N_{01} + N_{00}$.

The *precision* of a clustering solution is defined as the ratio of items, a clustering technique has computed right [Tan et al., 2009]:

$$\text{precision} = \frac{N_{11}}{N_{11} + N_{10}} \tag{6.35}$$

The *recall* is defined as the ratio of items, the system matched against the items expected by the true allocation:

$$\text{recall} = \frac{N_{11}}{N_{11} + N_{01}} \tag{6.36}$$

As both measures are contrary (greater precision decreases recall and greater recall leads to decreased precision), the optimal values are subjective. For this reason, both values are often merged into one value, the F-measure. The F-measure is the harmonic mean of precision and recall:

$$F = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{6.37}$$

Precision, recall and F-measure rate a clustering solution based on how well the items have been classified (classification-centric measures). In addition, the similarity between the clustering solution and the true allocation can be measured (similarity-based measures).

Various indices exist, measuring the similarity between C and A. One of the indices used most commonly is the Rand statistic [Gan et al., 2007]:

$$R = \frac{N_{11} + N_{00}}{N} \in [0, 1] \tag{6.38}$$

The Rand statistic and similar indices express how similar the clustering solution and the true allocation are. Therefore, higher index values express a better fitting of the clustering solution to the true allocation.

The main problem of ratings on data with known allocation is that the true allocation of the data must be known. If the true allocation is not known, it can be created for an example set of the data: First an expert selects a subset of the data that should be representative for the complete set. He then clusters this subset manually according to his expert knowledge. His clustering solution is then defined to be the true allocation of the data. However, creating a true allocation for an example set manually might be time-consuming, as the example set must be large enough to represent the complete dataset. Therefore, ratings for data with unknown allocation are needed.

6.6.2 Rating on Data With Unknown Allocation

Rating on data with unknown allocation can only use internal information of the clustering solution. As the “ideal” clustering solution is defined to provide high *cohesion* within the clusters and low *coupling* between the clusters, these measures are often used to rate a clustering solution. The *Silhouette Coefficient* combines both measures [Tan et al., 2009]. Algorithm 5 on the following page describes how to compute the silhouette value for one single point. The average silhouette

Algorithm 5 Computing the silhouette value for one item

```
1: Calculate average distance between $p_i$ to all other items in the same cluster $C_x$:
       $a_i = \underset{p_j \in C_x}{\text{average}}\left(d(p_i, p_j)\right)$
                                                        ▷ cohesion
2: With $p_i \in C_x$, calculate for each cluster $C_y \neq C_x$ the average distances between $p_i$ and $p_j \in C_y$.
   Take the minimum value with respect to all clusters:
       $b_i = \min_{C_y \neq C_x}\left(\underset{p_j \in C_y}{\text{average}}\left(d(p_i, p_j)\right)\right)$
                                                        ▷ coupling
3: The silhouette value for $p_i$ is then:
       $\text{silhouette}(p_i) = \frac{b_i - a_i}{\max(a_i, b_i)}$
                                                        ▷ $\text{silhouette}(p_i) \in [-1, 1]$
```

value of all points of a cluster leads to the silhouette value of this cluster:

$$\text{silhouette}(C_x) = \underset{p_i \in C_x}{\text{average}}\left(\text{silhouette}(p_i)\right) \tag{6.39}$$

Similarly, the silhouette value of a complete clustering solution C (called silhouette coefficient) is computed:

$$\text{silhouette}(C) = \underset{C_x \in C}{\text{average}}\left(\text{silhouette}(C_x)\right) \tag{6.40}$$

Kaufman and Rousseeuw [2009] propose a subjective, but widely accepted interpretation of silhouette values (Table 6.4). This interpretation is based on the observation, that the silhouette value

Silhouette coefficient	**Interpretation**
0.71 – 1.00	A strong structure has been found
0.51 – 0.70	A reasonable structure has been found
0.26 – 0.50	The structure is weak and could be artificial; please try additional methods on this dataset
< 0.25	No substantial structure has been found

Table 6.4: Proposed interpretation of silhouette coefficient [Kaufman and Rousseeuw, 2009]

is near to 1 if a_i is small (high cohesion) and b_i is big (low coupling). I.e., the distances to items within their own cluster are small while distances to other clusters are high. The items are therefore "well-clustered". A silhouette value around zero means that distances within a cluster are similar to distances to other clusters. Therefore, items could belong to other clusters, as well. The solution is therefore weak. Values below zero indicate that distances to other clusters are even smaller than within clusters.

This concludes the introduction of clustering approaches. In Chapter 11, clustering approaches will be used to support the identification of legacy code able to implement services.

Chapter 7

Link Analysis With Association Rules

In this chapter, the identification of association rules is described. Definitions, an algorithm and evaluation criteria for association rules are provided.

In addition to finding groups in data (cluster analysis), a goal of data mining is to identify associations between items (link analysis). These associations can be expressed in form of *association rules*. Association rules are defined as follows [Agrawal et al., 1993]:

Definition 7.0.1 (*Association rule*)

Given a set of items $P = \{p_1, \dots, p_n\}$, an association rule is a rule of the form

$$A \rightarrow C \tag{7.1}$$

with $A \subseteq P$, $C \subseteq P$ and $A \cap C = \varnothing$.
A is called the antecedent *and C is called the* consequent *of the rule.*
The rule is the read as "if all items in A occur, then all items in C occur, too.

Association rules describe which items in a set of transactions appear together. A *transaction* is a set of joint items, e.g., items that have been bought in one purchase. Then the whole purchase is one transaction and the goods which have been bought together are the items in this transaction.

Definition 7.0.2 (*Transaction*)

Let $T = \{t_1, \dots, t_n\}$ be a set of transactions. Each transaction t_i has a unique transaction id and contains a subset of items $t_i \subseteq P$.

Example: Association rules are often used in *recommender systems* of online shops. E.g., if a rule $notebook \rightarrow \{keyboard, mouse\}$ exists, customers who are thinking about buying a notebook will get a hint that "customers who bought a notebook also bought a keyboard and a mouse".

As association rule algorithms often compute a high number of rules, quality measures have been introduced to identify "good" rules. They are described in the following.

7.1 Quality Measures for Association Rules

Quality measures for association rules mainly rate how often the antecedents or consequents of a rule exist in the dataset [Lallich et al., 2007]. Three measures are often used to rate the quality of an association rule:

1. Support
2. Confidence
3. Lift

They are defined in the following.

7.1.1 Support, Confidence and Lift

Let $P = \{p_1, \dots, p_n\}$ be a set of items and $T = \{t_1, \dots, t_m\}$ with $t_i \subseteq P$ be the set of transactions. In addition, let $A \rightarrow C$ be an association rule as defined in Definition 7.0.1 on the preceding page.

Definition 7.1.1 (support(B))

The support *of a set of items $B \subseteq P$ is defined as the* relative frequency of B to T *[Möhring, 2009]:*

$$\text{support}(B) = \frac{|\, \{t_i \in T \mid B \subseteq t_i\} \,|}{|\, T \,|} \tag{7.2}$$

Definition 7.1.2 (support($A \rightarrow C$))

The support for an association rule is then defined as the joint *occurrences of A and C to T:*

$$\text{support}(A \rightarrow C) = \frac{|\, \{t_i \in T \mid (A \cup C) \subseteq t_i\} \,|}{|\, T \,|} \tag{7.3}$$

Summarizing, the support of an association rule describes in how many transactions the antecedent and the consequent occur together compared to the number of *all transactions*.

In contrast, the *confidence* of an association rule is defined as the frequency of joined occurrences of A and C related to the *antecedent* A:

Definition 7.1.3 (confidence($A \rightarrow C$))

$$\text{confidence}(A \rightarrow C) = \frac{|\, \{t_i \in T \mid (A \cup C) \subseteq t_i\} \,|}{|\, \{t_i \in T \mid A \subseteq t_i\} \,|} \tag{7.4}$$

$$= \frac{\text{support}(A \rightarrow C)}{\text{support}(A)} \tag{7.5}$$

Summarizing, the confidence of an association rule describes in how many transactions A and C occur together if A occurs.

The *lift* of an association rule compares the confidence of the rule to a confidence assuming that the antecedent and consequent are not correlated.

Definition 7.1.4 (lift$(A \to C)$)

If A and C were not correlated to each other, the support is computed as follows:

$$\text{support}*(A \to C) = \text{support}(A) * \text{support}(C) \tag{7.6}$$

Then, the confidence is computed as:

$$\begin{aligned} \text{confidence}*(A \to C) &= \frac{\text{support}(A) * \text{support}(C)}{\text{support}(A)} \\ &= \text{support}(C) \end{aligned} \tag{7.7}$$

The lift of the association rule is then:

$$\text{lift}(A \to C) = \frac{\text{confidence}(A \to C)}{\text{confidence}*(A \to C)} \tag{7.8}$$

$$= \frac{\text{confidence}(A \to C)}{\text{support}(C)} \tag{7.9}$$

A lift value of 1 means, that the confidence of the rule is equal to a confidence computed for the antecedent and consequent being independent. That is, the confidence is not better than the rule being randomly put together. Lift values greater 1 indicate, that the confidence of the rule is better than it would be for a random rule. Lift values smaller than 1 indicate that the rule is even worse than for a random rule.

These measures are used in algorithms to identify "interesting" rules. Given arbitrary threshold values, only rules with support, confidence or lift values greater than the thresholds are accepted as interesting rules.

One of the oldest algorithms to identify interesting association rules that is still used today is introduced in the following section.

7.2 The Apriori Approach

One of the oldest but still used link analysis approaches is the Apriori approach. The Apriori approach [Agrawal and Srikant, 1994] is based on the identification of *frequent itemsets*.

Definition 7.2.1 (*Frequent itemset*)

Let $P = \{p_1, \dots, p_n\}$ be the set of all distinct items occurring in the complete dataset and let the itemset Q be a subset of P ($Q \subseteq P$).
With ε being a user-defined threshold value, Q is called frequent itemset, *if $measure(Q) > \varepsilon$. The*

measure can be any measure to rate the quality of a set of items (Section 7.1).

Agrawal and Srikant [1994] used in their original approach the support as measure. Then, an itemset Q is called frequent if:

$$\text{support}(Q) > \varepsilon_{sup} \tag{7.10}$$

The basic idea of the Apriori approach is to compute all possible itemsets for P and to mark those itemsets with a support greater ε_{sup} as frequent. As the number of possible itemsets quickly reaches huge dimensions, pruning techniques are needed that are able to reduce the number of itemsets for which the support is calculated.

As pruning technique, the Apriori approach uses the *apriori property*.

Definition 7.2.2 (*Apriori property*)

Each subset of a frequent itemset is a frequent itemset, too [Agrawal and Srikant, 1994].

That means that an itemset with a non-frequent subset is non-frequent, too. In addition, a frequent itemset can be generated only out of a smaller frequent itemset [Möhring, 2009].

This characteristic is used by the Apriori approach to prune most of the uninteresting itemsets.

Today, various algorithms based on the apriori idea exist. In the following, the original algorithm proposed by Agrawal et al. [1993] is described.

7.2.1 The Apriori Algorithm

The Apriori algorithm has two stages:

1. Compute frequent itemsets
2. Create association rules

They are described in the following.

Stage 1 – Compute Frequent Itemsets

The first stage of the Apriori algorithm is to compute all frequent itemsets. The Apriori property is used here for more efficient computation of the frequent itemsets.

Let L_k be the frequent itemset with $|L_k| = k$. L_k contains all itemsets Q_i with $\text{support}(Q_i) > \varepsilon_{sup}$.

Algorithm 6 on the next page shows the pseudocode of the Apriori algorithm. It is described in the following [Han and Kamber, 2004].

In the first step of the algorithm, L_1 is initialized as the set of *frequent items*. An item p_i is called frequent if $\text{support}(p_i) > \varepsilon_{sup}$.

Algorithm 6 Apriori algorithm: Stage 1 – Compute frequent itemsets

```
Input: Dataset $D$, threshold value $\varepsilon_{sup}$
Output: Set of frequent itemsets $L = \{L_1, \dots, L_m\}$
 1: $k = 1$
 2: $L = \{\}$
 3: $L_1$ = set of frequent items
 4: repeat
 5:     $k = k + 1$
 6:     $C_k = L_{k-1} \bowtie L_{k-1}$                    ▷ Join step
 7:     $L_k = \{\}$
 8:     for all Itemset $M \in C_k$ do
 9:         if $\forall$ $(k-1)$-itemsets $S_i \subset M$ : support$(S) > \varepsilon_{sup}$ then $L_k = L_k \cup S_i$
10:         end if
11:     end for                                          ▷ Prune step
12:     $L = L \cup L_k$
13: until $L$ does not change anymore
```

The remaining algorithm is an iteration of *join* and *prune* steps as described in the following.

To find the frequent k-itemset L_k, a candidate k-itemset C_k is computed by *joining* L_{k-1} with itself: $L_{k-1} \bowtie L_{k-1}$ (*join* step).[1]

In the Apriori algorithm, two itemsets are allowed to be joined (are “joinable”) if the following condition is met: For comparison, all items of an itemset are compared in lexical ascending order. Then, two itemsets Q_i and Q_j are *joinable*, if their first $k-2$ items are equal. I.e., the join is applicable if:

$$Q_i[1] = Q_j[1] \wedge Q_i[2] = Q_j[2] \wedge \dots \wedge Q_i[k-2] = Q_j[k-2] \tag{7.11}$$

During the *prune* step, all non-frequent itemsets are removed from C_k. C_k is a superset of L_k containing all frequent k-itemsets but also containing non-frequent k-itemsets. Computing the support values for all itemsets in C_k to determine which itemsets are frequent would result in big computational effort. Therefore, the apriori property is used, to identify which itemsets are frequent: If a k-itemset $Q_i \in C_k$ contains at least one sub-itemset of the length $k-1$ that is non-frequent (i.e., it is not contained in L_{k-1}), then the whole k-itemset Q_i is non-frequent, too. It is then removed from C_k. After having checked all itemsets of C_k, C_k only contains frequent itemsets and is then defined to be L_k. The k-itemset L_k is then added to the set of all frequent itemsets L.

The iteration ends when no new frequent itemsets are added during a complete iteration (i.e., L does not change anymore).

[1]The Apriori approach was originally developed for mining association rules from relational databases. Therefore, the join is defined as in relational algebra.

Stage 2 – Computing Association Rules

During the second stage of the Apriori algorithm, the frequent itemsets are used to compute "strong" association rules. in the original algorithm, strong was defined as a confidence value greater than a given threshold (confidence$(A \rightarrow C) > \varepsilon_{conf}$).

Given the set of frequent itemsets L computed during stage 1, association rules are formed as follows (Algorithm 7).

Algorithm 7 Apriori algorithm: Stage 2 – Compute association rules

```
Input: Set of frequent itemsets $L = \{L_1, \ldots, L_m\}$, threshold value $\varepsilon_{conf}$
Output: Set of strong association rules $R$
1: R = { }
2: for all $L_i \in L$ do
3:     Generate all non-empty subsets $S = \{S_1, \ldots, S_n\}$ with $S_j \subseteq L_i$
4:     for all $S_j \in S$ do
5:         if confidence$(S_j \rightarrow (L_i - S_j)) > \varepsilon_{conf}$ then $R = R \cup \{S_j \rightarrow (L_i - S_j)\}$
6:         end if
7:     end for
8: end for
```

For each frequent itemset $L_i \in L$, all non-empty subsets $S_j \subseteq L_i$ are generated. For each of these subsets, check if a rule $S_j \rightarrow (L_i - S_j)$ has a higher confidence than the threshold:

$$\text{confidence}(S_j \rightarrow (L_i - S_j)) > \varepsilon_{conf} \tag{7.12}$$

All rules meeting this criterion are returned as strong association rules.

This concludes the introduction of link analysis with association rules. In Chapter 12, association rules will be used to support the identification of legacy code able to implement services.

Chapter 8

Data Mining Tool: PASW

In this chapter, the data mining tool PASW which has been used in this thesis is introduced briefly.

To be able to apply the techniques that have been introduced in Chapter 6 and Chapter 7, a tool is needed. In this thesis, IBM's PASW Modeler 14 has been used [SPSS, 2011].

PASW Modeler is a data mining tool specialized on mining useful information from large datasets. PASW provides a graphical user interface and allows to model data mining tasks in a visual way.

Figure 8.1 on the following page shows the main window of PASW.

The central part of the PASW GUI is the *data mining stream*. On this stream, all data mining tasks are modeled visually. In PASW, data mining tasks are provided as nodes. Tasks are grouped into seven categories:

Data sources. Load datasets in various formats (e.g., SQL or CSV)

Data operations. Select, aggregate or sort items of a dataset. Join multiple data sources

Fiel operations. Manipulate items (e.g., filter, rename, change level of measurement, manipulate attributes, create new attributes)

Diagrams. Visualize items in various diagrams (e.g., box plots histograms or radar charts)

Modeling. Create data mining models for classification, segmentation and link analysis (e.g., *k*-Means, TwoStep or Apriori-based association rules)

Output. Visualize data (e.g., as table or matrix) and compute basic statistics (e.g., mean, median or standard deviation)

Export. Export data into various formats (e.g., SQL or CSV)

On PASW's stream window, single data mining tasks of these categories are assembled. The task nodes are connected by links, representing the data flow between each task.

Example : In Figure 8.1 on the next page, a file is imported on the left, using the *import node*. The import node specifies data format and the level of measurement of the attributes. The imported dataset is forwarded to various nodes using the dataset as input.

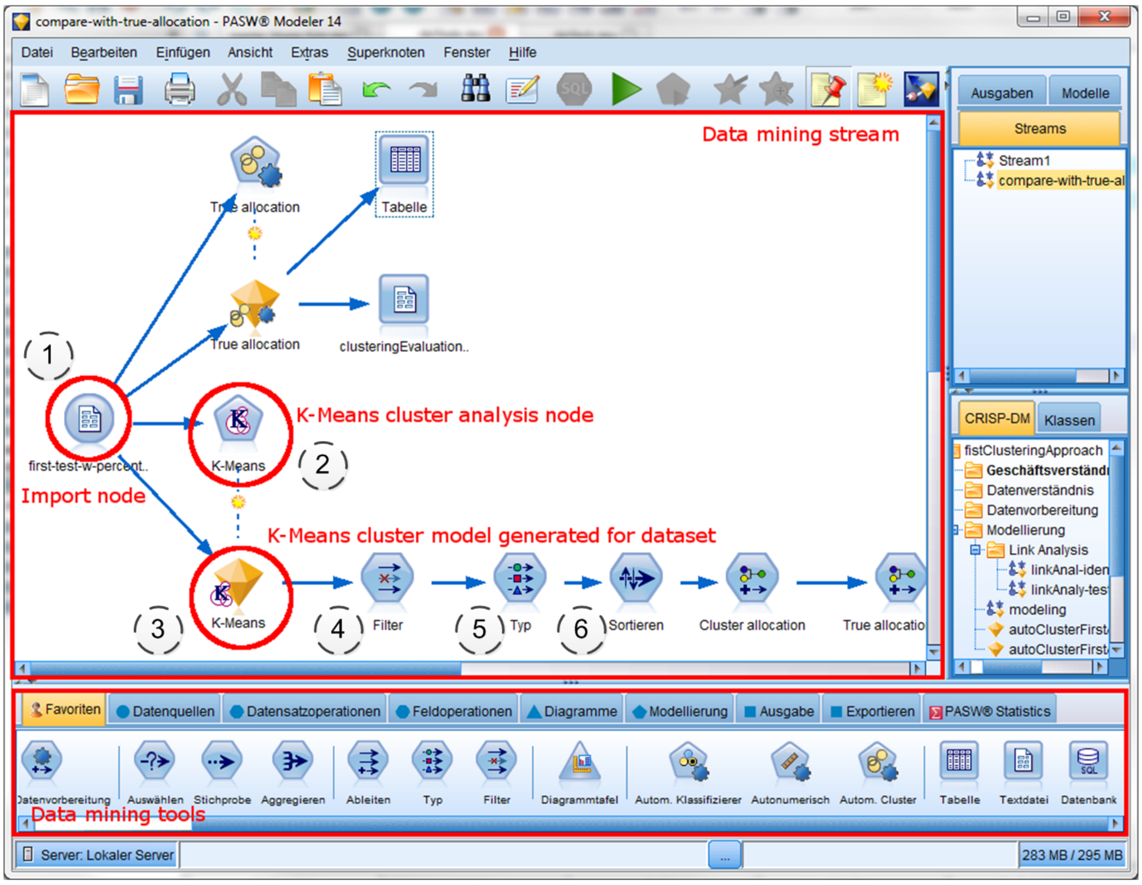


Figure 8.1: PASW workbench

Modeling nodes (e.g., the *K-Means cluster analysis node*) are used to adapt the parameters of a model. They are used to generate a model with the given parameters. This model is represented as a "nugget". In Figure 8.1 on the facing page, the *K-Means cluster analysis node* generated a *k*-Means model represented by *K-Means cluster model nugget* below the *K-Means cluster analysis node*.

The following example describes how to assemble single data mining tasks to cluster an post-process a dataset.

Example : In Figure 8.1 on the preceding page, a file (CSV format) is imported with the *import node* ①.

The dataset if forwarded to the *K-Means cluster analysis node* ②. In this node, parameters of the *k*-Means clustering algorithm are specified. A *k*-Means nugget is created, representing the *k*-Means clustering algorithm with the given parameters ③. The dataset is forwarded to this nugget and clustered by the *k*-Means algorithm. The output of the nugget ③ is the clustered dataset (the clustering solution is represented by an additional attribute containing the number of the cluster, an item belongs to).

The clustered dataset is forwarded to the *Filter node*. In the filter node ④, some uninteresting attributes are filtered out. The remaining dataset (without the filtered attributes) is forwarded to the *Typ* node. In the Typ node ⑤ a data format and a level of measurement for the cluster-solution attribute is selected.

As final step of this example[1], the dataset is forwarded to the *Sortieren* node where the dataset is sorted according to the natural order of the cluster-solution attribute.

By connecting single data mining tasks, complex data mining scenarios may be modeled in PASW.

Further information about using PASW to identify legacy code for service identification is provided in Chapter 11 and Chapter 12. In the following, the application of data mining techniques to identify legacy code for service implementation is explained.

[1]the remaining steps are not described here anymore.

Part III

Service Realization by Legacy Code

Chapter 9

Running Example

In this chapter, a guinea pig legacy system is introduced that was used to explore the possibilities of data mining techniques in identifying legacy code for service implementation. A business process for this legacy system is described.

In order to analyze how data mining techniques can be used to identify legacy code for service implementations, an example in the context of the SOAMIG project has been chosen.

The SOAMIG project[1] addresses semi-automatic migrations of legacy software systems towards Service-Oriented Architectures, based on model-driven techniques and code transformation. The following scenario that is used as example for this thesis, has been derived from one of the work-packages of SOAMIG.

9.1 Scenario

As part of the SOAMIG project, one task was to identify and to initially implement a set of exemplary services by extracting code from the legacy system. During the SOAMIG project, the exploration and extraction was supported by the *SoamigExtractor* tool [Erdmenger et al., 2011a].

The SoamigExtractor is a Java/Swing-based tool, visualizing the package and inheritance structure of a given legacy system. In both views, classes, fields and methods are shown. For methods, the call hierarchy can be navigated along, i.e., methods calling the selected method or methods the selected method is calling can be navigated to. Methods that should be extracted to the target architecture (a service implementation) are selected. The tool automatically calculates all required methods, classes and fields of the selected methods and includes them in the extraction process. Finally, the selected methods are extracted and stored as SOAMIG XML interchange format. Additional tools can then generate Java code from the XML files.

In the following, the business process to extract legacy code is introduced.

[1]http://www.soamig.de

9.2 Business Modeling

For this example, one business process has been identified: The extraction of legacy source code in order to implement a service (Figure 9.1 on the facing page).

The EXTRACTION business process starts with a request for extracting legacy code to implement a given service. The service developer working on the request first loads the TGraph representation of the legacy source code from the SOAMIG repository into the SoamigExtractor tool.

Next, he uses the SoamigExtractor to identify and select elements that should be extracted from the legacy system to implement the service. This step results in a selection of elements that should be extracted.

Optionally, he can decide to refactor some or all of the selected elements, then. The result is a selection of the refactored and unrefactored elements.

After the refactorings, the developer may decide to store the refactorings persistently in the SOAMIG TGraph. For this purpose, he can save the graph containing the refactored elements back to the SOAMIG repository.

Finally, all selected elements are exported into XML files. These files are then handed back to the requester who can use a different tool to generate Java code from the XML files.

In the following chapters, it is described how to identify legacy code that is able to implement services supporting the EXTRACTION business process.

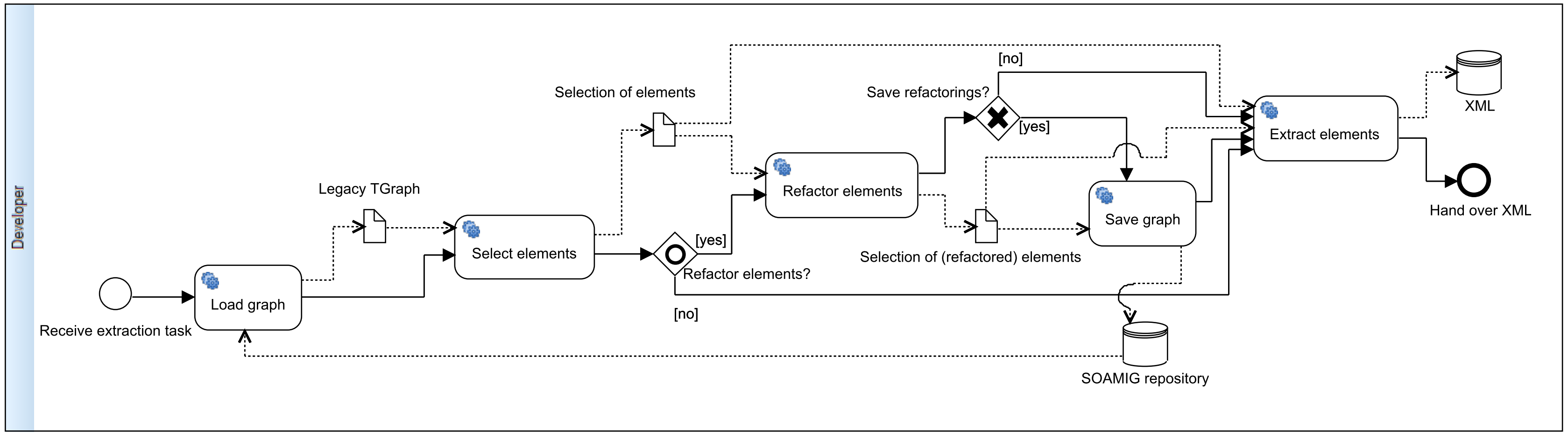


Figure 9.1: Exemplary business process: Extraction of legacy code for service implementation (BPMN Notation)

Chapter 10

Dynamic Analysis

In this chapter, dynamic analysis is introduced. It is motivated how business processes can be used for dynamic analysis scenarios. A tool-chain is described to execute the dynamic analysis.

Whereas static analysis techniques focus on analyzing legacy source code files, dynamic analysis techniques focus on analyzing properties of a *running* software system [Cornelissen, 2009]. For tracing the execution of a running system, the system must be extended to log which code it executes. One way not changing the overall behavior of the system is to instrument the system under observation using aspects (e.g., AspectJ). These aspects can be programmed to log which code is executed, then.[1]

In addition, a scenario is defined describing how to interact with the running system so it triggers all code of interest. To ensure completeness of the analysis results, it is important to ensure that all parts of interest in the code are really executed during a scenario.

In the following section, the definition of such scenarios is described in the context of identifying legacy code able to implement services.

10.1 Using Business Processes for Dynamic Analysis

As described, dynamic analysis traces the execution of code during a given scenario. The selection of appropriate scenarios influences the results of such an analysis.

For SOA migrations, interesting scenarios are provided by the business processes supported by a legacy system. As SOAs tightly relate code to business processes, tracing which legacy code is executed during what step of a process can be used for service implementation.

However, the business model provides a business-centric view on activities. In order to capture all legacy code that is executed during a business process, a more detailed description on how to use the legacy system during dynamic analysis is needed. Such a detailed description is given by the *workflow model*.

[1]As the logging of all code executions of a system generates a large amount of data, additional analyses are needed to evaluate the trace logs. For this reason, data mining techniques are applied to identify code for service implementation.

10.1.1 Workflow Models: An Implementation-Centric View on Business Processes

Figure 10.1 shows the relation of the business process model to the workflow model.

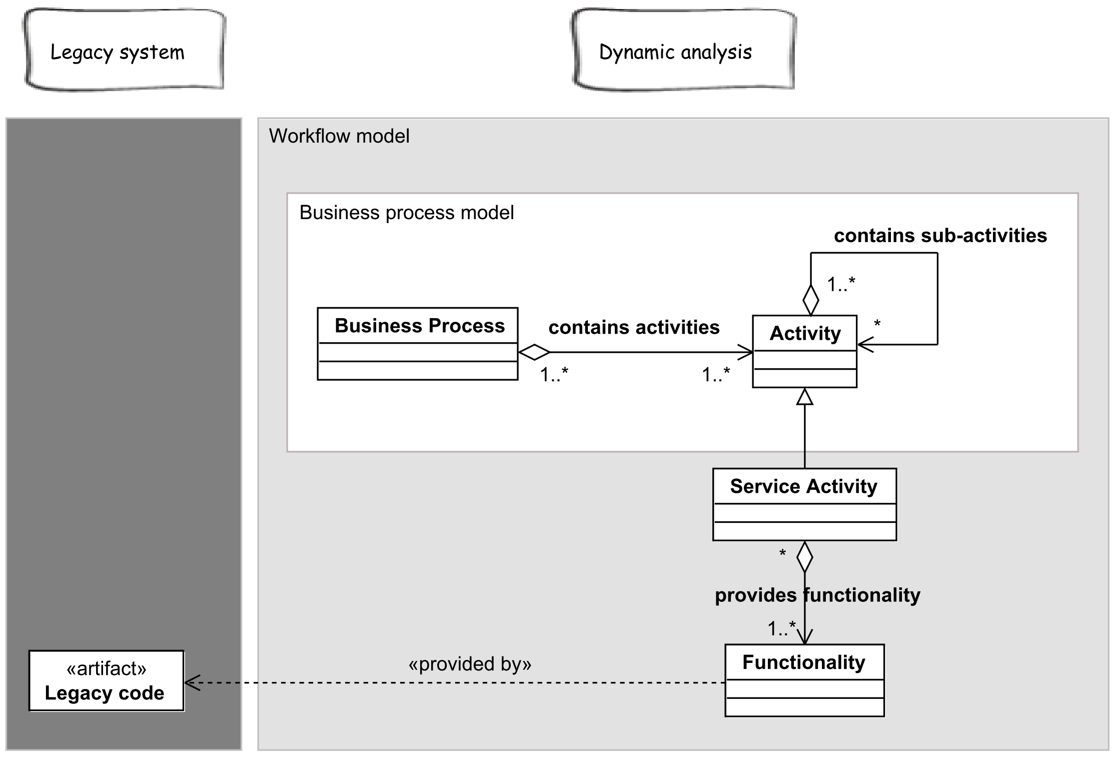


Figure 10.1: Relation of business model to workflow model

The workflow model is derived from the business process model. Therefore, it describes what activities are executed during a business process. In addition, the workflow model adds information about what *functionality* of the legacy system is required during this activity.

During dynamic analysis, it is traced which activity is executed. The details about functionality are used to give advice which functionality of the legacy system to execute during the dynamic analysis. This detailed advice ensures that no functionality of the legacy system is missed during dynamic analysis.

The following section introduces the workflow model that has been designed for the EXTRACTION business process.

10.2 Workflow Model for Extraction Business Process

For the workflow model, the business process model designed in Figure 9.1 on page 65 is enhanced by adding information about what functionality (stereotype `<<Functionality>>`) is executed during each activity.

Figure 10.2 shows the workflow model for the EXTRACTION business process that is supported by the SoamigExtractor tool.

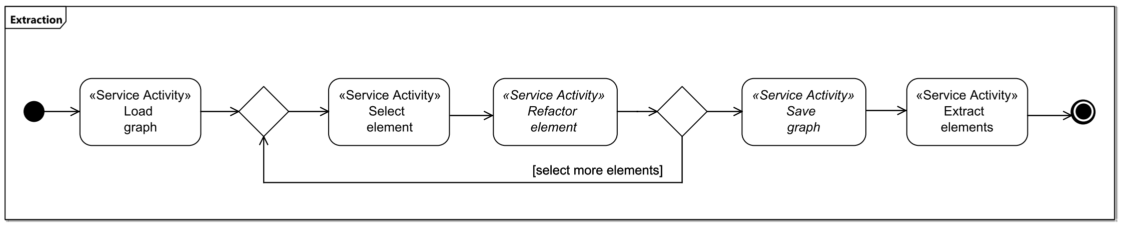

Extraction
«Service Activity» Load graph
«Service Activity» Select element
«Service Activity» Refactor element
«Service Activity» Save graph
«Service Activity» Extract elements
[select more elements]

Figure 10.2: Exemplary business process: Extraction of legacy code for service implementation

The first step is to load the TGraph representing the legacy system. The TGraph may be uncompressed as `.tg` file or gzip-compressed as `.tg.gz` file. Both formats can be handled by the SoamigExtractor (Figure 10.3).

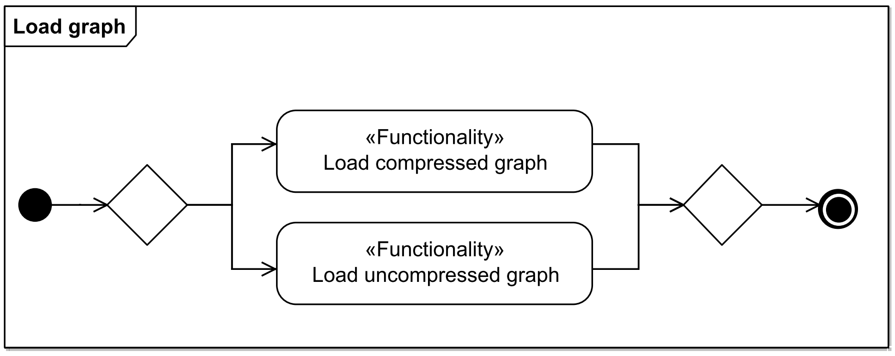

Load graph
«Functionality» Load compressed graph
«Functionality» Load uncompressed graph

Figure 10.3: Details of `load graph` functionality

After loading the graph, the core activity of this business process is executed: the selection of elements that should be extracted from the legacy system to implement a service. A user has three choices, how to select elements (Figure 10.4 on the next page):

1. Select single elements by hand (SELECT SINGLE ELEMENTS)
2. Select all elements that have been called during a workflow (SELECT BY WORKFLOW)
3. Select all elements that have been called during an activity of a business process (SELECT BY ACTIVITY)

All three techniques can be used to select elements that should be extracted. After having selected an element, various refactorings can be performed (Figure 10.5 on the following page):

- Move class to other package (COPYMETHODTYPE)
- Copy method to other class (MOVECLASSTYPE)
- Remove superclass of class (REMOVESUPERCLASS)

The process of selecting and refactoring elements is iterative. It ends when all desired elements have been selected. After having selected a set of elements that should be extracted from legacy

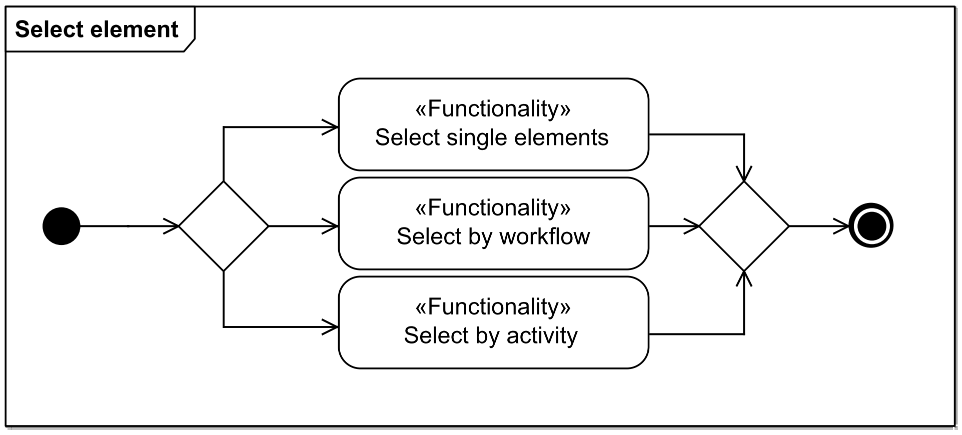


Figure 10.4: Details for activity `Select element`

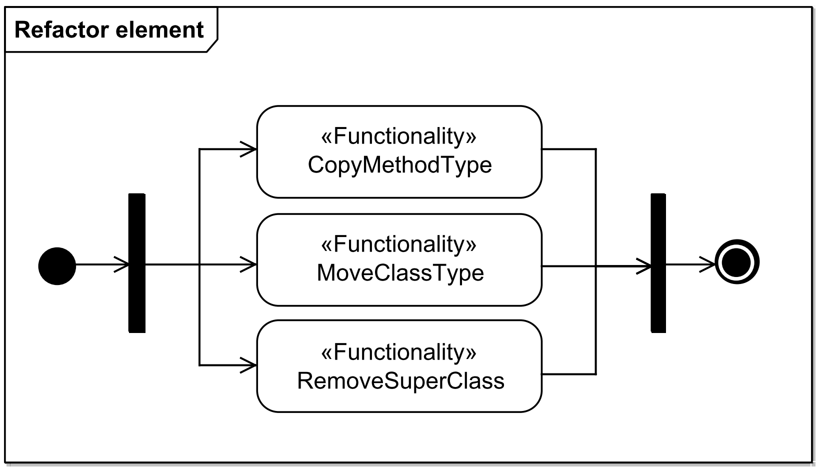


Figure 10.5: Details for activity `Refactor element`

code, the user may optionally store the graph containing the refactorings back to a graph file.

Finally, the selected elements are exported in XML files conforming to the SOAMIG exchange format.

This detailed workflow can now be used to perform the dynamic analysis. Although the workflows are ordered in sequence, this sequence should not be understood as strict ordering. The ordering only gives hints to a user, how to navigate best through the activities during dynamic analysis.

In the following section, the tool set-up of the dynamic analysis is described.

10.3 Set-Up for the Dynamic Analysis

During dynamic analysis, the following tasks are executed:

1. Trace the progress of the analysis scenario (which activity of the business process is executed when)
2. Log code executions of the legacy system (which code is executed when)
3. Store both pieces of information synchronously
4. Create trace links between legacy code and the activities of business processes

Figure 10.6 shows the tool set-up developed during the SOAMIG project to support these tasks [Fuhr et al., 2010].

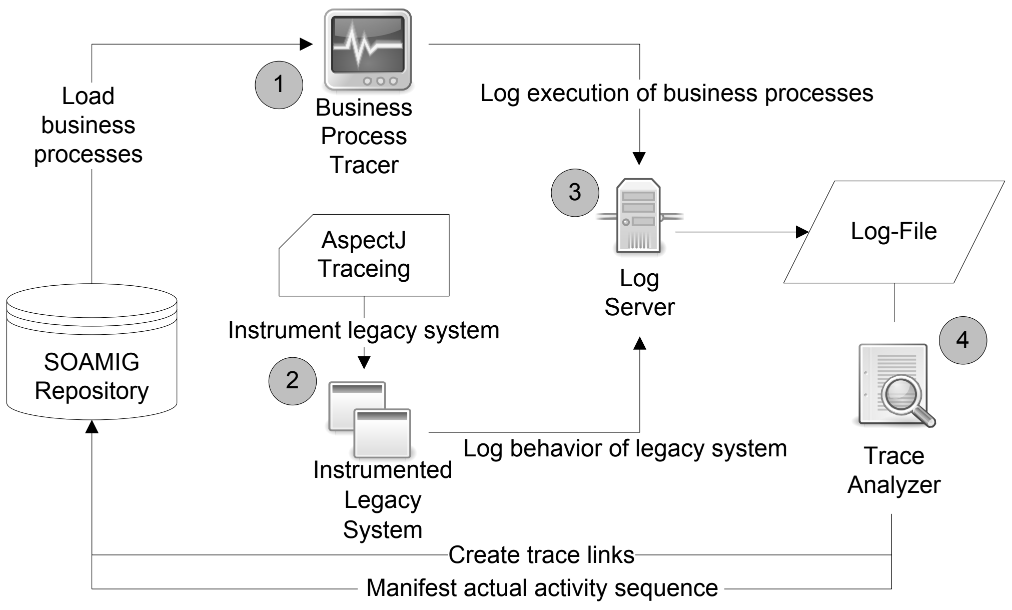
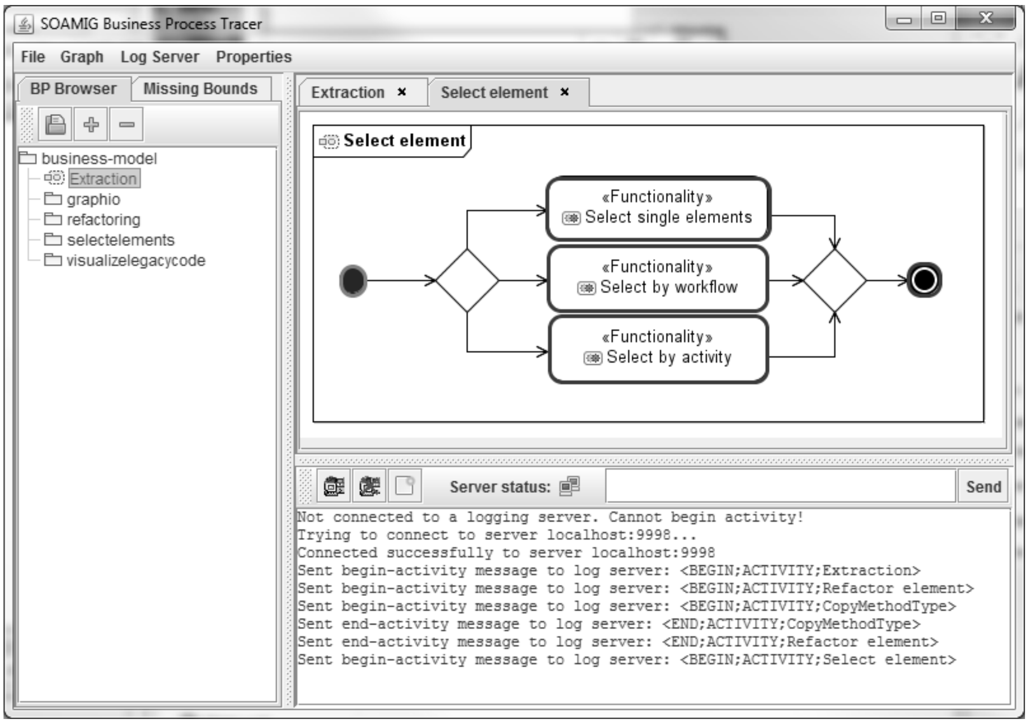


Figure 10.6: Set-up of the dynamic analysis

The Business Process Tracer. The Business Process Tracer ① is a graphical user interface developed during the SOAMIG project. The BPT loads the story workflows from the SOAMIG Repository and visualizes them as shown in Figure 10.7.



```
Not connected to a logging server. Cannot begin activity!
Trying to connect to server localhost:9998...
Connected successfully to server localhost:9998
Sent begin-activity message to log server: <BEGIN;ACTIVITY;Extraction>
Sent begin-activity message to log server: <BEGIN;ACTIVITY;Refactor element>
Sent begin-activity message to log server: <BEGIN;ACTIVITY;CopyMethodType>
Sent end-activity message to log server: <END;ACTIVITY;CopyMethodType>
Sent end-activity message to log server: <END;ACTIVITY;Refactor element>
Sent begin-activity message to log server: <BEGIN;ACTIVITY;Select element>
```

Figure 10.7: The Business Process Tracer tool

During the dynamic analysis, a user operates the tracer in parallel to the legacy system. In the tracer, an activity is selected and activated and then the user executes the functionality on the legacy system by using its user interface.

Whenever an activity is activated or deactivated in the tracer, a message containing a timestamp and that activity's name is sent to the Log Server.

Experiences in the SOAMIG project have shown that each business process can be executed in multiple ways when using the user interface of the legacy system. Therefore, the tracer does not enforce that the order in which activities are activated, matches the order in the idealized workflow model.

The Instrumented Legacy System. The legacy system is instrumented using AspectJ ②. The aspect defines pointcuts for the execution of methods and constructors. Before and after each pointcut, a message is sent to the Log Server. This message contains a timestamp, the name of the current thread and the qualified name and signature of the method or constructor that has been invoked.

The Log Server. The Log Server ③ receives the messages sent from the Business Process Tracer and the instrumented legacy system and stores them in log files. The Log Server provides a synchronization mechanism which guarantees the correct order of the logged messages, even though the legacy system and the Business Process Tracer are running on different machines with clocks ticking asynchronously.

The Trace Analyzer. So far, all components of the dynamic analysis framework dealt with recording the sequence of activity (de)activations and the corresponding method and constructor calls.

The Trace Analyzer's ④ job is to parse the log files and to create trace links between activities and methods. The trace links are then stored back to the repository.

The Trace Analyzer iterates over the log file entries and from the `begin activitiy` and `end activity` entries, it determines the activity in the business process model, which was executed at that time. Between the `begin` and `end` entry for an activity, there are method and constructor calls, which were executed in the legacy system while executing this activity. Therefore, a traceability link is created, connecting the activity in the business process model to the definitions of the called methods in the source code model. In case of constructor calls, the link leads to the class definitions from which objects are instantiated.

Additionally, the actual sequence of executed activities is integrated into the business process model by creating special edges between activities. Each path through the graph, consisting of activities and those edges represents one concrete instances of a business process (a *workflow*).

After integrating the static business process model with the static source code model by manifesting dynamically gathered information as edges in the repository, additional analyses can exploit these enhancements.

The remainder of this thesis describes how to use data mining techniques on the integrated model in order to identify legacy code that is able to implement services.

Chapter 11

Using Cluster Analysis to Identify Legacy Code for Service Implementation

In this chapter, clustering techniques are used to identify legacy code that is able to implement services. Two clustering approaches using different input data are presented and compared.

Using clustering techniques to identify code to implement services, clustering algorithms must divide legacy code into homogeneous groups so that each cluster can form a service.

The clustering approaches have to discriminate between

1. services that support exactly one activity (business services),
2. services that support multiple activities, i.e., they are used by other services (support services) and
3. services that provide general functionality to all other services, e.g., logging mechanisms (helper services).

In order to structure the process of identifying legacy code able to implement services by using data mining techniques, the CRISP data mining process (Section 5.1) has partially been applied. For each data mining approach presented in the remaining chapter, the following steps of CRISP are described:

1. Data preparation
2. Modeling
3. Evaluation

In the following, two clustering approaches are presented and compared to each other.

11.1 First Approach: Clustering Classes According to Usage in Activities

The first approach was to cluster classes of the SoamigExtractor according to how often they have been used in the activities of the example business process.

Example : The class `de.soamig.extractor.operations.refactorings.Refactoring` has been used during the activities as follows:

- 3% in activity Select element
- 97% in activity Refactor element

The following hypotheses have been established for the clustering approaches.

Hypothesis 11.1.1 (*Business services*)
Classes that are (almost) exclusively used in one activity implement business functionality for this activity.

Hypothesis 11.1.2 (*Support services*)
Classes that are used (almost) even in the same sub-set of activities belong to a support service used in these activities.

Hypothesis 11.1.3 (*Helper services*)
Classes that are used in all activities equally, belong to helper services providing cross-cutting concerns (e.g., logging).

These three types of services are visually exemplified in Figure 11.1.

Qualified name	% Activity 1	% Activity 3	% Activity 2
`de.example.Class1`	0,99	0,01	0
`de.example.Class2`	0,97	0,01	0,02
`de.example.Class3`	0,58	0,41	0,01
`de.example.Class4`	0,41	0,59	0
`de.example.Class5`	0,34	0,33	0,33

Figure 11.1: Visual clustering according to usage. The blue box is a business service, the red box is a composite service and the green box is a helper service

`Class1` and `Class2` are almost exclusively used in Activity 1. Therefore, they seem to implement business functionality that is only used in this activity. So, both classes should be put into a service supporting Activity 1. `Class3` and `Class4` are used almost evenly in Activity 1 and Activity 3. So they might be a support service supporting these two activities. Finally, `Class5` is used equally in all three activities and therefore seems to be a cross-cutting helper service.

For these types of services, legacy code able to implement them will be identified by the clustering approach as described in the following.

11.1.1 Data Preparation

To be able to apply PASW's clustering abilities, data must be exported from the TGraph that had been created during dynamic analysis into CSV format. For this purpose, a Java program has been written that extracts all necessary information and stores them as comma-separated values.

In this first approach, for each class of the SoamigExtractor, it has been computed how often the class has been used in all activities. As first try, absolute values have been used. However, it turned out that absolute values lead to non-interpretable clustering solutions as they are not standardized. Therefore, usage has been computed in percentage values.

To be able to evaluate the clustering solutions, an expert created an ideal clustering solution (called "true allocation"). According to this solution, each class in the data set has been allocated to one of the following seven clusters of the true allocation:

1. load: The cluster representing a service to load graphs
2. save: The cluster representing a service to save graphs
3. extract: The cluster representing a service to extract elements as XML files
4. select: The cluster representing a service to select elements
5. refactor: The cluster representing a service to refactor elements
6. data: The cluster representing a helper-service for data access
7. gui: The cluster representing a helper-service supporting visualization and the GUI

During Evaluation, this true allocation will be used to evaluate the quality of the clustering solutions.

Summarizing, the data set used in this first approach has the following structure:

- `Qualified name`: The qualified class name
- `PER_Select element`: Usage of the class in the activity SELECT ELEMENT in percentage
- `PER_Extract elements`: Usage of the class in the activity EXTRACT ELEMENTS in percentage
- `PER_Save graph`: Usage of the class in the activity SAVE GRAPH in percentage
- `PER_Load graph`: Usage of the class in the activity LOAD GRAPH in percentage
- `PER_Refactor element`: Usage of the class in the activity REFACTOR ELEMENT in percentage
- `True allocation`: The "true cluster" the class belongs to, defined by an expert

11.1.2 Modeling

Figure 11.2 on the following page shows the modeling of the clustering approach in PASW.

Node ① is the import node loading the csv-file that has been described in Section 11.1.1. Node ② is the "Data Audit" node, providing general statistics about the variables of the data set. Node ③ is the "Auto Clustering" node. This node is used to simultaneously generate, explore and rate multiple clustering solutions. In this node, clustering algorithms, settings and evaluation criteria can be selected.

As clustering algorithms, *TwoStep* (an hierarchical-agglomerative approach, see Section 6.5.1) and *k-Means* (a partitioning approach, see Section 6.5.2) have been selected. The settings for both algorithms were as follows.

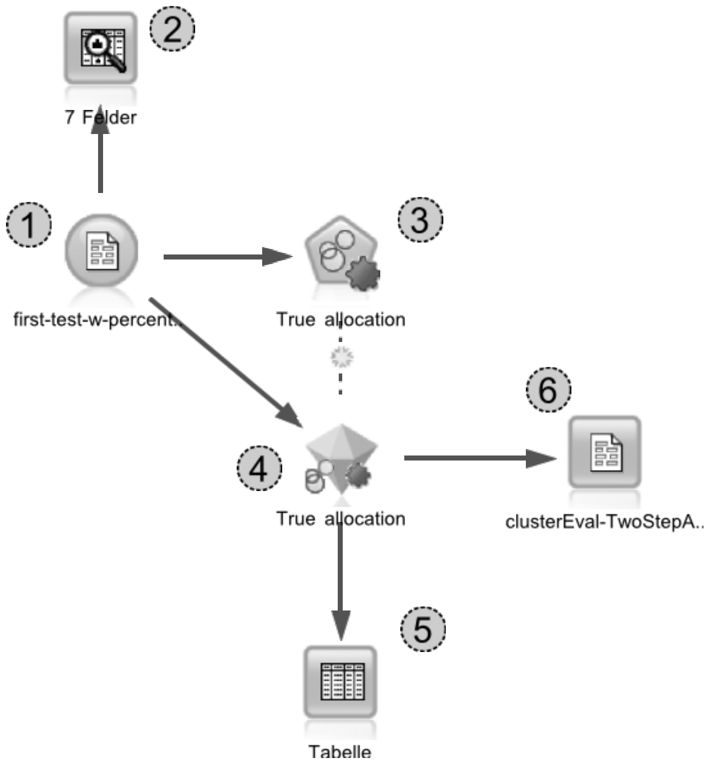


Figure 11.2: Modeling of the clustering in PASW

Settings for TwoStep were:

- Standardize numeric fields
- Number of clusters: automatically computed and fixed
  - Automatically computed: between 2 and 15 clusters
  - Fixed number of clusters: $2,3,4,5,6,7,8$
- Distance measure: Euclidean distance and Log-Likelihood
- Clustering criterion: Bayes' Information Criterion (BIC) and Akaike's Information Criterion (AIC)

Settings for *k*-Means were:

- Fixed number of clusters: $2,3,4,5,6,7,8$

As first evaluation criterion, the silhouette value (see Section 6.6.2) has been used. Solutions with a silhouette value smaller than $0,71$ have been dismissed.

Executing the Auto Clustering node generates node ④ containing all clustering solutions meeting the evaluation criterion (silhouette bigger than $0,71$). PASW generated five solutions meeting the initial evaluation criterion. They have been explored and evaluated during the Evaluation phase as described in the following.

File	Rand Statistic	Jaccard	Precision	Recall	F-Measure	Mean of Coefficients
first-approach-TwoStep-auto-clusters.csv	0,731	0,424	0,430	0,965	0,595	**0,629**
first-approach-k-Means-6clusters.csv	0,806	0,435	0,519	0,730	0,606	**0,619**
first-approach-k-Means-8clusters.csv	0,824	0,434	0,559	0,661	0,606	**0,617**
first-approach-k-Means-7clusters.csv	0,816	0,425	0,543	0,661	0,596	**0,608**
first-approach-k-Means-5clusters.csv	0,711	0,370	0,401	0,826	0,540	**0,570**

Table 11.1: Evaluation coefficients for the five clustering solutions that have been computed by PASW

11.1.3 Evaluation

During Evaluation, the clustering solutions generated by PASW have been evaluated.

For a first impression, node ⑤ visualizes the data set containing the computed clustering solution as table.

Node ⑥ exports the data set containing the computed clustering solutions as csv-files. These exported files have been imported in a Java tool computing various quality measures for the clustering solutions. These measures are used to get a first impression about which clustering solutions are helpful and which are not.

Quality Measures for Clustering Solutions

As the true allocation of the items is contained in the data set, quality measures for known allocations (see Section 6.6.1) can be computed. For this purpose, a Java application has been written, reading the true allocations and the computed clustering solutions for the data set. The application computes various measures as introduced in Section 6.6.1. Table 11.1 shows the coefficients computed by the application (sorted according to the mean of the coefficients in descended order).

Parallel to these coefficients measuring the quality of the solutions in comparison to the true allocation defined by an expert, PASW computes the silhouette values for all solutions (sorted in descended order):

1. k-Means, 8 clusters: $silhouette = 0,9$
2. k-Means, 7 clusters: $silhouette = 0,885$
3. k-Means, 6 clusters: $silhouette = 0,883$
4. k-Means, 5 clusters: $silhouette = 0,778$
5. TwoStep: $silhouette = 0,723$

The numbers indicate, that all clustering solutions have similar quality according to the coefficients. However, looking at the silhouette values, another ordering on the quality is applied. This is a hint that the true allocation may not match the usage of the classes.

Which solutions can be used to form useful services now depends on the interpretation of the clusters. In the following, three representative solutions are interpreted with the help of PASW.

Interpreting the TwoStep Solution

The solution with the highest mean of coefficients is a solution computed by the TwoStep algorithm. Although the mean of coefficient was the highest, this solution had the smallest silhouette value (0,723). In this solution, the number of clusters has been computed automatically. As results, three clusters have been formed. As distance measure, Log-Likelihood has been used. The clustering criterion was the Akaike's Information Criterion (AIC).

The Auto Clustering node of PASW supports the visual interpretation of the clustering solution. Figure 11.3 on the facing page shows the three clusters that have been computed automatically by the TwoStep algorithm. For each cluster, the five input attributes (PER_Refactor element, PER_Load Select element, ...) are listed. For each attribute, the diagrams show the relative occurrences of values for the classes in the given cluster.

Figure 11.4 on page 82 provides a more detailed view on such a relative occurrence diagram for the attribute `PER_Refactor element` in Cluster-3. On the x-axis, all possible values of the attribute are shown. As the attribute `PER_Refactor element` contains percentage values about how often a class has been used in this activity, the range is $[0,1]$. The y-axis denotes how many classes (in percent) with the given usage value are located in Cluster-3.

This view helps interpreting the clusters in the following way: Cluster-3 does not contain any of the classes that are often (with more than 70 %) used by the REFACTOR ELEMENT activity. Also, Cluster-3 contains most of the classes that are used by the activity REFACTOR ELEMENT rarely (with 0 % – 60 %). Therefore, the classes in this cluster are not highly related to the REFACTOR ELEMENT activity.

In addition to the relative occurrence, the absolute distribution can be viewed (Figure 11.5 on page 82). This view underlines, that most of the classes that have low usage values in the activity REFACTOR ELEMENT, are located in Cluster-3. No classes with high values are located in Cluster-3.

Looking at Figure 11.3 on the next page, e.g., Cluster-3 can be interpreted as follows. Cluster-3 contains classes that are almost exclusively used in the SELECT ELEMENT activity (the right part of the diagram is at 100 %). Therefore, this cluster may form a business service supporting the SELECT ELEMENT activity. However, this cluster contains classes that are often used in the LOAD

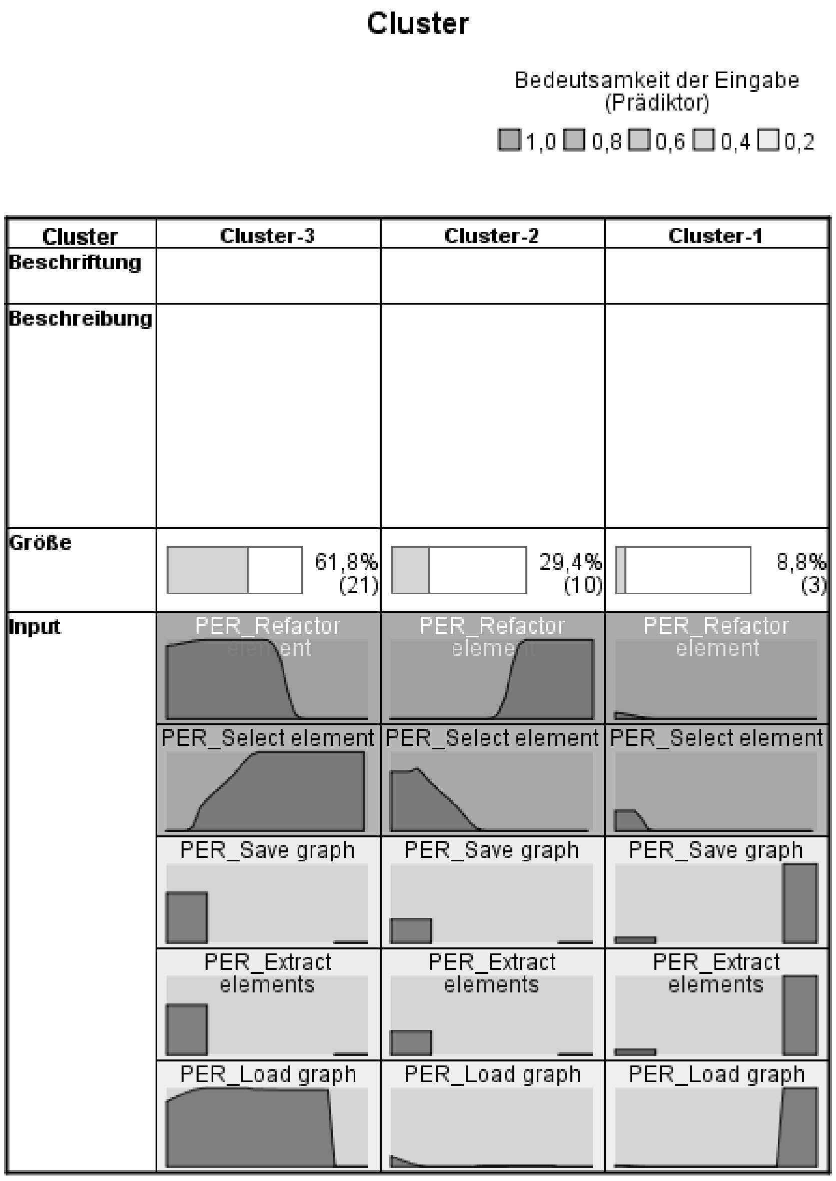


Figure 11.3: First approach: Relative occurrence view on clustering solution computed by TwoStep algorithm

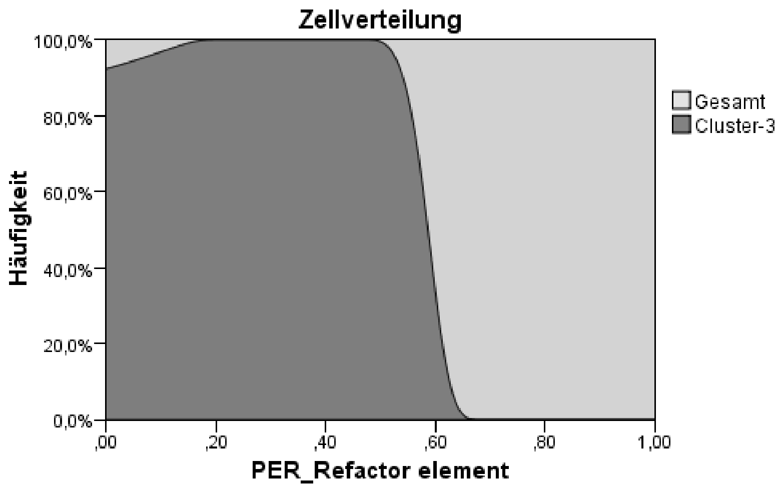


Figure 11.4: First approach: Detailed relative occurrence view on attribute PER_Refactor element in Cluster-3

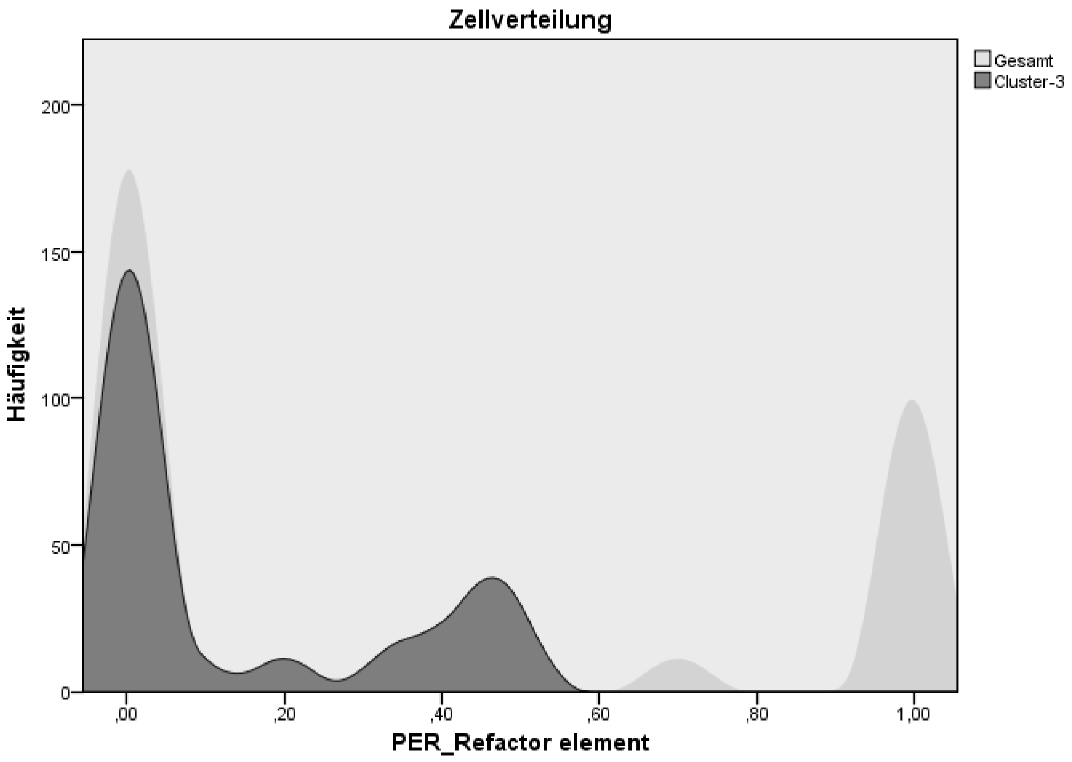


Figure 11.5: First approach: Detailed absolute occurrence view on attribute PER_Refactor element in Cluster-3

GRAPH activity (0 % to 80 %) and classes that are sometimes used in the REFACTOR ELEMENT activity (0 % to 60 %). So this cluster may not be shaped very well as it contains classes that are used in other activities, too.

To distinguish if this cluster more supports REFACTOR ELEMENT or SELECT ELEMENT, the cluster-comparison view shown in Figure 11.6 on the following page can be used. For each attribute, the view shows the median of the whole data set (the black vertical line) as well as the medians of the clusters (colored squares). The view shows that classes in Cluster-3 have a smaller median in the usage values of *Refactor element* than the overall median. In contrast, Cluster-2 has a higher median in the *Refactor element* attribute. For the *Select element* attribute, the median of Cluster-2 is lower and the median of Cluster-3 is higher than the overall median. This strengthens the interpretation, that Cluster-3 supports the SELECT ELEMENT activity whereas Cluster-2 supports the REFACTOR ELEMENT activity.

The remaining clusters can be interpreted as follows. Cluster-2 is more well-shaped. It contains classes that are almost exclusively used in the REFACTOR ELEMENT activity. These classes are not used in other activities very often (the other diagrams have no high values on the right part of the diagrams). So this clusters may provide a definition for a business service supporting the REFACTOR ELEMENT activity.

Finally, Cluster-1 contains classes that are solely used in one of the three activities SAVE GRAPH, LOAD GRAPH or EXTRACT ELEMENTS. Cluster-1 may therefore provide a "Graph-IO" service supporting these three activities.

In addition to this interpretation, the clustering solution can be compared to the true allocation defined by the expert (Figure 11.7 on page 85).

The three allocations load, save and extract have been put into one cluster by the TwoStep approach. Cluster-2 contains all classes of the refactor allocation plus most of the gui classes. Cluster-3 contains all classes of the select allocation plus all data classes. This view explains why the TwoStep 3-clusters solution got the highest values in the quality coefficients: As these coefficients only rate how many item-pairs are in the *same cluster* and how many are not, ratings are high for this solution.

In the following, two solutions computed by the *k*-Means algorithm are interpreted and compared to the TwoStep solution.

Interpreting the k-Means 6-Cluster Solution

The solution with the second highest mean of coefficients was computed by the *k*-Means algorithm. As parameter, a fixed number of 6 clusters has been set. This solution had a silhouette value of $0,883$, promising a more well-shaped solution than the TwoStep solution.

Looking at Figure 11.8 on page 85, the clusters can be interpreted as follows.

Cluster-1 contains classes that are almost exclusively used in the SELECT ELEMENT activity and therefore could form a business service to support this activity. Cluster-2 only contains one class used during the SAVE GRAPH activity and could form a business service for saving graphs. Cluster-3 contains a class used during the LOAD GRAPH activity and Cluster-4 contains a class used only in

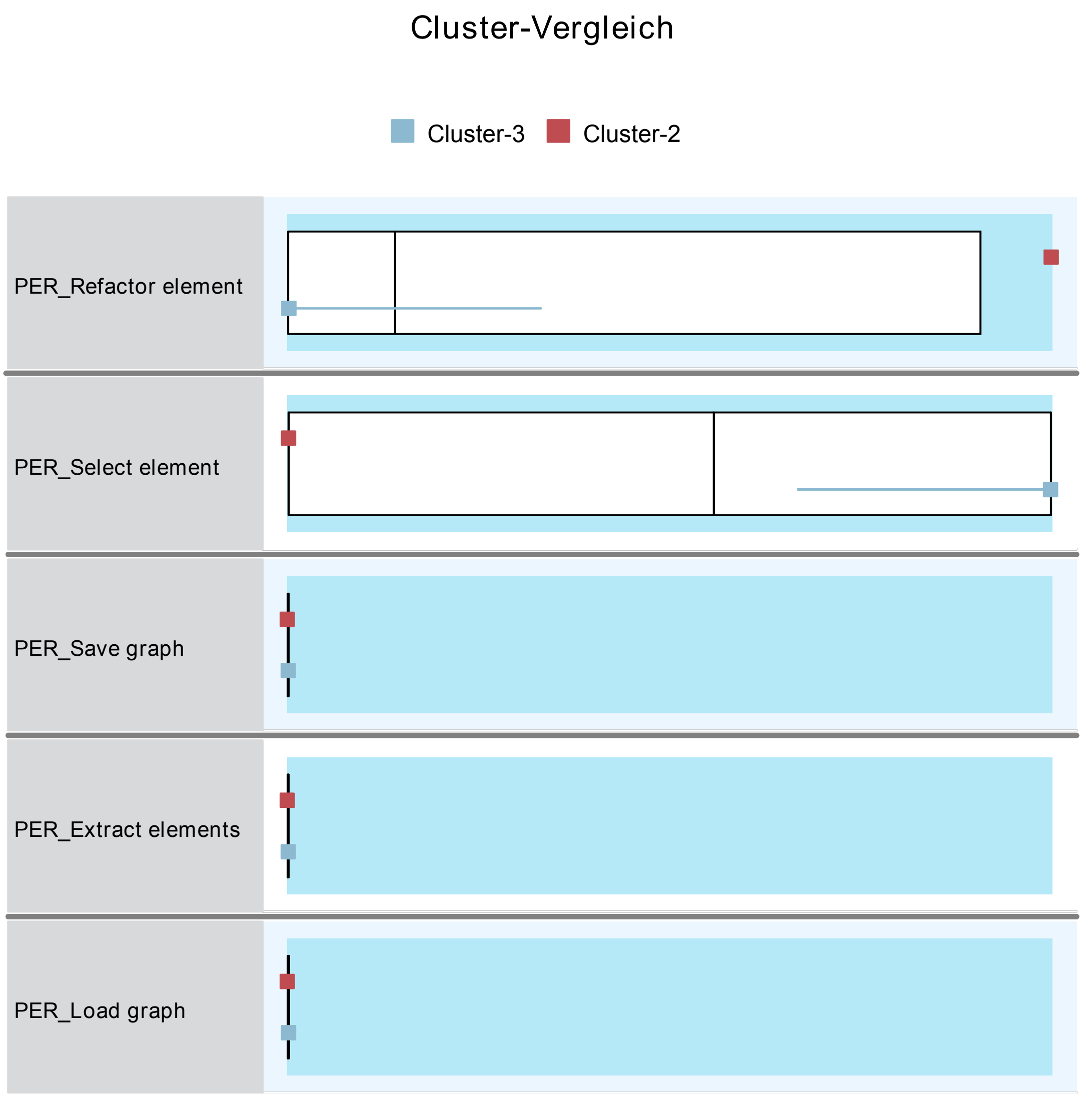


Figure 11.6: First approach: Comparing Cluster-2 and Cluster-3 of the TwoStep solution

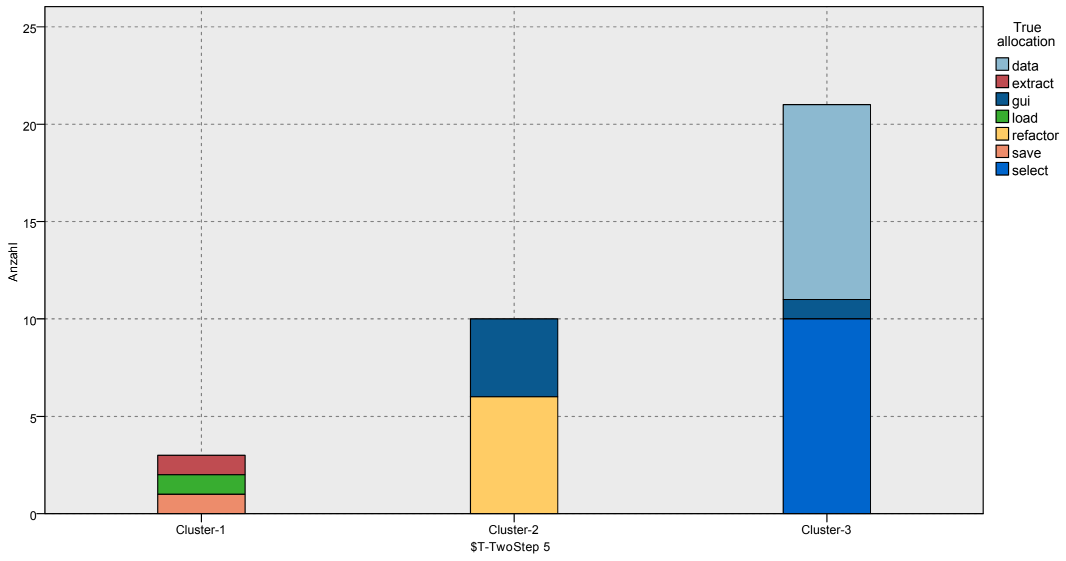
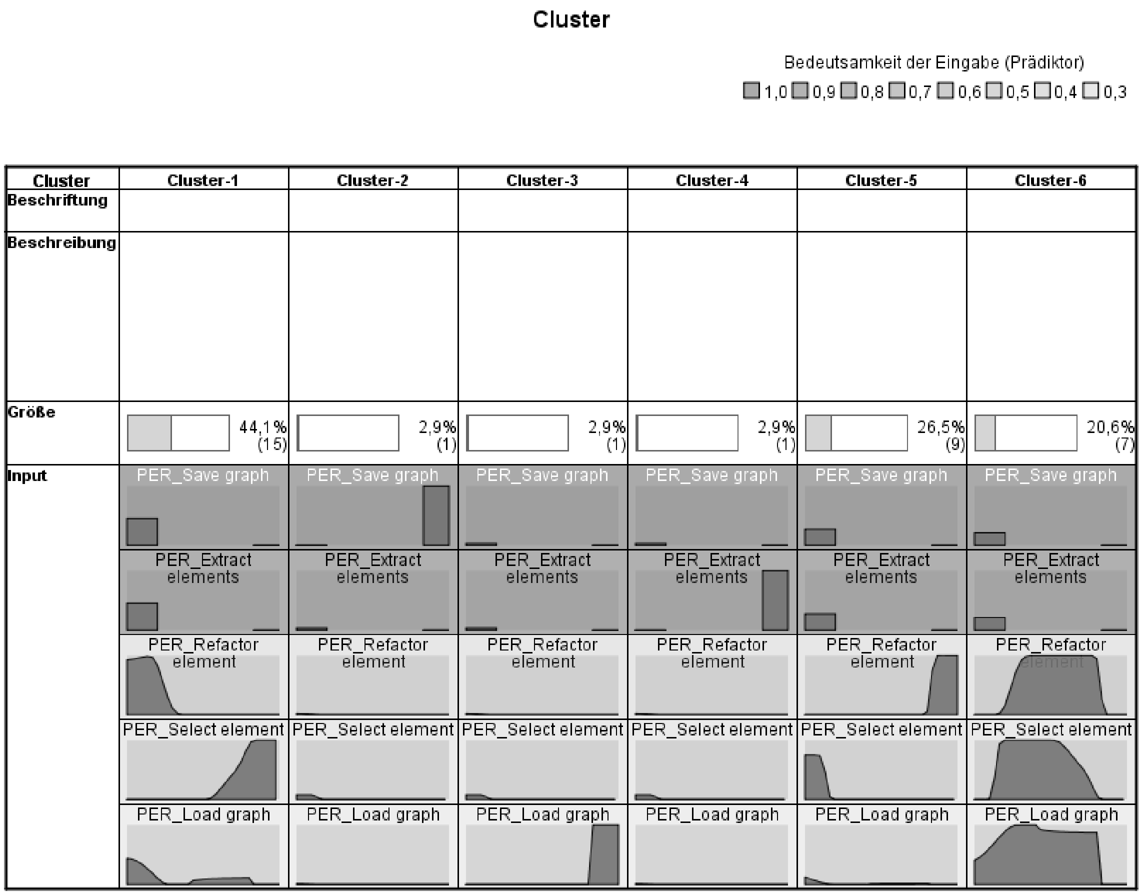


Figure 11.7: First approach: Comparing the TwoStep 3-clusters solution to the true allocation



Cluster	Cluster-1	Cluster-2	Cluster-3	Cluster-4	Cluster-5	Cluster-6
Beschriftung						
Beschreibung						
Größe	44,1% (15)	2,9% (1)	2,9% (1)	2,9% (1)	26,5% (9)	20,6% (7)
Input	PER_Save graph	PER_Save graph	PER_Save graph	PER_Save graph	PER_Save graph	PER_Save graph
	PER_Extract elements	PER_Extract elements	PER_Extract elements	PER_Extract elements	PER_Extract elements	PER_Extract elements
	PER_Refactor element	PER_Refactor element	PER_Refactor element	PER_Refactor element	PER_Refactor element	PER_Refactor element
	PER_Select element	PER_Select element	PER_Select element	PER_Select element	PER_Select element	PER_Select element
	PER_Load graph	PER_Load graph	PER_Load graph	PER_Load graph	PER_Load graph	PER_Load graph

Figure 11.8: Relative occurrence view on k-Means 6-clusters solution

EXTRACT ELEMENTS. They can form services to support their activities, respectively. Cluster-5 contains classes that are used most often during REFACTOR ELEMENT forming a business service for this activity.

So far, all clusters have been well-shaped, providing explicit business functionality to their activities. In contrast, Cluster-6 seems to contain classes, that are spread over the three activities REFACTOR ELEMENTS, SELECT ELEMENT and LOAD GRAPH. Therefore, this cluster may form a support service used during these three activities.

Figure 11.9 compares the clustering solution to the true allocation.

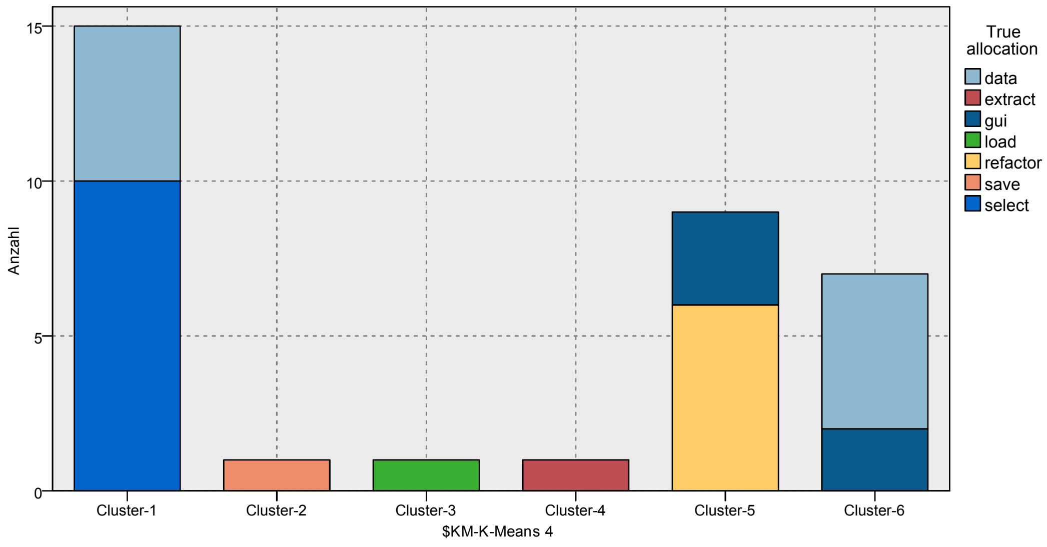


Figure 11.9: First approach: Comparing the k-Means 6-clusters solution to the true allocation defined by the expert

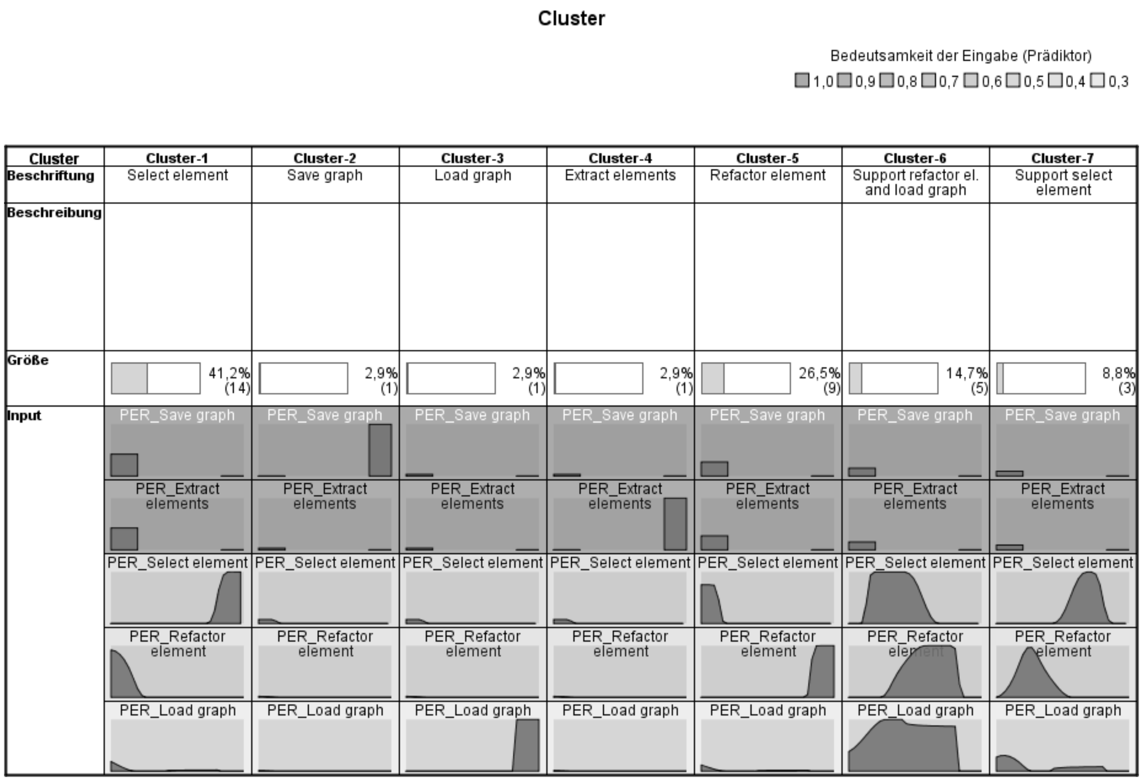
Cluster-2, 3 and 4 perfectly match the expert's intuitive allocation of legacy classes to services. Cluster-1 and 4 match the true allocation, too. However, they contain additional classes which had been allocated to the data or gui service by the expert. Finally, Cluster-6 is a mix of data and gui classes.

In the following, a solution splitting the last, mixed-up cluster, is presented.

Interpreting the k-Means 7-Cluster Solution

The *k*-Means 7-clusters solution is similar to the *k*-Means 6-clusters solution. The well-shaped clusters are identical. Only the last cluster that mixed-up classes to support REFACTOR ELEMENTS, SELECT ELEMENT and LOAD GRAPH has been divided into two separate clusters.

Figure 11.10 on the facing page shows the relative occurrences view on the 7-clusters solution. In this view, the clusters have been named according to their interpretation. In contrast to the 6-clusters solution, Cluster-6 of the 7-clusters solution focuses on supporting the activities REFACTOR ELEMENT and LOAD GRAPH whereas Cluster-7 focuses on supporting SELECT ELEMENT.



Cluster	Cluster-1	Cluster-2	Cluster-3	Cluster-4	Cluster-5	Cluster-6	Cluster-7
Beschriftung	Select element	Save graph	Load graph	Extract elements	Refactor element	Support refactor el. and load graph	Support select element
Beschreibung							
Größe	41,2% (14)	2,9% (1)	2,9% (1)	2,9% (1)	26,5% (9)	14,7% (5)	8,8% (3)
Input	PER_Save graph	PER_Save graph	PER_Save graph	PER_Save graph	PER_Save graph	PER_Save graph	PER_Save graph
	PER_Extract elements	PER_Extract elements	PER_Extract elements	PER_Extract elements	PER_Extract elements	PER_Extract elements	PER_Extract elements
	PER_Select element	PER_Select element	PER_Select element	PER_Select element	PER_Select element	PER_Select element	PER_Select element
	PER_Refactor element	PER_Refactor element	PER_Refactor element	PER_Refactor element	PER_Refactor element	PER_Refactor element	PER_Refactor element
	PER_Load graph	PER_Load graph	PER_Load graph	PER_Load graph	PER_Load graph	PER_Load graph	PER_Load graph

Figure 11.10: First approach: Relative occurrence view on k-Means 7-clusters solution

A comparison to the true allocation (Figure 11.11 on the next page) shows that the classes belonging to the true allocations data and gui that formed one cluster in the 6-clusters solution, have been split up into two clusters. However, this splitting did not follow the true allocation. So now both smaller clusters contain classes of both allocations.

Producing even more clusters (as in the *k*-Means 8-clusters solution) does not lead to solutions that are interpretable more easily. Also, the *k*-Means 5-clustering solution does not add new possibilities for interpretation. Therefore, these approaches are not interpreted in detail. In the following, the results of this first clustering approach are wrapped up.

11.1.4 Summary of the First Approach

As summarizing conclusion of this first approach it can be noted that the non-ambiguous classes that are (more or less) distinctly used in one of the activities have been separated well. However, the approach has problems with splitting up the helper classes (belonging to the true allocations gui and data). The approach could not match the expert's intuition.

However, analysis of the legacy code and interviews with the expert indicated, that some of the helper classes are exclusively used in one of the activities (e.g., the class `StringTextField` is allocated to the true allocation gui as it is a plain GUI class. However this GUI element is solely used in refactoring dialogs and therefore put into the refactorings cluster by the approaches). Hence, allocating them to these clusters might be a better fit than implicated by the true allocation.

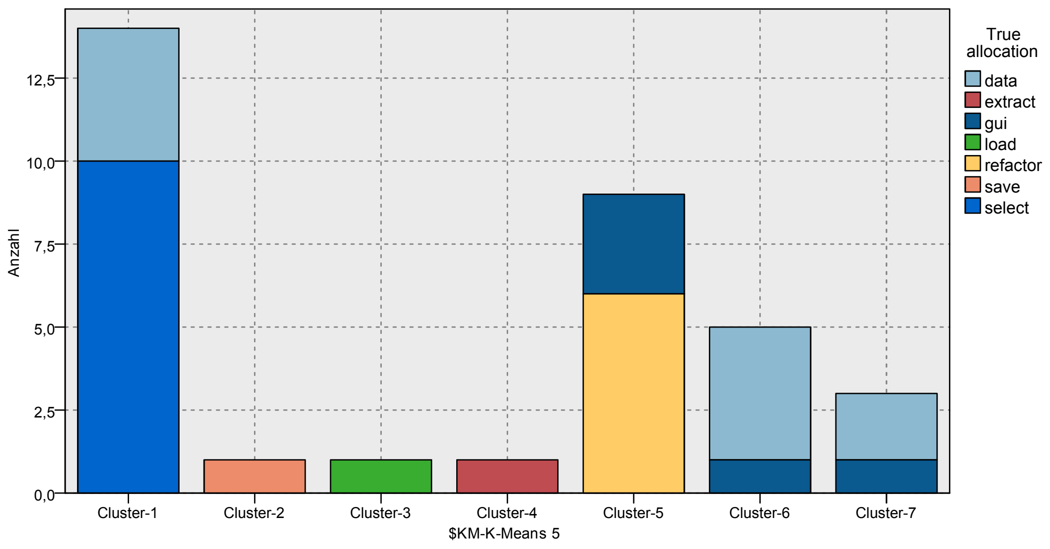


Figure 11.11: First approach: Comparing the k-Means 7-clusters solution to the true allocation defined by the expert

In the following, another approach is presented, trying to better fit the expert's intuition by adding additional information other than the usage values.

11.2 Second Approach: Clustering Classes According to Usage in Activities Extended by Knowledge about Packages

The first approach of clustering the data according to the usage values showed good results in allocating classes that have been used almost exclusively in any of the activities. To achieve a better allocation of classes that have been used in multiple activities, this approach adds a new input variable: the package names of the classes.

Hypothesis 11.2.1 (*Knowledge about Package*)

Knowledge about the package of a class leverages the clustering of classes that are used in multiple activities.

In this section, the three clustering approaches

1. TwoStep
2. *k*-Means, 6 clusters and
3. *k*-Means, 7 clusters

are applied to a data set that has been extended by an attribute containing the qualified package name of a class. The results are compared to the clustering solutions computed in the first approach.

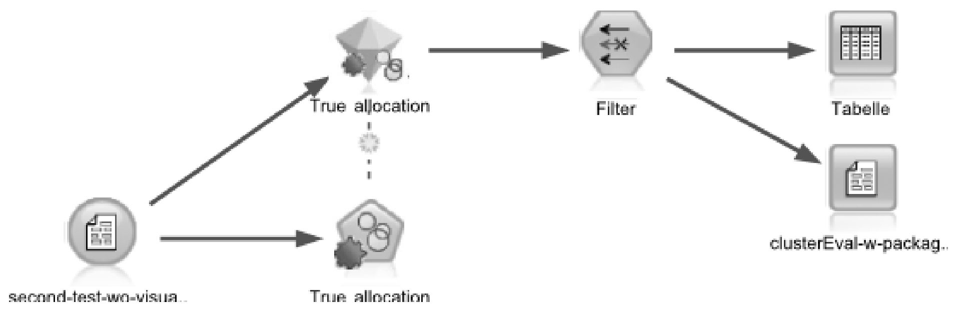


Figure 11.12: Second approach: Modeling

11.2.1 Data Preparation

In addition to the attributes used in the first approach (Section 11.1.1), the package of a class is retrieved by the graph extractor tool. The input data in the second approach contains the following attributes:

- `Qualified name`: The qualified class name
- `Qualified package name`: The qualified package name
- `PER_Select element`: Usage of the class in the activity SELECT ELEMENT in percentage
- `PER_Extract elements`: Usage of the class in the activity EXTRACT ELEMENTS in percentage
- `PER_Save graph`: Usage of the class in the activity SAVE GRAPH in percentage
- `PER_Load graph`: Usage of the class in the activity LOAD GRAPH in percentage
- `PER_Refactor element`: Usage of the class in the activity REFACTOR ELEMENT in percentage
- `True allocation`: The "true cluster" the class belongs to, defined by an expert

11.2.2 Modeling

The modeling set-up is similar to the modeling of the first clustering approach (Section 11.1.2). Figure 11.12 shows the modeling in PASW. As input file, the csv-file (with package names) is loaded. The auto clustering node has been set to generate the three models that have been interpreted in Section 11.1.3 with the same settings. The models and settings are:

1. *TwoStep* (compute number of clusters automatically, distance measure = Euclidean, clustering criterion: AIC)
2. *k-Means* (number of clusters = 6)
3. *k-Means* (number of clusters = 7)

The three clustering solutions are evaluated in the following.

11.2.3 Evaluation

Table 11.2 shows the quality for both approaches (without packages and with packages as input attribute).

File	Rand Statistic	Jaccard	Precision	Recall	F-Measure	Mean of Coefficients
second-approach-TwoStep-auto-clusters.csv	0,916	0,699	0,727	0,948	0,823	0,822
second-approach-k-Means-7clusters.csv	0,866	0,540	0,647	0,765	0,701	0,704
second-approach-k-Means-6clusters.csv	0,841	0,497	0,587	0,765	0,664	0,671
first-approach-TwoStep-auto-clusters.csv	0,731	0,424	0,430	0,965	0,595	0,629
first-approach-k-Means-6clusters.csv	0,806	0,435	0,519	0,730	0,606	0,619
first-approach-k-Means-7clusters.csv	0,816	0,425	0,543	0,661	0,596	0,608

Table 11.2: Quality measures for both clustering approaches

Comparing the mean of the coefficients, the second approach computes better results than the first approach. However, silhouette values are far smaller than with the first approaches:

1. First approach (without packages)
   a) k-Means, 7 clusters: $silhouette = 0,885$
   b) k-Means, 6 clusters: $silhouette = 0,883$
   c) TwoStep: $silhouette = 0,723$
2. Second approach (with packages)
   a) k-Means, 6 clusters: $silhouette = 0,542$
   b) TwoStep: $silhouette = 0,534$
   c) k-Means, 7 clusters: $silhouette = 0,485$

These number indicate that the second approach may better fit to the true allocation but that the true allocation may not reflect the true usage of the classes during the activities.

The three clustering solutions are interpreted in the following; they are compared to the solutions of the first approach, too.

Interpreting the TwoStep Solution with Packages

The relative occurrences view on the TwoStep clustering solution shows a similar picture compared to the first approach in a way that the graph IO activities (Load graph, Save graph and Extract elements) are still put together into one cluster (Figure 11.13 on the following page). Also, one cluster for the REFACTOR ELEMENT activity has been built.

In contrast to the TwoStep solution of the first approach, the cluster supporting SELECT ELEMENT has been split up into two clusters. Cluster-3 now only contains classes, that are exclusively used in the SELECT ELEMENT activity. Figure 11.14 on page 93 shows the absolute distribution of classes in Cluster-3. All classes of the `operations.selection` package are located in this cluster, supporting the interpretation that this clusters supports the SELECT ELEMENT activity.

The new cluster, Cluster-4, now contains classes that are often used in SELECT ELEMENT as well as REFACTOR ELEMENT. Contrary to the plain usage values, these classes have been put into an own cluster because of their packages. All classes in Cluster-4 belong to one of the packages `extractor`, `gui` or `types`. So this cluster may build a supporting service for the SELECT ELEMENT and REFACTOR ELEMENT activities.

Comparing the TwoStep solution to the true allocation, Cluster-3 is now a perfect fit with the select allocation (Figure 11.15 on page 94). Cluster-2 covers the refactor allocation. Cluster-1 combines the allocations load, save and extract. Cluster-4 covers the complete data allocation and parts of the gui allocation.

Interpreting the k-Means 6-Clusters Solution with Packages

The *k*-Means 6 clusters solution of the second approach is very similar to the first approach (Figure 11.16 on page 94).

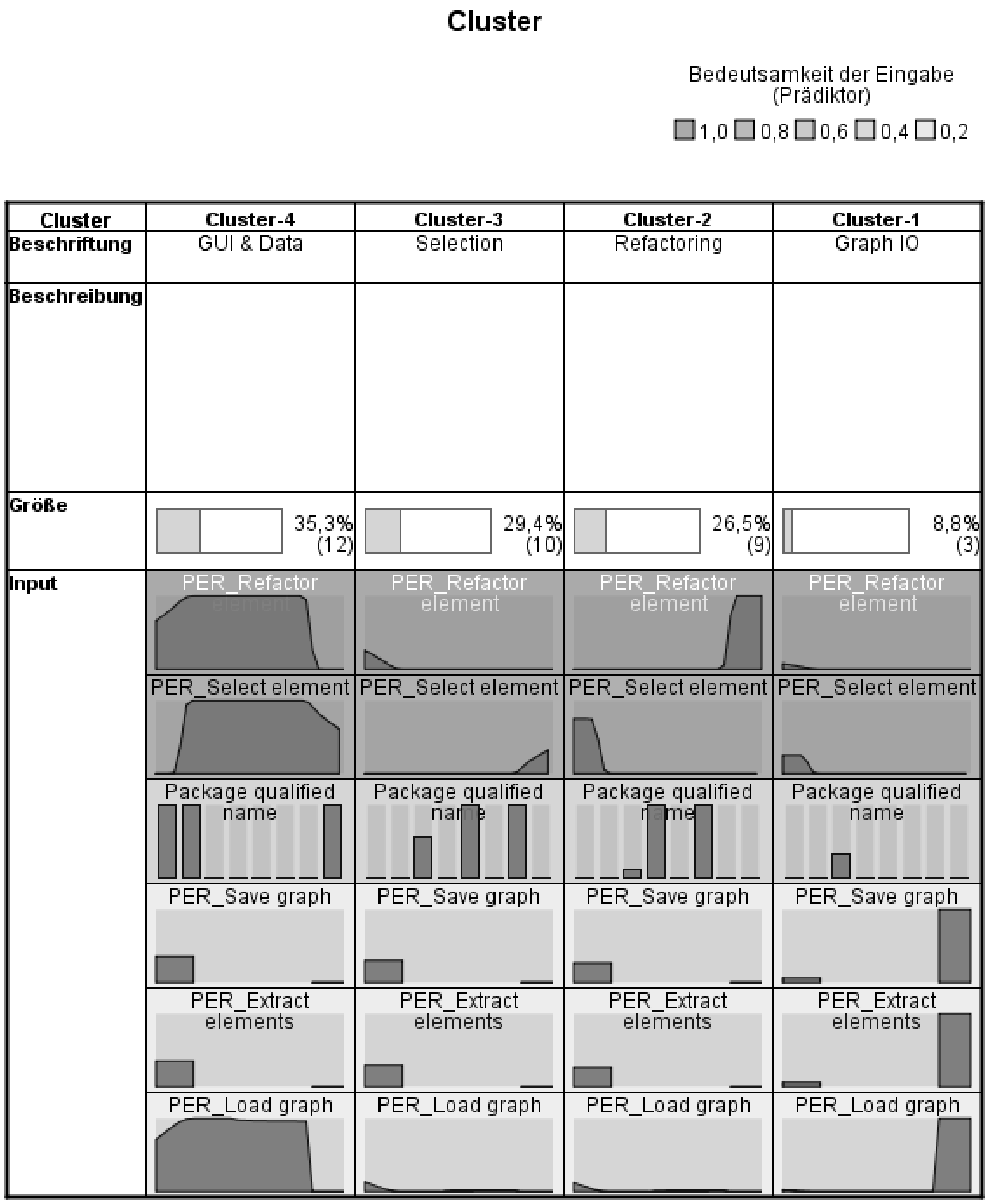
Only the cluster containing the select allocation has now less classes belonging to the data allocation. These classes have been moved to the supporting cluster.

Interpreting the k-Means 7-Clusters Solution with Packages

The *k*-Means 7 clusters solution is very similar to the first approach, too ((Figure 11.17 on page 94)).

In the second approach, some data classes that had been allocated to the select cluster now are put into the data cluster. In addition, the two clusters with the data and gui allocation are now separated very well.

Looking at the classes that are still mis-allocated, it can be stated that these classes are really exclusively used in the SELECT ELEMENT activity and the REFACTOR ELEMENT activity. Therefore,



Cluster	Cluster-4	Cluster-3	Cluster-2	Cluster-1
Beschriftung	GUI & Data	Selection	Refactoring	Graph IO
Beschreibung				
Größe	35,3% (12)	29,4% (10)	26,5% (9)	8,8% (3)
Input	PER_Refactor element	PER_Refactor element	PER_Refactor element	PER_Refactor element
	PER_Select element	PER_Select element	PER_Select element	PER_Select element
	Package qualified name	Package qualified name	Package qualified name	Package qualified name
	PER_Save graph	PER_Save graph	PER_Save graph	PER_Save graph
	PER_Extract elements	PER_Extract elements	PER_Extract elements	PER_Extract elements
	PER_Load graph	PER_Load graph	PER_Load graph	PER_Load graph

Figure 11.13: Clustering solution for the TwoStep algorithm including packages

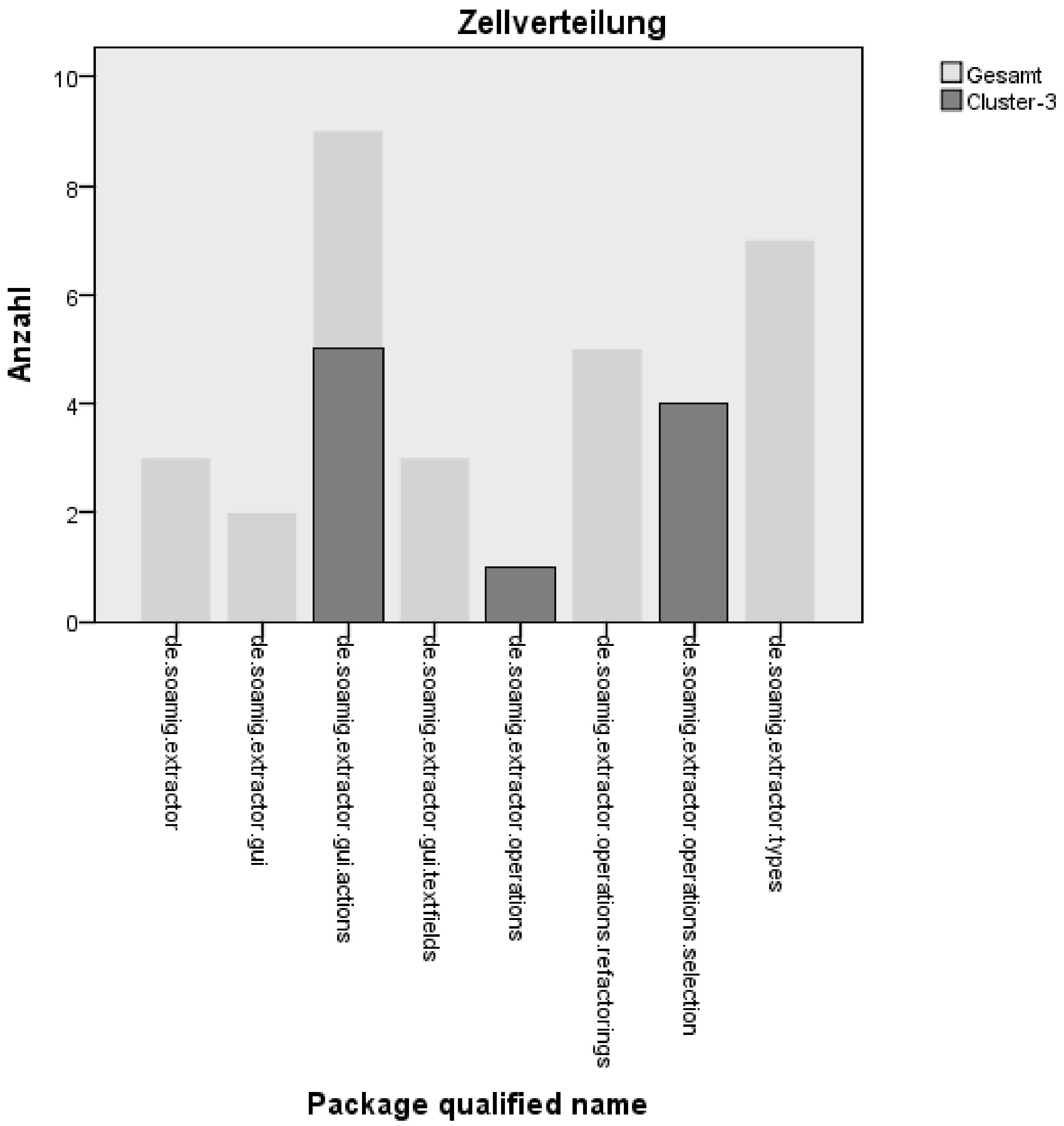


Figure 11.14: Absolute distribution of classes in Cluster-3

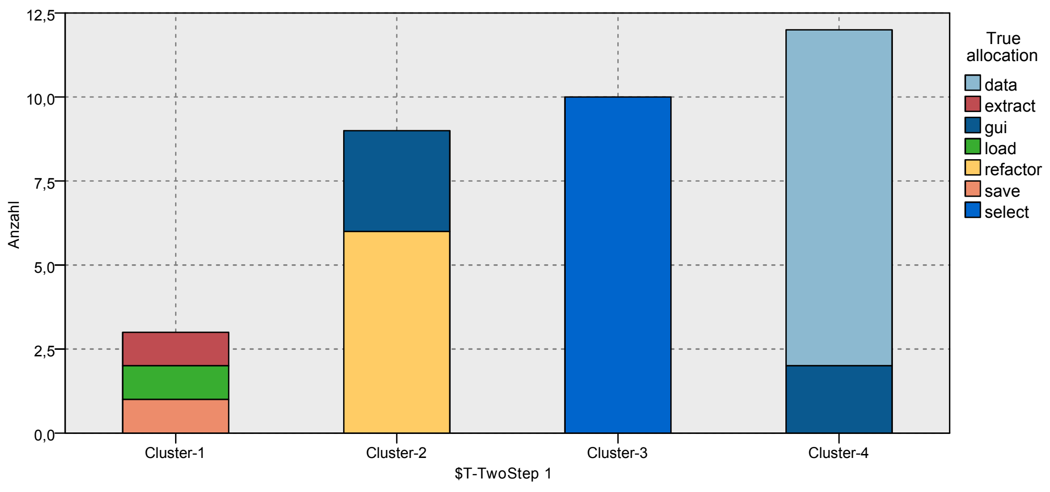
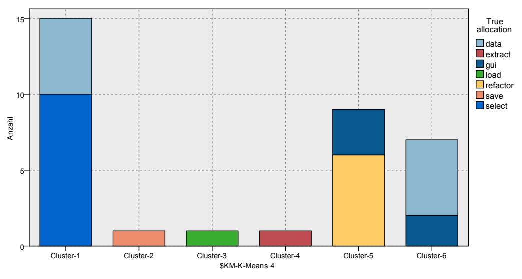
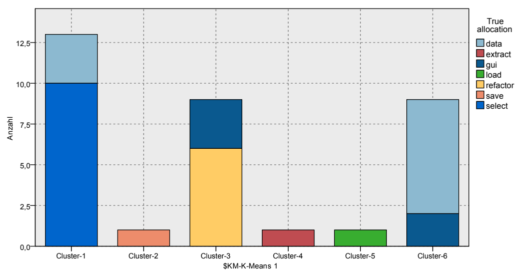


Figure 11.15: Comparing the TwoStep solution of the second approach to the true allocation

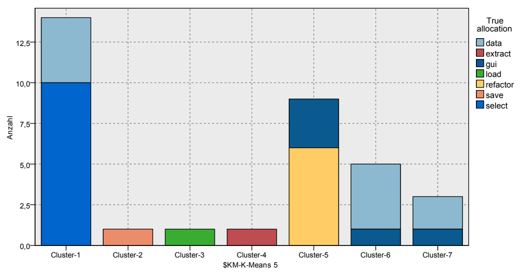


(a) without packages

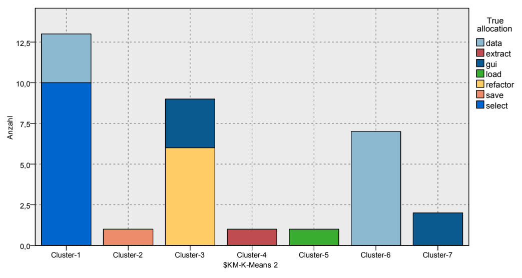


(b) with packages

Figure 11.16: Comparing the k-Means 6 clusters solution of the first approach (without packages) and the second approach (with packages)



(a) without packages



(b) with packages

Figure 11.17: Comparing the k-Means 7 clusters solution of the first approach (without packages) and the second approach (with packages)

these classes do fit well into these clusters and the true allocation might have to be adapted to fit this clustering solution.

11.2.4 Summary of the Second Approach

Summarizing, the introduction of package information especially helped to separate the supporting clusters. The business clusters are not affected by the additional information.

However, the package information distracted the clustering algorithms from the usage values. The solutions might now better fit the package structure. But if this structure does not group classes well, it might destroy good clustering solutions that had been based on the usage values.

As best practice, a legacy analyst should trade off very well if package information should be included in clustering. If the package structure of a legacy system groups classes according to their functionality, the information might support the clustering. But if the classes are structured badly, the package information might mislead the clustering algorithms.

11.3 Discussing the Suitability of Clustering Approaches for Identifying Legacy Code able to Support Services

For the first approach, three hypotheses have been set up (Hypothesis 11.1.1, Hypothesis 11.1.2 and Hypothesis 11.1.3 on page 76) stating, that clustering approaches can separate legacy code into clusters able to form three types of services.

As described in Section 11.1, legacy code for *business services* can be identified very well. As the legacy code is (almost) exclusively used is one activity, clustering approaches achieved a good allocation of this code.

The clustering solutions had more problems with the allocation of legacy code for *support services*. As this code is used almost evenly in multiple activities, the clustering solutions did not always allocate code well (compared to the true allocation defined by an expert). The first approach only using the usage values did not match the expert's intuition very well. However, manually expecting the legacy code revealed that the mis-allocated classes where often used explicitly in one of the activities. Therefore, the clustering solution might be a better fit than the expert's intuition. The second approach adding information about the package names achieved a better matching to the true allocation. However, if packages are structured badly in other legacy systems, this information might distract the clustering algorithms.

For the third type of service – *helper services* – no code could be identified in this legacy system. The reason might be, that there is no code that is used in all services (e.g., logging mechanisms).

A problem still remaining is to identify the optimal number of clusters without knowing the legacy code. Four of the six presented solutions where computed by the *k*-Means algorithm which requires a given number of target clusters. A first hint for the number of clusters might be the number of activities. However, that only covers the number of resulting business services and ignores support service and helper services. Therefore, an understanding of how many services will be needed may be necessary, first.

However, two of the six solutions where computed by the TwoStep algorithm. This algorithm is able to compute the optimal number of clusters automatically. As these solutions did fit to the expert's intuition, too, this may be a promising approach to allocate code without knowing the number of target clusters.

Summarizing, the clustering approaches achieved good results in allocating legacy code to clusters that can form services to support the activities. In the following chapter, link analysis is explored on its suitability for identifying legacy code to implement services.

Chapter 12

Using Association Rules to Identify Legacy Code for Service Implementation

In this chapter, two association rules approaches are introduced. Results are presented and their usefulness is discussed.

The motivation of association rules is to identify relationships between items of a dataset (Chapter 7). As the identification of association rules is based on frequencies, multiple workflows are needed. For this thesis, eight different workflows have been generated. They vary in the activities that were executed as well as the functionality that was executed during the activities. Therefore, the workflows represent a broad coverage of variance of how the SoamigExtractor tool can be used. Association rule identification aims at finding rules that apply to most of the workflows and therefore are valid for most scenarios the tool can be used for.

Choosing which data should be analyzed and how this data if formatted for input to the algorithm influences the identification of association rules. In the following, two variations on the input format are presented.

12.1 First Approach: Identify Legacy Code Supporting Activities

The first approach aims at identifying legacy classes that are almost exclusively used during one activity. This approach should generate results quite similar to the clustering approaches (Chapter 11) except that it does only identify code exclusively used in one activity (business service).

12.1.1 Data Preparation

For the first approach, the following attributes have been exported from the TGraph containing the legacy code, business processes and trace links:

workflowId. The id of the workflow
activity. The name of the activity
class. The qualified name of a class

So, each line of the dataset describes, what class has been called by what activity during which workflow.

12.1.2 Modeling

After importing the file containing the data as described in Section 12.1.1, the dataset is analyzed by the Apriori algorithm to identify association rules. PASW's Apriori implementation can handle two input formats: The classical transaction format as described in Chapter 7 as well as regular datasets (containing no transactions). Using a regular dataset (the dataset used for the first approach is regular as it does not contain transactions), PASW interprets each item as own transaction.

In addition, the algorithm has been limited to identify rules of the form

$$class \rightarrow activity \tag{12.1}$$

The rule is read like "If the class was `class` then it was called during the activity `activity`".

The minimal support ε_{sup} has been set to 0[1]. The minimal confidence ε_{conf} has been set to $80,0\%$.

12.1.3 Evaluation

Applying the Apriori algorithm to the first approach resulted in 22 rules matching the given criteria ε_{sup} and ε_{conf}. The rules are listed in Figure 12.1.

Antezedent	Consequent	Support %	Confidence %	Lift
class = de.soamig.extractor.gui.actions.GenerateXMLAction	activity = Extract elements	0,62	100,00	23,07
class = de.soamig.extractor.gui.actions.SaveGraphAction	activity = Save graph	1,24	100,00	13,46
class = de.soamig.extractor.gui.actions.OpenGraphAction	activity = Load graph	2,48	100,00	4,97
class = de.soamig.extractor.operations.refactorings.RemoveSuperClass.MethodTypeComparator	activity = Refactor element	0,31	100,00	3,40
class = de.soamig.extractor.operations.refactorings.CopyField	activity = Refactor element	0,62	100,00	3,40
class = de.soamig.extractor.operations.refactorings.RemoveSuperClass	activity = Refactor element	0,93	100,00	3,40
class = de.soamig.extractor.operations.refactorings.MoveClassType	activity = Refactor element	0,93	100,00	3,40
class = de.soamig.extractor.gui.textfields.StringTextField	activity = Refactor element	0,93	100,00	3,40
class = de.soamig.extractor.gui.textfields.TextField	activity = Refactor element	1,24	100,00	3,40
class = de.soamig.extractor.operations.refactorings.SwitchMethodCalls	activity = Refactor element	1,24	100,00	3,40
class = de.soamig.extractor.gui.textfields.VertexTextField	activity = Refactor element	1,24	100,00	3,40
class = de.soamig.extractor.operations.refactorings.CopyMethodType	activity = Refactor element	1,24	100,00	3,40
class = de.soamig.extractor.gui.actions.RefactoringAction	activity = Refactor element	1,55	100,00	3,40
class = de.soamig.extractor.types.ExtWorkflowImpl	activity = Select element	0,93	100,00	2,76
class = de.soamig.extractor.operations.selection.SelectWorkflow	activity = Select element	0,93	100,00	2,76
class = de.soamig.extractor.gui.actions.SelectByWorkflowAction	activity = Select element	0,93	100,00	2,76
class = de.soamig.extractor.gui.actions.SelectElementAction	activity = Select element	0,93	100,00	2,76
class = de.soamig.extractor.operations.selection.SelectActivity	activity = Select element	1,55	100,00	2,76
class = de.soamig.extractor.operations.selection.SelectMethod	activity = Select element	2,17	100,00	2,76
class = de.soamig.extractor.gui.actions.SelectionAction	activity = Select element	2,48	87,50	2,42
class = de.soamig.extractor.gui.actions.SelectByTypeAction	activity = Select element	1,86	83,33	2,30
class = de.soamig.extractor.types.ExtActivityImpl	activity = Select element	1,86	83,33	2,30

Figure 12.1: Result of first association rules approach: Mapping classes to activities

The antecedents of the result all have small support values. They result from the fact, that the classes (antecedents) are used exclusively in their activities. For this reason, the minimum support

[1] An explanation why these values have been chosen is provided in Section 12.1.3

has been set to 0. In addition, the confidence values of the rules are all beyond 80 % (as $\varepsilon_{conf} = 80\%$). As an example, the rule

$$de.soamig.extractor.gui.actions.GenerateXMLAction \rightarrow Extract\ elements$$
$$(\text{with support} = 0.62, \text{confidence} = 100)$$

can be interpreted as: "In 100% of the times, the class `GenerateXMLAction` was called, it was called during the activity EXTRACT ELEMENTS. Compared to all class calls, the class `GenerateXMLAction` was called in 0.62% of the calls."

The high lift values (all bigger than 1) indicate, that the rules are not based on random nature but result from the dataset. The higher a lift value is, the more expressive a rule is.

Comparing the result to the results of the clustering solutions shows, that the rules propose an allocation similar to the clustering solutions for the case of explicitly used classes. However, the clustering solutions put additional classes into the clusters, that are not covered by the association rules.

Compared to the true allocation defined by the expert, the results suffers from the same limitations as the clustering approach: Some classes belonging to the GUI are exclusively used in one of the activities. Therefore, they are allocated to this activity instead of a helper service.

Summarizing, the first association rules approach identified classes that are explicitly used in one activity. The result can be used to check the correctness of a clustering solution. If both data mining techniques result in similar outcomes, this supports the correctness of the results.

The second association rules approach will highlight a different possibility how association rules can be used: It identifies legacy code belonging together.

12.2 Second Approach: Identify Joint Legacy Code

The second approach aims at identifying legacy code that is used together across all activities. Legacy code that is always used together, independent of the activity in which it is used, may indicate that this code should not be split up on different services.

12.2.1 Data Preparation

For this, data has been brought into transaction format. That is, each activity of a workflow has been defined as one transaction (because it has been the smallest, atomic unit during dynamic analysis). Same activities during different workflows are defined as different transactions, too. For each transaction, the classes called during this workflow and activity have been defined as items. The input dataset has the following format:

transactionId unique id composed of workflow id, activity name and increasing counter (added if an activity has already been executed during the workflow)

items A set of qualified class names of classes called during the transaction

12.2.2 Modeling

PASW's Apriori algorithm has been configured to handle transaction data as provided by the input dataset. When using the transaction format, the algorithm tries to identify association rules between *items* (here classes).

Similar to the first association rules approach, the Apriori algorithm has been set-up as follows:

- Number of items in antecedent and consequent: one item each
- Minimum support $\varepsilon_{sup} = 0.0\%$
- Minimum confidence $\varepsilon_{conf} = 80.0\%$

The algorithm generates rules of the form

$$Class \rightarrow Class \tag{12.2}$$

if classes are often ($> 80\%$) used together in any activity.

12.2.3 Evaluation

Applying the algorithm to the transaction dataset results in 403 rules. Representative parts of the result are interpreted in the following.

Figure 12.2 shows all rules having the class `SelectByActivityAction` as antecedent.

Antecedent	Consequent	Support	Confidence	Lift
de.soamig.extractor.gui.actions.SelectByActivityAction = T	de.soamig.extractor.gui.actions.SelectByTypeAction = T	11,11	100,00	4,50
de.soamig.extractor.gui.actions.SelectByActivityAction = T	de.soamig.extractor.gui.actions.SelectionAction = T	11,11	100,00	3,38
de.soamig.extractor.gui.actions.SelectByActivityAction = T	de.soamig.extractor.gui.GUIUtils = T	11,11	100,00	1,00
de.soamig.extractor.gui.actions.SelectByActivityAction = T	de.soamig.extractor.SoamigExtractor = T	11,11	100,00	1,00
de.soamig.extractor.gui.actions.SelectByActivityAction = T	de.soamig.extractor.types.ExtActivityImpl = T	11,11	100,00	4,50
de.soamig.extractor.gui.actions.SelectByActivityAction = T	de.soamig.extractor.types.JavaPackageTreeNodeImpl = T	11,11	100,00	1,08
de.soamig.extractor.gui.actions.SelectByActivityAction = T	de.soamig.extractor.types.MethodTypeTreeNodeImpl = T	11,11	100,00	1,13

Figure 12.2: Result of second association rules approach: Identifying associated functionality

All rules have a confidence value of 100 %, that is, if the class `SelectByActivityAction` has been called, then the consequent class has *always* been called, too. However, this does not allow the interpretation, that both classes are always used together!

Example: The rule *SelectByActivityAction* $\rightarrow$ *GUIUtils* has a confidence of 100 %. That means, if `SelectByActivityAction` has been called in an activity, then the class `GUIUtils` has always been called in that activity, too.

However, a rule for the opposite direction (*GUIUtils* $\rightarrow$ *SelectByActivityAction*) does not exist because when `GUIUtils` has been called, `SelectByActivityAction` has not always been called!

As an consequence, an absolutely secure interpretation as "is always used together" requires for two classes $Class1$ and $Class2$ that two rules exist in the result set:

$$Class1 \rightarrow Class2 \wedge \tag{12.3}$$

$$Class2 \rightarrow Class1 \tag{12.4}$$

In addition, the lift value of a rule may indicate if it is a strong rule, even if the opposite rule is not in the result set. Lift values of rules for classes that do not belong together seem[2] to be near to 1. That is, these rules are not better than rules that were computed based on independent items. In contrast, rules for classes that really belong together seem to have higher lift values.

Taking this into account and looking at the result excerpt in Figure 12.2 on the facing page, only the classes (1) `SelectByTypeAction`, (2) `SelectionAction` and (3) `ExtActivityImpl` seem to belong to the `SelectByActivityAction` class. To find a complete set of classes belonging together, rules have to be analyzed transitively. That is, starting from `SelectByActivityAction`, the consequents of the three associated classes mentioned above must be analyzed, if rules exist that have one of the consequent as antecedent (Figure 12.3).

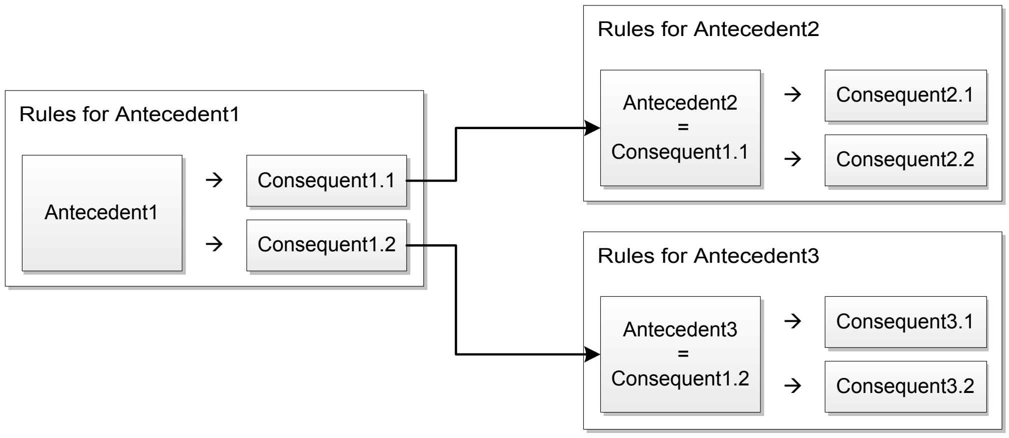


Figure 12.3: Transitive processing of association rules

Transitive processing[3] of the rules for the `SelectByActivityAction` result in the following classes belonging to `SelectByActivityAction`:

1. [...].`gui.actions.SelectByTypeAction`
2. [...].`gui.actions.SelectionAction`
3. [...].`operations.Operation`
4. [...].`operations.selectionSelectActivity`
5. [...].`operations.selectionSelectMethod`
6. [...].`operations.selectionSelectionCommand`
7. [...].`types.ExtActivityImpl`

Analyzing the legacy code and including the expert's intuition, these classes indeed belong together. However, legacy code covering the functionality "select activity by workflow" is missing. A reason could be that this functionality has not been executed very often during the workflows.

[2]The result set has been examined manually
[3]Here, this processing is performed manually

In addition, when it was executed, it was the only select functionality that was executed. Therefore, the result is not surprising, as no workflows exist that used together the select by workflow functionality and the other select functionalities.

Summarizing, the second association rules approach can be used to identify legacy code that was commonly used together. The granularity of the result depends on the granularity of the activities: Coarse-grained activities lead to larger sets of code that has been used together.

12.2.4 Discussing the Suitability of Association Rules for Identifying Legacy Code able to Implement Services

Link analysis with association rules could be used successfully to support the identification of legacy code for service implementation. The first approach was used to check the correctness of the clustering solutions. The second approach was able to identify legacy code belonging together, independent of the activity it was used in.

However, the second approach is influenced by the granularity of activities. If they are modeled too coarse-grained, too much functionality will be identified as belonging together. Compared to the clustering approaches, it may be necessary to define activities more fine-grained.

This chapter concludes the application of data mining techniques on the legacy system. The following chapter will summarize the thesis and will provide a brief outlook on tasks open for future research.

Chapter 13

Conclusion and Outlook

In this thesis, data mining techniques have been evaluated on their suitability for identifying legacy code able to implement services.

After motivating the context of the thesis, the current state of research has been depicted and the contribution of this thesis has been described.

In Part I, the principles of Service-Oriented Architecture have been introduced. Following, a generic architecture for SOAs and the basic concepts of SOAs have been defined.

A process for SOA migration projects has been introduced. The four phases (i) preparation, (ii) conceptualization, (iii) migration and (iv) transition have been introduced and the seven core disciplines (i) Business Modeling, (ii) Legacy Analysis, (iii) Target Architecture, (iv) Strategy Selection, (v) Realization, (vi) Testing and (vii) Cut Over have been described. In addition, this thesis has been brought into the context of such a SOA migration process.

In Part II, data mining has been introduced. The CRISP-DM reference process for data mining has been described. The three categories of data mining approaches – segmentation, classification and link analysis have been explained and their suitability for identifying legacy code able to implement services has been explored briefly. In the following, two data mining techniques have been described in more detail: cluster analysis and link analysis with association rules.

The theory about cluster analysis included the description of similarity and dissimilarity measures for various levels of measurement. Three types of clustering algorithms have been introduced: hierarchical, partitioning and fuzzy clustering algorithms. For the hierarchical type, one agglomerative and one divisive clustering algorithms has been presented. In addition, the TwoStep algorithm has been described. For the partitioning type, the k-Means algorithm has been introduced and for the fuzzy type, the fuzzy k-Means algorithm has been depicted. To conclude the chapter about cluster analysis, the interpretation of clustering solutions has been explained. Ratings on data with known allocation (precision, recall, F-Measure, Rand's index) as well as a rating on data with unknown allocation (silhouette value) have been defined.

The theory about link analysis included the definition of quality measures for association rules (support, confidence and lift) and the explanation of the Apriori approach.

In addition, the data mining tool PASW has been introduced briefly.

In Part III, cluster analysis and link analysis have been applied to an example legacy system in order to identify legacy code able to implement services. The guinea pig legacy system (the SoamigExtractor) has been described and an exemplary business process supported by the tool has been defined. A more detailed description of business processes – workflow models – have been introduced for dynamic analysis. The set-up of the dynamic analysis has been described, too.

For cluster analysis, two approaches have been applied. The first approach analyzed information about the usage in percent of legacy classes during the activities of the exemplary business process that had been captured during dynamic analysis. The results showed, that this approach computed clustering results that were useful to identify legacy code for service implementation. However, the results did not exactly match the expectations of an expert who would have allocated some classes to different services (contrary to their usage). For this reason, information about the package of a class has been added in the second approach. Adding this piece of information lead to results that matched the expert's intuition more precisely.

For link analysis with association rules, two approaches have been applied, too. The first approach tried to identify legacy code for service implementation (similar to the clustering approaches). Results showed that association rules lead to solutions similar to the clustering solutions. That supported the correctness of the clustering solutions. The second approach tried to identify legacy code belonging together. The rules were able to identify legacy code that belongs together. However, the approach generated many rules. So the results would have to bee post-processed to be really useful.

To wrap-up the results of the thesis: the two data mining techniques cluster analysis and association rules have been applied to the identification of legacy code for service implementation, successfully. Cluster analysis resulted in solutions that were more easily interpretable than the results of association rules. However, both approaches are promising techniques to support the identification of legacy code for service implementation.

In future research, the results of this thesis must be confirmed on industrial systems. It must be evaluated if the techniques can deal with bigger, more complex and more eroded legacy systems. In addition, it must be explored if other programming languages than Java affect the suitability of these techniques.

Appendix A

Common Usage of placeholders

A.1 Distance Functions

In the context of distance functions or similarity functions, the following placeholders are used.

Placeholder	Meaning
$p_i[a], q_i[a]$	One data item (= point = object). Each attribute of the item is described as one dimension a.
d	The dimension of an attribute (number of attributes).
$P = \{\dots\}$	A set of items
Subscripts $_i$, $_j$ and $_k$	Subscripts used to describe any item of a set.
Subscripts $_m$, $_n$ and $_o$	Subscripts/superscripts used to define any size of a set. In addition, stands for the last item of the set.
$C = \{C_1, \dots, C_k\}$	A set of k clusters
$C_x = \{p_i, \dots p_n\}$	A cluster C_x containing items
Subscripts $_x$, $_y$ and $_z$	Subscripts used to describe any cluster of a set.
Subscript k	Subscript used to define the size of a set of clusters
$s(p_1, p_2)$	A similarity function computing the similarity between point p_1 and point p_2.
$d(p_1, p_2)$	A distance function computing the distance between point p_1 and point p_2.
d_{L_m}	The Minkowski distance function where m is a parameter (here, m does not stand for the size of a set). Based on the values of m, the resulting functions have special names.

Appendix B

Examples

This section provides detailed examples for various data mining techniques. As the examples take much space and would therefore interrupt the flow of reading, they have been put in the appendix.

B.1 Example for Fuzzy k-means Algorithm

In this example the items given in Table B.1 ought to be clustered using the fuzzy *k*-means algorithm that had been described in Section 6.5.3.

ID	LOC	Inheritance depth
c1	25	3
c2	27	3
c3	29	2
c4	40	0
c5	48	1
c6	52	2

Table B.1: Example data set containing six items with two ratio-scaled attributes each.

The example data set contains six items, representing classes. Each item has two ratio-scaled attributes. *LOC* denotes the lines of code of the class and *inheritance depth* represents the height of the inheritance tree of the class.

The items can be noted as vectors alternatively:

$$\begin{aligned}
p_1 &= \{25;3\}\\
p_2 &= \{27;3\}\\
p_3 &= \{29;2\}\\
p_4 &= \{40;0\}\\
p_5 &= \{48;1\}\\
p_6 &= \{52;2\}
\end{aligned}$$

The fuzzy *k*-means clustering process starts with two initialization steps.

Step B.1.1 (*Set initial cluster means*)

First, the user has to decide how many clusters should be computed. Here, we decide to compute 2 clusters: $k = 2$.
The initial cluster means (seeds) are set to:

$$\mu_1(C_1) = \{30;3\}$$
$$\mu_1(C_2) = \{50;2\}$$

This assignment is chosen arbitrarily.

Step B.1.2 (*Compute initial membership matrix*)

We have to decide how to choose the parameter q (controlling the "fuzziness" of the clusters) and which distance function to use. In this example, we decide to set q to $q = 2$ and to use the *Euclidean distance* to compute distances between items p_i and cluster means $\mu(C_j)$:

$$d(p_i, \mu(C_j)) = \sqrt{\sum_{a=1}^{d} (p_i[a] - \mu(C_j)[a])^2}$$

Now, the initial membership matrix U is computed. For each item, the degree of membership to each seed is computed using the following formula:

$$u_{q=2}(p_i, C_j) = \frac{\left(d^2(p_i, C_j)\right)^{-\frac{1}{2}}}{\sum_{l=1}^{k} \left(d^2(p_i, C_l)\right)^{-\frac{1}{2}}}, \ i = 1, \ldots, n \wedge j = 1, \ldots k \tag{B.1}$$

The upper part of the fraction computes the squared distance to the mean of C_j. The lower part sums up the distances of p_i to *all* C_j. The formula therefore computes the degree of membership for p_i to each C_j.
Now, we compute the initial membership matrix U_1:

U_1	C_1	C_2
p_1	0.83	0.17
p_2	0.88	0.12
p_3	0.94	0.06
p_4	0.49	0.51
p_5	0.11	0.89
p_6	0.08	0.92

Table B.2: Initial membership matrix U_1

After these two initialization steps, the algorithm iteratively updates the cluster means (update step) and the membership matrix that declares the degree of membership of each item to each cluster. Therefore, this step is called the assignment step.

The iteration stops if the membership matrix does not change anymore significantly.

Step B.1.3 (*Recompute cluster means*)

Based on the initial membership matrix, the cluster means are re-computed using the following formula:

$$\hat{\mu}(C_j) = \frac{\sum_{i=1}^{n} u_q(p_i, C_j) p_i}{\sum_{i=1}^{n} u_q(p_i, C_j)} \tag{B.2}$$

The new cluster means are then:

$$\mu_2(C_1) = \{30.3; 2.19\}$$
$$\mu_2(C_2) = \{45.05; 1.39\}$$

Step B.1.4 (*Update membership matrix*)

Now, the membership matrix is updated according to the new means using Equation B.1 on the preceding page:

U_2	C_1	C_2
p_1	0.79	0.21
p_2	0.84	0.16
p_3	0.92	0.08
p_4	0.35	0.65
p_5	0.14	0.86
p_6	0.24	0.76

Table B.3: Updated membership matrix U_2

Step B.1.5 (*Check termination criterion*)

After this first iteration it is checked whether the termination criterion is fulfilled not not. For this purpose, the difference between the old membership matrix U_1 (the initial membership matrix) and the updated membership matrix $\hat{U}$ is computed:

$U_1 - U_2$	C_1	C_2
p_1	0.04	-0.04
p_2	0.04	-0.04
p_3	0.01	-0.01
p_4	0.15	-0.15
p_5	-0.03	0.03
p_6	-0.16	0.16

Table B.4: Checking the termination criterion $U_1 - U_2$

We decide to set the termination criterion to $\varepsilon = 0.1$. Because the largest value of this matrix

$(u(p_4, C_1) = 0.15)$ is bigger than ε, the clustering process continues.

Step B.1.6 (*Recompute cluster means*)

During the update step, the cluster means are recomputed. They are now:

$$\mu_3(C_1) = \{31.21; 2.24\}$$
$$\mu_3(C_2) = \{43.65; 1.34\}$$

Step B.1.7 (*Update membership matrix*)

During the assignment step, the membership matrix is updated according to the new means:

U_3	C_1	C_2
p_1	0.75	0.25
p_2	0.80	0.20
p_3	0.87	0.13
p_4	0.30	0.70
p_5	0.21	0.79
p_6	0.29	0.71

Table B.5: Updated membership matrix U_3

Step B.1.8 (*Check termination criterion*)

Next, the termination criterion is checked again:

$U_2 - U_3$	C_1	C_2
p_1	0.04	-0.04
p_2	0.05	-0.05
p_3	0.06	-0.06
p_4	0.05	-0.05
p_5	-0.06	0.06
p_6	-0.04	0.04

Table B.6: Checking the termination criterion $U_2 - U_3$

The largest value is now $u(p_3, C_1) = 0.06$, and therefore the termination criterion is fulfilled: $\max(\mid U_2 - U_3 \mid) = 0.06 < 0,1 = \varepsilon$. The clustering process terminates and the membership matrix U_3 represents the clustering solution.

Bibliography

[Agrawal et al. 1993] AGRAWAL, Rakesh ; IMIELIŃSKI, Tomasz ; SWAMI, Arun: Mining association rules between sets of items in large databases. In: *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data ; Washington, DC, May 26 - 28, 1993* Bd. 22.2. Washington, D.C., United States : ACM Press, 1993, pp. 207–216. – ISBN 0-89791-592-5

[Agrawal and Srikant 1994] AGRAWAL, Rakesh ; SRIKANT, Ramakrishnan: Fast Algorithms for Mining Association Rules in Large Databases. In: *Proceedings of the 20th International Conference on Very Large Data Bases*. Hove, East Sussex : Morgan Kaufmann, 1994, pp. 487–499. – ISBN 1-55860-153-8

[Arsanjani et al. 2008] ARSANJANI, A. ; GHOSH, S. ; ALLAM, A. ; ABDOLLAH, T. ; GANAPATHY, S. ; HOLLEY, K.: SOMA: A Method for Developing Service-Oriented Solutions. In: *IBM Systems Journal* 47 (2008), No. 3, pp. 377–396

[Bacher et al. 2004] BACHER, Johann ; WENZIG, Knut ; VOGLER, Melanie: *SPSS TwoStep Cluster - A First Evaluation*. 2004

[Berry and Linoff 2004] BERRY, Michael J. A. ; LINOFF, Gordon S.: *Data mining techniques: For marketing, sales, and customer relationship management*. 2. ed. Indianapolis, Ind. : Wiley, 2004. – URL `http://www.gbv.de/dms/hbz/toc/ht014016276.pdf`. – ISBN 0471470643

[Burton Group 2009] BURTON GROUP: *SOA is Dead*. 2009. – URL `http://aboutus.burtongroup.com/pr/bg/SOA-is-dead.aspx`. – last visited: 29.01.2009

[Chapman et al. 1999] CHAPMAN, Pete ; CLINTON, Julian ; KERBER, Randy ; KHABAZA, Thomas ; REINARTZ, Thomas ; SHEARER, Colin ; WIRTH, Rüdiger ; CONSORTIUM, CRISP-DM (Ed.): *CRISP-DM 1.0: Step-by-step data mining guide*. 1999

[Chiu et al. 2001] CHIU, Tom ; FANG, DongPing ; CHEN, John ; WANG, Yao ; JERIS, Christopher: A robust and scalable clustering algorithm for mixed type attributes in large database environment. In: *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, 2001, pp. 263–268. – ISBN 1-58113-391-X

[Cornelissen 2009] CORNELISSEN, Sebastiaan Gijsbert M.: *Evaluating Dynamic Analysis Techniques for Program Comprehension*, Technische Universiteit Delft, The Netherlands, Dissertation, 2009

[Erdmenger et al. 2011a] ERDMENGER, Uwe ; FUHR, Andreas ; HERGET, Axel ; HORN, Tassilo ; KAISER, Uwe ; RIEDIGER, Volker ; TEPPE, Werner ; THEURER, Marianne ; UHLIG, Denis ; WINTER, Andreas ; ZILLMANN, Christian ; ZIMMERMANN, Yvonne: SOAMIG Project: Model-Driven Software Migration towards Service-Oriented Architectures. In: FUHR, Andreas (Ed.) ; RIEDIGER, Volker (Ed.) ; HASSELBRING, Wilhelm (Ed.) ; BRUNTINK, Magiel (Ed.) ; KONTOGIANNIS, Kostas (Ed.): *Joint Proceedings of the First International Workshop on Model-Driven Software Migration (MDSM 2011) and Fifth International Workshop on System Quality and Maintainability (SQM 2011)*, CEUR-WS.org, 2011 (Workshop Proceedings)

[Erdmenger et al. 2011b] ERDMENGER, Uwe ; FUHR, Andreas ; HERGET, Axel ; HORN, Tassilo ; KAISER, Uwe ; RIEDIGER, Volker ; TEPPE, Werner ; THEURER, Marianne ; UHLIG, Denis ; WINTER, Andreas ; ZILLMANN, Christian ; ZIMMERMANN, Yvonne: The SOAMIG Process Model in Industrial Applications. In: MENS, Tom (Ed.) ; KANELLOPOULOS, Yiannis (Ed.) ; WINTER, Andreas (Ed.): *Proceedings of the 15th European Conference on Software Maintenance and Reengineering*. Los Alamitos : IEEE Computer, 2011, pp. 339–342

[Fuhr et al. 2010] FUHR, Andreas ; HORN, Tassilo ; RIEDIGER, Volker: Dynamic Analysis for Model Integration (Extended Abstract). In: *Softwaretechnik-Trends* 30 (2010), No. 2, pp. 70–71. – URL `http://www.soamig.de/paper/fuhr_horn_riediger_wsr10.pdf`. – last visited: 15.06.2010. – ISSN 0720-8928

[Gan et al. 2007] GAN, Guojun ; MA, Chaoqun ; WU, Jianhong: *ASA-SIAM series on statistics and applied probability*. Bd. 20: *Data clustering: Theory, algorithms, and applications*. Philadelphia, Pa. : SIAM [u.a.], 2007. – URL `http://www.gbv.de/dms/hbz/toc/ht015254647.pdf`. – ISBN 9780898716238

[Gartner Group 1996a] GARTNER GROUP ; GARTNER GROUP (Ed.): *Service-Oriented Architectures: Part 1: SSA Research Note SPA-401-068*. 1996

[Gartner Group 1996b] GARTNER GROUP ; GARTNER GROUP (Ed.): *Service-Oriented Architectures: Part 2: SSA Research Note SPA-401-069*. 1996

[Hammer and Champy 2006] HAMMER, Michael ; CHAMPY, James: *Reengineering the corporation: A manifesto for business revolution*. New York, NY : HarperCollins, 2006 (Collins Business Essentials). – ISBN 978-0-06-055953-3

[Han and Kamber 2004] HAN, Jiawei ; KAMBER, Micheline: *Data mining: Concepts and techniques*. 7. Dr. San Francisco, Calif : Kaufmann, 2004 (The Morgan Kaufmann series in data management systems). – ISBN 1558604898

[Hartigan and Wong 1979] HARTIGAN, J. A. ; WONG, M. A.: A K-Means Clustering Algorithm. In: *Applied Statistics* 28, (1979), pp. 100–108

[IBM Corporation 2007] IBM CORPORATION: *IBM Rational Method Composer*. 2007

[Kaufman and Rousseeuw 2009] KAUFMAN, Leonard ; ROUSSEEUW, Peter J.: *Finding Groups in Data: An Introduction to Cluster Analysis: An Introduction to Cluster Analysis*. Wiley-Interscience, 2009. – URL `http://www.lob.de/cgi-bin/work/suche2?titnr=257169418&flag=citavi`. – ISBN 9780470317488

[Kontogiannis et al. 2007] KONTOGIANNIS, K. ; LEWIS, G. A. ; SMITH, D. B. ; LITOIU, M. ; MÜLLER, H. ; SCHUSTER, S. ; STROULIA, E.: The Landscape of Service-Oriented Systems: A Research Perspective. In: *Proceedings of the International Workshop on Systems Development in SOA Environments*, IEEE Computer Society, 2007

[Lallich et al. 2007] LALLICH, Stéphane ; TEYTAUD, Olivier ; PRUDHOMME, Elie: Association Rule Interestingness: Measure and Statistical Validation. In: GUILLET, Fabrice (Ed.): *Quality measures in data mining* Bd. 43. Berlin, Heidelberg, New York : Springer, 2007, pp. 251–275. – ISBN 978-3-540-44911-9

[MacKenzie 2006] MACKENZIE, KenMcCabe FrancisBrown Peter FMetz R. ; MACKENZIE, C. M. (Ed.) ; LASKEY, Ken (Ed.) ; MCCABE, Francis (Ed.) ; BROWN, Peter F. (Ed.) ; METZ, Rebekah (Ed.): *Reference Model for Service Oriented Architecture 1.0: Committee Specification 1.* 2006

[Marchetto and Ricca 2008] MARCHETTO, Alessandro ; RICCA, Filippo: Transforming a Java Application in a Equivalent Web-Services Based Application: Toward a Tool Supported Stepwise Approach. In: *Proceedings Tenth IEEE International Symposium on Web Site Evolution, October 3-4, 2008, Beijing, China (WSE 2008)*, IEEE Computer Society, 2008

[Martin 2010] MARTIN, Wolfgang: *SOA Check 2010: Status Quo und Trends im Vergleich zum SOA Check 2007 bis 2009.* 2010

[Möhring 2009] MÖHRING, Michael: *Lectures on Data Mining I: Winter semester 2009/2010.* 2009

[Rajlich and Bennett 2000] RAJLICH, Václav T. ; BENNETT, Keith H.: A Staged Model for the Software Life Cycle. In: *Computer* 33 (2000), No. 7, pp. 66–71. – ISSN 0018-9162

[Ronen et al. 2007] RONEN, I. ; AIZENBUD, N. ; KVELER, K.: *Service Identification in Legacy Code Using Structured and Unstructured Analysis.* 2007. – URL `http://www.haifa.ibm.com/Workshops/ple2007/present/Service_identification_in_legacy_%20code.pdf`

[Schmitt 2004] SCHMITT, Ingo: *Multimedia-Datenbanken: Retrieval, Suchalgorithmen und Anfragebearbeitung.* Magdeburg, Otto-von-Guericke-Universität, Dissertation, 2004

[Schulze 2007] SCHULZE, Peter M.: *Beschreibende Statistik.* 6., korr. und aktualisierte Aufl. München, Wien : Oldenbourg, 2007. – ISBN 9783486582208

[Sneed et al. 2010] SNEED, Harry M. ; WOLF, Ellen ; HEILMANN, Heidi: *Softwaremigration in der Praxis: Übertragung alter Softwaresysteme in eine moderne Umgebung.* 1. Aufl. Heidelberg : dpunkt.Verl., 2010

[SPSS 2001] SPSS: *The SPSS TwoStep Cluster Component: A scalable component enabling more efficient customer segmentation.* 2001. – URL `http://www.spss.ch/upload/1122644952_The%20SPSS%20TwoStep%20Cluster%20Component.pdf`

[SPSS 2006] SPSS: *Clementine 10.1 Algorithms Guide.* Integral Solutions Limited, 2006

[SPSS 2011] SPSS: *IBM SPSS Modeler - Data mining, text mining, predictive analysis.* 2011. – URL `http://www.spss.com/software/modeler/`. – last visited: 24.02.2011

[Stevens 1946] STEVENS, Stanley S.: On the Theory of Scales of Measurement. In: *Science* 103 (1946), No. 2684, pp. 677–680. – ISSN 0036-8075

[Tan et al. 2009] TAN, Pang-Ning ; STEINBACH, Michael ; KUMAR, Vipin: *Introduction to data mining.* Boston : Pearson Addison-Wesley, 2009. – ISBN 9780321321367

[Thomas et al. 2010] THOMAS, Oliver ; LEYKING, Katrina ; SCHEID, Michael: Serviceorientierte Vorgehensmodelle: Überblick, Klassifikation und Vergleich. In: *Informatik Spektrum* 33 (2010), No. 4, pp. 363–379

[Wahli 2007] WAHLI, Ueli: *Building SOA Solutions Using the Rational SDP*. IBM International Technical Support Organization, 2007 (IBM Redbooks). – ISBN 0738486213

[Zhang et al. 1996] ZHANG, Tian ; RAMAKRISHNAN, Raghu ; LIVNY, Miron: BIRCH: An Efficient Data Clustering Method for Very Large Databases. In: JAGADISH, H. V. (Ed.) ; MUMICK, Inderpal S. (Ed.): *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada*, ACM Press, 1996