Topic:

A consolidated approach for service analysis

Master’s Thesis

in the subject Information Systems
at the Institute of Information Systems, Chair of Interorganisational Systems

<table>
<thead>
<tr>
<th>Institution</th>
<th>Principal Supervisor</th>
<th>Associate Supervisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queensland University of Technology, Australia</td>
<td>Prof. Dr. Rosemann</td>
<td>Dr. Dumas</td>
</tr>
<tr>
<td>Westfälische Wilhelms-Universität Münster, Germany</td>
<td>Prof. Dr. Klein</td>
<td>Dr. Riemer</td>
</tr>
</tbody>
</table>

Submitted by: Thomas Kohlborn
Dorfstrasse 103a
44534 Lünen
(+49) 2306 54581

Submitted: 2008-03-04
Acknowledgements

I dedicate this thesis to my parents and my sister, who have helped me along the way during my studies with their patience, faith and inspiration. I also want to thank my girlfriend Nadine, who has helped me through all the ups and downs while writing this thesis.

I would also want to thank the following people for their contribution to this thesis:

- Prof. Dr. Michael Rosemann: For the opportunity to write my thesis at QUT and to get involved in the projects and activities of the BPM group. Furthermore, I want to thank him for the invaluable ideas, advice and guidance and for all the contacts he provided for my research.

- Dr. Marlon Dumas and Dr. Taizan Chan: For their ideas, support and critical questions about my work.

- Everyone at the BPM group. In particular: Jan Recker, Stephan Clemens, Dr. Jan Mendling, Jessica Prestedge, Guy Redding, Christian Flender and Marcello La Rosa for their ideas and the time spent in leisure activities.

- Stephen McIlvenna: For proof reading my thesis and for all the leisure time activities.

- The Service Ecosystem’s project members: Dr. Alistair Barros, Dr. Paul York, Prof. Dr. Peter Bruza and Christian Prokopp: For the discussions and ideas.

- Prof. Dr. Stefan Klein: For the interesting assignment and the possibility to write my thesis at QUT.

- Dr. Kai Riemer: For his help and administrative support at the University of Münster.

I also want to thank Philipp Heltewig, Jan Heck and Stefan Winkens for their ideas, support, encouragements, valuable insights that helped me along the way and a remarkable leisure time.
Abstract

Dynamic market conditions and changes in customer demand are the major reasons why companies adapt the concepts behind service-orientation. Service-orientation on the business level enables organisations to expose and offer certain operations as business services to their business partners to facilitate on-demand collaboration opportunities. To support the responsiveness and agility of such an on-demand organisation, service-orientation on the technical level fosters the utilisation of software services. These software services encapsulate and expose application functionalities that can be reused and composed based on the business needs to react on changes in the environment. Although the benefits of service orientation are prevalent in literature, comprehensive approaches to identify and analyse business and software services are missing.

This thesis represents a major effort to analyse and consolidate relevant approaches for service analysis. Based on an extensive literature review of 30 relevant, existing service analysis approaches, a study is conducted to evaluate these approaches.

The finding of the analysis is that none of the existing approaches is comprehensive enough to provide a guideline for an organisation to identify and analyse business and software services. However, based on the analysis of the approaches and additional sources, an integrated, comprehensive approach for service analysis is proposed that combines the strengths of the existing approaches, as well as extends and adapts certain aspects that are of importance for an organisation in order to provide a comprehensive guideline for service analysis.

The contribution to theory and practice is significant. Firstly, several authors stated that a comprehensive approach for service analysis is missing and thus a research gap exists which this thesis is attempting to fill. Secondly, this approach will provide an organisation with a methodology to not only understand and document its existing capability from a service perspective, but more importantly, to identify potential new services that may be provided on the business and technical levels.

Keywords: service-orientation, service analysis, service identification, service-oriented organisation, SOA
“It is not the strongest of the species that will survive, or the most intelligent. It is the one most adaptable to change.”

Charles Darwin
# Table of Contents

1 Introduction ................................................................................................................... 1  
1.1 Motivation .............................................................................................................. 1  
1.2 Research questions ................................................................................................. 4  
1.3 Research design ..................................................................................................... 4  
1.4 Outline ................................................................................................................... 6  

2 The context of service analysis ..................................................................................... 7  
2.1 Services .................................................................................................................. 7  
2.2 Service-oriented architectures ............................................................................. 13  
2.3 Service analysis ................................................................................................... 19  

3 Analysis of existing approaches ................................................................................. 20  
3.1 Overview .............................................................................................................. 20  
3.2 Evaluation of existing approaches ....................................................................... 26  
3.2.1 Overview .................................................................................................... 26  
3.2.2 SOA Concept .............................................................................................. 26  
3.2.3 Service delivery strategies .......................................................................... 27  
3.2.4 General analysis approach .......................................................................... 29  
3.2.5 Identification process ................................................................................. 32  
3.2.6 Service classification .................................................................................. 33  
3.2.7 Service design principles ............................................................................ 38  

4 The consolidated approach ......................................................................................... 42  
4.1 Overview .............................................................................................................. 42  
4.2 The derivation of business services ..................................................................... 45  
4.2.1 Overview .................................................................................................... 45  
4.2.2 The preparation phase ............................................................................. 49  
4.2.3 The identification phase ............................................................................. 59  
4.2.5 The prioritisation phase .............................................................................. 65  
4.3 The derivation of software services ..................................................................... 69  
4.3.1 Overview .................................................................................................... 69  
4.3.2 The preparation phase ............................................................................. 76  
4.3.4 The detailing phase ..................................................................................... 85  
4.4 A comprehensive example ................................................................................... 90  
4.4.1 Overview .................................................................................................... 90  
4.4.2 The derivation of business services ........................................................... 91  
4.4.3 The derivation of software services ........................................................... 98  

5 Conclusion ................................................................................................................ 105  
5.1 Research implications ........................................................................................ 105  
5.2 Research limitations ........................................................................................... 106  
5.3 Further research potential .................................................................................. 106  


Bibliography ................................................................................................................... 108
Appendix ....................................................................................................................... 120
  A Technical perspective on SOA ................................................................. 120
  B The Strategic Alignment Model .............................................................. 122
  C Service analysis approaches ............................................................... 124
  D Service design principles ................................................................. 150
  E Detailed overview of the consolidated approach ................................. 164
  F Event-driven Process Chains and the operation model ...................... 166
  G Concluding declaration .................................................................. 168
List of Figures

Figure 1-1: The structure of the thesis ................................................................. 6
Figure 2-1: Service provider and consumer .......................................................... 7
Figure 2-2: Classification of services ................................................................. 8
Figure 3-1: A clustering approach for service analysis approaches ................. 29
Figure 4-1: Phases for the business and software service analysis .................. 44
Figure 4-2: The preparation phase for business services ................................. 45
Figure 4-3: Translating business context into SOA value ................................. 46
Figure 4-4: The identification phase for business services ......................... 49
Figure 4-5: Generic capabilities .......................................................................... 52
Figure 4-6: Hierarchy of capabilities ................................................................. 52
Figure 4-7: A stakeholder model ........................................................................ 56
Figure 4-8: The detailing phase for business services ........................................ 60
Figure 4-9: Business service details ................................................................. 60
Figure 4-10: The prioritisation phase for business services ............................. 66
Figure 4-11: Service Classification Matrix ....................................................... 66
Figure 4-12: Classification of business services ............................................... 68
Figure 4-13: The preparation phase for software services ............................. 70
Figure 4-14: The identification phase for software services ........................... 77
Figure 4-15: Ownership of data ........................................................................ 78
Figure 4-16: Relationship between services and processes ................................ 80
Figure 4-17: Logic encapsulated in services ..................................................... 81
Figure 4-18: The detailing phase for software services .................................... 85
Figure 4-19: Mapping from services to application functionality ................... 88
Figure 4-20: The decomposition of capabilities ............................................... 92
Figure 4-21: The domain model ....................................................................... 94
Figure 4-22: Detailing the quote service ............................................................ 96
Figure 4-23: The financial AHP model for Dash Inc. ........................................ 97
Figure 4-24: The outcome of the AHPs ............................................................. 98
Figure 4-25: Analysing the “Create quote (variant)” process ......................... 100
Figure 4-26: The identified services ............................................................... 102
Figure 4-27: Output of the consolidated approach .......................................... 104
**List of Tables**

Table 1-1: Classification of research topic ................................................................. 5
Table 3-1: Service analysis approaches ......................................................................... 25
Table 3-2: Software service typology ........................................................................... 34
Table 4-1: The three steps for domain decomposition .................................................. 54
Table 4-2: Consolidation of capabilities and service domains ....................................... 57
Table 4-3: Analysis of line of visibility and interaction .................................................. 82
Table 4-4: Mapping of domains and capabilities ........................................................... 94
Table 4-5: Detailing of task, entity and utility services .................................................. 103
Table 4-6: Detailing of process services ....................................................................... 103
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHP</td>
<td>Analytical Hierarchy Process</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>BPM</td>
<td>Business Process Management</td>
</tr>
<tr>
<td>B2B</td>
<td>Business-to-business</td>
</tr>
<tr>
<td>CEO</td>
<td>Chief Executive Officer</td>
</tr>
<tr>
<td>CRUD</td>
<td>Create, Read, Update and Delete</td>
</tr>
<tr>
<td>CSCW</td>
<td>Computer Supported Cooperative Work</td>
</tr>
<tr>
<td>DBMS</td>
<td>Database Management System</td>
</tr>
<tr>
<td>EA</td>
<td>Enterprise Architecture</td>
</tr>
<tr>
<td>EAI</td>
<td>Enterprise Application Integration</td>
</tr>
<tr>
<td>EJB</td>
<td>Enterprise Java Bean</td>
</tr>
<tr>
<td>EPC</td>
<td>Event-driven Process Chain</td>
</tr>
<tr>
<td>ESB</td>
<td>Enterprise Service Bus</td>
</tr>
<tr>
<td>ESSI</td>
<td>European Semantic Systems Initiative</td>
</tr>
<tr>
<td>GDP</td>
<td>Gross Domestic Product</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HSB</td>
<td>Human Service Bus</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>ISDL</td>
<td>Interaction Systems Design Language</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>OAR</td>
<td>Options Analysis for Reengineering</td>
</tr>
<tr>
<td>OASIS</td>
<td>Organisation for the Advancement of Structured Information Standards</td>
</tr>
<tr>
<td>OMG</td>
<td>Object Management Group</td>
</tr>
<tr>
<td>OOAD</td>
<td>Object-oriented Analysis and Design</td>
</tr>
<tr>
<td>OWL</td>
<td>Web Ontology Language</td>
</tr>
<tr>
<td>PIM</td>
<td>Platform Independent Model</td>
</tr>
<tr>
<td>PSM</td>
<td>Platform Specific Model</td>
</tr>
<tr>
<td>ROI</td>
<td>Return On Investment</td>
</tr>
<tr>
<td>SAM</td>
<td>Strategic Alignment Model</td>
</tr>
<tr>
<td>SBDMS</td>
<td>Service-Based Data Management Systems</td>
</tr>
<tr>
<td>SCOR</td>
<td>Supply-Chain Operations Reference</td>
</tr>
<tr>
<td>SDCM</td>
<td>Service Delivery Capability Model</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>SLI</td>
<td>Screen Logic Integration</td>
</tr>
<tr>
<td>SOA</td>
<td>Service-oriented Architecture</td>
</tr>
<tr>
<td>SOAD</td>
<td>Service-oriented Analysis and Design</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
<tr>
<td>USM</td>
<td>Unified Service Model</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Service Description Language</td>
</tr>
<tr>
<td>WSML</td>
<td>Web Service Modeling Language</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Motivation

Initial situation

Due to new communication and delivery channels, global competition\(^1\) and deregulation of markets, companies from all over the world are enabled to conduct business with each other. In this constantly changing environment, organisations struggle to reduce market cycles, enhance customer satisfaction, seize competitive advantage and respond effectively to threats and opportunities.\(^2\) Nevertheless, not only the environment in which companies have been operating in for years has changed, customer demands have changed as well. While mass production of standardised goods might have been appropriate to suit the customer’s needs a few years ago, individualised products and mass customisation are focused on by companies nowadays.\(^3\) The demand for new products has resulted in a shorter time to market that influences the design and execution of business functions and processes.\(^4\)

To deal with changes and to enhance the agility and responsiveness of an organisation, recent ideas promote the restructuring of organisations based on core competencies that differentiate them from their competitors.\(^5\) Competencies are typically composed of several capabilities of an organisation, which in turn are formed through the coordination and integration of activities and processes.\(^6\)

Since companies focusing on core competencies are highly specialised, they need to collaborate with external partners and suppliers to deliver a complete product or service to the end user.\(^7\) Thereby, a business network or ecosystem of collaborating companies could emerge whereas each company focuses on its core competency as well as on collaboration to fulfil customer demand.\(^8\) Hereby, the main challenges emerge how to model and communicate the interactions between the different business partners as well as how to inte-

---

1 Cp. Watson et al. (1997) for an analysis of key issues for different countries and the implications for organisations.
8 A recent survey of Chief Executive Officers (CEOs) across different industries and countries conducted by IBM supports the assumption that the primary targets for organisations are increased responsiveness and enhanced collaboration nowadays. Please refer to http://www-935.ibm.com/services/us/gbs/bus/html/bcs_ceostudy2006.html.
grate the business partners into the own operations and processes, since partners may rely on different operations of one organisation.

One possible solution to this problem is the identification of the business processes that are in focus of how to integrate the business partner. Business processes\(^9\) have governed intra- and inter-enterprise operational modelling for decades and have provided the basis for standardisation and transformation efforts, such as outsourcing\(^10\). However, since business processes typically span over many different functions within an organisation, the issue of modularity and standardisation comes into play, which is needed to expose specific operations to the business network. DAVENTPORT predicts that in the coming years specific processes and capabilities will be outsourced to reduce costs and gain flexibility as well as acquire external expertise. However, in order to outsource internal capabilities, comparability between the internal and external capabilities and processes needs to be assured since the variability in how organisations define processes makes it difficult to contract and communicate about them across organisations.\(^11\)

Based on these considerations, a new way of modelling collaboration and interaction between the partners within a business network is necessary to exploit deeper levels of commonalities across companies and enhanced agility.\(^12\) Business services are seen as an adequate approach to encapsulate specific operations of one organisation that can be exposed to the network. Each service provides a specific contract, which describes the commitments for using that service and provides comparability between different business services of multiple organisations. Hereby, a standardised way of providing business operations as business services that can be used by service consumers within the network is facilitated. This leads to on-demand interaction with business partners, customers, suppliers and further stakeholders of an organisation, since these stakeholders no longer need to negotiate how the interaction between them and the organisation is managed and how integration can be eased. Hereby, the interaction costs between the different participants can be reduced which can lead to the reorganisation of the business structure and the portfolio of offered goods and services to the network.\(^13\) These considerations facilitate the plug-and-play and connect-disconnect properties of the network. Therefore, business services may lead to enhanced agility and responsiveness of an organisation.\(^14\)

---

9 Throughout this thesis, a process is defined as “a completely closed, timely and logical sequence of activities which are required to work on a process-oriented business object.” Becker, Kahn (2003), p. 4.
The emergence of business services as a new way to model and analyse the interactions and relationships in a business network under market conditions can also be extended within the organisation. Malone argues that decentralisation within an organisation as found in markets has several advantages (e.g. encourages creativity and motivation; accommodates flexibility and individualisation) that have to be managed carefully to utilise the advantages of flexibility and responsiveness.15 Eisenstat et al. visualise organisations as markets of resources and opportunities where executives are not only required to negotiate with owners of resources but also compete for resources with other projects and units.16

Such an on-demand business needs to leverage its existent technology to the fullest.17 The concept of service-orientation has manifested itself in the concept of service-oriented architectures in the technical domain of an organisation. Hereby, functionalities of existing applications are comprised by software services that can be composed based on the actual business needs. Changes in the business environment may result in changes of the business process structure that can then rearrange existing software services in order to meet the new business requirements. The result is an agile technical infrastructure that can be leveraged by business processes that in turn utilise the software services of an organisation. The vision for software services goes even further: Software will be developed as services as a long term vision for service evolution.19 Since SOA and services are envisioned to be greatly beneficial for organisations in various areas, Gartner Research predicts that SOA will be used in part in more than 80% of new, mission-critical applications and business processes designed in 2010 (0.7 probability).19 Furthermore, Gartner Research predicts that SOA will be adopted widely in 2 to 5 years as the concept has gone past the trough of disillusionments.20

**Problem statement**

Although service-orientation is affecting different levels within an organisation, no unified way of how to identify and analyse services exists up to this point. Different approaches have been formulated by scholars as well as by industry vendors, but the scope and depths of these approaches vary widely. The need for a consolidated approach has been proposed in a number of publications.21

---

1.2 Research questions

Based on the problem statement described in the previous chapter, the following research questions have been proposed to address the research gap:

1. What are underlying concepts of relevant, extant methods and approaches for identifying and analysing services in an organisation?

2. How can these different concepts of methods and approaches be consolidated, enhanced, and extended to provide a comprehensive guideline for the analysis and identification of services in an organisation?

The first question addresses the need to identify a comprehensive, relevant set of service analysis approaches that needs to be analysed in order to compare the relative strengths and weaknesses of the individual approaches. Based on the answer to the first question, the second research question focuses on the compatibility of the underlying concepts for service analysis associated with each approach. If the analysis results in the identification of certain gaps or shortcomings, the analysed approaches will be extended and enhanced by an analysis of additional references that are not directly related to the identification of services. The consolidated approach would provide an organisation with a methodology to not only understand and document its existing capability from a service perspective, but more importantly, to identify potential new services that may be provided.

1.3 Research design

A research design can be regarded as a guideline to get from the initial research questions to the derivation of conclusions or answers for these questions. Thus, a research design addresses the steps that have to be enforced in order to derive the conclusions. To enhance the overall comprehension of this thesis the underlying research methodology and the research context are explicated.

Research Methodology

For the foundation of this thesis, a design science approach is chosen. Hereby, an artefact is created by the application of the design science approach on the research questions. Hence, this thesis can be positioned within the design science classification scheme as pro-

---

posed by MARCH and SMITH.\textsuperscript{24} Please refer to Table 1-1 for an overview of the scheme and the classification of the topic for this thesis.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
\textbf{Research outputs} & \textbf{Build} & \textbf{Evaluate} & \textbf{Theorise} & \textbf{Justify} \\
\hline
\textbf{Constructs} & & & & \\
\hline
\textbf{Model} & & & & \\
\hline
\textbf{Method} & X & (X) & & \\
\hline
\textbf{Instantiation} & & & & \\
\hline
\end{tabular}
\caption{Classification of research topic}
\end{table}

\textsuperscript{25} Cp. Hevner et al. (2004), p. 79.

The method or the consolidated approach is the artefact that will be built during the course of this thesis. In design science, a method can be seen as a formal algorithm or an informal, textual description of “best practice” approaches or any combination of these two.\textsuperscript{25} Due to time constraints, the evaluation will not be part of the thesis, but will be conducted at a later point in time. A comprehensive example will be proposed at the end of this thesis that visualises and explicates the different parts of the consolidated approaches. Additionally, the example can be seen as a proposed way to validate the approach in practice.

\textit{Research Context}

This thesis was developed as part of the deliverables of the work package “Service Analysis and Design” of the research project “Service Ecosystems Management for Collaborative Process Improvement”, which is conducted by the Queensland University of Technology in collaboration with the industry partners SAP Research and Queensland Government and is sponsored by the Australian Research Council (ARC Linkage Grant: LP0669244).
1.4 Outline

The remainder of the thesis is structured as follows: Chapter 2 will set the foundation for this thesis. The understanding of services and SOA will be explained, as well as the scope of service analysis. Chapter 3 will examine extant approaches that have been proposed for service analysis. The approaches will be compared with a specific set of criteria and evaluated regarding their strengths, weaknesses and compatibilities. If certain aspects for service analysis are insufficiently represented by the existing approaches, further analysis and additional criteria will be presented to extend and enhance the concepts of the existing approaches. Based on this analysis, a consolidated approach for the identification of business and software services will be presented in Chapter 4 that combines the strengths of the analysed approaches. At the end of the chapter, a comprehensive example will be provided in order to apply and visualise the phases and steps for service analysis based on the consolidated approach. Chapter 5 will provide the research implications of this thesis as well as its limitations. Furthermore, the potential for further research based on this thesis will be explicated. Figure 1-1 visualises the structure of this thesis.

![Figure 1-1: The structure of the thesis](image-url)
2 The context of service analysis

2.1 Services

2.1.1 Overview

The term service had emerged with the evolution of economic change long before the Internet was invented. Economies made the transformation from an agricultural state, through industrialisation to a service economy.\textsuperscript{26} Thus, some economies heavily rely on the service industry as a contributor to the Gross Domestic Product (GDP) nowadays. In Germany, for example, the tertiary sector has grown relative to the primary and secondary sector in recent years and forecasts do not predict any stagnation.\textsuperscript{27} Hence, the biggest proportion of the GDP comes from the service industry and is therefore the most important contributor.

However, there is no commonly agreed upon definition of the term service.\textsuperscript{28} Today the term can be found in different areas with different meanings under different circumstances.\textsuperscript{29} Even in the area of science, there are several definitions and understandings of services, which are applicable under specific conditions. However, the definition of a service provided by the World Wide Web Consortium (W3C) captures the essence of a service: \textit{“A service is an abstract resource that represents a capability of performing tasks that form a coherent functionality from the point of view of providers entities and requesters entities.”}\textsuperscript{30} Hereby, the provider entity (in the following: service provider) performs some kind of action on the behalf of the consumer entity (in the following: service consumer) at some time and place and through some channel.\textsuperscript{31} This relationship is visualised in Figure 2-1.

![Service provider and consumer](Source: Adapted from Krafzig, Banke, Slame (2006), p. 14.)

**Figure 2-1:** Service provider and consumer

\textsuperscript{26} Cp. Yahia et al. (2006), p. 22.  
\textsuperscript{27} Cp. Haller (2005), p. 3.  
According to the scope of this thesis, a differentiation between the business science perspective and the computer science perspective is necessary. Figure 2-2 provides a classification scheme for services to serve as a guide through the explanations for the remainder of this chapter.

**Figure 2-2:** Classification of services

### 2.1.2 Business perspective on services

Traditionally the term service has been the focus of researches in business science.\(^{32}\) Even though a standard definition is missing, most scholars could at least agree upon a certain set of constituent features of a *traditional service*. These key features can be derived by looking at various definitions and descriptions proposed in recent scientific research approaches.\(^{33}\) AHLERT and EVANSCHITZKY propose that a service has three distinct features based on the definitions found in literature, namely intangibility, integration of an external factor and synchronous production and consumption of the service.\(^{34}\) Another feature of a service is perishability, meaning that a service cannot be stored.\(^{35}\)

These features lead to the conclusion that least two entities have to interact where one entity performs a certain action on behalf of the other one and that services can be seen as assets.\(^{36}\) ZEITHAML and BITNER agree on the intangibility characteristic of a service. However, they also state that services can also include secondary, tangible elements.\(^{37}\) This fact has also been mentioned by SPATH and DEMÜB: As a first step, they differentiate services based on the service consumer into investment services and end-consumer services. Consumers of investment services are organisations that use the service as an input for their respective production processes. Investment services can further been split up into pure

---


investment services and hybrid products, which encompasses a tangible good and an intangible service.\textsuperscript{38} Hence, services are ubiquitous in our economy, since tangible products are combined with intangible services to provide a combined good for the fulfilment of customer needs.

However, the characteristics of traditional services do not include any references to technological advances in time. The advancements in technologies, like the invention of the Internet and its use in the business environment, have been the main drivers for the emergence of a new term of services, namely \textit{e-services}. Three main views on this kind of services can be identified:\textsuperscript{39}

- Services use the Internet as a user interface to bring service consumer and service provider together (e.g. online shop).\textsuperscript{40}
- Customers can access business specific functions via the Internet in order to strengthen the customer-service provider relationship.\textsuperscript{41}
- Services can be distributed and accessed over electronic networks, which are not only limited to the Internet.\textsuperscript{42}

BAIDA, GORDIJN and OMELAYENKO state that an e-service is also seen as an Internet-based version of the traditional service by some researchers.\textsuperscript{43} These kinds of synonyms make it difficult to differentiate between the intended meanings of the term.

On a more abstract level than the \textit{e-service} and the \textit{traditional service}, the term \textit{business service} can be encountered. Hereby, the focus lays on the grouping of operations of an organisation that can be accessed on-demand.\textsuperscript{44} No differentiation is made, if the service is an actual traditional service, a hybrid product or an e-service, for example. Sometimes this term is also used in the area of computer science, to describe a technical implementation of business related activities.\textsuperscript{45}

In business science, the term web service can be encountered as well. However, this term is typically used in computer science and will be described in the following chapter.

\begin{thebibliography}{99}
\bibitem{Roland} Cp. Roland, Kannan (2003), p. 38.
\bibitem{Baida1} Cp. Baida, Gordijn, Omelayenko (2004), p. 3.
\end{thebibliography}
2.1.3 Technical perspective on services

In general, the term web service refers to a technology approach for implementing a SOA.\textsuperscript{46} Therefore, definitions of this term are rather technical and describe the application of certain standards. The following definition has been proposed by the W3C:

\textit{“A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.”}\textsuperscript{47}

Three elements can commonly be identified in definitions of web services, namely software/applications, certain functionalities and the Internet.\textsuperscript{48} These elements are also partly applicable to e-services. However, the e-service definitions in the area of computer science are used inconsistently. Thus, some scholars use the terms e-service and web service synonymously while others use the term e-service to abstract from certain elements of the web service definition. However, since web services use certain standards for the implementation, broader definitions of services can be identified that abstract from the implementation-specific standards, but agree on the other elements. Krafzig, Banke and Slama, for example, define a service as a:

\textit{“[...] remotely accessible, self-contained application module, which is accessible to a user by an application frontend.”}\textsuperscript{49}

This definition describes the notion of a software service, since it abstracts from standards and implementation issues that are addressed by most definitions of web services. Since this definition focuses on the area of computer science, the term “service” as used in this definition cannot be regarded as a traditional service. The notion of a software service is not consistently used in scientific literature, since most authors just refer to software services as services. However, the idea is that software services can be described in an implementation-independent and “semantic” fashion.\textsuperscript{50} Hence, software services can be seen as the parent term for e-services and web services.

\textsuperscript{46} The term SOA will be discussed in Chapter 2.2.
\textsuperscript{47} W3C (2004).
2.1.4 Consolidated perspective

As presented on the previous chapters, there is no clear definition of the term service, neither in the area of business science nor in the area of computer science. The term service by itself without any prefixes is sometimes used as a synonym for web service as well as software services as well as for traditional services. Some scholars even use the term web service and e-service synonymously\(^{51}\), while others speak about an e-service as an abstraction of the specific realisation of a web service.\(^{52}\) Thus, the term e-service is not used consistently in the area of computer science and will not be used any further in the course of this thesis. However, two specific derivations of the term service will be used, namely business service and software service.

**Business service**

The term business service is used to describe a specific set of actions that are performed by an organisation.\(^{53}\) Since the operations of an organisation can be analysed on different granularity levels, business services can represent these operations on different levels as well. Hereby, one has to abstract from any distinction between investment services or end-consumer services. A business service comprises a certain amount of business operations including a description of its behaviour rendered by other business services on which it depends.\(^{54}\) The business logic that is encapsulated inside a business service is accessible through a set of business service operations that share the associated activities among themselves.\(^{55}\) These operations are exposed to the service’s environment so that they can be invoked by various service consumers including other business services.\(^{56}\) Furthermore, for each business service there has to be a business service contract specifying the commitments made to and needed from the service’s potential consumers.\(^{57}\) As mentioned before, this concept assumes that a business represents some amount of business operations that delivers value to its service consumer. It abstracts from the realisation of the business service operations, as it may or may not leverage existing IT infrastructure. Another meaning of the term business service can be found in literature as well. Hereby, a business service is seen as a reusable combination of IT components that deliver a business-oriented service.

---

\(^{54}\) Cp. Sanz, Nayak, Becker (2006b), p. 63. Erl supports this proposition by stating that “[…] an organisation that carries out tasks associated with its purpose or business is also providing a service. As long as the task or function being provided is well-defined and can be relatively isolated from other tasks, it can be distinctively classified as a service.” Erl (2007), p. 68.
Thus, a business service in regards to that terminology is a software service, since it is implemented and realised as an IT solution.58

**Software service**

In this thesis, the term *software service* will be used to describe a part of an application system, which can be consumed separately by several entities.59 A software service consists of a *service contract*, at least one *service interface* and an *implementation* including data and business logic.60 An informal specification of the service including its purpose, functionality, constraints and usage can be defined in a service contract. Additionally, the service contract may define a formal interface that provides an abstraction of specific technology dependent characteristics of the service (e.g. network protocol, programming language, etc.) It also may include additional information regarding detailed semantics on functionality and parameters. A contract provides a formal definition of the service endpoint, each service operation, the input and output messages supported by the service and the rules and characteristics of the service.61 The interface exposes available operations of the service to service consumers. Therefore, the logic along with the data can be accessed via an interface that has to fulfil the requirements stated in the service contract. The implementation of the service comprises the required business logic as well as the required data. The implementation can be seen as the technical realisation that fulfils the service contract.62

**Comparison between business and software services**

The idea behind business and software services is obviously similar as indicated by the common terminology. Both have some kind of contract that is used to specify the commitments made by the specific service. In addition, the contract clearly defines the expected outcome of each of the exposed operations. However, since a business service comprises a substantially large amount of an organisation’s operations under each of its business service functions, the business service contract of a business service describing the interaction of the service and its consumers is much richer and more complex than the contract of a software service.63 The focus of the contract is also different between the two different types of services. The contract of a software service details the syntax and semantics of invoking the service including the format of message exchange, the software component involved, etc. A business service contract describes aspects that are relevant to the

---

The context of service analysis

communication between business entities (e.g. service pricing and billing methods, underlying operational processes, etc.). A business service contract needs to be explicit enough such that an organisation can operate and rely on a business service partner. Nonetheless, the contracts of both types of services encapsulate a description of the operations that can be invoked by service consumers. Based on the commonalities in the terminology of the service characteristics, Nayak et al. proposed a Unified Service Model (USM) that visualises the relationships between the different elements of a service.

2.2 Service-oriented architectures

2.2.1 The concept of a SOA

There is no commonly agreed upon definition of what a SOA is or what it actually looks like. Most publications by scholars, standardisation consortiums or vendors allocate the concept of a SOA to a purely technical domain. For example, the W3C describes a SOA as a form of distributed system architecture, where a distributed system consists of diverse concrete software agents in different environments that work together to perform certain tasks. Krafzig, Banke and Slama define a SOA as a software architecture [...]. A SOA can also be seen as a multiple-layer, distributed information system, in which parts of the application logic of an organisation are encapsulated and exposed as [software] services as proposed by Legner and Vogel. Schemm et al. propose three different perspectives on such a technical SOA, namely SOA as a middleware solution, SOA as a paradigm for structuring information systems and SOA as a layered distributed software architecture. The technical definitions of a SOA can be allocated to either one of these perspectives.

However, the allocation of a SOA to a technical domain falls too short since the whole concept of a SOA can be applied on the business level as well. Thereby, technical concerns only play a secondary role. One of the few definitions that are applicable on a broad scope has been formulised by the Organisation for the Advancement of Structured Information Standards (OASIS). OASIS defines a SOA as a “paradigm for organising and util-

72 Please refer to Appendix A.
74 Cp. Steen et al. (2005), p. 133.
izing distributed capabilities that may be under control of different ownership domains.\textsuperscript{75}

Hereby, services are the mechanisms by which the needs and capabilities are brought together.\textsuperscript{76} Thus, SOA can be seen as a paradigm for organising capabilities on the business domain as well as on the technical domain.

Two specific aspects of a SOA must be taken into account when analysing and designing services, namely visibility and interaction. One has to ensure that the service provider and consumer are able to see each other. This is true for any type of relationship between service provider and consumer regardless of the specific instantiation of these entities (e.g. applications or humans). The visibility is influenced by three factors, namely willingness, reachability and awareness. The service consumer needs to be aware of the existence of the service provider; both need to be willing to interact with each other and the service provider needs to be reachable for the service consumer.\textsuperscript{77} The interaction is dependent on the visibility because service consumer and provider cannot interact if they do not see each other. For a successful interaction, the service consumer needs to know the type of inputs and outputs of the service and the actions that can be performed against the service.\textsuperscript{78}

\textbf{2.2.2 Constituent Elements}

Three constituent elements of a SOA can be identified, namely services, a service repository and a service bus, which will be described in the following part.\textsuperscript{79}

\textit{Services}

Services are regarded as the core element of a SOA. As previously analysed, the concept of a SOA is equally applicable for the business domain as well as for the technical domain and thus comprises business and software services.

\textit{Service Repository}

A service repository facilitates the discoverability of services since it is used to provide additional information about services. The concept of a service repository is equally applicable for the IT domain for software services, as well as for the business domain for business services.\textsuperscript{80} However, the information provided within the service repository may be different since the information that is needed to interact with a software service is more

\textsuperscript{78} Cp. OASIS (2006), pp. 15 ff.
The context of service analysis

technical than for business services, which focus primarily on non-technical issues. The provided information can encompass for example information about the service provider, contact persons, fees, security issues and available service levels. The service repository may also include a rating functionality to provide support for discriminating between services. If the service repository is exposed beyond the boundaries of an organisation, detailed information about service versioning, service subscription, billing and other legal issues need to be addressed by the service repository. If the number of services not belonging to a service repository exceeds an amount that is no longer manually manageable, a service repository is needed to handle and manage those services. Especially when new services are developed or new business goals should be executed by services or service compositions, a service repository inhibits the development of redundant services. A service repository does not explicitly address any specific technology. A printed stack of papers describing each service’s attributes can also be seen as a service repository. Nonetheless, a database is typically used to save the descriptions of services.

Service Bus

The service bus in general is used to connect service consumers and service providers. Since a SOA is most often seen as a technological approach, the service bus, or Enterprise Service Bus (ESB), is typically referred to as a set of products and concepts to leverage communication between different software services. The ESB may also provide common functionalities, such as logging, auditing, security and transactions. An ESB avoids the proliferation of point-to-point connections with custom-built application interfaces at both ends. Thus, all services connect to the ESB to communicate with each other in an independent fashion. However, the concept of a service bus as a support and integration layer for autonomously acting services is equally applicable on a business level. Bieberstein ET AL. propose the Human Service Bus (HSB) with analogues objectives as the enterprise service bus in the IT domain. Since business services may be decomposed into finer-grained business services that can be assigned to teams, departments, etc., the HSB supports the collaboration between business services by providing an integration layer. The HSB can be used to advertise business services, offer workflow tools and planning tools to support the

coordination across business services. These tools may also comprise other tools to support Computer Supported Cooperative Work (CSCW), for asynchronous interaction, such as project databases, portals and forums, and synchronous interaction, such as instant messaging, webcasts and electronic whiteboards.

2.2.3 Proposed benefits of a SOA

General benefits of a SOA

Numerous benefits of SOA have been stated, but validation of these benefits by case studies can hardly be found. Krafzig states that enhanced flexibility and cost efficiencies of applications are proposed benefits of a SOA. One of the reasons for these proposed benefits is founded on the reusability potential of services. If functionality or operations can be reused instead of redeveloped, a SOA seems to be more cost efficient. However, vom Brocke analysed existing case studies that addressed the calculation of the “Annual Return On Investment (ROI)” for SOA projects. He concluded that the calculations described in these studies are not comprehensible. SOA is also proposed to be advantageous, when various applications on different platforms and technologies have to communicate with each other based on the adherence of specific standards. Therefore, one promise of a SOA is the inter- and intraorganisational integration of heterogeneous application systems that need to be integrated across ownership boundaries. Various benefits have been proposed by Richter et al., including the flexible adaptation of new communication channels. Hereby, service consumers can be granted access to business logic that is accessible through standardised service interfaces via multiple communication channels. Additionally, based on the idea to group functionalities within services that can be composed based on business requirements, a SOA can provide foundation for organisational agility and adaptability to enhance the responsiveness of an organisation towards (unforeseen) changes in its environment. However, Chen states that a SOA alone cannot achieve the desired organisational agility and must rely on sound alignment between different do-

---

mains.104 ERRADI, ANAND and KULKARNI; ARSANJANI and ERL argue similarly and propose that the benefits of SOA are a better business and IT alignment and organisational agility.105 These two proposed benefits directly relate to the problem statement in Chapter 1.1. Thus, they will be analysed further in the following part.

**Strategic business and IT alignment and agility**

Earlier studies indicate that a close business and IT alignment will result in higher business value for an organisation. The Strategic Alignment Model (SAM) developed by HENDERSON and VENKATRAMAN has been used to argue that the strategic alignment between the business and IT domains results in higher business value and increased payoffs on the IT infrastructure.106 The authors propose that an organisation is typically made up of four domains. A basic differentiation is made between the business and IT domains, which can be further be divided into an external and an internal domain. The external business domain and the IT domain focus on their respective strategy towards their environment. Hereby, a strategy reflects decisions that align resources and capabilities with environmental threats and opportunities.107 The internal domains focus on the structure and processes of the business and IT domains. However, strategic decisions should consider all four domains of an organisation. If an organisation focuses primarily on external issues (business and IT strategies without considering the internal domains), the difficulties of designing the underlying processes of the internal domains could be underestimated. Thus, the SAM advocates cross-functional relationships that consider external and internal domains, as well as business and IT domains. The driver behind the alignment might be the IT strategy that influences the design of the internal IT processes that consequently influence the organisational structure of the business. This concept describes the multivariate cross-domain perspective that works on the premise that strategic alignment at an organisational level can only occur when three of the four domains are in alignment.108 However, one has to keep in mind that strategic alignment is not an event, but rather a process of continuous adaptation, innovation and differentiation.109

Based on the SAM, different approaches have been used to analyse and validate the model. Alignment between the IT strategy and business strategy can be measured in different ways.110 A possible approach is to analyse how the IT plans support the business plan and

---

106 Please refer to Appendix B for a description of the Strategic Alignment Model.
110 For a comprehensive overview about strategy and alignment, please refer to Avison et al. (2004).
vice versa. Nevertheless, since planned strategy can be different from the realised strategy, other approaches have been focussing on performance metrics of a company. These metrics may include market share, revenue growth, return on investment, net profits and product quality. Recent studies investigate the relationships between strategic alignment, IT flexibility and IT effectiveness. However, Tallon and Kraemer proposed a perception-based approach to IT business value using multiple process-level measures at various points throughout an organisation. As part of their studies, they concluded that the ability to realise payoffs from IT is due to a presence of strategic alignment.

This thesis will focus on the business strategy as the driver. This perspective assumes that the articulated business strategy shapes the internal organisational processes as well as the IT infrastructure. The relationships between the domains can be described as followed: The business strategy articulates the business requirements based on the analysis of the external environment. Business processes and organisational structures are established to fulfil the business requirements. To accomplish this objective the IT capabilities have to be leveraged as effectively and efficiently as possible. If the environment is rapidly changing and business requirements change as well, processes may become less effective through increased discrepancy between requirements and practice. Depending on the business agility, the processes may be adjusted, so that they retain their effectiveness. Nevertheless, frequent changes to the organisation’s processes can result in a misalignment between business and IT, resulting in decreased business agility. This in turn may impede the ability to change the processes based on changing requirements, since the IT capabilities cannot be changed as rapidly as required. A possible solution to solve this problem is to enhance IT agility that results in better business and IT alignment. Thus, if an organisation wants to enhance its agility, the underlying IT will serve as one of the core enablers.

Service-orientation as a concept to respond to environmental changes should not and must not focus solely on the IT domain of an organisation as addressed by most scholars, but should consider the changes that need to take place within the business domain as well. Furthermore, it has been shown that a close business and IT alignment delivers higher

---

115 Cp. Tallon, Kraemer (2002), p. 14. Their studies also identified a paradox. Companies that pursue ever-higher levels of strategic alignment may realise an adverse outcome in which greater strategic alignment leads to a decline in payoff from IT. This might be due to the inability of the organisation to be flexible regarding changes in the external environment since the IT becomes the limiting factor. However, service-orientation is supposed to overcome this “paradox”, but no empirical validation could be identified.
value to the organisation. This alignment can be achieved via different approaches as proposed by Chen.\textsuperscript{119} Software capabilities should be exposed as business-meaningful services.\textsuperscript{120} Hence, software services within the IT domain should represent or align with services in the business domain to bridge the gap between business and IT. Furthermore, services are used as a binding concept between the domains on a communicational level.\textsuperscript{121}

2.3 Service analysis

Based on the broad research area of services and SOA that is addressed in this thesis, service analysis cannot be allocated to distinct phases of lifecycles that have been proposed in academic literature. These lifecycles lack validation and a clear separation of service lifecycles and SOA lifecycles cannot be identified. Leaving these considerations aside, even between the proposed lifecycles, a unified definition of service analysis is not anticipated. For example, Marks and Bell state that “service analysis is performed on candidate business services in order to simplify complexities of service abstraction and apply logical modelling operations to better understand candidate services.”\textsuperscript{122} Papazoglou and van den Heuvel propose that “service-oriented analysis is a phase during which the requirements of a new application are investigated.”\textsuperscript{123} In addition, ERL states that service-oriented analysis is a stage where individual services are modelled and service layers are mapped out.\textsuperscript{124}

For this thesis, service analysis is seen as a comprehensive phase, where the concept of service-orientation is applied to analyse the capabilities provided by an organisation and to identify services consequently. This notion includes the analysis of the impact of service-orientation on the business level as well as on the technical level. All activities that result in the identification of services are comprised in this phase including the specification of a service’s operation that may change the composition of the set of services. Service identification is typically one of the first tasks within any SOA initiative and errors made during this phase can flow through detailed design and implementation phases consequently.\textsuperscript{125} The outcome of this approach is a first draft of a SOA.

\textsuperscript{119} Cp. Chen (2008).
\textsuperscript{121} Cp. Chen (2008), pp. 152 ff.
\textsuperscript{122} Marks, Bell (2006), p. 58.
Analysis of existing approaches

3 Analysis of existing approaches

3.1 Overview

Based on the broad scope of this thesis an extensive literature review has been conducted using online academic libraries and search engines such as IEEE, EBSCO, Springer, Pro-Quest, ACM, Emerald and ScienceDirect\textsuperscript{126} and using available resources of the libraries at the Queensland University of Technology and Westfälische Wilhelm-Universität Münster. The search terms have been manifold since there is no standardised terminology in this area of research.\textsuperscript{127}

3.2 Comparison of existing approaches

Various approaches regarding service analysis and design have been proposed by different scholars and researches. Table 3-1 provides an overview of a set of 30 approaches including specific characteristics for each approach that are described and analysed in Appendix C. These approaches serve as the underlying basis for the consolidated approach. However, additional approaches have been described in Appendix C.e that do not directly relate to the analysis and identification of services. For example, SPROTT ET AL. and SEWING, ROSEMANN and DUMAS describe different characteristics of processes that serve as primary indicators for the suitability of a process to be service-enabled.\textsuperscript{128}

Different criteria have been selected in regards to the requirements of the consolidated approach and the comparability of the different approaches. These criteria can be used to compare and analyse the existing approaches:

- 	extit{SOA Concept:} The first criterion characterises the approaches based on the underlying concept of a SOA. A differentiation is made between approaches that focus primarily on the derivation of business services and approaches that address the derivation of software services or both.

- 	extit{Delivery strategy:} Three strategies can be distinguished to deliver a SOA. The top-down strategy (T) comprises the analysis of business requirements to derive services that are closely aligned with the organisation’s business logic.\textsuperscript{129} The bottom-


\textsuperscript{127} Please refer to Chapter 2.

\textsuperscript{128} These approaches have not been included in Table 3-1, but references are made during the description of the consolidated approach, when the concepts of these approaches are applicable.

up strategy (B) focuses on the derivation of services based on an analysis of the legacy systems on a as-needed basis. The bottom-up strategy is generally suited to develop services that accomplish application specific requirements. Requirements can comprise point-to-point integration between different systems (e.g. legacy systems). The existing IT-infrastructure and application systems of a company serve as a starting point for the bottom-up strategy. The meet-in-the-middle strategy (M) combines both strategies. Thus, services are identified and analysed in a top-down manner, whereas the existing legacy systems are analysed in a bottom-up approach to evaluate the readiness for service enablement.

- **General analysis approach:** The approaches can be clustered into two general classes, based on their focus. The first class encompasses approaches that focus on the business domain. These approaches identify and analyse services from a business perspective (e.g. processes, use cases, domains, capabilities, etc.). The second class focuses mainly on the IT domain to derive services based on existing applications. This criterion describes the background of the delivery strategy (e.g. what business artefacts are used for service analysis).

- **Identification process:** The various approaches can be compared according to the respective structure of their identification process for services. Approaches may describe the identification of services on a very abstract level or on a more detailed level including different activities and steps. Some approaches cover more phases than the service analysis phase. This is the reason why a detailed description of the different phases and/or steps for each method has not been included for the overview in Table 3-1. However, the number of phases/steps can be used as an indicator for the degree of prescription of the specific approach.

- **Service classification:** The approaches also differ in their description of different types of services based on different objectives. Some approaches only provide guidelines to derive a service in general, whereas other approaches distinguish between different types of services. Similar to the last criterion, there is no common classification framework for services. Hence, only generic descriptions of the different service types that are distinguished within one approach are annotated in Table 3-1.

---

133 As explained previously, no consolidated, unified life cycle for a SOA or a service could be identified in recent literature. If an author of an approach describes more than the service analysis phase, the approach itself will include a specification of a proposed life cycle. See for example Erl (2005); Papa-zoglou, van den Heuvel (2006).
• **Design principles**: Another criterion that can be used to compare the different approaches is the description of design principles. Most approaches describe desired characteristics of services, such as loose coupling, but fail to describe how these characteristics can be achieved. However, some approaches detail specific design principles and their application in the service analysis phase. Since the design principles are not described using a common terminology by the different authors, only the number of described principles per approach is provided in Table 3-1.

• **Documentation**: Furthermore, the approaches differ in their documentation. Most approaches are illustrated using fictitious examples, others use real case studies and a few derive services on a purely theoretical level without providing any examples. This criterion is used to analyse which approaches have already been applied in practice and which approaches have purely been derived on a theoretical basis.

• **Proprietary**: The concept behind SOA has been facilitated and supported by vendors, such as IBM, SAP, Sun and Microsoft. These organisations also have their own methodologies and concepts to identify and analyse services. These approaches typically lack insights into details, whereas non-proprietary approaches are openly available.

• **Stakeholder**: Since one of the primary issues of service-orientation is the analysis of the interaction between service consumers and providers, which is based on the utilisation of services, the analysis of stakeholders of the organisation should be addressed by the approaches as well.
## Analysis of existing approaches

<table>
<thead>
<tr>
<th>Approach author (year)</th>
<th>Criteria</th>
<th>SOA concept</th>
<th>Delivery strategy</th>
<th>General analysis approach</th>
<th>Additions and Comments</th>
<th>Identification process</th>
<th>Service classification / layers</th>
<th>Service design principles</th>
<th>Documentation</th>
<th>Proprietary</th>
<th>Regards to stakeholder</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Business services</td>
<td>T</td>
<td>Capability decomposition</td>
<td>Originally for software services</td>
<td>3 models</td>
<td>n.a.</td>
<td>Implicit</td>
<td>Example</td>
<td>Yes</td>
<td>External / Internal actors / owners</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Business services</td>
<td>T</td>
<td>Domain decomposition</td>
<td>Business component model</td>
<td>4 steps</td>
<td>3 basic service types</td>
<td>Implicit</td>
<td>Example</td>
<td>Yes</td>
<td>n.a.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Business service</td>
<td>T</td>
<td>Domain decomposition</td>
<td>Comments and gaps</td>
<td>10 steps</td>
<td>Generic business service</td>
<td>Implicit</td>
<td>Example</td>
<td>Yes</td>
<td>n.a.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Business service</td>
<td>T</td>
<td>Entity analysis</td>
<td>Service identification and clustering</td>
<td>4 steps</td>
<td>Business service as part of hierarchy</td>
<td>Implicit</td>
<td>Example</td>
<td>No</td>
<td>n.a.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Business-Software service</td>
<td>T</td>
<td>Domain decomposition</td>
<td>3 analysis phases</td>
<td>6 service types</td>
<td>4 service types</td>
<td>Implicit</td>
<td>Example</td>
<td>No</td>
<td>n.a.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software service</td>
<td>T</td>
<td>Process models</td>
<td>3 analysis phases</td>
<td>4 software service types</td>
<td>4 service types</td>
<td>Implicit</td>
<td>Example</td>
<td>No</td>
<td>n.a.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software service</td>
<td>M</td>
<td>Process and entity models</td>
<td>IT suitability</td>
<td>11 service types; 3 layers</td>
<td>5 service types</td>
<td>Implicit</td>
<td>Example</td>
<td>No</td>
<td>n.a.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software service</td>
<td>T/M</td>
<td>Process and domain analysis</td>
<td>IT analysis</td>
<td>5 service types</td>
<td>5 service types</td>
<td>Implicit</td>
<td>Example</td>
<td>Yes</td>
<td>Implicit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software service</td>
<td>M</td>
<td>Processes and solution maps</td>
<td>IT analysis</td>
<td>Not structured</td>
<td>10 guidelines</td>
<td>Implicit</td>
<td>Example</td>
<td>Yes</td>
<td>Implicit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software service</td>
<td>M</td>
<td>IT criteria</td>
<td>Example</td>
<td>3 phases</td>
<td>Example</td>
<td></td>
<td>Example</td>
<td>Yes</td>
<td>Implicit</td>
</tr>
</tbody>
</table>

### Notes
- **SOA concept:** Business services, Business service, Business-Software service, Software service, Software service, Software service, Software service, Software service
- **Delivery strategy:** T, T, T, T, T, M, T/M, M, M
- **General analysis approach:** Capability decomposition, Domain decomposition, Domain decomposition, Entity analysis, Domain decomposition, Domain decomposition, Domain decomposition, IT suitability, IT analysis, IT analysis, IT criteria
- **Additions and Comments:** Originally for software services, Business component model, Comments and gaps, Service identification and clustering, 4 drivers; 16 indicators for services
- **Identification process:** 3 models, 4 steps, 10 steps, 4 steps, 3 phases, 8 analysis phases, 3 analysis phases, 3 analysis phases, Not structured, 3 phases
- **Service classification / layers:** n.a., 3 basic service types, Generic business service, Business service as part of hierarchy, 6 service types, 4 software service types, 4 service types, 11 service types; 3 layers, 5 service types, 5 service types
- **Service design principles:** Implicit, Implicit, 2, n.a., Implicit, 7, 4, 8, 2, 10 guidelines
- **Documentation:** Example, Example, Example, Example, Case study, Case study, Example, Case study, Example
- **Proprietary:** Yes, No, Yes, Yes, No, No, No, No, Yes
- **Regards to stakeholder:** External / Internal actors / owners, External actors, n.a., External actors, Internal owner, Actors, n.a., n.a., Implicit
## Analysis of existing approaches

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Criteria</td>
<td>SOA concept</td>
<td>Delivery strategy</td>
<td>General analysis approach</td>
<td>Technical-driven service analysis</td>
<td>Additions and Comments</td>
<td>Identification process</td>
<td>Service classification / layers</td>
<td>Service design principles</td>
<td>Documentation</td>
<td>Proprietary</td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td>Business-driven service analysis</td>
<td>Domains and processes</td>
<td>IT analysis</td>
<td>Not structured</td>
<td>2 service types</td>
<td>Implicit</td>
<td>Example</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>T</td>
<td>Process decomposition</td>
<td>Process reфactoring</td>
<td>Portfolio analysis, code restructuring</td>
<td>Not structured</td>
<td>One generic service</td>
<td>n.a.</td>
<td>Example</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Software services</td>
<td>M</td>
<td>Use case refactoring</td>
<td>Use case</td>
<td>Code restructuring</td>
<td>6 phases</td>
<td>2 service types</td>
<td>Implicit</td>
<td>Case study</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>B</td>
<td>Starting point business operation</td>
<td>Business requirements</td>
<td>Multiple approaches, but no details</td>
<td>3 phases</td>
<td>One generic service</td>
<td>Implicit</td>
<td>Example</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>B</td>
<td>General analysis approach</td>
<td>IT analysis</td>
<td>Process analysis</td>
<td>2 phases</td>
<td>One generic service / decomp.</td>
<td>Implicit</td>
<td>Case study</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td>Technical-driven service analysis</td>
<td>IT analysis</td>
<td>Domain analysis</td>
<td>3 phases</td>
<td>2 service types</td>
<td>Implicit</td>
<td>Example</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td>Additions and Comments</td>
<td>Comparison with objects and components</td>
<td>Goal modelling</td>
<td>12 phases</td>
<td>2 service types</td>
<td>Implicit</td>
<td>Example</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td>Identification process</td>
<td>Features</td>
<td>Case study</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td>Service classification / layers</td>
<td>3 steps</td>
<td>2 phases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td>Service design principles</td>
<td>Implicit</td>
<td>3 phases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td>Documentation</td>
<td>Implicit</td>
<td>5 phases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td>Proprietary</td>
<td>Implicit</td>
<td>3 phases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td>Regards to stakeholder</td>
<td>Implicit</td>
<td>12 phases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td></td>
<td>Implicit</td>
<td>2 service types</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td></td>
<td>Implicit</td>
<td>2 service types</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td></td>
<td>Implicit</td>
<td>3 phases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td></td>
<td>Implicit</td>
<td>5 phases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td></td>
<td>Implicit</td>
<td>3 phases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td></td>
<td>Implicit</td>
<td>12 phases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td></td>
<td>Implicit</td>
<td>2 service types</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td></td>
<td>Implicit</td>
<td>2 service types</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td></td>
<td>Implicit</td>
<td>3 phases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td></td>
<td>Implicit</td>
<td>5 phases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td></td>
<td>Implicit</td>
<td>3 phases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Software service</td>
<td>M</td>
<td></td>
<td>Implicit</td>
<td>12 phases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------</td>
<td>-----------------</td>
<td>---------------</td>
<td>---------------------------------------</td>
<td>------------------------</td>
<td>----------------------------------</td>
<td>--------------</td>
<td>------------</td>
<td>-------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Criteria</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOA concept</td>
<td>Software service</td>
<td>Software service</td>
<td>Software service</td>
<td>Software service</td>
<td>Business-Software service</td>
<td>Software service</td>
<td>Software services</td>
<td>Software services</td>
<td>Software services</td>
<td>Software services</td>
</tr>
<tr>
<td>Delivery strategy</td>
<td>T/M</td>
<td>M/B</td>
<td>T</td>
<td>M</td>
<td>B</td>
<td>T/M</td>
<td>M</td>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business-driven service analysis</td>
<td>Event-driven service identification</td>
<td>Business processes</td>
<td>Activity Diagrams</td>
<td>Process analysis</td>
<td>Business requirements</td>
<td>Goal modelling</td>
<td>Use cases</td>
<td>Business requirements</td>
<td>MDA, Use cases</td>
<td></td>
</tr>
<tr>
<td>Technical-driven service analysis</td>
<td>IT analysis</td>
<td>Decomposition of systems</td>
<td>Application portfolio analysis</td>
<td>Code restructuring</td>
<td>IT analysis</td>
<td>IT analysis</td>
<td>IT analysis</td>
<td>Class diagrams</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Additions and Comments</td>
<td>Adaptable to process analysis</td>
<td>EJB</td>
<td>Adaptable to process analysis</td>
<td>Own language (ISDL)</td>
<td>Extensive approach, no details</td>
<td>Extensive approach, but no details</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identification process</td>
<td>4 steps</td>
<td>Not structured</td>
<td>8 steps</td>
<td>5 phases</td>
<td>5 phases</td>
<td>7 steps</td>
<td>n.a.</td>
<td>n.a.</td>
<td>3 models</td>
<td></td>
</tr>
<tr>
<td>Service classification / layers</td>
<td>2 generic service types</td>
<td>n.a.</td>
<td>One generic service</td>
<td>2 generic services</td>
<td>n.a.</td>
<td>2 generic service types</td>
<td>Composed /atomic services</td>
<td>n.a.</td>
<td>n.a.</td>
<td>One generic service</td>
</tr>
<tr>
<td>Service design principles</td>
<td>Implicit</td>
<td>Implicit</td>
<td>3</td>
<td>n.a.</td>
<td>Implicit</td>
<td>Implicit</td>
<td>n.a.</td>
<td>n.a.</td>
<td>implicit</td>
<td></td>
</tr>
<tr>
<td>Documentation</td>
<td>Example</td>
<td>Example</td>
<td>Example</td>
<td>Case study</td>
<td>Running example</td>
<td>Few examples</td>
<td>Applied in industry, n.a.</td>
<td>n.a.</td>
<td>Example</td>
<td></td>
</tr>
<tr>
<td>Proprietary</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Regards to stakeholder</td>
<td>Trigger events</td>
<td>n.a.</td>
<td>Implicit</td>
<td>Actors</td>
<td>n.a.</td>
<td>Implicit</td>
<td>n.a.</td>
<td>Internal stakeholders</td>
<td>n.a.</td>
<td>Actors</td>
</tr>
</tbody>
</table>

Table 3-1: Service analysis approaches
3.3 Evaluation of existing approaches

3.3.1 Overview

This chapter evaluates the different criteria that have been used to compare the identified approaches. The criteria “documentation”, “proprietary” and “regards to stakeholder” do not directly influence the reasoning about the structure and scope of the consolidated approach. However, they are used to analyse and evaluate the respective approach and are utilised throughout the remains of the chapter. The other criteria are used to evaluate the existing approaches and the consequences for the consolidated approach.

3.3.2 SOA Concept

3.3.2.1 Evaluation

SOAs containing primarily business services are less prevalent than SOAs for IT infrastructure. Jones; OASIS; Sehmi and Schwegler propose approaches that do not directly apply on the concept of a business service. Nonetheless, the underlying concepts can be adopted for the identification of business services. Flaxer and Nigam; IBM explicitly define business services, but a detailed approach for the identification of these services is missing. Kaabi, Souveyet and Rolland identify services based on goal-modelling that can lead to the identification of business services that can be supported by software services. However, the approach has not been applied in practice and is not very detailed regarding the scope of the approach. Another promising approach has been proposed by OASIS, but the “SOA Blueprints” are published as a draft version including comments and ideas. Additionally, there are a lot of gaps and placeholders. However, the underlying idea encompasses the concepts published by Jones that can be adapted to derive business services as well as technical considerations that lead to the derivation of software services. All other approaches focus on the derivation of software services, although the term business service is used to distinguish between services that encapsulate business logic and services that encapsulate application logic.

3.3.2.2 Consequences for the consolidated approach

Having analysed the different approaches, one can argue that no approach is comprehensive enough to cover both SOA concepts. The need for a consolidated approach has been formulated by some scholars, but instead of proposing a consolidated approach, they advocate another service analysis method that adds to the proliferation of these approaches. Thus, the consolidated approach described in the next chapter will focus on the derivation of business services that can be supported by software services to achieve a close business and IT alignment. Hereby, the strengths of the analysed approaches will be combined and enhanced with additional information found in literature that is applicable in the service analysis phase to derive a consolidated approach that can be used to specify a first draft of a SOA.

3.3.3 Service delivery strategies

3.3.3.1 Evaluation

The service delivery strategy is dependent on the underlying SOA concept. Approaches, which address the analysis of business services, postulate a top-down strategy for the delivery of services. However, none of these approaches is comprehensive and detailed enough to provide an analysis of the existing application landscape. A bottom-up strategy for business services could not be identified. Approaches addressing the derivation of software services postulate one of the three described delivery strategies.

The top-down approach is supposed to derive a high quality SOA that is built on well-designed services and service compositions. However, depending on the size of the company and on the scope of the SOA initiative, a top-down strategy may consume many resources, such as money and time, without showing an immediate outcome since the upfront analysis has to be conducted before actually deriving services. Sehmi and Schwegl er propose a pure top-down approach that describes how a business model can be implemented using software services. However, details are not openly available.

Contrarily, a pure bottom-up strategy to deliver software service typically comprises activities that analyse existing legacy systems in order to define fine-grained services that can be linked to business processes and business requirements.

---

An interesting point is that most approaches postulate a meet-in-the-middle strategy that takes into account business requirements as well as existing legacy systems to combine the advantages of both strategies. Thus, the advantages of a high quality SOA have to be weighed against reality constraints applied by the legacy systems. ARSANJANI as well as ZIMMERMANN, KROGDahl and GEE describe how an overarching approach could look like, but they fail to go into details since their approaches are proprietary.

### 3.3.3.2 Consequences for the consolidated approach

A consensus can be found in literature that postulates that the derivation of services should always be aligned with business requirements. Business services will be derived in a top-down fashion, since none of the analysed approaches provides any insights of how to define business services in a bottom-up manner. Since most organisations will have existing application systems in place that cannot be easily replaced, a SOA initiative should also address existing systems. Hence, a meet-in-the-middle strategy will be focussed by the consolidated approach for the derivation of software services to make this approach suitable and applicable for most organisations. Thus, software services are derived by analysing the underlying business logic of the business services in a top-down manner. The business logic can be visualised by processes, use-cases and entity models that serve as the foundation for the identification of software services. During the identification, the applications have to be analysed in a bottom-up manner to identify technical weaknesses and examine the suitability for service enablement. Furthermore, the software services identified based on the business logic have to be mapped to the application logic to identify misalignments.

A meet-in-the-middle strategy will provide several advantages, if services are derived in a top-down manner while existing systems are analysed from the bottom-up. Firstly, if extant legacy systems are analysed to act as a basis for software service identification, one will derive services that may not be needed by the organisation or may not be aligned with the business needs. The benefits of a close business and IT alignment will not be reaped. Secondly, by decomposing business processes, one may identify reengineering potential or parts of processes that do not fit the business needs anymore. By analysing software services based on applications first, without analysing the parent business processes, the reengineering opportunity will be missed. Thirdly, the context around each service is thor-

---

145 After a comparison between a top-down and a bottom-up strategy, Gold-Bernstein, Ruh (2004) and Marks, Bell (2006) explicitly state to use a meet-in-the-middle strategy for service identification.
ously analysed in order to maximise reusability potential and opportunities for compositions across different business processes.\textsuperscript{147}

Thus, business services will be derived in a top-down manner, whereas software services will be derived using a meet-in-the-middle strategy to guarantee business alignment as well as the technical feasibility of the software services.

3.3.4 General analysis approach

3.3.4.1 Evaluation

The analysed approaches can roughly be clustered according to the business and IT domain. Approaches within the IT domain focus solely on the derivation of software services, whereas approaches within the business domain focus on business and/or software services (refer to Figure 3-1). A strict distinction between the approaches and domains is hardly possible, since most approaches refer to both domains based on the meet-in-the-middle delivery strategy. For example, Erl analyses process models to derive services. However, he also stresses the fact that existing applications should be analysed as well.\textsuperscript{148} Hence, the classification scheme provided below can only be seen as a rough clustering of the analysed approaches based on their primary focus.

\textbf{Figure 3-1:} A clustering approach for service analysis approaches

This thesis does not focus on the transformation between different types of models or artefacts of an organisation, but rather focuses on the consolidation of the underlying concepts for service identification. Thus, the main challenge is to analyse the approaches in terms of their commonalities and compatibilities. The approaches related to the derivation of business services, are all compatible to a certain extent since the underlying idea is similar, as it will be shown in the following chapter.

The scopes of the methods for the derivation of software services vary widely. Approaches related to the IT domain can be clustered according to the treatment of legacy code. Non-invasive legacy approaches encompass methods that do not change the structure of the legacy system.\textsuperscript{149} They propose to build wrappers around the functionalities and components of the legacy system, so that they can be used in a service-oriented environment.\textsuperscript{150} Invasive legacy approaches aim at self-contained software services that encapsulate the functionalities provided by the legacy systems.\textsuperscript{151} Invasive approaches, such as reengineering or componentisation, deliver complete services with clear boundaries and interfaces. The aim is to rebuild an existing application based on services, so that the application itself can be retired after service-enablement. However, the risk of these approaches is very high. A tool-based legacy code analysis has to be conducted to identify all the interactions with the application across users and other systems. Such an invasive approach should only be conducted if there is a real need to have fully encapsulated services. One of the needs could be that each service should be deployed in a different environment. ERRADI, ANAND and KULKARNI proposed a matrix to guide the decision of how to service enable legacy systems. A legitimate approach could be to start wrapping legacy systems to allow communication within a service environment.\textsuperscript{152} Subsequent projects should focus on reengineering these wrapped applications based on business needs. Different options may be suitable for service enabling legacy systems, e.g. the Options Analysis for Reengineering (OAR) might be a useful guideline.\textsuperscript{153} However, since the consolidated approach only utilises the application portfolio on an analysis level, the considerations about the handling of the source code of the legacy systems have to be made during the realisation of the services, which is beyond the scope of this thesis.

The other approaches for the derivation of software services are clustered based on the way business logic is represented that should be encapsulated in software services. Most of the approaches use business process models to show how the work within an organisation is performed.\textsuperscript{154} Additionally, process models may include further information that is important for identifying and specifying services. Organisational units and domains can be annotated as well as underlying systems that are leveraged by the process. In addition, one can visualise which business entities are involved in which part of the process. QUARTEL,
DIJKMAN and VAN SINDEREN also state that a business process can implicitly define the business requirements that are needed to derive business-aligned software services.155

Further approaches visualise business logic by modelling use cases and related task trees.156 A use case corresponds to the activities and goals of an elementary business process and hence, the concepts behind the identification of services of these approaches can be consolidated with the approaches focusing on process decomposition.157 ZIMMERMANN, KROGDAHL and GEE also state that processes are more suitable for the identification of services than use cases since one has to analyse more than one system at a time to derive adequate services.158 Activity diagrams as used by WINKLER for the identification of services are derived on the basis of process models.159 Hence, this approach can be adapted to business process models as well as the approach by GOLD-BERNSTEIN and RUH. They propose that a table with events occurring during the operations of an organisation is used as the foundation to derive services. These events can also be represented within a business process if they are brought in a sequential and logical order.160

Entity-driven approaches encompass methods that describe how to derive services based on business logic represented primarily in entity models. However, no approach can solely be classified as an entity approach, since they also focus on other ways of service identification. For example, ERL primarily focuses on the derivation of software services based on process models. Secondarily, he also describes that entity models can be used as well for service analysis, but he fails to go into details.161

Other approaches describe the identification and analysis of services based on the concepts of Model Driven Architecture (MDA) or goal-modelling, for example.162 Some of these concepts can be applied within the consolidated approach and will be referenced accordingly.

### 3.3.4.2 Consequences for the consolidated approach

The consolidated approach will derive business services based on all the approaches that have been analysed. However, since no approach directly addresses the identification of business services, the underlying concepts of the approaches have to be adapted and en-

---

hanced by analysing additional related sources. An approach that details how to derive software services based on business services is missing as well. Some approaches propose that it is possible, but they fail to provide any details.\(^{163}\) Thus, the consolidated approach will show a way to relate both domains to achieve a close business and IT alignment. Software services will be derived based on process decomposition, since process models can provide the additional information, which can be used to identify services (e.g. roles, systems, etc.). Furthermore, most of the service analysis concepts of the other approaches can be adapted or related to the process-oriented approaches. Once the services have been identified using a top-down delivery strategy, they will be mapped to the application portfolio in a bottom-up manner to identify the support and shortcomings of the existing systems (meet-in-the-middle).

3.3.5 Identification process

3.3.5.1 Evaluation

Some of the approaches do not provide any structured guideline or process to derive services. These approaches give general suggestions about what to do, but do not provide information on how it should be done.\(^{164}\) However, most of the analysed approaches provide some kind of procedural model to identify business services or software services. The steps or phases for the identification are very approach-specific. For example, RAHMANI ET AL. propose an Model Driven Architecture (MDA) approach that focuses on the derivation of three specific models (three phases) to identify services.\(^{165}\) JONES on the other hand proposes four steps within his identification framework that can be used to identify services.\(^{166}\) However, for a comprehensive service analysis phase, one has to broaden the scope of the identification phase to a preceding preparation phase and a subsequent detailing phase as similarly proposed by KLOSE, KNACKSTEDT and BEVERUNGEN as well as by KOHLMANN and ALT for the derivation of software services.\(^{167}\)

3.3.5.2 Consequences for the consolidated approach

Since the consolidated approach should provide a comprehensive guideline for an organisation for the introduction to service-orientation, different phases and activities should be provided to derive a procedural model for the derivation of business and software services. Thus, the consolidated approach will be divided into two parts. The first part focuses on

---

\(^{163}\) For example, Arsanjani (2004).

\(^{164}\) For example, cp. Zimmermann, Krogdahl, Gee (2004).


the derivation of business services whereas the second part addresses the derivation of software services. Both parts will consist of a preparation phase, an identification phase and a detailing phase. The preparation phase comprises activities that lay the foundation for service identification, which is focused on by the service identification phase. The detailing phase comprises activities that are not directly related to the identification of services, but necessary to derive a sound draft of a SOA (e.g. verification of services). An additional phase, the prioritisation phase, will be described as a fourth phase during the derivation of business services to provide the link between the two parts.

3.3.6 Service classification

3.3.6.1 Evaluation

The classification of services is not standardised. Hence, different authors propose different classification schemes based on the scope of their proposed approaches. Some approaches only provide guidelines to derive services in general, others distinguish between basic types of services and a few provide a classification scheme with descriptions of the objectives of each service.

In the case of business services, no classification scheme based on the analysed approaches could be identified. For example, ARSANJANI as well as SEHMI and SCHWEGLER propose decomposition approaches that can be used to identify services, but no classification for services is provided. If the SOA concept of software service is addressed by the approaches, different levels of details are observable. For example, IVANOV and STÄHLER; SNEED; WINKLER; ZHANG, LIU and YANG do not provide any classification and thus propose approaches that generally identify software services. ZIMMERMANN, KROGDHAHL and GEE provide a rough differentiation between software services, whereas ERL; KLOSE, KNACKSTEDT and BEVERUNGEN; KOHLMANN and ALT as well as SAP provide more detailed classification schemes. However, the naming of the different services is not standardised, which leads to a proliferation of homonyms and synonyms for services and their meanings.

---

172 Cp. Erl (2005); Klose, Knackstedt, Beverungen (2007); Kohlmann, Alt (2007); SAP (2005); Zimmermann, Krogdahl, Gee (2004).
3.3.6.2 Consequences for the consolidated approach

The consolidated approach should provide a most detailed description of how to describe and identify services, so that it can be used as a comprehensive guideline. Hence, a classification scheme for software and business services should be provided based on a consolidation of existing classifications. Since not all analysed approaches provide such a scheme, additional literature has been analysed that provide insights into the problem area. As presented in Chapter 2.1 a general distinction between software services and business services can be made. Firstly, a classification scheme of software services will be presented, since scientific literature provides more insights into this area. Afterwards, analogous to the service classification scheme for software services, a classification scheme for business services will be presented.

Software services

After analysing the different approaches, it is observable that different authors proclaim a different service concept and name the services in different ways. However, the underlying concept or objective of the different types of services allows for a consolidated terminology that is followed by this thesis. Additionally, further criteria can be used to differentiate between different service characteristics. Table 3-2 shows the different criteria and their respective characteristics.

<table>
<thead>
<tr>
<th>Service type</th>
<th>Business related</th>
<th>Technical related</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Granularity</strong></td>
<td>Business process</td>
<td>Process step/Task</td>
</tr>
<tr>
<td><strong>Composition</strong></td>
<td>Composite service</td>
<td>Atomic service</td>
</tr>
<tr>
<td><strong>Interaction style</strong></td>
<td>Synchronous</td>
<td>Asynchronous</td>
</tr>
<tr>
<td><strong>Exchange patterns</strong></td>
<td>Request/Response</td>
<td>Notification</td>
</tr>
<tr>
<td><strong>State</strong></td>
<td>Stateless</td>
<td>Stateful</td>
</tr>
<tr>
<td><strong>Accessibility</strong></td>
<td>Within the organisation / internal</td>
<td>Outside the organisation / external</td>
</tr>
</tbody>
</table>

Source: Adapted from Legner, Vogel (2007), p. 373.

Table 3-2: Software service typology

In general, one can differentiate between the service types based on the underlying source of functionality. Business related services are identified and specified based on business requirements and documentations that can include business processes and tasks as well as business entities. Technical related services are rather business-logic agnostic realised as

---

utility services providing generic functions. A service can furthermore be composed of other services to expose a specific set of functionality itself or be atomic.¹⁷⁴ Services can additionally be differentiated according to the style of interaction, to the way of information exchange patterns and to the way state information is managed. The accessibility of a service can be used to classify services based on their intended service consumers. Thus, a service may be exposed to external or to internal service consumers or to both.

Based on the service type and granularity concerns, this thesis distinguishes between four service types, namely utility service, entity service, task service and process service that will be described in the following. Before the types of services will be explained in more detail, one has to keep in mind that the derivation of software services will be based on process analysis and entity model analysis, as explained in the previous chapter.

- **Utility service**: Utility services are typically business-logic agnostic.¹⁷⁵ Their main objective is to provide reusable, cross-cutting functionalities related to processing data within legacy application environments.¹⁷⁶ Thus, they provide operations such as event-logging, notification, transformations or import/export that can be drawn from various application systems within an organisation.¹⁷⁷ Other terms related to these types of services are “application services” and “infrastructure services”.¹⁷⁸

- **Entity service**: In almost every organisation, there will be certain documents describing its business entities (e.g. customer, invoice, order, etc.). These business documents can be seen as the underlying basis for the definition of entity services that represent a business-centric service with a service boundary encompassing one or more business related entities.¹⁷⁹ An entity service might be responsible for the creation of business objects like “Order” including the abidance of business rules and data completeness.¹⁸⁰ Other terms related to that type of service is data-centric service, for example.¹⁸¹

- **Task service**: A task service is directly related to a business task of a process. It is modelled for specific processes to meet immediate requirements of the organisation and therefore contains specific business logic.¹⁸² In addition to the encapsulated

business rules, task services may be used for complex calculations that have been encapsulated in libraries and business frameworks traditionally.\textsuperscript{183} As an illustration, one can imagine a pricing engine that contains all the necessary information for calculating a product price based on certain terms and conditions.\textsuperscript{184} Encapsulated in a task service one may provide centrally accessible functionality that is used consistently throughout the organisation.\textsuperscript{185} Other terms found in literature are “engine services”\textsuperscript{186}, or “logic-centric services”\textsuperscript{187}.

- **Process service**: A process service can act as the parent controller of entity, task and utility services\textsuperscript{188}. Thus, it invokes their operations based on the underlying process, which it represents.\textsuperscript{189} The process logic and related business rules are transferred to the sphere of control of a process service. Hence, the related logic is defined and embedded in a process definition, which is accessible to the process service. The orchestration logic (which services have to be invoked in which order) could be part of the individual services, but depending on the complexity of the scenario, this could result in business rules encapsulated in services influencing the orchestration across various service instances.\textsuperscript{190} Process services control and maintain the state of the process for their clients and thus are stateful to a certain extent.\textsuperscript{191}

**Business services**

The specification of a business service is made by deciding what kind of business operations should be grouped together based on certain criteria. Thus, a business service can be used to model and structure the organisation. The main question that has to be asked when identifying business services is “what does the organisation actually do?” The answer for this question is naturally dependent on the desired granularity level of the organisation. A classification scheme for business services is not provided by the approaches focussing on the business service as the underlying SOA concept. However, BIEBERSTEIN ET AL.

\textsuperscript{186} Cp. SAP (2005), p. 19.
\textsuperscript{188} Cp. Klose, Knackstedt, Beverungen (2007)
\textsuperscript{189} Cp. SAP (2005), p. 18.
\textsuperscript{190} Cp. Nadhan (2004), p. 44.
business services on different hierarchical level dependent on the value contribution of the service and the related service owner\textsuperscript{192} that is responsible for a specific service:\textsuperscript{193}

- **Team services**: The most fundamental type of services is team services. These services are precisely defined to deliver value related to an organisation’s core competencies. Thus, the underlying tasks to achieve this goal have a narrow focus, such as “functional testing of component A in product X”. The related service owner is responsible for keeping the service operational and for ensuring that the service meets contractual requirements.

- **Departmental services**: The aggregation of team services leads to the definition of departmental services to deliver core objectives of the organisation. Thus, senior managers have to understand these business objectives in order to group and streamline the team services accordingly. A departmental service could be exemplified as “testing of component A in product X”.

- **Business-unit services**: Departmental services are aggregated to business-unit services that execute the tactical goals of a company (e.g. testing product X). However, they can also directly compose team services. A director should be the service owner, since this role encompasses executive power that is used for locating bottlenecks, defining new emerging services or evaluating the composed underlying services.

- **Divisional services**: Strategic objectives are addressed by divisional services that compose the underlying services (e.g. manage product A). Senior-level executives, such as general managers, are responsible for reaching the desired objectives.

- **Group services**: The overall mission and goals of an organisation are focused on by group services that manage the strategies by managing underlying services (e.g. software portfolio services). The top management of an organisation defines the periodic goals and sets the overall strategy.

By defining these different services, including different processes and activities, redundancies might be identified that can be removed to streamline the organisation. Furthermore, the services can be rated based on their strategic value contribution that may lead to the elimination of existing services or the creation of new services. However, an organisation

\textsuperscript{192} Considerations for service ownership are addressed by the governance structure of an organisation. Cp. Malinverno (2006), p. 4. Hence, the described roles can only be regarded as a possible allocation.

\textsuperscript{193} Please refer to Bieberstein et al. (2005), pp. 698 ff for the following description.
may increase its agility and responsiveness towards new opportunities and threats, by defining and structuring its operations on different operational levels.\footnote{Cp. Bieberstein et al. (2005), p. 699.}

### 3.3.7 Service design principles

#### 3.3.7.1 Evaluation

After analysing the presented methods, it is obvious that most of the approaches address the identification and analysis of services together with a certain set of service design principles. However, the extent to which these principles are described varies widely among the analysed approaches. Most of the approaches state certain desired service characteristics (e.g. loose coupling), but do not detail on how these characteristics can be achieved.\footnote{Cp. for example Kohlmann, Alt (2007), p. 182; Rahmani et al. (2006), p. 580; Arsanjani (2004).} Hence, service design principles can be used as a guideline to derive meaningful services in the context of a service-oriented architecture. If the services are derived by applying the design principles to a certain desired extent, the services will feature the corresponding service characteristics.\footnote{Cp. Erl (2007), pp. 28 f.} However, there is no commonly agreed upon set of design principles, similar to the classification of service types. Based on the different focuses of the multiple analysed approaches the respective design principles are not all compatible with each other. Some are named identically but are applied differently by the authors (homonyms) or have different names but a closely related content (synonyms). Others are only applicable to a certain extent based on the underlying assumption about the service types.

#### 3.3.7.2 Consequences for the consolidated approach

The design principles as described in Appendix D are derived from the literature review encompassing the analysis of the presented approaches for service analysis. Since most approaches reference a certain set of design principles or just mention desired service characteristics without describing how to achieve the desired characteristics, additional papers and monograms have been included in the derivation of a coherent set of design principles that do not have a primary focus on the identification of services, e.g. LEGNER and VOGEL, Erl as well as KRAFZIG, BANKE and SLAMA.\footnote{Cp. Erl (2007); Krafzig, Banke, Slama (2006); Legner, Vogel (2007).} Nine design principles could be identified and consolidated, namely contract, abstraction, autonomy, coupling, statelessness, cohesion, discoverability, reusability and composability. These design principles are applicable from the identification of the service to the implementation of the service logic to a certain extent. For example, the service design principle of abstraction describes what information
should be published for the potential service consumer and guides the service developer in the decision-making of what information should not be visible to the service consumer.\footnote{\textit{Cp. Erl (2007), pp. 212 ff.}} This decision will typically be made, if the service has been realised or implemented and details about the service environment are available. However, since not all of the service design principles are applicable during the service analysis phase, only a subset of the analysed approach will be described in this chapter to an extent that is useful for service analysis.\footnote{\textit{Cp. Inaganti, Behara (2007), p. 1.}} This subset of design principles should be applied during the service analysis and identification, since errors made in the service analysis phase can pervade through to the service implementation activities.\footnote{\textit{Cp. Inaganti, Behara (2007), p. 1.}} Hence, if these design principles are applied in early stages of the SOA initiative, any misalignments or errors can be avoided.

\textit{Contract}

A service contract “\textit{establishes the terms of engagement, providing technical constraints and requirements as well as any semantic information the service owner wishes to make public.}”\footnote{\textit{Cp. Erl (2007), p. 126.}} To allow services to interact with each other and to be invoked by their service consumers, they need to share a formal contract that defines the terms of information exchange and the commitments made by both parties. Service contract design is very important since the agreement to a service contract means to establish a dependent relationship between the service consumer and provider.\footnote{\textit{Cp. Legner, Vogel (2007), p. 373.}} The contract encompasses a description of the functional and non-functional characteristics of a service.\footnote{\textit{Cp. Zacharias (2005), p. 45; O'Sullivan, Edmond, Hofstede (2002), pp. 120-127.}} The more technical characteristics encompass a description about the operations that are exposed to the service environment and can be accessed by service consumers.\footnote{\textit{Cp. Sehmi, Schwegler (2006b). The technical characteristics are described in the service interface of a software service that is part of the service contract. Krafftzig, Banke, Slama (2006), p. 59.}} Hence, this design principle is applicable in the service analysis phase, as the detailing of the operations is related to the definition of the interface.\footnote{\textit{Cp. Gold-Bernstein, Ruh (2004), p. 130 and the scope for service analysis in this thesis as described in Chapter 2.3.}}

\textit{Autonomy}

Autonomy refers to the level of independence of a service. This means a purely autonomous service has full control over its environment, which results in increased reliability and predictability, since external unpredictable influences are minimised.\footnote{\textit{Cp. Erl (2007), pp. 294 f.}} Thus, a ser-
vice’s contract should not overlap with any other contract and the underlying realisation or implementation of one service should reside under the control of that service.²⁰⁷ This principle is equally applicable on software and business services. Service normalisation is another aspect that aims at designing the operations in a non-redundant manner.²⁰⁸

**Coupling**

This service design principle is applicable on different elements of a service and a SOA in general and refers to the level of dependency between two or more elements. The observable type of coupling can be identified as the interdependency of multiple services and service compositions. However, other types of coupling or dependency can be encountered. For example, the coupling between the service implementation to proprietary vendor technology, the coupling between the service implementation to the service contract and vice versa and the coupling between the service to one or more business processes.²⁰⁹ However, in this thesis coupling is referred to as the dependency of one service to another service, when not stated otherwise. As soon as one service calls an operation provided by another service, the service is dependent on the functionality offered by the other service’s operation and the services are coupled. Services that are not dependent on the other services have a high reusability and maintainability potential. Thus, the coupling between services should be as loose as possible.²¹⁰ This goal can be achieved by specifying a well-defined contract for each service that serves as the only access point to the service and its underlying logic.²¹¹ This way a simple interchange between services can take place. For example, an existing service may be substituted by a third-party service because the external provider offers better performance. If different versions of a service exist, the business requirements will determine which service will be used depending on a certain scenario (e.g. external or internal accounts depending on the purchase order). Furthermore, one can influence the coupling of two or more service by changing the allocation of a service’s operations to another service. Hence, the coupling of services should already be addressed in the service analysis phase.

**Cohesion**

The idea about cohesion has emerged from the area of software engineering, since it refers to the concept of grouping elements based on their functional relatedness to perform a cer-

tain task.\textsuperscript{212} On this abstract level, the design principle is applicable to business and software services. For example, PAPAZOGLOU and VAN DEN HEUVEL refer to service cohesion as the strength of functional relatedness between operations in a service.\textsuperscript{213} They propose three levels of service cohesion, whereas the functional cohesion is the most preferable.\textsuperscript{214} Other types of cohesion can be analysed by regarding the input and output parameters of an operation or the sequence of operation invocations.\textsuperscript{215} The functional relatedness can be interpreted as the interdependencies between operations or resources.\textsuperscript{216}

\textit{Reusability}

The principle of reusability has a straightforward underlying concept: to make the service useful for more than one single purpose.\textsuperscript{217} Thus, services should be applicable in different situations and under unforeseen circumstances and be used by different service consumers.\textsuperscript{218} This can be achieved by defining an agnostic functional context and defining the encapsulated logic as generic as possible, so that the service is not bound to one single usage scenario. Furthermore, the service contract should encompass generic and extensible operations, so that they are not restricted to one specific type of input parameters, for example.\textsuperscript{219} Software services can be used to centralise certain logic that can be accessed within and across an organisation, to enhance the consistency and availability of functions and data.\textsuperscript{220} This description fits the purpose of business and software services. Three different kinds of reusability related to a software service can be distinguished. Firstly, the \textit{tactical reusability} focuses on the specification of service operations for immediate requirements that must be fulfilled by the service. Secondly, the \textit{targeted reusability} focuses on the implementation of service operations that have the highest potential of being reused under more than one scenario. Lastly, \textit{complete reusability} of the service can be focussed on. Hereby, the service provides a set of operations that address known requirements.\textsuperscript{221}

\textsuperscript{221} Cp. Erl (2007), pp. 265 f.
4 The consolidated approach

4.1 Overview

Preliminary considerations

This chapter will provide a detailed description of a comprehensive approach for the identification and analysis of both business and software services. This approach will provide an organisation with a methodology to not only understand and document its existing capability from a service perspective, but more importantly, to identify potential new services that may be provided to potential service consumers (at the business and technical levels). The consolidated approach will combine the strengths of the analysed approaches as well as extend and adopt certain facets to provide a comprehensive procedural model for service analysis. The business strategy is treated as an external input that serves as the basis for the consolidated approach. Consecutively, business services will be derived that can (partly) be supported by IT functionalities represented by software services.

Outline

Based on the findings of the previous chapter, the consolidated approach will cover the derivation of business and software services to combine both SOA concepts. For the derivation of business services, a top-down approach will be pursued to identify services based on business requirements and existing business structures. A meet-in-the-middle strategy will be pursued to derive software services based on business needs as well as on the analysis of existing legacy systems within an organisation. The approach is subdivided into two parts. Both parts will further be structured by introducing certain phases including specific activities to make the approach more manageable and understandable. An example explicating the different phases and activities will be provided at the end of this chapter. This procedure has been chosen since the provision of examples for each activity directly after the respective description would hinder the understanding of the overall approach, as it would break up the flow of argumentation.

Part I: The identification and analysis of business services

The first part covers the identification and analysis of business services by detailing, adapting and consolidating existing service analysis approaches that focus on the business domain of an organisation. This part will be structured into four distinct phases, each comprising a specific set of activities that may use the outputs of previous phases as inputs. Refer to Figure 4-1 for an overview of the inputs, activities and outputs for each phase. The preparation phase serves as the foundation for service identification. Hereby, the SOA
strategy will be defined, as well as the information base and a business ontology. Having built the foundation, the service identification phase comprises activities that lead to the derivation of a set of business services. This set of services serves as an input for the detailing phase that encompasses activities to analyse the consumer interaction with each business service as well as the relationship of one business service to other business services. When all business services have been detailed and analysed, the prioritisation phase proposes certain activities that enable an organisation to structure their services based on specific measures. The last phase is used as a link to the second part, which focuses on the derivation of software services.

**Part II: The identification and analysis of software services**

The second part of the consolidated approach will address how software services can be identified and analysed that support business services to achieve a close business and IT alignment. Similar to the first part of the consolidated approach, this part will be structured into a preparation, an identification and a detailing phase, each comprising specific activities. Refer to Figure 4-1 for an overview of the second part of the consolidated approach. The preparation phase processes the classified business services, which have been the output of the last phase of the first part, and defines the scope of the derivation of software services that should support a certain set of business services. Additionally, the processes underlying each business service and serving as the main business artefacts for the derivation of software services have to be decomposed and analysed along with an application analysis. A suitability analysis of the processes and applications will provide an understanding of which processes can actually be service-enabled. The identification phase takes the outcome of the preparation phase and identifies software services. The detailing phase encompasses verification, operation detailing, functionality mapping and the establishment of a service inventory. The outcome of the second part is a first draft of a SOA comprising the identified set of services.

In the following, the consolidated approach will be presented as visualised in Figure 4-1. A more detailed overview of the different phases, their inputs, activities and outputs is visualised in Appendix E.
Figure 4-1: Phases for the business and software service analysis
4.2 The derivation of business services

4.2.1 Overview

The derivation of business services can be subdivided into four phases. The preparation phase lays the groundwork for the actual identification of business services. The detailing phase takes the identified services as an input and specifies them in more detail. Since business services do not all deliver the same value to an organisation, the prioritisation phase comprises activities that classify services according to their value.

4.2.2 The preparation phase

4.2.2.1 Overview

The preparation phase encompasses all activities that are related to laying the basis for the identification and specification of services. Similar to any project, the driver behind the SOA initiative has to be defined. Due to the multitude of different terminologies in the area of service-orientation, a business ontology has to be established. An information base that is used as the starting point for service analysis has to be established and aligned with ontologies defined previously. These activities including their inputs and outputs are visualised in Figure 4-2 and further detailed during the course of this chapter.

![Figure 4-2: The preparation phase for business services](image)

223 Similar to the preparation phase proposed by Klose, Knackstedt, Beverungen (2007).
4.2.2.2 Development of a SOA strategy

The specification of a SOA strategy should be focussed on at the beginning of a SOA initiative to set the scope and objectives of the following analysis efforts. The development of a SOA strategy can be structured according to Figure 4-3.

Source: Adapted from Marks, Bell (2006), p. 85.

Figure 4-3: Translating business context into SOA value

The first step for each organisation that wants to invest in service-orientation is to understand its environment, the business driver and context. This step provides the direct link between the business strategy and the SOA initiative. Thus, one has to conduct a business context analysis, including an assessment of the external environment, a review of the current business strategy including the current and future business initiatives and an analysis of the market position including a customer and product analysis. An organisation has to understand the position in its specific market as well as the external drivers of the organisation in order to analyse and communicate the proposed benefits it aims to achieve by the introduction of SOA and services. Most of the needed information may already be available because of strategic planning efforts conducted earlier. However, the objective here is to find key areas of the business context or problem space that have to be supported by the SOA initiative. This provides the underlying basis for the identification of business and IT imperatives as well as for the SOA strategy. Business and IT imperatives are organisational challenges that need to be addressed properly or otherwise they may imperil the organisation (e.g. business processes are error-prone, inconsistent customer channels, etc.). These imperatives serve as the motivation or driver for the SOA initiative that should be documented in a SOA strategy. Based on this strategy, SOA value drivers may be proposed that describe how the imperatives will be addressed and how the outcome of

---

227 Since the information could already be available, the business context analysis has been classified as an input in Figure 4-1 and Figure 4-2.
the initiative might be measured.\textsuperscript{231} Examples of the value drivers have been described in Chapter 2.2.3 that focussed on the proposed benefits of SOA. The proposed steps help to focus on appropriate business challenges that serve as the foundation for service analysis.

Executive support and sponsorship that is critical for the SOA initiative may also be built upon the business context analysis and the specification of a SOA strategy.\textsuperscript{232}

The SOA strategy should not be seen as a static document, but should always be aligned with the business strategy and business requirements that are based on external and internal factors. Additionally, since a SOA initiative is most often an incremental approach with multiple projects implementing services over time, the SOA strategy should be revised after each project.\textsuperscript{233}

4.2.2.3 Establish business vocabulary

A business ontology should be established and maintained throughout the SOA initiative. Since a SOA initiative may affect various parts of an organisation, a common vocabulary helps to avoid misunderstandings between any involved parties.\textsuperscript{234} The Object Management Group (OMG) published a guideline to establish a vocabulary including business rules that could be used as a starting point.\textsuperscript{235} More advanced than the development of a common vocabulary is the development of an ontology for the organisation. The term “ontology” originated in philosophical science. However, in the area of information systems, Z\U{u}NIGA states that an ontology can be seen as the formal representation of knowledge within a specific domain.\textsuperscript{236} GRUBER defines an ontology as “an explicit specification of a conceptualization.”\textsuperscript{237} Hereby, linguistic expressions have to be defined that different actors have agreed on.\textsuperscript{238}

As a result, an ontology is founded and relations between the classified information is established. This is an essential step towards the integration of the business perspective and the technical perspective within an organisation. Based on the size of the company and on the scope of the SOA initiative, this activity may consume many resources, such as money and time, without resolving in immediate outcomes.\textsuperscript{239}

\textsuperscript{231} Cp. Marks, Bell (2006), pp. 78-91.
\textsuperscript{235} Cp. OMG (2008).
If the ontology is supposed to be processed by a software system, the linguistic expressions have to be formally defined. Different languages have been proposed, for example the Web Service Modeling Language (WSML) by the European Semantic Systems Initiative (ESSI)\textsuperscript{240} or the Web Ontology Language (OWL) recommended by the W3C\textsuperscript{241}.

4.2.2.4 Establish information base

When the business drivers have been identified, the SOA strategy has been defined, executive support is guaranteed and the ontology has been established, the business documents that serve as the underlying basis for the analysis and identification of business and software services have to be compiled.\textsuperscript{242}

Based on the broad scope of this consolidated approach, multiple business artefacts documenting the business structure and operations of an organisation may be useful for the service analysis phase. Thus, these documents may comprise details about the enterprise architecture, capabilities, organisational structure and responsibilities, the stakeholders, business processes and entities. Furthermore, interviews with top and senior management should be conducted to enhance and validate the information provided by the documents. Business documents related to the IT domain of the organisation, such as application documentation, are also needed in later phases of the consolidated approach. These documents may therefore be collected at this stage as well. Having the ontology defined, the existing information base has to be updated using the vocabulary provided by the derived classification scheme.

During the next phases of the consolidated approach, it may be necessary to extend the information base by acquiring additional documents or conducting further interviews in order to gain access to details that have not yet been identified. However, the scope and depth of this phase is highly organisation specific, since not all organisations may have the documents or roles mentioned above. This thesis assumes an ideal situation, where all documents provide the required level of detail and sufficient information for the analysis and identification of services.

\textsuperscript{240} Cp. ESSI (2005).
\textsuperscript{242} The collection of documents related to the SOA initiative is proposed by Klose, Knackstedt, Beverungen (2007); Kohlmann, Alt (2007); Erl (2005).
4.2.3 The identification phase

4.2.3.1 Overview

Based on the outputs of the preparation phase and the service design principles described in Chapter 3.3.7, the identification phase comprises the activities that are related to the identification and analysis of business capabilities and business services. The identified and analysed business capabilities serve as the foundation for the identification of business services. Hereby, the domain analysis along with a stakeholder analysis and analysis of the interactions and entities is conducted to support the identification of business services. (Refer to Figure 4-4).

Figure 4-4: The identification phase for business services

4.2.3.2 Capability analysis

Preliminary Considerations

Before the actual business services can be identified, the working definitions of a service and a SOA are recapitulated for this thesis: In general, a service is seen as a representation of a capability that forms a coherent functionality from the point of view of provider entities and requester entities. SOA is a paradigm for organising and utilising distributed capabilities that may be under control of different ownership domains. As stated in the definition above, a service is associated with organisation’s capabilities. Hence, before the actual business services can be derived, capabilities of the organisation need to be identified and analysed.

---

243 Please refer to Chapter 2.3.2.
OASIS defines a capability as something that provides a real world effect.\textsuperscript{244} A real world effect can be the response to a request for information or the change in state of some defined entities.\textsuperscript{245} SEHMI and SCHWEGLER argue the same way, as they see a business capability as a particular ability or capacity that a business may possess or exchange to achieve a specific purpose or outcome (real world effect). A capability describes what a business does to create value for customers. However, it is not concerned about how the actual business function is actually achieved, but rather with its externally visible behaviour and its expected level of performance.\textsuperscript{246} HOMANN defines that a capability “abstracts and encapsulates the people, process/procedures, technology and information into essential building blocks necessary to facilitate performance improvements and redesign analysis.”\textsuperscript{247} Thus, the complete organisation can be seen as a federation of capabilities that interact with one another to achieve a valuable outcome for the business network.\textsuperscript{248} Capabilities exist not only in the private sector, but also in the public sector related to government agencies. The Department of Finance and Administration within the Australian Government published a Service Delivery Capability Model (SDCM) that provides a framework to describe the capabilities required to deliver services.\textsuperscript{249} Hereby, a capability is described as a “container” for the people, processes, facilities and equipment, information and communication technology, knowledge and governance\textsuperscript{250} similar to the description of a business capability.\textsuperscript{251} Hence, capability analysis is applicable in the private and public sector. However, not all capabilities provide the same value for an organisation. LONG and VICKERS-KOCH distinguish between threshold capabilities and core capabilities. Threshold capabilities include support and basic capabilities that are common to most companies in a particular organisation. Core capabilities focus on critical and cutting-edge capabilities that are owned by an organisation to provide sources for the present and future competitive advantage of the organisation.\textsuperscript{252} This perspective on capabilities supports the assumption that capabilities can be seen as the building blocks of an organisation. Capabilities can be used to develop a business model that is particularly stable against changes in the external and internal environment of an organisation because capabilities are not affected by changes in the organisational structure of an organisation, or by changes in the flow of the underlying processes as long as the purpose or outcome remains the same.

\textsuperscript{244} Cp. OASIS (2006), p. 28.
\textsuperscript{245} Cp. OASIS (2006), pp. 18 f.
\textsuperscript{247} Homann (2006).
\textsuperscript{248} Cp. Cherbakov et al. (2005), p. 654.
\textsuperscript{251} The SDCM is used by government agencies to develop and plan new policies. Cp. Australian Government (2006), pp. 11 ff. This perspective is beyond the scope of this thesis. However, the reference has been chosen to show that capability analysis is not restricted to the private sector.
Different approaches to identify capabilities within an organisation can be found in literature. However, most approaches take the capabilities as a given factor, since they are highly organisation specific, and state that these capabilities have to be aligned with the organisation’s strategy.\(^{253}\) There are unstructured approaches describing how to identify capabilities. HAFEEZ, ZHANG and MALAK interviewed several senior managers of an organisation that provided the necessary insights and knowledge to identify the organisation’s capabilities.\(^{254}\) A structured approach has been proposed by SEHMI and SCHWEGLER, who published a framework for capability decomposition.\(^{255}\) However, this framework is very closely aligned with the Process Classification Framework (PCF) published by the American Productivity & Quality Center (APQC).\(^{256}\) Although this might seem contradictory at first sight since capabilities should abstract from any processes, the idea behind both approaches is quite similar: The PCF does not detail how the specific processes are executed, it just classifies the processes based on their purpose and outcome, similar to the framework proposed by SEHMI and SCHWEGLER. Process classification frameworks abstract from the process flow, specific activities, utilised resources, etc. and can therefore be used to identify organisation specific capabilities. Another process classification framework has been developed by the SUPPLY-CHAIN COUNCIL, namely the Supply-Chain Operations Reference (SCOR) model that can be used as a guideline for capability identification as well.\(^{257}\) However, one must not focus on specific process activities or process flows of one process, but rather analyse the outcome and value propositions of the processes as representations of capabilities.

**Capability decomposition**

Each organisation will have their own specific set of capabilities and ways of classifying processes and enterprise operations. However, the presented way of classifying capabilities is industry neutral and allows the application across different industries and organisations.\(^{258}\) It can be used as a guideline to show how organisation specific models can be used as an input for service analysis.

Capabilities can be modelled on different granularity levels to establish a hierarchy of capabilities. On the highest level, the capabilities can be grouped according to general activities of organisations. Figure 4-5 visualises these generic clusters.

\(^{253}\) For example, cp. Day (1994)
\(^{256}\) Cp. APQC (2006).
\(^{257}\) Since the AQPC Process Classification Framework is openly available, it will serve as the basis for capability decomposition. For more information about the SCOR model, please refer to Poluha (2007).
These capabilities are within the physical boundary of an organisation and are common to most companies. HAGEL III and SINGER state that an organisation is typically made up of a customer relationship business, a product innovation business and an infrastructure business.\(^\text{259}\) This proposition supports the clustering of the capabilities, since capabilities 2-4 can be matched to a product information business, capability 4 relates to the customer relationship business whereas capability 6 can be associated with the infrastructure business. However, capability 1 relates to all three businesses since they all need some type of strategy to manage their operations. The top-level process clusters of the SCOR model (plan, source, make, deliver and return) could be associated accordingly.

Each of these general clusters can further be decomposed based on specific business characteristics. Thus, one can derive a hierarchy of capabilities until fine-grained business capabilities can be identified. This could be done by interviewing several senior managers of an organisation and using the PCF as a guideline for identifying capabilities on different granularity levels. Refer to Figure 4-6 which exemplifies the decomposition of the fourth cluster (deliver products and services) into finer-grained business capabilities.

4. Deliver products and services
   4.1 Provide service
   4.2 Advanced planning
   4.3 Procurement
      4.3.1 Sourcing and supplier contract management
      4.3.2 Purchasing
         4.3.2.1 Request resources
         4.3.2.2 Acquire resources
   4.4. Produce product
   4.5. Logistics

The depth of the decomposition is based on the business needs that can be addressed by the SOA strategy. Business capabilities should be described and detailed by certain attributes. Questions that have to be answered comprise for example:

- What are the performance requirements?
- What are the inputs and outputs?
- Is the capability part of why customers, suppliers and partners do business with the organisation?

These attributes serve as the underlying basis for the reasoning of appropriate boundaries of business services as well as an overview of how parts of the contract need to be designed related to the service performance.

**Relationships between capabilities**

A capability model also visualises the relationships between all the different capabilities of an organisation. These relationships can be distinguished into three different types of connections between capabilities. An *input/output* connection between two capabilities demonstrates a consumer/provider interaction. A *supporting* connection is used when information is passed from one capability to another. This one-way connection is typically used for exception handling, inventory returns, verifications and inspection capabilities. These would typically require supporting connections. The *control* connection is used to model the impact of one capability on the performance of another capability. In this situation, one capability sets the policies and performance requirements for the other one.

**4.2.3.3 Business service identification**

**Preliminary considerations**

Having built different layers of business capabilities, one has to identify the boundaries of a business service that encapsulates certain capabilities and exposes them to the service environment. Each capability identified can potentially be used for the specification of a business service. Hence, the service boundary would comprise one single capability. However, depending on the granularity levels of the capabilities this can lead to proliferation of business services. Thus, some capabilities should be grouped together to form a business service that provides cohesive functionalities and is loosely coupled to other business ser-

---

261 Refer to Homann, Tobey (2006) for the following description of the different types of connections between capabilities.
services. Hence, the design principles as described in Chapter 3.3.7 can be used as a guideline to define such service boundaries. Four indicators can be used to analyse capabilities according to the principles, namely the function, the owner, the underlying business entities and the relationships between different capabilities.

The domain decomposition approach

In the following part, a domain decomposition approach will be presented that groups capabilities according to the indicators mentioned above. Ownership considerations can be taken into account during domain decomposition as well. Thus, this phase is complementary to governance consideration of the SOA, since SOA governance also addresses the ownership concerns of single services.\textsuperscript{262} A domain, or service domain, can be described as a sphere of control that contains a collection of tasks to achieve related goals.\textsuperscript{263} This definition fits precisely into the presented argumentation and description of the consolidated approach, since the collection of tasks is reflected in the definition of business capabilities.

Domain decomposition, including the analysis of functional areas and its subsystems in an organisation, has been recognised as another starting point for service identification.\textsuperscript{264} The domain decomposition approach proposes to establish a domain model on different levels of granularity that visualises the interactions among the different domains as well as the interactions external to the domains.\textsuperscript{265} Each level is derived by decomposing a service domain on the previous level. Therefore, it is possible to navigate through the levels of the model by drilling down or rolling up on a specific domain on a certain level.\textsuperscript{266} The different levels of the domain model can be derived by identifying the domains, the external stakeholders and the interactions. The steps are summarised in Table 4-1.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>What</td>
<td>Definition of service domains (scope)</td>
</tr>
<tr>
<td>2.</td>
<td>Who</td>
<td>Specification of actors / stakeholders</td>
</tr>
<tr>
<td>3.</td>
<td>Why</td>
<td>Analysis of the reason of interaction</td>
</tr>
<tr>
<td>4.</td>
<td>How</td>
<td>Allocation of capabilities</td>
</tr>
<tr>
<td>5.</td>
<td>Next</td>
<td>Next iteration of decomposition</td>
</tr>
</tbody>
</table>

Table 4-1: The three steps for domain decomposition

\textsuperscript{264} Cp. Jones (2006); OASIS (2005); Kohlmann, Alt (2007).
\textsuperscript{265} Cp. OASIS (2005), p. 5.
\textsuperscript{266} Cp. OASIS (2005), p. 6.
Step 1: Definition of service domains

Starting at the highest level of granularity (level 1), the first step involves the definition of the service domain’s scope, meaning what is the boundary of a service domain within an organisation that comprises certain capabilities (the “what”). The objective of a service domain is to represent what the organisation does and to place boundaries between the domains. Thus, services that will be provided by one domain are owned and managed by this specific domain. On this highest level, the replacement of such a domain would have minimal impact on the other domains. As a starting point, the different divisions of an organisation or organisational functions can be used as a starting point for the specification of coarse-grained service domains. However, additional factors may be used to define the boundary of coarse-grained domains according to the indicators. Flaxer and Nigam proposed an approach that identifies all business tasks associated with the life cycle of one business entity. This approach has been utilised by IBM to identify business components that define the structure of an organisation and can be regarded as equivalent to service domains on a coarse-grained level. Each component is responsible for one specific entity. Thus, the boundary between two domains on the highest level of granularity may be based on the differentiation between core business entities. There are several ways to derive these entities. One way is to think about the main “nouns” of an organisation (e.g. customer, order, etc.) or the respective business service; what are the important entities and elements of an organisation? These can include the main business documents and transaction areas. Therefore, business artefacts are not an explicit requirement to identify entities, but they may be helpful for the identification. Business artefacts may encompass entity relationship models, process models or object/class diagrams generated during application development projects.

Step 2: Specification of actors / stakeholders

The next step focuses on the interaction between different service consumers and service domains (the “who”). Actors can be distinguished between service consumers (e.g. persons, services, etc.) and other service providers, who offer certain capabilities. The actors considered in this step are not the internal facilitators of the services but rather external

---

consumers. Thus, the objective is to identify and understand the external stakeholders of the organisation.

The term stakeholder has been introduced by FREEMAN, as he defined a stakeholder as any group or individual, who can affect or is affected by the achievement of the organisation’s objective. In recent years, this definition has been revised because of its very wide scope and interpretational potential. Thus, stakeholders should have legitimate claims regardless of their power to influence the firm. Stakeholder analysis can be regarded in three different perspectives, namely descriptive, instrumental and normative. However, a thorough discussion about these perspectives lies beyond the scope of this thesis. DONALDSON and PRESTON published a stakeholder model that visualises different classes of stakeholders with legitimate interests in an organisation. This model goes beyond an input/output relationship between the organisation and its stakeholders, but takes into account the interests of different groups regarding an organisation (refer to Figure 4-7).


**Figure 4-7:** A stakeholder model

Another approach that tries to classify different stakeholders of an organisation has been published by MITCHELL, AGLE and WOOD. This approach tries to classify stakeholders according to three specific attributes, namely power, legitimacy and urgency. Based on the possible overlaps of these three attribute classes, eight classes of stakeholders can be distinguished. The authors propose that the stakeholder salience increases with the number of

---

attributes.\textsuperscript{283} However, a practical classification framework is not provided. Thus, for an analysis of different stakeholders, both approaches can be combined to identify the classes of stakeholders and the attributes for each stakeholder in one class. However, at this stage the stakeholder analysis is primarily used to support the identification and analysis of domains and to guide the understanding of the capabilities that are actually utilised by the organisation’s stakeholders.\textsuperscript{284}

Step 3: Analysis of the reason of interaction

After the identification of the service domains and the actors, the reason for the interactions between the domains themselves and between the domains and the actors have to be identified (the “why”).\textsuperscript{285} This step aims at understanding the reason why various domains and actors interact.\textsuperscript{286} Interactions themselves are physical or involve the integration of IT (e.g. manual, semi-automated, and automated interactions). The identification of the reasons why domains and actors communicate has been utilised by KAABI, SOUVEYET and ROLLAND to describe the goals or requirements of different actors and identify the services to consequently achieve these goals.\textsuperscript{287} This is the first step that relates the perspective of the service consumer and the service provider.

Step 4: Allocation of capabilities

Based on this first level of the domain model, one may already allocate capabilities to certain domains. The domain decomposition and the capability decomposition are orthogonal to each other. Both can be conducted independently. A table consolidating the approaches is shown in Table 4-2. The generic capabilities are decomposed into finer-grained levels as well as the domains with the external stakeholders.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
\textbf{Domains} & \textbf{Capabilities} & \textbf{Capability}_A & \textbf{Capability}_B \\
\hline
& DomainA & CapabilityA1 & CapabilityB1 \\
\hline
& DomainA & CapabilityA2 & CapabilityB2 \\
\hline
DomainB & DomainB & CapabilityB1 & CapabilityB2 \\
\hline
DomainC & DomainC & CapabilityC1 & CapabilityC2 \\
\hline
\textbf{Ex. Stakeholder} & & & \\
\hline
\end{tabular}
\caption{Consolidation of capabilities and service domains}
\end{table}

\textsuperscript{284} Only stakeholders that are external to the organisation will be regarded at this stage, since the interactions between the organisation and its external service consumers are the objective of this phase.
The allocation should be made regarding the support of the capability towards the documented interactions (functional indicator). Furthermore, one may also identify the core business entities and the associated underlying data of each capability. If the capabilities share common data sets that are commonly manipulated and have real time effects, the cohesiveness from a data perspective is high and serves as an indicator for a close relationship between the respective capabilities. For example, the business capabilities “Accounts payable” and “Accounts receivable” may manipulate the same data sets and share the same business entities, which make them good candidates for a grouping into one service domain.\footnote{Cp. Homann, Tobey (2006).} Hence, the identified business entities of a capability can be compared with the underlying business entities of the service domain to guide the allocation of capabilities to domains. All capabilities should be cohesive within one service domain regarding their functions and entities. Furthermore, the coupling between capabilities within one domain should be higher than between domains. Hence, the frequency or density of the interaction of the capabilities can be used as an indicator as well. Capabilities that show frequent interactions may belong to the same domain.\footnote{Cp. Homann, Tobey (2006).}

Step 5: Next iteration of decomposition

The elements in the first level of the domain model become more concrete on the next decomposition level (level 2). Each identified service domain has to be decomposed applying the first three steps of the domain decomposition approach. One purpose of the decomposition into finer-grained domains is the ability to navigate through the levels of a domain model.\footnote{Cp. Jones (2006), p. 45.} Hereby, the operating departments within a division can be used as a starting point for analysis of level 2.\footnote{Cp. Jones (2006), p. 48.} The capabilities that have been allocated to service domains on the previous level have to be classified to the domains on the next granularity level. Similar considerations have to be made regarding the coupling and cohesion principles. Thus, the capabilities allocated to one domain should be cohesive within the domain and loosely coupled towards capabilities of other domains. Entities and interactions between capabilities may serve as the indicators for the compliance with the design principles. On the lower abstraction levels, each step of the approach has to be applied on each specific domain of the developed diagram derived on the previous level.\footnote{Cp. Jones (2006), p. 46.} By doing so, a deeper understanding of the problem domain is achieved as well as an analysis of the different external representations of the domains.\footnote{Cp. OASIS (2005), p. 17.}
Conclusion

Each service domain can be regarded as a provider of one or more business services. Hence, the domains can be used to group services according to the classification of business services given in Chapter 3.3.6. Hence, high-level domains and their related high-level capabilities serve as the basis for group services. Finer-grained domains and capabilities can be assigned to finer-grained business services. Thereby, the organisation itself can be described as a conglomeration of business services. Since each domain on level \( k \) is decomposed separately, it is possible that one identifies domains that are shared on level \( k+1 \). This leads to the identification of business services that are reused under different circumstances within the organisation.

By analysing the different actors and interactions, one can also identify certain virtual services, which are not core to the organisation and do not belong to a direct domain. Virtual services resemble a combination of internal services to provide an external view for the customer (e.g. customer portal). Therefore, they can be seen as interaction points with the underlying services of the system. Additionally, they indicate where business logic can be coordinated and contingently simplified, since they do not comprise any business logic themselves. The actual realisation of the business logic is done in separate services.

4.2.4 The detailing phase

4.2.4.1 Overview

At this stage, capabilities, domains and business service boundaries have been identified. This phase comprises activities concerned with the relationships and interactions between business services. The input for this phase comprises the identified business services as well as a set of business artefacts that are useful for the detailing phase. For each business service out of the set of services that have to be detailed, one has to define the specification, the resources, the operation model and the interaction model. The contract can be seen as a part of the service’s resources. On the one hand, the contract can be used to derive the service specification, which details the interaction and operation model. On the other hand, the interaction model, operation model and the service specification can be defined as the basis for an adequate service contract. The output of this phase is a model consisting of different interacting business services. Figure 4-8 visualises the inputs, activities and outputs of this phase.

---

294 Virtual services are very closely aligned with the conception of e-services, as explained in Chapter 2.1.3.
Figure 4-8: The detailing phase for business services

As a template for the service details that have to be specified during this phase, one can refer to Figure 4-9. A business service exposes one or more operations to its environment that can be utilised according to the business service contract. Each operation has two underlying models, namely the interaction model and the operation model that describe the behaviour of the specific business service operation. Each of the visualised elements has to be addressed properly. The elements will be explained in further detail in separate sub-chapters.

Source: Adapted from Sanz, Nayak and Becker (2006b), p. 63.

Figure 4-9: Business service details

---

4.2.4.2 Business service resources

Each business service has an owner, who is responsible for the specific service. Depending on the granularity of the business service, the owner can be the organisation itself, a business unit, or even finer-grained domains that are responsible for providing the service to its environment. The owner of that specific business service is related to the ownership domains that have been identified in the previous phase.

As described in the previous phase, a business service comprises one or more capabilities based on ownership concerns, functions, related interactions and entities. At this stage, the underlying processes of the capabilities have to be identified. The Process Handbook explicating different relationships between processes as described by Malone, Crowston and Herman can be used as a guideline to analyse the interdependencies between different processes. The value or output of one capability can be based on one single process as well as an interaction between different processes. Consequently, the design of a business process across a business network or within a single organisation (e.g. order-to-cash process) can be modelled as a choreography of interacting business services that may be under different ownership domains. However, there is no explicit need to manage and control end-to-end processes to achieve a desired goal, since each business service acts as defined in the service contract. Thus, if a business goal is not achieved, the related business service can be identified and changes can be made, instead of analysing the deficiencies of a complex process. The scope of the process will be dependent on the granularity of the business service and the underlying capabilities. The scope of a process underlying a group service is potentially wider than a process underlying a team service. As already defined in Chapter 1.1, a process is seen as “a completely closed, timely and logical sequence of activities which are required to work on a process-oriented business object.” Since an end-to-end process involves the processing of several entities (e.g. quote, order, invoice, etc.), it is typically composed of several processes conforming to the definition stated above. Sub-processes perform a distinct set of actions on one entity. As described, high-level service domains may be specified by identifying the underlying business entities, hence processes as defined can be directly related. However, since service domains can be decomposed, sub-processes are typically associated with finer-grained domains and capabilities.

---

4.2.4.3 Business service specification

A business service specification encapsulates all elements that are relevant for a potential service consumer and thus have to be consequently exposed. These elements are the basis for the business service contract and can be seen as its realisation. The specification should contain a service preamble including the service name, a description and provider information. Furthermore, the specification should contain the exposed business service operations that are associated with that business service as well as an interaction model specifying the sequence of commitments from various partners. Additionally, specific terms and conditions that govern the use of the business service should be provided as well including policies and rules.\(^{302}\)

The business service operations are related to a specific business goal. Each operation is made visible to the service’s environment. The operation itself can be realised by the organisation itself or by external partners that implement the operation based on provided requirements or the underlying operation model.\(^{303}\)

4.2.4.4 Operation Model

The operation model of a business service operation details how the specific operation is executed. Hereby, the invocation of operations of other business services to achieve the desired goal stated in the business service contract has to be addressed as well as the processes utilised by the business service operation. The model can be used to communicate the service operations to the service implementer. The operational model also facilitates the reuse of business operations as described by business processes. The model can consist of processes, sub-processes or activities that solely belong to one business operation. These processes can as well be reused by other business operations comprised by the business service.\(^{304}\) This model does not need to be published to the service environment, since it is not necessary for the service consumer to understand how the specific outcome of a service is realised.

Some principles should be regarded while modelling the interaction between business services to ensure consistency and autonomy of the services. Business services at level k should not use business services from any coarser level (k-1 or above). Additionally, at any level, a business service can relate to business services offered by external providers.\(^{305}\)

---

The operation model itself orchestrates the underlying processes and the involved resources of the business service based on the functionality that should be provided by a business service operation. An orchestration describes the behaviour that a service provider performs internally to realise a certain provided service. This may include the cooperation with external business partners and the interaction with other services, although the focus lays on the fulfilment of one service operation. The result of an orchestration is a description of the activities and their parameters that may be used by other services to realise the provided service operation. However, an orchestration is intended to be executed. Thus, it can also be called an executable process.

4.2.4.5 Interaction Model

The interaction model as part of the service contract is used to define the commitments that have to be made by the service consumer and provider. However, the interaction model can be derived by service blueprints known from the area of service marketing.

Service blueprinting

The derivation of the interaction model is closely related to service blueprinting, which is addressed in the area of traditional service marketing and quality management. A service blueprint can be regarded as a two-dimensional model that represents the actions and tasks between the service provider and the service consumer in a chronological way. The horizontal axis represents the chronology of actions, whereas the vertical axis distinguishes between different areas of actions. The areas are distinguished by describing so-called “lines”.

The “line of interaction” reveals the division of activities between the organisation and its customer. The more activities performed by the organisation, the less activities reside in the area of responsibility of the customer. Tasks formerly performed by customers will then be performed by the organisation, if the line is adjusted towards the organisation. On the other hand, the customer will gain more control and responsibility if functions are transferred from the organisational sphere of control.

The “line of visibility” determines how much of an organisation’s activities are visible to the customer. If the line of visibility is shifted towards the customer, the customer may

---

gain a better understanding of how the task is actually performed inside the organisation. This may leverage the customer satisfaction by understanding why certain processes have such a long cycle time (e.g. order tracking).

The “line of internal interaction” distinguishes between front office and back office activities. All supporting activities that are necessary for the front office in order to deliver the service in an adequate manner are below the line if internal interaction.

Implications for the interaction model

With regards to the interaction model, the line of interaction is the main focus of the analysis. Services are always invoked by a service consumer. The participation can range from a one-time invocation of the service to a deep integration of the service consumer into the service delivery process of the service provider.311 Thus, one has to define which part of the service delivery has to be executed by the service consumer and what parts are actually executed by the service provider. Increased consumer participation can lead to more efficiency since the consumer carries out tasks that would otherwise have to be carried out by the organisation. However, this increase in efficiency needs to be based on good management on the service provider side, since missing or delayed consumer contribution influences the cost, time and tasks of carried out by the organisation.312

The organisation has to analyse what parts of the service can or should be managed by the customer and how the exception handling mechanisms have to be defined. One solution to overcome the risks associated with extending the line of interaction towards the consumer is an extensive standardised description of the service contract.313

Each business service has a service contract that governs the relationship between the service consumer and the service provider.314 The contract can be used to define the interaction model, or the interaction model can be used to define the contract. Thus, the relationship between the contract and the interaction model is bidirectional. The interaction model is the explicit way a business service defines how joint collaboration will be realised.315 As shown in Figure 4-9 the interaction can be modelled as a process, indicating that the interaction between the service provider and consumer is purely behavioural. The interaction model is obviously based on the operational model, although the focus is different. The interaction model visualises what tasks have to be provided or executed by the service con-

sumer and what tasks belong to the domain of the service provider. The tasks provided by the service consumer do not have to be known in detail since they solely belong to the domain of the service consumer. Only the inputs and outputs as defined in the business service contract bear a meaning for the interaction model. Hence, an interaction model for a business service operation represents a choreography that can be seen as the external view on a collaboration between a service provider and its consumer to achieve a certain goal. It abstracts from any tasks that the service provider performs internally, but rather describes the interactions between service providers and between service providers and service consumers that contribute to the common goal. Thus, a choreography does not describe how one single service interacts with other services, but it describes how a certain goal can be achieved through a sequence of commitments between service consumers and providers.

4.2.5 The prioritisation phase

4.2.5.1 Overview

A value classification enables the organisation to gain an understanding of which business services contribute most to the value of an organisation and the competitive advantage. This information is fundamental to make sourcing and funding decisions. A value chain or business network analysis may enable the organisation to understand its core critical functions as well as supporting functions. Each business service has been described in terms of its interactions, inputs and outputs. Based on this information and a value classification it may be beneficial for an organisation to outsource certain business services, such as accounting or payroll. Refer to Figure 4-10 for an overview of this phase including the inputs, activities and outputs.

4.2.5.2 Service value analysis

Ad hoc service classification

JONES proposed a classification matrix based on the business driver for service change and the decision driver for approval of change. He distinguishes four classes for the value of services. Please refer to Figure 4-11 for a visualisation of the different classes.

The services with the highest value deliver clear advantage to the organisation over its competitors. The business driver is the identification of change in the specific market. Opportunity driven services, on the next value level, are driven by identifying opportunities and are measured by their ROI. In the next cluster, one may group services that are only going to be changed in case of real issues at hand that make changes inevitable. Changes are driven by the cost of change. On the lowest value level, the key separation between the
services is the service level (e.g. performance levels, etc.). These commodity services have little value to the organisation, since they are chosen by market specific conditions.  

*The analytical hierarchy process*

The ad hoc service classification might be plausible, but it is not structured since the decision drivers for change are represented by an arbitrarily chosen factor. The Analytical Hierarchy Process (AHP) is a structured approach that takes into account financial and non-financial data along with the decomposition of a complex problem into finer-grained problems that can be “solved” more easily. Hafeez, Zhang and Malak propose the application of the AHP to determine the key capabilities of an organisation. Thus, this approach can easily be adapted to classify business services. However, if the business services have recently been deployed by the organisation, the measures for the calculation may not be available. In this case, the capabilities comprised in one service can be used as an input for the AHP as well.

The AHP for business services can be structured into three phases. The *first phase* addresses the determination of performance measures and the identification of business services. The performance measures should comprise financial as well as non-financial measures that are aligned with the strategy and objectives of the organisation (e.g. cost leadership). As inputs for the definition of certain measures, financial reports or documents about the business context can provide valuable information. Hereby, the measures that have been proposed by Jones can be included as well. The identification of business services addresses the number of services on different decomposition levels to be evaluated.

Once the measures and services have been identified, they have to be analysed using the AHP in the *second phase*. The AHP can be used for quantitative and qualitative analysis in scientific research, which results in an optimal outcome between decision alternatives. Here, the AHP is applied based on four principles, namely decomposition, prioritisation, synthesis and sensitivity analysis. First, the decision problem is decomposed into finer-grained elements. Second, the elements on one granularity level are compared in a pairwise manner and ordered. The next principle focuses on the hierarchic composition of the different priorities to provide the overall assessment of available alternatives. The last principle addresses the analysis of the stability of the outcome by applying a sensitivity analysis. Additionally, the AHP provides a consistency check that ensures that the judgement of the

---

322 Please refer to. Hafeez, Zhang, Malak (2002) for the following description.
decision-makers is consistent during the pairwise comparison. These four principles have to be applied on the AHP models developed for the financial and non-financial performance evaluation. The models should be composed as follows: On the top of the hierarchy, the goal is modelled. Hence, for the top-node for the financial AHP model the goal could state “Financial performance contribution”. On the second hierarchy level, the financial measures are annotated and on the third level, the different business services or capabilities are stated. On lower hierarchy levels, the constituent services or capabilities that are related to respective parent service and capability can be modelled. Based on the hierarchy models the other three principles can be applied.

The third phase consolidates the results of the two AHP models of the previous phase by developing a two-dimensional matrix. Please refer to Figure 4-16 for a visualisation.

![Figure 4-12: Classification of business services](image)

Source: Adapted from Hafeez, Zhang, Malak (2002), p. 46.

Figure 4-12: Classification of business services

The business services in the top-right corner are important based on financial and non-financial measures. The AHP can be used to group the business services based on different measures that have to be agreed on before and are based on business specific characteristics. The outcome can be used to guide and coordinate upcoming SOA projects, since the development of software services will not be implemented all at one time, but over a number of different projects. Furthermore, one may analyse if it is not more suitable for the

---

organisation to outsource some of its business services that are not profitable and core to the business.

However, one needs to keep in mind that the pairwise comparison of different measures, is purely subjective in nature since the comparison is made by humans. Further criticism can be encountered in recent literature. For example, it is seen as problematic that the verbal responses given during the pairwise comparison of the different alternatives are transformed to a numerical scale since the meaning of the verbal expressions are interpreted differently by humans.\textsuperscript{327} Hence, the top management of an organisation should be interviewed and discrepancies between the given information should be discussed collectively.

4.3 The derivation of software services

4.3.1 Overview

At this stage, a set of business services on different granularity levels have been identified. The derivation of software services can be structured into three distinct phases, namely a preparation phase, an identification phase and detailing phase, which encapsulate different specific activities. Please refer to Figure 4-1 for an overview of the phases for software service derivation.

4.3.2 The preparation phase

4.3.2.1 Overview

This phase is focussed on comprising activities, so that the service identification phase is leveraged as good as possible. Thus, the scope of the software service enablement has to be defined, the application portfolio has to be analysed, the processes underlying the different business services have to be modelled in an adequate way, and finally the suitability of processes and applications to be service-enabled has to be analysed as well. Refer to Figure 4-13 for an overview of the inputs, activities and outputs of the preparation phase.

4.3.2.2 Define the scope

The last phase of the first part of the consolidated approach classified the business services according to their value for the organisation in order to provide support for sourcing decisions. The derivation of software services should begin with business services that are considered important and valuable for the organisation and that would benefit the most from the achieving the goals related to the SOA value drivers identified earlier in order to derive quick sustainable advantages and benefits.\[328\] Since business processes typically have to interact in certain ways to deliver the proposed value to the customer, one may analyse the interaction model to identify which business services should be prioritised for software service enablement. Furthermore, the scope of the software service enablement should be aligned with the goals and drivers stated in the SOA strategy. Thus, software service enablement can begin by analysing the business and IT imperatives that are addressed by the SOA strategy. Based on that analysis, one needs to identify the business services that relate to the stated goals. For example, the SOA strategy states that service-enablement should enhance the customer experience on all channels. Hence, business services have to be identified that are exposed to external stakeholders. These business services serve as the foundation for the identification of software services.

Not all business services can or should be realised the same way, since some of these services may require the application of technology; others may be purely executed by manual tasks.\[329\] However, as described in Chapter 4.2.4, each business service comprises one or more processes that serve as the basis for the analysis and identification of software services. Ideally, each process should be supported by one service that in turn composes finer-

---

\[328\] Cp. Marks, Bell (2006), pp. 91-98.
grained services to achieve a close business and IT alignment. Hence, the underlying processes have to be analysed and decomposed.

4.3.2.3 Application analysis

Preliminary Considerations

An application analysis or existing system analysis has to be conducted in order to understand the applications and functions that are available and the systems that are suitable candidates for service-enablement. The analysis of the application portfolio of an organisation typically bears multiple applications that often have been developed independently. These applications are typically operating in silos with overlapping and redundant functionalities and data. The targets for this bottom-up approach will typically be redundant business logic and multiple copies of data entities, which in general may result in a high licensing and maintenance cost. Therefore, one has to understand the applications that provide the functionalities and data records in order to derive adequate services related to the business requirements. The application documentation that have been collected in the previous phase are used as the input for the application analysis.

The scope of the application analysis is based on the scope of the software service enablement. However, the analysis of a single application at a time has two main disadvantages. First, there is no analysis of redundancies between applications and therefore opportunities for streamlining the application portfolio might be missed. Second, other applications might utilise the specific application, but the interfaces are not exposed by the analysis of one single application. Thus, by making changes to the application’s interfaces or functionality, other dependent applications might not work properly anymore. The scope of the application analysis should be broad enough to cover the concerns described above.

Application portfolio analysis

The application portfolio analysis may include interviews and questionnaires in order to document key attributes of the application. Hereby, the main (internal) stakeholders of the legacy systems are interviewed to gain an understanding of what the legacy systems actually do. The stakeholders typically include owners and current end users of the legacy system. Thus, interviews should address the capturing of information about legacy systems,
end users, owners, contractors and the technological issues.\textsuperscript{336} The business artefacts that have been collected during the preparation phase of business services are validated by the conducted interviews to assure that the artefacts describe the actual situation within the IT domain. Additionally, for each application the business value, technical health, technical flexibility and the associated costs (e.g. operational costs, development costs, etc.) should be identified to gain more information about the portfolio composition.\textsuperscript{337} Hereby, the relative quality and maturity of the systems (e.g. outstanding problems, change history, user satisfaction and future requirements) can be addressed as well. One may create a list of characteristics about the existing applications, which may be updated throughout the analysis phase.

\textit{Components and functionalities analysis}

Once the characteristics of the legacy systems have been captured and the composition of the application landscape has been exposed and evaluated, the capabilities of existing systems should be analysed.\textsuperscript{338} Thereby, components\textsuperscript{339} and functionalities of the applications are identified and documented. Interviewing techniques are a legitimate starting point to gather information about these components. Thereby, based on the characteristics that have already been identified, one may question technical personnel about the architecture, design paradigms, code complexity, level of documentation, module coupling and existing interfaces between applications and end users (including dependencies between other applications and commercial products).\textsuperscript{340} This step is highly dependent on the organisation specific application landscape. Considerations have to be made concerning the application’s underlying technology and the availability of the application’s source code (e.g. which programming language has been used during the development). Furthermore, the layering of an application may have a huge impact on the ease of service enablement of the application.

A thorough analysis of the application by using automated tools for scanning the source code and gathering metadata about the application may complement the interviews.\textsuperscript{341} The main goal is to gain an understanding of the composition of the application itself and to derive comprehensive attributes for each application.\textsuperscript{342} The automatically derived attributes complement the interviews as a basis to identify the applications that belong to the

\textsuperscript{336} Cp. Lewis et al. (2005), p. 9.
\textsuperscript{339} Components are binary units of independent production, acquisition, and deployment that interact to form a functioning system. Szymerski (2002), p. 3.
\textsuperscript{341} For example, WebSphere Studio Asset Analyzer: http://www-306.ibm.com/software/awdtools/wsaa/
operational environment of an organisation as well as the components and functionalities
that are provided by each application. In addition to the documented characteristics of the
applications, one may establish a component table that captures component or functionality
specific characteristics. Redundant functionality may be exposed by this analysis. Thus,
one may consider consolidating certain applications as well as migrating or deleting redund-
ant functions to derive a more consistent application landscape.343

4.3.2.4 Process decomposition

Preliminary considerations

The main sources for the derivation of software services are business process models or
other business artefacts related to the organisation’s processes (e.g. capability models, or-
ganisational charts, etc.). These models are used by many organisations to document the
way the actual work is performed and may have been the outcome of formerly conducted
Business Process Management (BPM) projects.344 Nevertheless, to achieve a high-quality
set of software services, one has to make sure that the modelled processes reflect the actual
business of the organisation because services cannot make up for a misalignment between
the actual working environment of an organisation and its processes. Thus, prior to the
engagement in software service identification as part of a SOA initiative, one may have to
consider revisiting and reengineering the organisation’s processes and process models.345

Process detailing

The process models should have been built upon certain modelling conventions that make
them more suitable in the service analysis phase to identify and design adequate services.
Three different aspects have to be addressed by the conventions.346

The process models might be modelled on different granularity levels. Depending on how
thoroughly a process classification has been conducted, one can differentiate between three
granularity levels.347 Business processes at the most abstract level of granularity can be
decomposed into sub-processes and subsequently into atomic process steps. The processes
identified during the detailing phase of the business services can be regarded as sub-
processes that are part of one or more abstract processes. For the identification of software

343  Cp. Hess (2005), p. 39. The migration of functions and the retiring of complete application is a major
change in the application landscape and has to be analysed thoroughly in advance. However, the main
goal of identifying redundancies within an application landscape is to raise awareness of these situations
and deal with them appropriately in the service analysis phase.


services, these sub-processes have to be decomposed. Since elementary process steps cannot be decomposed any further without increasing the understanding of the overall process logic, the identification of software services will start at this granularity level. The originally documented business process model may describe the process steps on another granularity level. To provide support for the adequate granularity, one may apply frameworks or mapping proposals.\textsuperscript{348}

Once the process has been decomposed into its constituent process steps, one can annotate roles or organisational units. The organisational units involved in the process should be modelled related to their specific functions.

### 4.3.2.5 Suitability analysis

#### Preliminary considerations

The suitability analysis comprises activities related to the assessment of processes and applications in regards to service enablement. A mapping between the processes and the applications serves as the foundation for the suitability analysis.

#### Process-to-application mapping

After the application landscape has been analysed and a set of processes has been identified related to the scope of the SOA initiative, a mapping between the processes and the applications has to be conducted in order to identify the processes that may actually leverage the existing application landscape.\textsuperscript{349} From the perspective of the process, the mapping enhances the understanding about the processes and activities that are dependent on certain applications and where redundant implementation of application functionality is located. From the application perspective, the mapping shows the business functions that are supported by each application.\textsuperscript{350} The relationships between business processes and applications need to be determined as a prerequisite for service-enablement.\textsuperscript{351}

The information gathered during the mapping activity has to be used to detail and extend the information provided by the process models. The process models should describe the actual IT support for each function. The functions may be completely conducted by manual activities or be fully automated. Semi-automated functions may involve a dialog between the application / IT system and its user.\textsuperscript{352} By extending the process models regarding the

IT support for each function, one can also annotate the process related applications or components for each function or process.

Suitability assessment

Not all processes and applications can be service-enabled based on specific constraints. Hence, one needs to analyse the suitability of service enablement for processes as well as for the applications. The information gathered about the applications and their functionalities have to be analysed in order to decide if an application is feasible to be service-enabled or not. Based on this information, one can rank applications according to their suitability for service-enablement. General characteristics, such as the health of the system, the maturity and future plans can be taken into account for the decision making process. However, technical characteristics should be regarded as well. Technical requirements for service-enablement are dependent on the choice of the technology that will be implemented (e.g. web services). Since implementation issues are not in scope for this thesis, technical requirements will not further be analysed. Nonetheless, as a general requirement one should analyse if an interface for the application’s functionality can be established.

Once the process has been broken down into its constituent, elementary process steps and the IT support for each function has been annotated based on the process-to-application mapping, one has to identify the processes that cannot be automated by software services. SPROTT ET AL. proposed business process characteristics, to conclude what processes might benefit the most from service-enablement. One of the proposed benefits of a technical SOA is the integration of heterogeneous application platforms and technologies. Hence, the more actors and stakeholders are involved in executing a certain cross-functional process, the greater the benefit from service-enablement is since different stakeholders may utilise different technologies for participating in that process. Services are hereby used to integrate the different technologies of the participants. If a process needs real-time information, service-enablement may be used to develop centralised services that provide all service consumers with consistent data. If the process is dynamic, it may also benefit from service-enablement since the process can be reconfigured by combining different services to fulfil new business requirements. Instead of proposing qualifying criteria for processes, SEWING, ROSEMANN and DUMAS propose a set of disqualifying criteria regarding the

356 The authors propose the criteria for web services.
357 Please refer to Chapter 2.2.3.
358 Cp. Sprott et al. (2003), pp. 70 f.
service-enablement suitability of business processes.\textsuperscript{359} The proposed criteria that make a process not suitable for service-enablement are:\textsuperscript{360}

- The process consists of only manual tasks that cannot be digitised,
- human intelligence or sophisticated interpretation is required,
- it is an isolated process that is stable and efficient and cannot to be leveraged,
- the process is (partly) performed by a legacy system that cannot be service-enabled.\textsuperscript{361}

The potential benefits of services based on processes with these characteristics can be considered as low and thus analysis and reengineering effort may not be justified. The remaining processes can then be matched against criteria that measure whether the business needs match with the drivers for service-enablement and whether the technical requirements for service adoption can met by the existing infrastructure.\textsuperscript{362} SAP proposed further criteria that could be used to assess the benefits of service-enablement for a business process.\textsuperscript{363}

4.3.3 The identification phase

4.3.3.1 Overview

The identification phase is purely focussed on the derivation of software services. Hereby, two main sources of information are needed. On the one hand, entity models are needed to derive the entity services related to the core business objects of each business service. On the other hand, detailed process modelled should be in place to be analysed for service-enablement.

\textsuperscript{359} Cp. Sewing, Rosemann, Dumas (2006). This approach consider web services as an implementation technology for a SOA.
\textsuperscript{360} Cp. Sewing, Rosemann, Dumas (2006), pp. 23 f.
\textsuperscript{363} Cp. SAP (2005), pp. 24 ff.
Figure 4-14: The identification phase for software services

4.3.3.2 Entity analysis

The entity analysis starts by examining the primary entities of one business service and thus approaches the service identification from a data modelling perspective.\textsuperscript{364} Based on an entity analysis, one can identify objects that should be represented as entity services, which in turn contain a certain set of operations that are necessary to process the related entity.\textsuperscript{365} Entity services are very generic and reusable in nature since they are not tightly coupled to processes, meaning that the provided interface of that service is not process specific.\textsuperscript{366} Since these services may not contain any process logic, they require a parent service or controller, what makes them dependent to a certain extent. The identification may be based on simple brainstorming techniques when no business artefacts are in place. Another possibility to define the boundary of an entity service, one has to analyse the actual context of the service. This can be achieved by examining process models. Processes might be analysed to define the entities that are processed and the operations that are used for processing the entity. The method proposed by Ivanov and Stähler states that services should be made out of (sub-) processes that handle only one specific business entity.\textsuperscript{367} This actually implicitly refers to the derivation of entity services.

Class-diagrams in combination with process models can also be used as an entry point for the identification of services.\textsuperscript{368} Since the granularity level of classes is too fine-grained for a direct transformation to (entity) services, one has to analyse the behaviour of each object

\textsuperscript{367} Cp. Ivanov, Stähler (2005).
\textsuperscript{368} Cp. Zimmermann, Krogdahl, Gee (2004); Rahmani et al. (2006), pp. 581 ff.
to abstract objects and operations into services.\textsuperscript{369} Thereby, process models might be analysed to identify the related operations to certain objects. Classes visualized in the class-diagram should be aggregated to enhance the granularity level. For example, the class “Work order” and “Work order item” might be combined to create the “Work order service”.\textsuperscript{370} Aggregated classes may be transformed to entity services. Unlike class models, the identified services, or service models representing all derived services, do not visualize any interaction, business rules or business events. Therefore, a parent service or controller needs to be developed to ensure the invocation of services and their operations at the right time based on business rules.\textsuperscript{371}

KRAFZIG, BANKE and SLAMA refer to entity services as data-centric services.\textsuperscript{372} An entity service might guarantee that the related entity (e.g. order) is properly created to all the business rules for data completeness and validation.\textsuperscript{373} Their main purpose is to handle persistent data including the storage and retrieval of data, locking and transaction management. Thus, entity services act similarly to the traditional data access layer of traditional applications. However, where data access layers serve the entire organisation, entity services enforce vertical layering of data since they encapsulate one business entity.\textsuperscript{374} Any service that needs specific data access must use the interface of that entity service. Figure 4-15 visualizes the concept of data ownership.

\begin{center}
\textbf{Figure 4-15: Ownership of data}
\end{center}

\begin{center}
\textsuperscript{370} Cp. Rahmani et al. (2006), p. 581 refer to these kind of entities as “value objects”.
\textsuperscript{373} Cp. SAP (2005), p. 19.
\end{center}
Typically four operations of entity services can be identified, namely create(), read(), update() and delete() (CRUD-operations). These operations should not be interpreted literally. For example, instead of the operation “Read”, one may rather use the operation “List” since it fits better to the context of the service. Hence, the operations “GetCustomer” or “UpdateCustomer” are not services themselves, but rather invocation points within a service. Nevertheless, there are certain considerations that go along with the development of entity services since cross-references between services are not allowed based on the service design principle of autonomy. To exemplify the problem of cross-references, one may visualize the scenario where a new customer wants to order a certain set of products from the online shop of an organisation. A new order has to be created as well as a new customer. Thus, entities managed by entity service must be sufficient self-contained or must contain a unique identifier that is used to determine relationships between objects. Alternatively, MINHAS and VOGT propose to design certain packages for predefined queries and operations that can be encapsulated and exposed as one service. VOGELS states that garbage collection, dynamic object creation, state management, dynamically created object references, and a variety of reliability and transactional mechanisms as known from the object-oriented systems, have to be addressed when using services for objects.

Since the transition from a central data layer towards a vertical sliced SOA with entity services acting as data access points, the assignment of entities to entity services becomes a main design decision with major impact on many characteristics of the resulting applications including the integrity and consistency of data. Entity services may also encapsulate more than the CRUD-operations based on the principle of reusability.

4.3.3.3 Process analysis

Preliminary considerations

Given the fact that processes have been modelled or adapted based on the conventions described in the last phase, a set of processes within a business service has been identified to be suitable based on various factors. Now, each process needs to be further adapted to be suitable for specific factors for service-orientation. Subsequently, the process is analysed...
and a set of software service can be identified based on the application of the service design principles described in Chapter 3.3.7.

Before services are derived from the process model, one should understand the relationships between services, processes, operations and messages. Figure 4-16 illustrates the relationships between these elements from the perspective of a process.

![Figure 4-16: Relationship between services and processes](source: Adapted from Erl (2005), p. 289.)

A process model describes the logical sequence of activities related to the accomplishment of one specific business task. An instance of the process can be seen as an execution of the process model. A process instance can compose certain services to achieve the specified task. The instance is not necessary defined by the composed services since it may only use a subset of the functionality provided by the services. The functionality of the service can only be accessed through the operations that are exposed through the interface of a service. Nevertheless, based on the underlying process flow, the process instance invokes a certain set of operations in a certain way to achieve its task. Operations may send and receive messages in order to execute their operations. They can be seen as single logical units of work.

Given the fact that proven process models are in place, one can derive services from the process logic. The main difficulty in deriving services based on process models is the decision about the right scope of logic that will be implemented by a service. Figure 4-17 provides an illustration of different services with different encapsulated logic.

---

Figure 4-17: Logic encapsulated in services

Services can encapsulate a task based on a single process step, a sub-process comprising multiple steps or even the entire process logic. Thus, when deriving services from process models, one has to ensure that the underlying logic and their implications are well understood. Service design principles that have been described and analysed in Chapter 3.3.7 are directly focusing on the issue of grouping certain process steps to services.

Visibility and interaction analysis

The concept of service blueprinting has already been introduced in Chapter 4.2.4. The line of interaction has been used to derive the interaction model for business services. At this stage, the lines of interaction and visibility have to be analysed again. KLOSE, KNACKSTEDT and BEVERUNGEN applied the findings of traditional service marketing research that focuses more on the integration of the customer into the service processes of an organisation, into the field of process analysis and software service identification. Since the interaction model has already been defined along with the analysis of the line of interaction, this model can also be used as an input or guide of further analysis. The interaction model shows the commitments made by the service consumer and provider. It is aligned with the operation model that details how a business service operation utilises the underlying business processes. The interaction model already details the tasks that lay in the sphere of control of the service consumer. The objective here is to identify how these tasks can be integrated in the business processes of the organisation and how these tasks can be supported by software services.

Source: Adapted from Erl (2005), p. 34.

The line of interaction specifies the parts or functions of the process that may be taken over by the service consumer. Especially with multiple channels facing the consumer, one has to decide what process functions may reside in the sphere of control of the service consumer. The line of visibility defines how much of the process should be visible for the stakeholders. The stakeholders may comprise external business partners (e.g. customers, suppliers, etc.) as well as internal partners (e.g. subsidiaries). These stakeholders have already been identified and analysed during the business service detailing phase described in Chapter 4.2.4 and can be used as an input for this activity.

Since each function within a process may be a potential candidate for the redesign towards visibility, the focus should reside on areas that are of strong interest to the customer. Furthermore, functions that are carried out with a high level of perfection are suited better to make them visible to the customer, than functions that concise a lot of error handling and exceptions. As exemplified in Table 4-3, one needs to analyse each function of a process in terms of the takeover and visibility potential regarding the stakeholders.

<table>
<thead>
<tr>
<th>Function to be evaluated</th>
<th>Inter-organisational unit / IT System</th>
<th>Service analysis - Line of interaction / visibility</th>
<th>Group of stakeholders</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Takeover</td>
<td>Other facilities</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Commercial representative</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Visibility</td>
<td>Customer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Supplier</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Service provider</td>
</tr>
<tr>
<td>Function</td>
<td>Org. Unit System</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td>X</td>
<td>X1 : only customer specific operations</td>
</tr>
</tbody>
</table>

Space for annotations to note specifics or constraints

Source: Adapted from Klose, Knackstedt, Beverungen (2007), p. 1807.

Table 4-3: Analysis of line of visibility and interaction

By abstracting from the interaction of the customer of the organisation, one can shift the focus towards internal and external business partners. If functions, which have been performed by the organisation before, are going to be performed solely by external stakeholders, the information flow will have to be restricted or redirected in a proper manner. Different questions need to be asked by analysing a function based on its takeover and visibility potential (e.g. Which channels should be used to provide customers with the required information? / Which activities is the customer capable to provide in a timely man-

---

By analysing process functions based on their visibility and interaction potential with stakeholders, one can identify potential groupings of functionality that must/should be explicitly exposed to the organisation’s stakeholders by means of services. One of the goals and benefits of a SOA is the facilitation of customer integration by providing new communication channels. By exposing certain services this goal can be achieved. However, as pointed out by Tate, Hope and Johnstone, who explored the impact of multi-channel interactions on the line of visibility and on customer perception on quality, a consistent customer experience throughout the multitude of communication channels has to be facilitated in order to enhance customer satisfaction.

The software service identification process

The identification of software services can be structured into different steps. The outcome of this process is a first set of potential software services that has to be adapted and revised during subsequent activities. The identified services can be classified according to the classification scheme presented in Chapter 3.3.6.

Step 1: Identify service operation candidates

Once the process has been decomposed into its fine-granular process steps, one should identify potential service operations. Each process step can be regarded as a potential service operation. However, all process steps that represent solely manual tasks or process steps that are executed by a legacy system that cannot be service-enabled have to be excluded from the potential logic that can be encompassed by a service.

Step 2: Abstract process logic

One has to abstract the process logic from the process model that serves as the foundation for any composition of services later on. One may abstract business rules, conditional logic, exceptional logic and sequence logic. Some process logic may not actually be visualised by an individual action or process step. Depending on the modelling conventions (e.g. modelling language) an organisation has used while deriving the process models, some process steps might consist of an condition and an action (e.g. if A occurs, then perform B). One can simply shift the condition as information that has to be regarded when

---

393 The described steps have been adapted from Erl (2005), pp. 399-415.
identifying the service operations within a process service and leave the action as part of the process.\textsuperscript{396}

Step 3: Define logical contexts

The remaining process steps should be grouped based on their logical context. Thus, the identified context confines the service boundary. Hereby, the design principle of \textit{service cohesion} plays the most important role.\textsuperscript{397} The objective is to group operations together that are functional related as this is seen as the strongest form of cohesion. Furthermore, one should analyse the business entity that is processed in order to relate the respective entity service.

Step 4: Refinement

During this step, further design principles have to be applied to refine the operations and the boundaries of the preliminary service candidates. Hence, the service operations based on the underlying business logic and process steps have to be analysed for potential readjustments. The design principle of \textit{reusability} can be applied to specify further operations within the boundary of a service. Depending on the scope of the service identification, one may address the targeted reusability and define service operations that have the highest potential to be consumed in different scenarios. However, the added operations should still relate to the logical boundary of the service. The design principle of \textit{coupling} can also be applied to identify sequential dependencies between operations. Sequential operations, which are only depended in one way, may be combined inside a service.\textsuperscript{398} One may also identify process steps that are recurring within that process, which can be grouped together into a single service. New services may be created as well depending on new logical contexts that may be identified. For example, if two service candidates both provide some kind of transformation operation of documents, it may be advisable to define a new transformation service, which encapsulates these operations. The visibility and takeover potential annotated to the process steps may provide a guideline for the grouping of process steps and consequently for the definition of task and entity services. CHANG and KIM propose to develop two different services, if the service invokers are different since the services could be invoked at different times.\textsuperscript{399} Furthermore, one can identify services that are purely technology related and business-logic agnostic. Thus, these services can be classified as utility services.

\textsuperscript{397} Cp. Chapter 3.3.7.
The consolidated approach

Step 5: Define compositions

Once the services have been identified, they have to be “tested” to identify further potential for enhancements and adjustments. Scenarios have to be developed than can be applied to the service in order to identify any chances for composition and consolidation of services and to evaluate the appropriateness of the service boundaries. Furthermore, one can discover in which situation logic is missing and can shift the related business rules to the task services or process services. Consequently, new services may be created. The main objective is to specify process services that compose the task, entity and utility services related to the underlying process. Based on the visibility and interaction analysis, one may create process services that are exposed to a specific set of stakeholders.400

4.3.4 The detailing phase

4.3.4.1 Overview

This phase comprises all activities that should be conducted after services have been identified based on processes and entity models. Please refer to Figure 4-18 for an overview of this phase.

Figure 4-18: The detailing phase for software services

One has to verify that already existing services do not overlap with the newly discovered ones. New services should be detailed further in order to identify additional overlaps with already existing services and in order to make the services more reusable and autonomous. The identified operations and services should be mapped on the existing application layer in order to identify missing functionalities or the need for additional services. These two

steps may lead to the identification of new services and therefore need to be part of the service analysis phase. The last step of the service analysis phase includes the exposure of the service candidates in a service inventory that will be used for further analysis and identification efforts.

4.3.4.2 Verification of service candidates

After the services have been identified, they have to be named according to the organisation’s guidelines and the function of the service has to be specified. If the function is already provided by another service, the service can be reused to avoid overlapping functionalities according to the design principle of autonomy. Additionally, one has to assess if the service meets the business needs of the process and roles.\textsuperscript{401} If the service or service operation already exists, one has to decide if the existent service can be reused or if it has to be extended or adapted. One should never offer operations with the same functionality in two different services. This leads to redundancies, which in turn lead to high maintenance costs and inconsistencies. Services may already be existent, if processes have recurrent process steps. Thus, by analysing different processes in terms of communalities, one can typically abstract from certain process steps and group them together to form a service, which can be shared by these processes.\textsuperscript{402} If the verification confirms that similar services are nonexistent within the organisation, the identified set of services has to be detailed further.

Nevertheless, one has to keep in mind to document each step in order to adjust the established ontology and to be able to understand which changes have been made when revisiting the services at the further stages of their development.

4.3.4.3 Operations detailing

A service can encompass one or more operations (see Figure 4-16). For each operation, one can identify the input and output parameters in order to minimise the coupling between different operations.\textsuperscript{403} If each operation is as loosely coupled as possible in terms of input and output parameters, the interdependencies between applications can also be minimised since operations might utilise multiple applications for the output creation. Input and output parameters are defined and described in the service interface. Therefore, FEUERLICHT and MEESATHIT propose an approach for the service interface design that build on the co-

\textsuperscript{403} Klose, Knackstedt, Beverungen (2007); Kohlmann, Alt (2007) detail the operations of the identified services. However, they fail to provide details.
hesiveness and coupling principles. They propose the general rule that only data that is used directly by a given operation should be exposed as parameters to maximise encapsulation (cohesiveness) and minimise coupling. Therefore, the data properties of the parameters have to be analysed. Since parameters are seen as data, functional dependencies between parameters and the application of data normalisation rules have to be considered. Functional dependencies are the theoretical basis for normal forms and the definition of data normalisation rules. Data normalisation rules are typically applied in database design to eliminate redundancies. These rules are applicable for the design of adequate service interfaces, so that the coupling between operations is reduced and redundant parameters are eliminated. Thereby, one can formulate requirements for operations to ensure well-designed service interfaces:

- All input parameters are used as data and must not control the operation execution
- Input and output parameters form a minimal set

A minimal set of parameters refers to the definition of parameters that are mutually independent from one another. Thus, the parameters cannot be derived from each other because of functional dependencies. Based on the type of operation, one can define further requirements for the parameters. Venners classified operations into three different types, namely state-view operations, state-change operations and utility operations. State-view operations, also named query type operations, return data in output parameters based on the data provided by the input parameters. State-change operations result in update, insert and delete transactions based on the provided input. These are therefore known as update type operations. Utility operations do not use any parameters (e.g. notifications). For state-view operations, an additional requirement can be formulated.

- Output parameters must be functionally dependent on the input parameters

The output of the operation has to be directly connected to the input parameters and must not contain any additional information. Data normalisation rules can be applied when considering the input parameters as key attributes and the output parameters as non-key attributes. Hence, non-key attributes have to be fully functional dependent on the key attributes to ensure cohesive, loosely coupled operations. Applying data normalisation techniques to

---

404 Please refer to Feuerlicht (2005); Feuerlicht, Meesathit (2004) for the following description about operation detailing.
the parameter sets of operations ensures low coupling and high cohesiveness. However, these normalisation techniques cannot directly be applied to state-change operations (update type), since these operations typically use the input parameters to create or change data sets (e.g. insert record) and return a value as an output parameters that may only contain values symbolising an acknowledgement or identifying the manipulated data set.\textsuperscript{410}

A service design can be enhanced by applying the principle of reusability. Reusability of a service is addressed, when the operations are too specific regarding their inputs and have to be redesigned to provide more generic input parameters related to business requirements. This may lead to the identification of operations that overlap within or between services, which leads to the specification of new services or operations. Hence, the newly identified services have to be verified to ensure that these services are not already existing.

4.3.4.4 Mapping

Once the desired functionalities of the applications using a top-down approach have been identified and the existing functionalities of the application landscape have been analysed from the bottom-up, a mapping between both approaches has to be conducted. Since the operations within the services have been adapted by incorporating desired business requirements, the gap analysis has to point out, to what extent the application can support the desired state.\textsuperscript{411} Figure 4-19 visualises a simplified mapping between potential services and the applications with their respective functionalities.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure419.png}
\caption{Mapping from services to application functionality}
\end{figure}

Source: Adapted from Inaganti, Behera (2007), p. 7.

For each operation candidate within the identified software service, one has to analyse the underlying processing requirements, especially the application logic that needs to be exe-
cuted to process the action of each operation candidate. Subsequently, one has to identify which application logic already exists in order to make decisions about the development of the specific logic, and the sourcing of the functionality by a third party service provider. Additionally, one has to identify if more than one application is needed to process the required actions.\footnote{Cp. Erl (2005), p. 413.} This part of the analysis can be supported by using the documents that have been developed during the previous phases.

One may also break down the application logic requirements into smaller steps in order to identify new operation candidates within a proposed service. The steps can be grouped in accordance to the design principle of cohesion. Thus, one may group the steps associated with a specific legacy system or with one or more solution components. In addition, one may also group these steps according to the type of function.\footnote{Cp. Erl (2005), p. 414.} However, it may be possible that all the operation candidates identified in the previous phase are supported by the application portfolio and do not need to be revised. Thus, one needs to analyse the original service compositions and identify if any changes need to be made concerning the inclusion of new services or operations.\footnote{Cp. Erl (2005), p. 414.}

The gap analysis may also result in missing functionality that cannot be provided by the existing functionality. Two different approaches can be identified for sourcing the desired functionalities that are not provided by existing applications. The first one is to develop the functionalities from scratch. This can be an enticing opportunities for developers to use new technology. Nevertheless, many resources have to be used to develop, test and deploy these new services. Additionally, services have to be reliable and therefore have to inhibit a certain quality standard.\footnote{Cp. Sneed (2006), p. 5.} The other approach is to buy in the new functionality from external organisations. The problem here is that these functionalities rarely fit perfectly on the requirements of any particular organisation and therefore adaptors have to be built in order to utilize the third party service. Furthermore, one is dependent on the external organisation in terms of maintenance and reliability.\footnote{Cp. Sneed (2006), p. 5.} As an alternative, one may also consider to develop function-adding services. These services provide the functionality of the original service and add the required new characteristics.\footnote{Cp. Krafzig, Banke, Slama (2006), p. 78.} Third party services may not expose their source code. Thus, in addition to an adaptor, one may also implement additional functionalities to modify the output of the original service. A function-adding service may also act as an evolutionary step since it can be used as a placeholder for the development of a new service that actually has the new function implemented.
If new application services have been identified, they have to be verified again, since it may be possible that the functionality is already exposed as a service.

4.3.4.5 Establish inventory of service candidates

Once a set of service candidates have been established, one should list them in a service inventory. This inventory must not be the actual service repository of an organisation or a domain, since the services are still preliminary. Services in a service repository are actual services that are completely specified and executable. However, since the service analysis phase may encompass the identification of services based on more than one process, the identified services must be repatriated to serve as a basis for further services that will be identified through the analysis of other processes. If there is no inventory of preliminary services during the service analysis phase, a proliferation of redundant services may be supported that do not match with the considerations addressed by the service design principles. For example, if services expose the same operations, the service design principle of autonomy is disregarded. If the services are not analysed and specified in accordance to the design principles, the overall quality of the service design will suffer.

For each service that has been identified and verified, service-specific details have to be documented. These details comprise the service identifier (e.g. the name) including a service description, the service type, input and output parameter and the service consumers. These details should be affiliated to a service list that can be used to discover services and facilitates the reuse potential of already existing services.

4.4 A comprehensive example

4.4.1 Overview

This chapter will provide a fictitious example of the service analysis efforts within a SOA initiative of a manufacturing organisation, Dash Inc. that has decided to analyse their business in terms of service-orientation. The organisation existed for several decades in the Australian market and has been conducting business with several suppliers and customers in different business networks. However, based on the dynamic market conditions and volatile customer demand, the top-management of the organisation has decided to adapt

---

420 The provided example will address all phases and activities that have been proposed by the consolidated approach. However, based on the scope and the focus of the approach not all activities will be addressed as expressive as they would be addressed in a real application scenario. The example will adapt examples and related figures published by Erl (2005); Hafeez, Zhang, Malak (2002); Jones (2006); Klose, Knackstedt, Beverungen (2007).
the organisation to these conditions and facilitate organisational agility. The organisation has been specialising in manufacturing standardised car dashboards for major car manufacturers, such as Volkswagen, Holden and Audi. It has also been delivering customised products to private end-consumers as well as to organisations that modify the car interiors.

One year ago, a project team was formed with the goal to analyse how the organisation could benefit from service-orientation. The example details the different activities of the project team.

4.4.2 The derivation of business services

4.4.2.1 The preparation phase

Based on the business context that had been analysed as part of a former project, the first task for the project team was to identify the business and IT imperatives. The business context had been analysed in order to realign the business strategy with the organisation’s external environment. The team had recognised that Dash Inc. was the market leader in manufacturing standardised and customised dashboards in the Australian market and the business strategy had addressed the need to maintain this position and to enhance customer interaction, especially in the area of customised products.

Members of the project team conducted several interviews with the top-management of the organisation to identify the business and IT imperatives that served as the underlying basis for the SOA strategy. They identified that especially the integration of customers in the organisation’s processes should be focussed on to enhance customer satisfaction. Furthermore, it was shown that the existing IT was very inflexible. If changes had occurred on a process level, the IT department would have needed a long time to implement these changes. Thus, time-to-market advantages disappeared. They also identified that although Dash Inc. was the market leader in its respective market, it was very hard to do business with the organisation, since it took a lot of time and effort to integrate the customer. IT imperatives were identified as well, namely the development costs of new functionalities impeded other important projects that needed funding. Based on these imperatives, a SOA strategy was formulated that addressed these imperatives. The SOA value drivers associated with the SOA strategy were defined as well, e.g. improve business agility, reduce development costs, improve customer integration and improve the business process visibility and control. Different approaches to measure the success of the SOA initiative related to the value drivers were proposed as well. For example, the advancements in the development costs were measured based on a comparison between the costs of past and present projects that focussed on the implementation of certain functionalities. Based on a presen-
tation of the SOA strategy, executive support for a comprehensive service analysis phase within a SOA initiative was granted.

The next task for the project team was to establish a business ontology to ensure a consistent use of terminology throughout the SOA initiative. They were able to access an existing ontology and decided to extend this ontology throughout the analysis phase.

The team also collected multiple documents describing the organisational structure of Dash. Inc.: Enterprise architecture models, business process models, organisational charts and entity models. After three interviews with senior executives and business analysts had been conducted, they decided these documents were up-to-date, as they had been developed shortly before the SOA initiative. Thus, they did not have to be realigned with the existing ontology. Furthermore, the team collected application documentation that described the existing application landscape.

4.4.2.2 The identification phase

As the first step towards the business service identification, the capabilities of the organisations needed to be explicated. Since the organisation had not documented any capabilities before, the project team decided to identify the capabilities of Dash Inc. by utilising the process classification framework that had been developed during past BPM projects. Hereby, they identified capabilities based on the main clusters of the process classification framework. The first cluster encompassed all processes that were related to the design and development of new products or services. The second cluster grouped capabilities based on the generation of demand and the third cluster grouped capabilities together that addressed the delivery of products and services. These clusters along with the results of interviews with senior managers of Dash Inc. were used to identify and order the capabilities as shown as an excerpt in Figure 4-20.

2. Generate demand
   2.1 Manage channel strategy
   2.2 Manage promotional activities
   2.3 Sales Management
      2.3.1 Manage outbound sales
      2.3.2 Manage inbound sales orders
      2.3.2.1 Manage quotes
      2.3.2.2 Determine stock
   2.5. Develop sales strategy

Figure 4-20: The decomposition of capabilities
After the capabilities had been identified, relationships between the capabilities were annotated to the capability model. Hereby, input/output connections between capabilities were modelled as well as control and support connections. Additionally, the team annotated different attributes for each capability, including the inputs/outputs, pre- and post conditions, and owners. After the capability model had been established, project members analysed the underlying entities, the existing owner of the capabilities and the relationships between the capabilities to specify the boundary for business services that were to be exposed to the service consumer. To guide the identification process, they applied the domain analysis approach.

The first step aimed at identifying the core domains of Dash Inc. The organisational chart that had been collected previously was used as a starting point. Nevertheless, since the project manager knew that in recent times the organisational structure of Dash Inc. had been subject to frequent changes, he decided to only use the organisational chart as a guideline for the domain identification. The project team identified the entities of the organisation by analysing the collected entity models and organised a brainstorming session with the top-management to find a consensus on what the business of Dash Inc. was focussed on. In this session, four primary domains were identified that represented the business of Dash Inc., namely finance, logistics, manufacturing and sales. For the next step, they analysed the documents about the stakeholders of the organisation that had been developed by the marketing team in former projects. Three general classes of stakeholders were identified to be relevant for the service analysis phase, namely customer, supplier and logistic companies. The customers of the organisation included end-consumers, car manufacturers and car customisers. The next step focused on identifying the interactions between the domains and between the domains and stakeholders. Hereby, the goals and requirements of the stakeholders were used to guide the identification of the reason of interaction. This information was also available based on the previously conducted analysis of the marketing team. After the first three steps had been executed, the following domain model was developed (see Figure 4-21).
After the model had been established, the identified capabilities were mapped to the domains. This was done by comparing the underlying entities of domains and capabilities in terms of communalities and analysing the support of the capabilities regarding the interaction between domains and actors. The relationships between the capabilities were used to analyse how frequent and intense the interactions between capabilities were, in order to argue about the allocation to certain domains. The mapping was supported by using the template visualised in Table 4-4.  

![Diagram of the domain model](image)

**Figure 4-21:** The domain model

<table>
<thead>
<tr>
<th>Domains</th>
<th>Capabilities</th>
<th>Sales management</th>
<th>Manage promotions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Determine stock</td>
<td>Manage quotes</td>
</tr>
<tr>
<td>Manufacturing</td>
<td>Testing</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>R&amp;D</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Production</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales</td>
<td>Presales</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>Logistics</td>
<td>Quoting</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Packaging</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Shipping</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finance</td>
<td>Credits</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Billing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supplier</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logistic Company</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

P = Provider; C = Consumer

**Table 4-4:** Mapping of domains and capabilities

---

421 Due to space constraints, only an excerpt is visualised.
The capabilities were allocated to the respective service domains. Additionally, potential provider-consumer relationships between domains and external stakeholders were indicated.

The first three steps were applied to decompose the domains even further until fine-grained service domains were identified. The capabilities that had been allocated to the high-level domains were reallocated to the finer-grained domains. The boundary for each business service was drawn based on entity mapping, ownership considerations and interaction analysis. Subsequently, each service domain was responsible for one or more business services. The ownership considerations were closely aligned with the governance structure of Dash Inc.

4.4.2.3 The detailing phase

Once the business services had been allocated to their respective service domains, they were detailed in order to understand their relationships with each other and their environment. As a first step, the project team detailed the resources of business services. In the following, the coarse-grained business service “Quote service” will be used to exemplify the activities that needed to be conducted. This service was owned and managed by the service domain “Quoting” within Dash Inc. and derived from the capability “Manage quotes”.

The “Quote service” was classified as a business-unit service within the group service “Sales”. The organisational objectives addressed in the business strategy were aligned with the specification of the business services. The owner of the sales service was the head of the sales department. Based on the capability that had been allocated to this service, the underlying business processes were annotated. As mentioned before, Dash Inc. had established a process classification framework previously that was not only used to identify the capabilities of Dash Inc. but also to identify the service’s underlying processes. The “Quote service” had four underlying processes as shown in Figure 4-22. A preliminary contract for the business service had not been specified at that time. That was the reason why the project team decided to specify the underlying elements first and subsequently derive details for the service contract.

The “Quote service” exposed one operation, namely “Quote processing”. The operation model for this service operation utilised all four processes and specified which business service had to be subsequently invoked.\(^\text{422}\) If a query was received from a customer, it would be entered into the system, before the technical feasibility was clarified. A feasible

\(^{422}\) The detailed operation model can be found in Appendix F.
query would be analysed regarding its relation towards a standardised product, a variant of a product or a fully customised product. Based on this differentiation, a quote was created and transmitted to the customer. If the customer accepted the quote, the business service “Order processing” would be invoked.

**Figure 4-22**: Detailing the quote service

The interaction model was the next element of the business service that had to be specified. This model, which was related to the operation model, detailed the interaction between the service provider and consumer. The first task for the service consumer was to send a query with adequate information to the organisation. The query could be sent via email or could be received by phone. Based on the SOA strategy, the project team had already considered to offer this service via the Internet to provide another communication channel between the customer and the organisation. If a query was received, the organisation would analyse the query for its feasibility and notify the customer in case the query was not feasible. If it was feasible, a quote would be created. The second task of the service consumer was to accept or decline the quote. Alternatively, the customer could also propose changes to the quote. Alternatively, the customer could also propose changes to the quote.

After the elements of the business service had been specified, the project team decided to postpone the detailed specification of the service until potential supporting software services were identified. The detailing of the business service was delegated to the top management and service owner, since they provided the specific knowledge about the details of the service, e.g. billing, provision, etc.
4.4.2.4 The prioritisation phase

At this stage, the project team had identified business services on different levels interacting with each other. Impressed with the work done so far, the top management decided to conduct a value analysis since they envisioned a lean organisation in the long term and wanted to know which business services were the most valuable for Dash Inc. and which could be subject for outsourcing decisions.

![The financial AHP model for Dash Inc.](image)

Since the business services had just been identified, the project team decided to conduct a value analysis based on the service domains and their underlying capabilities utilising the Analytical Hierarchy Process (AHP). Thus, different interviews were conducted with the top management of the organisation in order to find out what the financial and non-financial measures for a pairwise comparison of the different capabilities could be. Financial reports were also taken into account to identify which measures had already been calculated in the last fiscal year. The operating profit, sales growth and return on capital employed were chosen as financial measures. Customer satisfaction, market share and new product introduction were chosen as non-financial measures. Two AHP model were developed to derive the relative importance of the financial and non-financial importance of the distinct business capabilities. (The financial performance AHP model is shown in A pairwise comparison between the measures was conducted to identify the priorities of the different measures. The outcome of the AHP is visualised in Figure 4-24.)
Based on the outcome of the AHP, the project team concluded that the service “Logistics” had a low financial performance and a low non-financial performance, whereas the sales service had the exact opposite characteristics. Based on this information, the top-management decided to conduct further analysis of the value contribution of the “Logistics” service, to investigate the possibility of outsourcing its functionalities. However, this was out of scope for the project team.

![Diagram showing Key service region and key service cell with services like Logistics, Manufacturing, Sales, Finance]

**Figure 4-24:** The outcome of the AHPs

### 4.4.3 The derivation of software services

#### 4.4.3.1 The preparation phase

Based on the analysis of the value contribution of the underlying business service capabilities, the project team selected the “Sales” business service and its underlying services to define the scope of the software service enablement. The SOA strategy specified that one of the business imperatives addressed the need to integrate the customers into the business processes. The domain and stakeholder analysis proved that the sales domain directly interacted with the customers. Thus, the focus laid on finding software services to support the business services in terms of customer integration and flexibility.
An application analysis was conducted by the project team to identify the existing applications and their characteristics. The application documentation that had been collected previously was used as an input for the conducted interviews with application owners to prove the validity of the gathered documents. Once, the validity had been proven, further analysis was conducted to identify the existing components and functionalities. Data flow and invocation calls were traced throughout the applications to identify interfaces and interdependencies. Two documents were established to record the characteristics of the applications on the one hand and the identified functionalities on the other hand.

Based on the processes of the business service, detailed process models were developed. Since the chosen business services comprised multiple finer-grained services, the project team split up into different teams and utilised the knowledge of business analysts to define the processes in an adequate granularity. (In the following, the already described “Quote service” will be used as the foundation to derive software services. Specifically the process “Create quote (variant)” will be used to exemplify the further analysis steps.).

After the scope, the applications and the processes had been specified, a process-to-application mapping was conducted to analyse the suitability of the process and application for service-enablement. Hereby, the applications that were associated with the “Sales” business service were annotated to the respective process models. Furthermore, the project team identified which process steps were executed manually, semi-automatically or fully automatically.

As the next step, the project team conducted a suitability analysis for the application and processes. The application related to the “Quote service” had been developed in-house. Thus, the source code was available, as well as the knowledge to customise the source code if required. Interfaces could have been developed if required. Thus, the project team decided that the application was suitable to be service enabled. The suitability analysis of the process showed that it was already supported by existing IT and thus could be service-enabled as well.

4.4.3.2 The identification phase

As a first step, the entity underlying the business service was analysed in terms of its suitability to be service enabled. The “Order” entity was identified to be relevant for the business service by analysing the process model. Based on the targeted reusability considerations as described in Chapter 3.3.7 the “Order” entity service provided the CRUD-operations since these operations were most often to be utilised by other software services in different scenarios.
As the next step, the process were analysed further in terms of the interaction and visibility potential for stakeholders. Hereby the interaction model was used as an input to identify the process steps that could be executed by the service consumer and to analyse which results of which process steps should be made visible to the service consumer.

The process “Create quote (variant)” is visualised in Figure 4-25.\textsuperscript{423}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure4-25.png}
\caption{Analysing the “Create quote (variant)” process}
\end{figure}

\textsuperscript{423} For the visualisation of the process, the EPC notation has been used. Refer to Appendix F for an explanation of the different graphical elements.
The process would start, if either the product or variant were specified. The data necessary for creating a quote would be entered into the quotation system. Subsequently, two automatic activities were executed in parallel. On the one hand, the price for the product was calculated; on the other hand, the delivery date was determined. Afterwards, both results would be verified and modified, if required. As the last step, the quote was copied to a local network folder that was accessible by the top-management for controlling purposes.

The customer was allowed to enter his or her own quote data into the system. However, since Dash Inc. had to ensure that the data provided was accurate and detailed enough, the input data had to comply with the specification for dashboards provided by Dash Inc. Customers were allowed to calculate the delivery dates and prices independently of the availability of any account manager or sales representative. Furthermore, customers had the possibility to gain insights into the details of their own quotes.

Once the process had been analysed based on takeover and visibility potential, the software services were identified.

Step 1: Identify service operation candidates

Since all process steps within the process were supported or executed by IT, all steps were regarded as potential service operations.

Step 2: Abstract process logic

The project team identified the parallel split after the process step “Enter quote data” and documented this issue for further analysis regarding the composition of services.

Step 3: Define logical contexts

The underlying entity of this process had been identified as being the “quote” entity. Since all steps focussed on the manipulation of this entity, the first draft of the logical boundary of the service, based on regarding the service cohesion principle, encompassed all process steps.

Step 4: Refinement

Based on the principle of reusability, the project team decided to model the operations “Calculate price” and “Calculate delivery date” as two separate services. This way, both services could be utilised independently without invoking the complete entity service. Furthermore, both services were related to different underlying documents. For example, the “Calculate price” service was regarded as a task service that utilised different documents about prices based on the specific customer. Since the functionality of the entity service did not change if the customer changed, the operations “Calculate price” and “Calculate delivery date” were modelled as separate services.
delivery date” were outsourced to two separate task services. The “Calculate price” operation was grouped together with the “Modify price” operation to form the “Price” task service. Similarly, the “Calculate delivery date” operation and the “Modify delivery date” were comprised by the “Delivery date” service. The “Copy quote” operation comprised purely business-agnostic logic. Hence, the project team classified this operation as a separate utility service.

Step 5: Define compositions

The project team focused on a close business and IT alignment. Thus, the process was represented by one process service that composed the entity and utility services as well as the task services. Furthermore, the process service invoked the operations of the composed services based on the process flow. The interaction and takeover analysis of the process steps identified that the operations “Enter quote data”, “Calculate price” and “Calculate delivery date” were also executable by the customer. The project team decided to encapsulate these operations in a second process service that could be utilised by customers independently of any sales representatives or account managers. Thus, six services were identified as visualised in Figure 4-26.

![Figure 4-26: The identified services](image)

4.4.3.3 The detailing phase

After the services had been identified, the project team verified that the services did not already exist. Since the project team split up into different groups to identify services more efficiently, one team had already fed their identified services in the service inventory. The entity service had already been specified and could be reused. As the principle of targeted reusability had been taken into account, the needed operations were already defined. The other services were new and had to be detailed further.

To identify further reusability potential, the different operations were detailed regarding their input and output parameters. The project team decided that the utility service “Copy quote” should be made more reusable by extending the allowed parameters. Thus, the ser-
vice should not only copy quotes, but different data types. The following two tables were created (see Table 4-5 and Table 4-6).\footnote{424}

<table>
<thead>
<tr>
<th>Atomic services</th>
<th>Operation</th>
<th>Input Parameter</th>
<th>Output Parameter</th>
<th>Service Consumer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quote (Entity)</td>
<td>create()</td>
<td>quote data [payment and delivery conditions]</td>
<td>quoteID</td>
<td>CU (customer)</td>
</tr>
<tr>
<td></td>
<td>update()</td>
<td>quote data [payment and delivery conditions (delta)]</td>
<td>notification</td>
<td></td>
</tr>
<tr>
<td></td>
<td>read()</td>
<td>quoteID</td>
<td>quote data</td>
<td>CU</td>
</tr>
<tr>
<td></td>
<td>delete()</td>
<td>quoteID</td>
<td>notification</td>
<td></td>
</tr>
<tr>
<td>Price (Task)</td>
<td>calculatePrice()</td>
<td>materialID, values</td>
<td>price</td>
<td>CU</td>
</tr>
<tr>
<td></td>
<td>modifyPrice()</td>
<td>quoteID, new Price</td>
<td>notification</td>
<td></td>
</tr>
<tr>
<td>Delivery date</td>
<td>calculateDeliveryDate()</td>
<td>quoteID, values</td>
<td>delivery date</td>
<td>CU</td>
</tr>
<tr>
<td>(Task)</td>
<td>modifyDeliveryDate()</td>
<td>quoteID, new delivery date</td>
<td>notification</td>
<td></td>
</tr>
<tr>
<td>Copy (Utility)</td>
<td>copy()</td>
<td>data</td>
<td>notification</td>
<td></td>
</tr>
</tbody>
</table>

Table 4-5: Detailing of task, entity and utility services

<table>
<thead>
<tr>
<th>Process Service</th>
<th>Service Consumer</th>
<th>Function</th>
<th>Service</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter quote</td>
<td></td>
<td>enter quote data</td>
<td>quote</td>
<td>create()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>calculate price</td>
<td>price</td>
<td>calculatePrice()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>calculate delivery date</td>
<td>delivery date</td>
<td>calculateDeliveryDate()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>modify price, delivery date</td>
<td>price</td>
<td>modifyPrice()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>modify delivery date</td>
<td>delivery date</td>
<td>modifyDeliveryDate()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>copy</td>
<td>copy</td>
<td>copy()</td>
</tr>
<tr>
<td>Calculate quote</td>
<td>CU</td>
<td>calculate price</td>
<td>price</td>
<td>calculatePrice()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>calculate delivery date</td>
<td>delivery date</td>
<td>calculateDeliveryDate()</td>
</tr>
<tr>
<td></td>
<td></td>
<td>enter quote data</td>
<td>quote</td>
<td>create()</td>
</tr>
</tbody>
</table>

Table 4-6: Detailing of process services

After the operations had been detailed, the application functionalities were mapped on the identified service operations. By utilising the list of the application’s functionalities and components, the project team concluded that no changes to the application functionality had to be made others than providing an interface for the potential services. However, this was not in focus of the project team.

\footnote{424} Only the external stakeholders have been included as service consumers in the tables. The services themselves are consumed by the respective service domain and internal stakeholders of the organisation as well.
Finally, all the newly identified and detailed services were fed into the service inventory to provide other teams and projects with the possibility to reuse already identified services. As visualised in Figure 4-27, the first draft of a SOA for Dash Inc. was established.

Internal service domains were identified as well as external stakeholders. These actors utilised the business services that had been identified as well. The business services were owned and managed by their respective service domains. The processes underlying each business service were used to identify process services that provided IT-support for certain operations that were exposed by their related business service. The process services in turn composed finer-grained task, entity and utility services. Their functionalities were mapped on the IT systems to identify reengineering requirements.

**Figure 4-27:** Output of the consolidated approach

The top-management of Dash Inc. was more than satisfied with the work that had been done by the project team. Further projects were founded to analyse the first draft of the SOA to analyse the benefits for Dash Inc. in more detail.
5 Conclusion

5.1 Research implications

Globalisation, changing customer demand and dynamic market conditions are the main reasons why organisations nowadays are focussing on their core competencies. These highly specialised organisations are typically involved in several business networks to collaborate with other participants in order to fulfil customer demand. To cope with the dynamic environmental conditions and to expose standardised endpoints that can be used to leverage the offered operations or functionalities of a partner within the network, the concept of business services has been introduced. To support such an on-demand business, organisations have to leverage their existing application systems to the fullest. Hence, software service should be derived to support the business services, foster a close business and IT alignment and provide an organisation with the possibility to compose services based on the business needs.

Although especially software services have been focussed by academia and industry, extant approaches that have been published to identify and analyse services only cover certain aspects. Several authors state that a consolidated approach is needed to fill this gap, but instead of proposing such an approach, they add to the proliferation of service analysis approaches by proposing their own method.

This thesis provides a significant contribution to academia and industry. On the one hand, it provides a comprehensive analysis of the concepts behind 30 extant approaches. These approaches have been analysed regarding their strength and weaknesses in order to propose a structured, consolidated approach that combines the relative strengths as well as to propose enhancements and extensions based on the identified shortcomings of the extant approaches. This approach attempts to fill the research gap that has been identified by various scholars, who point out the need for a consolidated service analysis approach. Furthermore, this thesis shows how software services can be used to support business services to achieve a close business and IT alignment. On the other hand, this approach will provide an organisation with a methodology to not only understand and document its existing capability from a service perspective (e.g. how these different services are interrelated, across both the functional and technical boundaries), but more importantly, to identify potential new services that may be provided (on the business and technical levels of an organisation).
5.2 Research limitations

The development of the consolidated approach is based on the analysis of 30 published approaches related to the area of service identification and analysis. Hence, a larger amount of analysed approaches might have lead to the development of a consolidated approach that focussed on different issues. Since none of the analysed approaches explicitly describes how to derive business services, approaches had to be adapted that could be used to derive business services. If the scope of the analysed approaches had been extended to analyse approaches for the specification of business models and enterprise architectures, other ways to define business service could had been discovered.

Based on the time constraints of this thesis, an evaluation of this approach could not be conducted. The consolidated approach combines approaches that have successfully been applied in practice, but no proposition can be given if the consolidated approach is applicable for all organisations in the private and public sector. Hence, case studies need to be conducted that could lead to refinements of certain aspects of the consolidated approach. Different case studies within different organisations will provide valuable insights for the validity of the approach.

The design principles and the service classification framework are based on a literature review including the set of analysed approaches. However, due to the lack of existing case studies, different design principles and different classes with the classification scheme for services that might be more relevant in practice might be discovered. The design principles and the classes for services should be focused on as part of the case study for the validation of the consolidated approach.

5.3 Further research potential

As pointed out at multiple points within this work, service analysis touches different topics that can serve as the basis for further research. First of all, a consolidated lifecycle for a service and for a SOA is missing. This lifecycle should clearly identify the roles and the responsibilities within an organisation that relate to service-orientation. Additionally, the activities that are addressed by each phase within the lifecycle should be described.

A modelling language for the analysis and design of services is missing. No consensus could be found on how to model services or how process models could be annotated and extended to show which services are utilised by a process. A service analysis and design language will lead to comparability of different service designs as well as for the derivation of computer supported modelling of services.
Since service-orientation affects the whole organisation, further research can be conducted to address the organisational and cultural changes required for the implementation of the service concept. The business model or enterprise architecture of an organisation has to be aligned as well. This thesis provides first insights into this topic.

Further research in the area of value net analysis should also be conducted to identify areas within the business network that can be used to deliver further value for an organisation and its partners. Hereby, new services could be developed that fill out the identified gaps. Services could also be further analysed based on consumer requirements, such as expectation and perceived quality. These topics have been focussed by traditional service marking literature, but are equally applicable on software services.

Case studies should be conducted to provide empirical evidence supporting the proposed benefits of service-orientation. Since SOA is a recent paradigm, empirical evidence can hardly be found since they should compare the state of an organisation before the implementation of a SOA and after the implementation. Especially, the paradox identified by Tallon and Kraemer should be used as a starting point, to investigate if service-orientation can overcome the potential downsides of tight business and IT alignment related to the organisational agility.425

Furthermore, the metrics proposed by Heck can be used as a starting point to measure the coupling and cohesion of the services to identify optimal clusters of services.426 Clustering of services has also been proposed by Kohlmann and Alt, but they fail to go into details.427 However, the “coupling-distance” between each pair of service should be calculated and clustering algorithms should be applied to derive these service clusters. The optimal clusters can then be compared to the clustering that has been derived by allocating services to service domains. If there is a large discrepancy between the optimal and the imposed clusters, one can hypothesise that this discrepancy may lead to degraded levels of agility, since services are primarily utilised by domains and consumers that are not the service owners.

Bibliography


Appendix

A Technical perspectives on SOA

The technical perspective on SOA is prevalent in existing literature. SCHEMM ET AL. define three different types of SOA perspectives that will be explained in the following.428

SOA for process-oriented composition of information systems: The application landscape is divided into small, independent parts that expose their functionalities as services to other applications. However, these services are derived based on functional, process-oriented considerations. Therefore, services should be combined to provide flexible information systems that can easily adapt to changing business requirements. This is supposed to result in an increased agility of the organisation because the IT can more easily be aligned with business processes. Information systems can therefore be configured in design-time as well as in runtime combining different services based on the specific situation.429

SOA as a middleware approach: A middleware provides a software layer that comprises interfaces and protocols to support a transparent communication between distributed applications.430 A middleware can be used to hide the complexity and implementation details of the underlying systems and thus facilitates the development of distributed applications partly using existing functionalities. For the integration of a heterogenous application landscape within an organisation, comprehensive approaches subsumed under the term Enterprise Application Integration (EAI). For the integration of applications crossing the boundaries of an organisation, middleware concepts such as Business-to-Business (B2B) are applicable that provide a uniform business vocabulary and protocols. For the implementation of a SOA a middleware concept is needed that does not need to comprise more functionalities and mechanisms than are available in existing middleware concepts (e.g. EAI products).431 Thus, the middleware concept has been introduced in the SOA paradigm.432

SOA as an architectural paradigm: Hereby, a SOA is seen as a horizontal layered architecture that encapsulates functionalities of applications within a service layer. This layer is accessible by the process layer that can execute the underlying services to achieve their business task. User interaction is typically handled by web-based portals. Additionally, services can be clustered vertically based on functional relationships (e.g. customer ad-

ministration). Thereby, services are used to communicate across different domains. The vertical and horizontal layering of services is used to decrease the complexity of existing applications.433

The allocation of a set of perspectives of SOA as (implicitly) stated by some authors is visualised in Figure 5-1.

Figure 5-1: SOA paradigms

The Strategic Alignment Model (SAM) shown in Figure 5-2 has been developed by Henderson and Venkatraman to outline the alignment of the different areas of an organisation.

This model identifies the need to specify two types of integration between the business and IT domain. The link between the business strategy and IT strategy deals with the capability of IT to shape and support the business strategy, whereas the operational integration deals with the internal domains and the integration of organisational infrastructure and the IT infrastructure. The model encompasses four different domains of strategic choices, namely the business strategy, IT strategy, organisational infrastructure and processes and IT infrastructure and processes. The constituent components of the domains are scope...
competencies and governance at the external level and skills and processes for the internal domain. Architectures are an additional component in the internal IT domain, whereas administrative infrastructure is part of the internal business domain.

However, strategic decisions should consider all four domains of an organisation. If an organisation focuses primarily on external issues (business and IT strategies without considering the internal domains), the difficulties of designing the underlying processes of the internal domains could be underestimated. Thus, the Strategic Alignment Model advocates cross-functional relationships that consider external and internal domains, as well as business and IT domains. The driver behind the alignment might be the IT strategy that influences the design of the internal IT processes that consequently influence the organisational structure of the business. This concept describes the multivariate cross-domain perspective that works on the premise that strategic alignment at an organisational level can only occur when three of the four domains are in alignment.\footnote{cp. Avison et al. (2004), pp. 230 ff.}
C Service analysis approaches

To enhance the overview and guidance through the different methods found in literature, a clustering approach has been conducted. The criteria that have been used to derive these clusters are made regarding the focus of the different approaches. Nonetheless, the scope of multiple methods varies widely. Thus, some methods solely fit into one cluster, while others span different clusters. If approaches focus equally on multiple ways to derive services, these approaches cannot be assigned to one specific cluster and will be described and analysed in a separate chapter. Two general clusters could be identified, based on the underlying concept of service-orientation, namely “business-driven approaches” and “IT-driven approaches”. These two clusters will be further split up in the course of this chapter. A summary and short analysis will be given for each approach.

C.a The business-driven approaches

All approaches that focus on the business domain of an organisation can be clustered together. The approaches may address the business’s processes, the business model or business requirements. However, some approaches cover the derivation of business services as well as software services based on their broad scope of service analysis.

Since the business-driven approach encompasses methods for analysing business services as well as software services, the methods can be grouped into multiple categories based on their respective starting point for service analysis. The scope of each category will be described at the beginning of each category to provide an overview and guideline through the chapter.

C.a.a Capability-driven approaches

Overview

Capability-driven approaches focus on the derivation of services based on the organisation’s business model. Capabilities can be seen as what the organisation does to deliver value to internal and external stakeholders.

Method according to OASIS\textsuperscript{436} and Jones, Morris\textsuperscript{437}

The Organization for Advancement of Structured Information Standards (OASIS) has published a draft version of its “SOA Blueprints” that builds upon the work of Jones and

\textsuperscript{436} Cp. OASIS (2005).
MORRIS.438 A blueprint is seen as a structure including patterns that can be used within a SOA planning, design and implementation strategy. The approach is quite comprehensive, but since it is a draft version, there are many gaps and missing content. However, the concept itself is conclusive: a service model is established that serves as the basis for applying certain blueprints. The blueprints are then converted to guide the technical implementation phase. The service model contains the functional domains of an organisation that serve as containers for services that can be delivered to internal and external stakeholders of the organisation. Each domain can be decomposed into finer-grained granularity levels to establish a hierarchy of domains and services (level 0…*). Each granularity level is developed by identifying the domains, its external stakeholders and the interactions amongst the domains and between domains and stakeholders. Thereby, three types of services may be distinguished (virtual services, support services and shared services). For each domain, one can derive its capabilities that are specified in terms of input and output parameters and a description that may be used to define business services for each domain. Once the service model is established for the organisation, one can assign certain blueprints, namely service pipeline patterns, service patterns and service types. Service pipeline patterns define the technical requirements to execute the services that have been assigned to this pattern. Service patterns define five distinct software services, whereas supporting types focus on the technical realisation (e.g. synchronous or asynchronous invocation, exception handling, routing, etc.). The last step described by the “SOA Blueprints” is the maintenance and governance of the services. However, this approach does not cover the application of design principles to deliver adequate services.

Method according to SEHMI, SCHWEGLER439

The authors propose a comprehensive method for service-enablement that can be applied to analyse business services as well as software services. Starting at developing a business model that is made of capabilities of an organisation, the authors describe how to transform this model into a service model and into a technology model consequently. All three models have to be described regarding the boundary, autonomy, contract and policy of the modelled elements. The business model consists of capabilities of the organisation itself and the capabilities of the environment that interact with the organisation to deliver value. These capabilities can be modelled on different granularity levels to derive hierarchies of business capabilities that can further be specified by defining multiple attributes for each capability. The types of connection between capabilities are modelled as well before the transformation into a service model can take place. Hereby, the service contract is speci-

fied (e.g. input and output parameters, (non-)functional requirements, etc.). Thus, the capabilities serve as inputs for deriving business services. They are defined independent from any technology and implementation issues. However, the service model is used to identify the entities, messages and interfaces of the business capabilities. Thus, one has to define which data is passed between capabilities using messages and the interfaces that are exposed by capabilities. Furthermore, one has to define non-functional requirements of each service and the composition of services to support a capability. Finally, services that can be implemented or supported by IT are transformed into the technology model. This model identifies how each service will be implemented, hosted and deployed. This approach can serve as a guideline to derive business services and software services, but it lacks a detailed description of how to use the different models. No design principles are proposed and the description of different service types is missing as well.\(^{440}\)

\textit{Method according to Flaxer, Nigam}\(^{441}\)

These authors propose an approach that can be used to analyse and structure the business based on components, operations and business services that are supported by software services. Hereby, each component serves a distinct purpose and may collaborate with other components to fulfil a certain goal or objective. The aim is to find the groupings of cohesive business activities that together can be optimised as a unit. A business component, which can be seen as a business in its own rights, encompasses a certain set of business services that it offers to other components or to external parties. Hereby, different business operations may be identified that describe what the business actually does including activities, resources, people, technology and artefacts. The authors identify artefacts as a certain chunk of business information (e.g. documents, messages, etc.). Based on this idea they propose the “Artifact Method” that can be used to identify services and components. Hereby, the aim is to model what the business actually does as a network of business tasked that can be assigned to components. As the first step, the artefact has to be identified that is processed by a certain component. Next, all business tasks associated with this artefact have to be listed to establish the life cycle of the artefact. During the next step all other artefacts that are either consulted or modified during the processing of the original artefact by each of the listed business tasked have to be analysed. Hereby, one has to decide if these additional artefacts are maintained by the component at hand or some other component. The last step involves the repetition of the first three steps for all artefacts that the component is responsible for. If all artefacts have been identified and the responsible com-

\(^{440}\) Since this approach is based on the Microsoft Motion Methodology, the details of this approach are not openly available.

ponents are associated, the method ends. There are no explicit design principles and the authors only give ideas but do not go into details of how to apply these.

**Method according to IBM**\(^{442}\)

IBM proposes a component business model that is used to model the business structure based on the idea of internal and external specialisation. Hereby, certain components have to be identified that are based on the core competencies of an organisation. Components encompass five dimensions. The business purpose defines the logical reason for the existence of that specific component by defining the value it delivers to other components. Activities describe how the components actually achieve its purpose. Resources (e.g. people, knowledge, assets etc.) are modelled as well, supporting the activities. The governance model specifies what entity manages the specific component. Furthermore, business services that are provided or received by the component need to be annotated as well. The components are specified by applying the loose coupling and high cohesion principles. The components themselves should be as loosely coupled as possible, meaning that the service boundaries are clearly defined and connections to other components only last as long as needed. Cohesion refers to grouping together related activities, so that each activity solely belongs to one component. Hereby, two specific design principles are mentioned. However, how exactly components are identified is not shown.

**C.a.b Process-driven approaches**

**Overview**

Process-driven approaches focus mainly on the derivation of services based on process models. Additionally, two approaches are presented that examine the suitability of processes to be service-enabled.

**Method according to ERL**\(^{443}\)

The author gives a comprehensive approach that covers all phases of his proposed SOA-lifecycle.\(^{444}\) He focuses mainly on the derivation on software services. The starting point is the service analysis phase is a business requirements analysis and an existing systems analysis, followed by the service modelling phase, in which service candidates are derived by decomposing process models and applying service design principles. Service candidates

\(^{442}\) Cp. IBM (2005).


\(^{444}\) ERL proposes six phases, namely service-oriented analysis, service-oriented design, service development, service testing, service deployment and service administration. Erl (2005), pp. 358 ff.
are seen as a first proposal that becomes more concrete during the next steps of the life cycle. He describes a process consisting of twelve steps of how to derive different types of services based on process models. He proposes eleven types of services that are partly hierarchical and partly orthogonal to each other. In addition, he proposes three different layers on which different services can reside.

\textit{Method according to Ivanov, Stähler}\textsuperscript{445}

Ivanov and Stähler argue that a simple transformation of process elements leads to the definition of software services. They differentiate business process models into three different granularity levels, based on the processed business entity. On the highest level of abstraction, a process interacts with more than one business entity. In this case, the business process model is made up of sub-processes. On the medium level of abstraction, a process only handles one specific business entity. Therefore, the process model has the granularity of process steps. If furthermore the atomic activities are described in a process model, the lowest level of granularity is reached. Service identification starts on the highest level of abstraction, encapsulating sub-processes. Operations inside a service are derived from the process steps (medium level of granularity), whereas specific functionalities inside an operation are developed based on the atomic elements of a process. They do not differentiate between different service types or layers and do not apply design principles.

\textit{Method according to Klose, Knackstedt, Beverungen}\textsuperscript{446}

These authors propose three general phases for the identification of software services based on process models. The \textit{preparation phase} comprises activities that are performed to define the scope of the analysis, prepare the process models (model IT support for each function, attach organisational units and define granularity of process models) and define relevant internal and external stakeholders. The service \textit{analysis phase} comprises the service identification based on process models and a feasibility analysis of the underlying IT to support the execution of the service. Service identification comprises the analysis of the visibility and takeover potential of process function based on the stakeholder identification. The service feasibility is analysed by applying four design principles. Based on the analysis of visibility and takeover of functions and the analysis of service feasibility, the third phase, namely \textit{service categorisation}, focuses the IT point of view in a bottom-up approach. Thereby, services are classified as elemental services or composed services based on their internal composition. Elemental services cannot be decomposed into finer-grained

\textsuperscript{446} Cp. Klose, Knackstedt, Beverungen (2007).
services since they comprise atomic operations. Additionally, the authors distinguish between two types of elemental services based on their provided operations.

*Method according to Gold-Bernstein, Ruh*[^447]

These authors propose an event-driven service design. Thereby, certain events trigger certain actions that can be executed by services. Since events can be visualised using process models, this approach can be classified as a process-driven approach. According to the proposed approach, the scope of the SOA initiative has to be defined before in addition to a conducted stakeholder analysis. Four steps have to be executed to derive a proper set of services. The first step focuses on the development of a table that describes the response of the system to an event. The identified stakeholders are the entities that trigger these events. Since software services are focused on by this method, all events that trigger a manual activity have to be excluded. As a second step, one has to define service categories based on the responses of the systems that have been identified in the first step. However, no definition of a service category is given as well as a definition of an adequate granularity. The objective of the categories is to identify which functions are supported by the IT and which functions have to be newly developed. The third step encompasses steps to describe the categories in more detail, especially the technical implementation. The last step describes the interface of the service. Hereby, the input and output parameters, the operations and the concrete implementation details are specified. Additionally, one can describe non-functional requirements of a service by analysing the usage of that service by its consumers. The authors mention several characteristics of a service, but fail to go into details. However, specific service types or layers are not part of this method.

*Method according to SAP*[^448]

Based on a solution map that describes business scenarios in one industry on different hierarchical levels, one can identify business processes that can be used to derive so called “Enterprise Services”. SAP identified four objectives or business drivers for the service design: Improving user productivity, Support for new business practices, increasing efficiency of business processes and flexible deployment of a new application. The process of designing these services can be decomposed into three steps. The first step is called “discovery”. Hereby, based on business requirements different scenarios are analysed, which might benefit from service-enablement. After a promising business process has been identified, the design step may begin. This step encompasses activities the identification of the service(s) that must be designed to support the process. The last step, documentation, com-

prises activities that involve a proper integration of the service(s) in the service repository of an organisation. For each business driver, SAP defined design contexts or patterns that have been proven to enhance an application by adding services. Each design context has related indicators. These indicators are features of a process that can be used to identify if a process might benefit from a solution based on the design context. Thus, dependent on the business driver, one has to analyse the indicators for a given process in order to apply a certain design pattern. Indicators also support the analysis of the underlying information systems. Enterprise services can be classified in four service types, but no layers are proposed. However, design guidelines are given to support the right granularity of services.

*Method according to QUARTAL, DIJKMAN, SINDEREN*449

These authors propose four phases to derive software services based on process models. The first phase encompasses the analysis of processes to determine which processes are supported by the existing application portfolio. The second phase comprises an application analysis to specify which functionalities should be encapsulated in a service. Hereby, the process is decomposed to identify fully automated functions that can be performed solely by the information system and (semi-) automated functions that require communication with other systems or human interventions. Services are defined that specify the relationship between the service and the service’s environment (only semi-automated functions are used as the basis for service identification). The third step comprises a more detailed definition of these services and their underlying finer-grained services that comprise components of the system. The last step focuses on the implementation of the service and its underlying services. For modelling the process and the related services, the authors propose the Interaction Systems Design Language (ISDL). However, the proprietary language for modelling processes and services likewise can be seen problematic since processes may have other requirements that have to be modelled than services. Any service characteristics or service design principles are not described by the authors.

*Method according to KOHLMANN, ALT*450

These authors propose a service analysis methodology based on process model decomposition. They describe a service identification technique and a service clustering technique that are divided into four phases, namely a preparation phase, an analysis phase, a verification phase and a detailing phase. Starting with the *service identification* technique, the preparation phase comprises the gathering of all relevant models that are needed to identify services. These models include different granularity levels of business processes and a

network model that describes the roles and domains inside an organisation. The analysis phase identifies service in a top-down manner using process decomposition. Four design principles and service criteria are applied to identify a certain service. The verification phase encompasses the activities of analysing if the identified service already exists as well as examining if a service fits the business needs. The detailing phase specifies the service into more detail and allocates it to a specific service layer. Service clustering consists of the same phases, but they have a different focus. The preparation phase encompasses the gathering of the enterprise model and an interorganisational network model to identify the scope of service clustering. All services in the selected scope are the analysed concerning the cohesiveness, coupling and autonomy of the services to identify service clusters during the analysis phase. The verification phase takes the service clusters and identifies the areas of a process that are covered by the services. The detailing phase is the focussed on the differentiation between service consumer and provider and further details regarding the service clusters.

Method according to Papazoglou, van den Heuvel\textsuperscript{451}

These authors describe a service-oriented design and development methodology that covers all phases of their proposed service life-cycle consisting of one preparatory phase (planning) and eight distinct main phases, namely planning, analysis and design, construction and testing, provisioning, deployment and execution and monitoring. Thus, the focus of this approach lays on services derived from business processes. The planning phase focuses on feasibility, goals, rules and procedures of the SOA initiative to gather business requirements. The analysis phase focuses more on the identification of multiple alternatives for implementing business processes (e.g. wrapping systems). The phase is followed by the design phase that identifies and specifies services and business processes. The service construction phase comprises activities related to the realisation and testing of services and processes based on the specification that were made during the design phase. Service provisioning focuses on service metering, billing and rating to establish a provisioning model for all services that have been identified. The last phases deal with the execution and monitoring of the services that are concerned with the binding and invocation of the services. The authors apply two design principles, namely cohesion and coupling. In addition, they also give guidelines regarding the granularity of services.

Method according to Erradi, Kulkarni, Maheshwari\textsuperscript{452}

The authors propose an approach for identifying services, designing them and deciding the service granularity and layering. Thereby, they apply a certain set of design principles. For the service identification, a hybrid approach combining a top-down domain decomposition and a bottom-up application portfolio analysis is advocated. The domain decomposition starts by capturing high-level business processes as well as a mapping of the business processes to the applications. The processes are used to identify potential services that are realised by existing application functionality. Activities within a process are mapped to the functionality of the applications. Herby, possible redundancies and overlaps in the application portfolio are identified, in addition to functionality that can be shared between multiple processes. Furthermore, the application portfolio itself is analysed by a combination of tools to derive fine-grained functional modules that have to be clustered afterwards to derive reasonably coarse level activities that may be aligned with the services identifies in a top-down manner in the previous step. The authors propose to design one service that fulfils one business task. Furthermore, they argue that a process is modelled to achieve a specific business task as well and hence conclude that one service should represent one process. However, this service can compose finer-grained services, such as infrastructure services (e.g. authentication) or provisional services (e.g. billing). In addition, the external behaviour of the service has to be specified as well as the service consumers and service providers. Once the services are identified, they have to be classified according to their scope, since the non-functional requirements of different service classes might be different. The authors also refer to the service granularity and service design principles to derive services based on business needs. However, the authors do not describe a structured way of deriving services; they describe what should be done, but not how it should be done. Nonetheless, they include service design principles as well as a service classification scheme.

Method according to Chang, Kim\textsuperscript{453}

The approach proposed by these authors is focussed on a systematic way for service-analysis and design. The authors argue that service-orientation has to extend object-orientation as well as component-orientation. Thus, they compare all three approaches to derive a process consisting of five phases that takes service requirements as inputs and delivers service specifications, compositions and verified service components as deliverables. They propose five layers of a SOA, namely a business process layer, an unit service layer, a service interface layer and a service component layer. Hereby, unit services are seen as software services that are utilised by activities within a process. Since unit services

may be reused by more than one activity, there exist one-to-many relationships between activities and unit services. Once the services are identified based on business processes one has to specify the functionality, the interface, pre- and post-conditions and further constraints to derive the lower layers of the proposed model. The first phase, identifying business processes, is subdivided into three steps that comprise the identification of service requirements, potential services based on the requirements and the derivation of the business process for each potential service. The second phase, defining unit services, encompasses the analysis of the business process to identify commonalities and to derive unit services. Hereby, a certain set of design principles and guidelines are applied. After the identification, a service interface has to be defined that includes one or more operations. Further details, such as the input and output parameters have to be specified as well. The next phase, discovering services, aims at finding already available services internal or external to the organisation that are conform to the specification of the unit services. If existing service could not be found, the next step develops the services based on the specification. This can be achieved by either wrapping existing legacy functionality into services or by developing services from scratch. Afterwards the services and interfaces are registered in a service repository. The last step, composing services, aims at relating certain unit services to fulfil a certain business process task.

C.a.c Use case-driven approaches

Overview

Use case-driven approaches focus on service derivation on use case models. Use cases focus on the interaction between the interaction between the users and the related system.

Method according to Kim, Doh454

The authors propose a use case-driven approach to derive software services. Use case analysis is used to identify cohesive functionalities within the boundary of a system without specifying the specific implementation. Hereby, task trees are generated from use cases that are refactored afterwards to discover common behaviour in use case models. Refactoring has been applied to enhance the structure of source code455 and has been extended and applied on other models.456 The main goal is to eliminate redundancies, identify commonalities, reuse functionalities and select the right level of granularity and abstraction.457 Thus,
five refactoring rules, that can be used to extract service candidates from task trees.\textsuperscript{458} The \textit{decomposition refactoring} rule is applied when a task tree can be decomposed into smaller trees to decrease the complexity. The smaller task trees can be used to model a new use case. The \textit{equivalence refactoring} rule is applied when two use cases have the same task tree. Thus, they can be treated as one task tree. \textit{Generalization refactoring} is used when use cases share common primitive tasks that can be treated as a subset of a more general use case. \textit{Merge refactoring} is used when one task tree is too specific compared to another one, so that both task trees should be merged. \textit{Delete Refactoring} is applied when a task tree is defined, but not related to any other use case or actors. Thus, it can be deleted from the further service analysis. Design principles are implicitly applied by the refactoring rules. However, the authors do not define service types or layers.

\textit{Method according to STOJANOVIC, DAHANAYAKE, SOL} \textsuperscript{459}

The authors propose an approach for modelling and designing service-oriented architectures based on the concept of a service component and standard UML modelling constructs (use cases). Thereby, they provide a model-driven design approach to allow traceability between business requirements and software artefacts. The authors focus on the design of software services that are provided by a service component that represent encapsulated, autonomous software functionality. They describe the characteristics of a service component as well as the specification. However, they classify a service component into two classes, namely a business service component and an application service component. The latter one can furthermore be divided into three different classes, based on its function. A business service component realises specific parts of a business process and composes application service components. They can be identified by conducting a domain analysis as well as use case analysis. They propose various requirements of use cases in order to identify the scope of business service components. However, they do not propose any design principles explicitly.

\textit{Method according to Sun} \textsuperscript{460}

Sun has proposed a comprehensive methodology to build a “pragmatic and flexible SOA”\textsuperscript{461} named SOA Repeatable Quality methodology. The methodology consists of five phases, namely inception, elaboration, construction, transition and conception. The methodology addresses different perspectives of internal stakeholders that are involved in de-

\textsuperscript{458} Each refactoring rule is described by its parameters, pre- and postconditions and its process. For further details, please refer to Kim, Doh (2007), pp. 113 ff.
\textsuperscript{460} Cp. Sun (2006).
\textsuperscript{461} Cp. Sun (2006).
veloping the SOA. Furthermore, the methodology provides a set of artefacts to identify SOA drivers and a framework that provides tools and artefacts to derive a SOA solution. Starting at use case decomposition, fine-grained services and consumers can be identified. These fine-grained services are then composed to coarse-grained services. However, since it is a proprietary approach specific details are missing, especially for service analysis.

C.a.d Other approaches

Overview

The approaches described and analysed in this chapter cannot be assigned to one of the identified clusters, since they either propose service analysis as a comprehensive approach that addresses more than one perspective or they describe service analysis based on a specific perspective that cannot be subordinated in one of the clusters.

Method according to Zimmermann, Krogdahl, Gee\textsuperscript{462}

The authors describe a Service-oriented Analysis and Design (SOAD) approach that leverages existing models and approaches, such as Business Process Management (BPM), Enterprise Architecture (EA) and Object-oriented Analysis and Design (OOAD). To be adaptable to SOA, these approaches need to be extended to cover aspects like service orchestration, repositories and an enterprise service bus. SOAD proposes a meet-in-the-middle approach that combines a service derivation based on processes in a top-down manner as well as in a bottom-up manner using existing systems. They propose three major levels within a SOA; operations as single logical units of work, services as logical groupings of the operations and finally business processes. The authors mention more aspects that need to be addressed by service enablement, for example service classification, but they do not give any further guidance. The focus lies on deriving services based on a class diagram, but no structured approach is presented to understand the service derivation.

Method according to Arsanjani\textsuperscript{463}

The author describes a service-oriented modelling and architecture method that consists of three phases, namely identification, specification and realization of services. The identification phase comprises a top-down approach by conducting a domain analysis, a bottom-up approach by analysing existing systems as well as a goal analysis to align business requirements with the SOA initiative. Thus, a distinct allocation to one cluster cannot be conducted. The specification phase comprises the classification of services and the identi-

fication of components. The flow between services is analysed as well to understand the interactions between services. The realisation phase comprises activities that are focussed on the mapping between services and components. The method proposes starting points for service analysis, but no details are given how to perform the certain tasks. Design principles cannot be identified nor an explicit service classification scheme.

Method according to Kaabi, Souveyet, Rolland

The authors propose a method for service elicitation, distribution and orchestration in a declarative goal-driven manner. They focus on the collaboration of different organisations that expose certain services that composed fulfil a certain task. The need to understand the requirements of the organisations for a composite service and the modelling of the underlying cooperative process supporting the service provision is focussed by this approach. The basis is laid by a so called goal model or map, that visualises the intentions and strategies and the connections between them. An intention can be seen as a goal to be achieved by the performance of the process. A strategy is an approach to achieve an intention. Once a map has been established, the authors propose five steps to identify business services and software services. During the first phase, identifying services from the map, the authors propose to map each strategy modelled in the map to a business service that is supported by one software service. The second phase identifies the key actors that act as providers for the different services. The next phase aims at identifying parts of the legacy applications of the service providers that can be used to identify finer-grained software services that support a certain service identified in the previous steps. This can be done by highlighting the communication between actors involved in the service by using a sequence diagram. The authors propose different rules for identifying these finer-grained software services within the legacy system. The outcome of the application of the rules to the sequence diagram is a set of finer-grained software services that can be assigned to specific legacy systems of the actors. The fourth phase is concerned with the distribution of services (and their embedded finer-grained services) among the key actors. The authors propose to all services under the control of one provider should be gathered into a single composite service. Hence, each provider is assigned one composite service and its finer-grained software services. The last step focuses on the orchestration of the different services. Orchestration of services means to model the dependencies between the different services based on the underlying cooperative process. The described approach is comprehensive, but very complex. The authors do not apply any design principles, but identify business services, stakeholders and underlying software services.

Method according to WINKLER\textsuperscript{465}

WINKLER identifies services based on communalities of activities modelled in activity diagrams. Granularity levels of activity diagrams are developed by decomposing each single activity into finer-grained activities. Once the activity is decomposed into its atomic elements, the identification of commonalities between different activities is enforced. Sequential activities that are dependant on each other as well as activities occurring more than once on different levels can be encapsulated into different services. The author proposes four phases for service analysis and additional four phases for service design. Service layers are related to the granularity level of the activity diagrams.

Method according to MARKS, BELL\textsuperscript{466}

The authors propose three phases for identifying business services, namely SOA value modelling, business service identification and service modelling. The first phase aims in setting the scope for the SOA initiative by conducting a domain analysis. The second phase proposes a iterative cycle of identifying service candidates in a top-down manner as well as from a bottom-up perspective. The propose six specific ways for identification, namely business process analysis, core entity analysis, opportunistically via budgeted initiatives, business or domain expertise, pre-existing services and existing business applications. Thus, this approach cannot be assigned to one specific cluster, since the proposed ways to identify services cover a broad spectrum of alternatives. However, one may identify business services that may have potential value for the organisation using the proposed ways of identification. These service last phase aims at modelling the services that are actually implemented based on a design and analysis of service candidates, a reuse analysis, granularity analysis and horizontally and vertical business service analysis. They propose six ways to enhance the service granularity, before they analyse the actual implementation issues of the derived services. However, they do not classify services or introduce service layers.

Method according to Allen\textsuperscript{467}

The Service-oriented Process is part of the CBDi-SAE SOA reference framework. The process consists of four main areas, namely manage, consume, provide and enable. Each part can be broken down into different disciplines and even further into process units and sub-process units. The process follows a meet-in-the-middle strategy since business requirements are regarded as well as an analysis of legacy systems. However, since the process not only aims at service analysis, it is not very detailed about how to execute different

\textsuperscript{466} Cp. Marks, Bell (2006), pp. 99-149.
parts of the process. A service classification is missing as well as the description of any design principles.

Method according to Rahmani et al.\textsuperscript{468}

The authors propose an approach that uses the concept of the Model Driven Architecture (MDA) as proposed by the Object Management Group (OMG).\textsuperscript{469} The idea about MDA is to separate a Platform Independent Model (PIM) from the Platform Specific Model (PSM) and to perform transformations between these models via tools. The PIM is supposed to be independent from any platform, so that different PSMs for different platforms can be developed that are transformed into code in the last step. The proposed approach develops a PIM, before a PSM for a SOA (or a PIM for the next level) is generated. The next PSM is based on a specific platform or technology (e.g. web services). The PIM is developed using Unified Modeling Language (UML) models to model the system as the base for a SOA. The class diagram can then be transformed into a SOA model and finally into a PSM for the specific technology. The authors use a bottom-up approach, since the service derivation starts by modelling class diagrams of the actually system, which is transformed based on service-oriented concepts. However, design principles are not mentioned and neither are any classifications of the derived services.

C.b The IT-driven approaches

The IT-driven approach starts by analysing the application landscape and IT infrastructure of a company in order to derive services. These systems most often encapsulate important data records and business logic that cannot simply be replaced.\textsuperscript{470} Therefore, methods and approaches are described and analysed that focus on integrating legacy systems and their functionalities into a service environment while still meeting business requirements. Legacy modernization can mainly be classified into two areas, namely non-invasive legacy integration and invasive legacy transformation that will be described in the following chapters.\textsuperscript{471} Figure 5-3 provides an overview of the different modernisation approaches.

\textsuperscript{468} Cp. Rahmani et al. (2006).
\textsuperscript{469} Cp. OMG (2003).
\textsuperscript{470} Cp. Chen et al. (2005), p. 201.
Figure 5-3: Legacy modernisation approaches

C.b.a  Non-invasive service identification

*Overview*

The approaches described in this cluster focus on the identification of services without changing the legacy code of the existing systems.

*Wrapper of functionalities*

This approach aims at exposing intact functionality through non-invasive wrapping of legacy functionalities and defining proper interfaces for accessing these functionalities. The goal is to consolidate the legacy’s functionalities into the current and future applications.\(^\text{472}\) Thereby, the lifetime and reach of legacy systems can be increased since the wrapped functionalities can be used in the new environment (SOA). Interoperability might be enhanced as well since interfaces provide a way to hide the implementation details of the functionalities than can be accessed in a uniform manner. Since legacy code is not restructured or reengineered, fewer resources for the upfront analysis and design of the legacy system might result in reduction of integration cost.\(^\text{473}\)

One method for legacy integration is the development of wrappers. Wrapping legacy applications can be differentiated into three different approaches related to different layers of application architecture that are accessible. This kind of architecture is characterised by the division into the presentation layer, functional layer and data layer. Legacy integration may use any of these layers as a starting point for defining wrappers.

Wrapping of functionality on the presentation layer is even applicable if a clear separation of the different layers cannot be guaranteed, e.g. the business logic and the presentation layer cannot be clearly separated or if the application logic cannot be wrapped. Thus, one may consider wrapping or developing the Graphical User Interface (GUI) to derive a service for the specific legacy application. The underlying logic remains unchanged. This approach offers the identification and derivation of a service that can be accessed from other services. Nevertheless, the service itself cannot consume other services (one-way).

Screen Scraping may be used as a technique for collecting and extracting data from other programs displayed on the user display. Especially, screen scraping can be used for extracting data from web pages in order to use this data for further processing. Many tools are available that support the developers to design services based on a GUI, e.g. HttpUnit, OpenKapow. Microsoft published an article of how to create services that parse the contents of a web page. Screen Scraping can further be used to fill out any forms on web pages that automatically process the input afterwards or create a new GUI on top of legacy systems. A similar approach of how to expose a service from an existing mainframe on the presentation layer is Screen Logic Integration (SLI). Hereby, a sequence of screens is recorded, mapped to a function and then “played back” to the legacy system in order to be able to execute the function automatically. This can be achieved by using Shadow z/Services for Screens by DataDirect Technologies, for example. Various tools are available to integrate legacy application on a presentation layer that support the design and development of services. Sometimes it may be the only available option to integrate legacy application, when the legacy code is not available and there is no Application Programming Interface (API) available. Nevertheless, there is no integration of functions or data records, the reusability, flexibility and performance of the derived services is limited.

475 An approach for turning a character-based interface into a graphical interface has been proposed by Merlo et al. (1995).
477 For further information, please refer to http://httpunit.sourceforge.org
478 For further information, please refer to http://openkapow.com
482 For further information, please refer to http://www.datadirect.com
Legacy application may also expose various methods on the *functional layer* that can be wrapped into a service. Nevertheless, wrapping of certain functions is highly technology specific. There are multiple options depending on the used technology (e.g. CORBA, EJB) and the programming framework (e.g. .NET, JavaEE). Business logic integration requires that the presentation layer of an application is bypassed and interfaces of functions are wrapped into services directly. Three different methods can be distinguished, state-view, state-change and notifications. Based on the specific method, certain considerations have to be made concerning the data integrity and method invocation. Services encapsulating state-change operations implement functionalities that change data records (e.g. seat reservation), whereas state-view methods query a specific database. However, there may be various applications encapsulating similar functions and methods. Services derived with this approach can be accessed by other services and they can consume other services as well (two-way). An integration of different legacy application can be comprehensive because one can access the different data sets of each legacy system as well as the application logic. Nevertheless, one has to ensure the plausibility and integrity of the application logic that has to be integrated. This can be very complex and expensive approach, when interfaces are not available and the source code is not accessible.

Services or wrapper encapsulate functionalities that directly access operational or transactional data residing on databases without going through the business logic (*data layer*). These services offer one-way functionalities. A service accepts an incoming call and the parameters are verified and passed to a query. The results are returned to the service afterwards.

*Method according to NADHAN*  

NADHAN focuses on the technical derivation of software services. Thereby he proposes seven steps that have to be applied to identify and implement services. The steps comprise the identification, location, grouping, packaging, orchestration, routing and government of services. The identification of services and service providers are seen as the first step towards a SOA. He acknowledges that a transformation of each system’s function into a service results in a proliferation of services and therefore proposes to consolidate and rationalize services based on the invocation behaviour of the services. Seldomly used or similar services are therefore combined to form a unified service. However, he only differentiates services based on the characteristics of composed services (consisting of other services) or

---

elemental services. In addition, he proposes to conduct a domain analysis on the level of the business to group identified services and on the technical level to leverage domain specific characteristics.

C.b.b Invasive service identification

Overview

Invasive legacy modernisation approaches revitalise and streamline legacy systems to ease maintenance and extensions. These approaches concise a deep analysis of the code structures as well as an understanding of the system’s functionalities and their overall architecture.\footnote{Cp. Erradi, Anand, Kulkarni (2006a), pp. 257 ff.} The goal is to redesign, rearchitect and consolidate the activities to make the code modular and ease the gradual migration to a flexible architecture.\footnote{Cp. Mecella, Pernici (2001), p. 4.}

This is particularly important when multiple applications are running on different platforms communicating with each other. There may be different systems within an organisation that support similar business functions. Therefore, a simple wrapping and translation of each system’s functionality into a corresponding service may result in an extensive proliferation of services, which is difficult to manage and maintain.\footnote{Cp. Nadhan (2004), p. 41.} The aim is to identify and provide reusable services encapsulating certain functionalities of a system to other applications or users.

Before the actual analysis of specific legacy systems and applications, one can group potential services according to their domains. Domains can be technology specific and are therefore reliant on a given platform (e.g. Windows Service Domain). Since these platforms are most often vendor specific, one has to identify potential dependencies associated with that platform and other operating systems, in order to make proper use of the technology-specific capabilities. Application domains on the other hand encompass front-end applications belonging to an underlying system. Identification of potential services for that domain can help in the later stages of maintenance and administration of services, since the underlying system remains the same.\footnote{Cp. Nadhan (2004), p. 43.}

During the actual phase of analysing the application landscape, one may identify applications serving only one goal in an organisation. These legacy applications can be characterized as monolithic, single-tier, mainframe-based applications that have to be adapted and

\footnote{Cp. Erradi, Anand, Kulkarni (2006a), pp. 257 ff.}
\footnote{Cp. Mecella, Pernici (2001), p. 4.}
\footnote{Cp. Nadhan (2004), p. 41.}
\footnote{Cp. Nadhan (2004), p. 43.}
reengineered to work in a service-oriented environment.\textsuperscript{492} They can be categorized into three different clusters in regards to the degree of dependence on their environment:\textsuperscript{493}

- \textit{Programs that are not dependent on their environment}. These programs can be reused in any environment, which has a compiler to compile them. If programming languages like C/C++, COBOL or Fortran have been used for the application development, the reuse in different environments is solely dependent on available compilers.

- \textit{Programs that are partly dependent on their environments}. These programs, e.g. written in PL/I, Smalltalk or Forté, use runtime functions that are dependent on their environment. These applications can only be reused in other environments, if their runtime routines are replaced by compilable modules written in the host languages itself.

- \textit{Programs that are totally dependent on their environment}. All programs written in 4\textsuperscript{th} generation programming languages, e.g. ADS-Online, Natural or Oracle Frames, require a specific environment to run. These programs have to be kept in their native environment.

Thus, the more elementary programming language has been used for program development, the easier it is to reuse the software in another environment. Therefore, the programming language is an important factor to consider, if one has to decide which application has to be transferred to another environment, how applications are restructured and reengineered for service-enablement and which language is used for further application development projects.\textsuperscript{494}

\textit{Method according to Zhang, Liu, Yang}\textsuperscript{495}

The authors propose a method consisting of five steps for the identification and packaging of services within a legacy application. In the first step, the legacy system has to be evaluated in order to decide if service-enablement is suitable. One may apply different methods that help to evaluate the legacy system, for example the Options Analysis for Reengineering (OAR).\textsuperscript{496} The second step focuses on obtaining design and architecture information of the legacy system. Application modules in a legacy system are identified by system decomposition techniques. These modules can be further analysed and restructured by differ-

\textsuperscript{493} Please refer to Sneed (2006), pp. 3 ff for the following description of the different program types.
\textsuperscript{496} Cp. Bergey, O'Brian, Smith (2001); Smith, O'Brian, Bergey (2002)
ent algorithms and techniques.\textsuperscript{497} However, the outcome of this analysis step is the architecture of the legacy systems containing its components and connectors. The third step aims at identifying software services. Hereby, one has to examine the business requirements on a set of systems in the same application domain. The identification process starts by decomposing the application domain to acquire inherit system requirements. Hereby, certain business functions can be identified to be valuable and reusable and should therefore be encapsulated in services. Thus, these logical services are the result of the domain analysis and its requirements analysis.\textsuperscript{498} After the requirements have been identified, the next step aims at the identification of functionalities provided by the legacy system that can be encapsulated into services. The architecture of the legacy system examined before serves as an entry point. However, the architectural information may not be useful in analysing which functionality is provided by which components. Thus, one has to model and understand the legacy code in order to extract any valuable structured functionality. Many resources may be used for the reverse engineering process, namely the source code, graph user interfaces and further documents. The authors use an agglomerative hierarchical clustering technique to automatically identify classes, objects, inputs and outputs in order to derive a intermediate object-oriented model (for a procedural software system). The entities to be clustered are classes, entities and procedures. Identifier names are used to measure similarities between these elements. For example, an element for inheritance has heavy weight in clustering an object-oriented program. Thereby, redundant data and code is identified and abstract into classes and methods. After the algorithm has been used to structure the legacy code, one may obtain a dendogram. A cutting point has to be identified by the architect in order to define the boundary of a service based on the legacy code. Hereby, the complete code can be structured into several services that are loosely coupled to one another. Afterwards, the legacy functionalities have to be mapped onto the business requirements, in order to identify which services inhabit what kind of functionality and which functionality cannot be provided by the legacy system. Step four aims at service enabling the application. An external interface for each of the services has to be developed. Thus, they can be used by other applications. Rather than serving one specific purpose or application, they can now be leveraged by several other systems, whereby the inputs and outputs remain the same.\textsuperscript{499} The last step comprises the publication of the service and the choreography with other services. The authors do not apply design principles, nor do they classify the derived services. A linkage to process models is made indirectly.

\textsuperscript{497} Please refer to Zhang, Liu, Yang (2005).
\textsuperscript{499} Cp. Nadhan (2004), p. 44.
The authors propose a method to identify services based on analysing and restructuring existing applications by using features. Features are seen as a coherent and identifiable bundle of system functionalities or a single functionality that is visible to the user. Not all single functionalities can be regarded as features though. The authors propose certain rules that can be applied to determine if a functionality itself can be seen as a feature.\(^{501}\) Since applications typically consist of more than one feature, models may be developed visualizing the relationships between the features of an application.\(^{502}\) A wide variety of sources might be consolidated when constructing a feature model, including domain experts, textbooks, documentations, technology forecasts, etc. Once the different features and their relationships have been discovered, one has to locate the features in the source code of the legacy application. Feature location is a reengineering technology that aims at revealing the relationships between an implementation module and a particular feature. There are different ways for identifying the source code, which belongs to a feature.\(^{503}\) Test case based location techniques\(^{504}\) might be used in combination with program slicing techniques. Thereby small modules may be identified that serve one specific feature. Program slicing techniques are used to identify the relevant code and to slice of non-relevant variables and code segments.\(^{505}\) After the source code has been identified, it has to be aggregated into modules.\(^{506}\) A service can be regarded as a local grouping of one or more features depending on the context and complexity of the feature. As a guideline for the grouping of features to services, one may consider single features to act as service operation and the service itself as the context of these operations.

CHEN ET AL. propose to develop delegation classes for each service in order to bridge the gap between the legacy source code and the self-contained service. Thus, a service operation is implemented by a public method inside a delegation class as a service interface exposed to the service consumer.\(^{507}\) Specific “glue” code intercept the incoming calls and invokes the adequate operations or methods. Design principles or linkages to business requirements are not provided by this method nor are service classifications or layers.

\(^{500}\) Cp. Chen et al. (2005).
\(^{501}\) For a complete listing of the rules (6), please refer to Chen et al. (2005).
\(^{503}\) Cp. Li et al. (2005), p. 338.
\(^{504}\) Cp. Wilde et al. (2003).
Method according to SNEED\textsuperscript{508}

The author proposes a technical approach to identify software services based on restructuring existing legacy code. As a starting point, the author proposes to take business functions that have to be provided by the IT. Business operations provided by applications may be identified by performing an application portfolio analysis. The goal is to identify all the operations that are supported by the application and that are valuable to the business.\textsuperscript{509} These operations have to be identified and extracted from the legacy code, before they can be wrapped and exposed as services.\textsuperscript{510} Depending on the size of the program, a domain expert is required to analyse the code structure by using reverse engineering tools. These tools help in analysing business operations by discovering the results of the operations. The variables, which are returned by the functions processing the business operation, are used to identify the functions themselves. Functions might be scattered throughout several blocks of code in several applications. Additionally, a code block might be used by several functions. Thus, n:m relationships between code blocks and operations may exist. A data flow analysis based on the output of the operation identifies all statements, variables and functions that have been used in producing that result. These code segments are then copied together with their referring variables (code stripping).\textsuperscript{511} The code is then reassembled into an own module with its own interface. The original input arguments are the input parameters and the output arguments are then output parameters. These modules can then be exposed as software services. Thereby, the complete legacy application may be replaced by a series of services that are loosely coupled to each other. These services can then be leveraged independently regarding the usage intensity (load balancing). However, this approach requires a lot of inside knowledge of the domain expert regarding the underlying code structure and business logic structure. Furthermore, the granularity of the service is dependent on the business requirements. Considerations concerning the linkage to processes or the application of design principles are not provided by the author.

C.c Other approaches

Overview

The approaches described in this cluster do not directly relate to the service identification and analysis. However, they can be used to enhance and extend the existing approaches.

\textsuperscript{508} Cp. Sneed (2006).
\textsuperscript{511} Cp. Sneed, Erdos (1996).
Method according to ZACHARIAS\textsuperscript{512}

This method is focussing on the decomposition of information systems into components, services and operations. Components are cohesive functional units, but the author misses to define the granularity of these components. Thus, the definition of the boundary of one component lies in the sphere of the service developer. Components encompass one or more services that relate to business functions that may own certain data sets. Each service consists of at least one or more operation(s) that define(s) specific transactions. The author describes how to design services with Enterprise Java Beans (EJB). Several service characteristics are mentioned, but principles are only stated implicitly. No differentiation between layers and types of services is made. The author also proposes to transform each function inside a process model into a service. The executing entity is herby the service provider. In addition, input and output parameters have to be defined for each function. Hence, this approach cannot clearly be allocated to one specific class of service analysis approaches.

Method according to FEUERLICHT, MEESATHIT\textsuperscript{513}

These authors focus mainly on the design of the interfaces of software services. Nonetheless, they base this approach on the fact that each activity within a process can potentially encapsulated by a software service. They apply design principles to design adequate services. Additionally, they exemplify how to use data normalisation techniques to design proper interfaces.

Method according to SPROTT ET AL.\textsuperscript{514}

The authors do not propose a comprehensive method for service identification, but they analyse business process characteristics, to conclude which processes might benefit most from service-enablement.\textsuperscript{515} They propose that the more actors and stakeholders are involved in executing a certain cross-functional process, the greater is the benefit from service-enablement since different stakeholders may utilize different technologies for participating in that process. Services are herby used to integrate the different technologies of the participants. If a process needs real-time information, service-enablement may be used to develop centralised services that provide all service consumers with consistent data. Services should also be used for automated processes that do not need human interventions to execute. If the process is dynamic, it may also benefit from service-enablement since the process can be reconfigured by combining different services to fulfil the new business re-

\textsuperscript{514} Cp, Sprott et al. (2003), pp. 70 f.
\textsuperscript{515} The authors propose the criteria for web services. Nevertheless, the criteria are generic enough to be applied on software services in general.
quirements. Because of the scope and focus of this approach, no service types, layers or design principles are proposed.

**Method according to Sewing, Rosemann, Dumas**

These authors propose four phases that can be used to assess the suitability of processes to be service-enabled. Given a certain set of processes, the first phase involves the mapping of the process’s characteristics to certain disqualifying criteria. For example, processes that solely consist of manual activities are excluded as well as processes that run stable in isolation so that the risk of changing this process is not justified. The second phase comprises the classification of the remaining processes into four clusters by applying organisation’s independent criteria. Thereby, processes can be identified that are less suitable for service-enablement and hence are excluded from the further analysis. The remaining processes are mapped against organisation specific criteria in the third phase. The criteria may include the assessment of the strategic importance of service-enablement for an organisation. The last phase comprises the determination of the priorities of the remaining processes for service-enablement.

**Method according to Subasu, Ziegler, Dittrich**

Subasu, Ziegler and Dittrich proposed an idea of how to service-enable Database Management Systems (DBMS). They argue that today’s DBMS are mainly large and heavyweight monolithic systems that come with a growing set of fixed extensions. These characteristics lead to high maintenance costs and inflexibilities in regards to adaptability to new requirements and software evolution issues. On the other hand Service-Based Data Management Systems (SBDMS) are supposed to handle different data types, to provide different methods for adapting new database features and to be flexible in regards to the extensibility to the system over time. Derived from older approaches of database architecture models, the authors propose four different functional database layers.

- **Storage Service Layer**: Services on this layer work on a byte level in close cooperation with the file system of the operating system. Their purpose is to find and update specific data sets and optimize storage methods for different data types. Additionally, they have to handle the physical specification of each storage device inside the database.

---

517 The authors assess the suitability of processes to be supported by web services. However, the criteria are also applicable for software services.
• **Access Service Layer:** More complex access paths, mappings and particular extensions for special data models are located in this layer. Additionally, it provides representations of data records and navigation functionalities for logical record structures based on joins.

• **Data Service Layer:** This layer represents logical data structures like tables or views. These data structures do not provide any procedural interfaces to the underlying database.

• **Extension Service Layer:** This layer is used to design tailored extensions (e.g. creating new services based on available services from other layers, provide functionalities for new data types). Application specific services can be found in this layer.

By service enabling the database system, one can adapt its services to application specific functionalities and not the other way around. Thereby, specific services may be shared by different applications. Access rights are not restricted to data types or user roles, but rather are formulated for single services. Distinct services provide bridging capabilities between internal and external services in order to provide a high flexibility, adaptability and extensibility of the complete database.
**D Service design principles**

The design principles are based on a thorough analysis of different sources. Since no explicit design principles for business services are existent in literature, the principle for software services will be described and analysed first, before analogues considerations are made for business services. For each design principle, the applicability within the service analysis phase will be evaluated as well.

**D.a Contract**

*Software service*

A service contract “establishes the terms of engagement, providing technical constraints and requirements as well as any semantic information the service owner wishes to make public.” To allow services to interact with each other and to be invoked by their service consumers, they need to share a formal contract that defines the terms of information exchange and the commitments made by both parties. Service contract design is very important since the agreement to a service contract means to establish a dependent relationship between the service consumer and provider. The contract comprises a technical interface, which describes the endpoints, each operation of the service, its input and output parameters and the specific data types, and a document detailing the non-technical characteristics. The non-technical characteristics can be recorded in a specific document, e.g. a service level agreement (SLA), which can be seen as a document that establishes a contract about the service quality, such as availability, accessibility and performance. Since it describes non-technical service characteristics, it is not technical but rather legally binding as it extends the technical interface of a service. However, the aim of this design principle is to design services that are in the same service repository in a consistent standardised manner including the specification of functional service expressions (e.g. naming of elements), service data representation (e.g. input/output data type) and service policies (e.g. validation of assertions). Furthermore, one has to ensure that the logic implemented by the service is only accessed through the service interface. Thus, a separation between the interface and the underlying logic has to be enforced that results the chance to change the interface independently from the implementation and vice versa. This leads to the considerations of providing the interface for a service separately from the service implementation, so that the

---

organisation providing the interface for a service does not have to be identical with the organisation providing the service implementation. Other authors relate the separation of interface and implementation to service autonomy\textsuperscript{526} or service coupling.

\textit{Business service}

This principle is fully applicable for software service as well as business services. However, the specific design and content of a service contract may be different, since a software service has a technical focus, whereas the business services focuses on business operations. However, the business service characteristics have to be described in a standardised manner throughout the organisation to support the interoperability between different business services as well as to provide a consistent way of providing business services to external partners. SLAs have to be addressed as well as the interaction points with the service consumer.

\textit{Applicability regarding service analysis}

The service analysis phase is focussed on the identification of services and a preliminary specification of their operations. Hence, this design principle is partly applicable, as the detailing of the operations is related to the definition of the interface.\textsuperscript{527} However, the contracts of services have to be carefully designed and described based on business specific requirements that go beyond the scope of this thesis.

\textbf{D.b Abstraction}

\textit{Software service}

As explained previously, a service consists of a service contract including one or more interfaces and the implementation logic that can be distributed on various applications. Nonetheless, the underlying logic of a service is not relevant for the service consumers. All that is exposed to the service’s environment is the service contract including the service interfaces.\textsuperscript{528} Therefore, services act as black boxes since they hide their details from the outside world. This service design principle focuses primarily on information hiding, so that the service consumer has the right amount of information that is needed to use the service.\textsuperscript{529} The information about services is limited to what is published in the service description documents that define the interaction requirements and constraints.\textsuperscript{530} Thus, the

service description can abstract different types of information about the specific service. ERL applies this design principle on four distinct types of information, namely technology information, functional information, programmatic logic information and quality of service information.\textsuperscript{531} This kind of information is typically provided by SLAs. However, one has to decide what essential information \textit{needs} to be published for each service in order to allow a service consumer to interact with the service in the desired way. One may also differentiate between internal consumers and external consumers respective to the organisation to restrict the published information about a service.

\textit{Business service}

This service design principle can also be applied to business services. Each business service has to be described in a certain way, including the business service contract, the interactions and the underlying operations. One has to decide what information has to be published in order to set the right incentives concerning the usage of the service. Hereby, one probably has to distinguish between external service consumer and internal service consumer, since internal service consumers may be allowed to see the underlying processes whereas external service consumers may be restricted to see the business service contract including information about the inputs, outputs and additional service requirements.

\textit{Applicability regarding service analysis}

The principle of abstraction has to be regarded when the service contract is defined. As described previously, the service contract can only be defined after the services are identified. Hence, it is not used in the further course of this thesis.

\textbf{D.c Autonomy}

\textit{Software service}

Autonomy refers to the level of independence of one service. This means a purely autonomous service has full control over its runtime execution environment, which results in an increased reliability and predictability, since external unpredictable influences are minimised.\textsuperscript{532} Thus, a service’s contract should not overlap with any other contract and the underlying runtime environment of one service should reside under the control of that ser-

Service normalisation is another aspect that aims at designing the operations in a non-redundant manner.\textsuperscript{534}

Two specific types of autonomy can be distinguished. On the one hand, the \textit{runtime autonomy} represents the control of service over its environment. This is especially important for service compositions, since the composition is only as autonomous as the autonomy of each single service. If the performance of one service in a composition is heavily dependent on external factors, than the outcome of the complete service composition will not be very predictable. On the other hand, the \textit{design-time autonomy} describes the autonomy of the service owner over the service design. As soon as the service is exposed to its environment, service consumers bind themselves to the service contract. This may influence the control the service owner has over the changes he wants to make to the service, since the obligations made in the contract have to be fulfilled.\textsuperscript{535}

\textit{Business service}

The principle of autonomy can be applied on the level of the exposed functions. Thus, the exposed functions should not overlap with already existing services. The runtime considerations cannot be directly applied, since the definition of a business service is always dependent on the underlying realisation, e.g. manual process, software services, application packages, etc.

\textit{Applicability during service analysis}

Especially the application of this design principle to derive non-overlapping service contracts can be used during service analysis. Hereby, the service boundaries have to be set to avoid the replication of redundant functionalities. However, the autonomy of a service is typically in the focus during the design and realisation of a service because these phases address the runtime environment of a service including the definition of the autonomy of the implementation logic.

\textbf{D.d  Coupling}

\textit{Software service}

This service design principle is applicable on different elements of a service and a SOA in general and refers to the level of dependency between two or more elements. The observ-
able type of coupling can be identified as the interdependency of multiple services and service compositions. However, other types of coupling or dependency can be encountered. For example, the coupling between the service implementation to proprietary vendor technology, the coupling between the service implementation to the service contract and vice versa and the coupling between the service to one or more business processes.\textsuperscript{536} However, in this thesis coupling is referred to as the dependency of one service to another service. As soon as one service calls an operation provided by another service, the service is dependent on the functionality offered by the other service’s operation and the services are coupled. However, services that are not dependent on the other services have a high reusability and maintainability potential. Thus, the coupling between services should be as loose as possible.\textsuperscript{537} This goal can be achieved by specifying a well-defined contract for each service that serves as the only access point to the service and its underlying logic.\textsuperscript{538} This way a simple interchange between services can take place. For example, an existing service may be substituted by a third-party service because the external provider offers better performance. If different versions of a service exist, the business requirements will determine which service will be used depending on a certain scenario (e.g. external or internal accounts depending on the purchase order). The design principle addressing the service contract is closely related to this goal since services should communicate and interact with their service consumers as independent as possible.\textsuperscript{539} In addition to a well-defined contract, one should minimise the consumer coupling requirements, since a tight coupling between a service and its consumer might hinder the evolution and changeability of the service.

\textit{Business service}

As mentioned before, coupling can be seen as a type of dependency. Hence, the coupling of software services can be extended to the realm of business services, since business service might need to interact with each other to achieve a certain goal or task. Similar considerations have to be made concerning the changeability of a business service.

\textit{Applicability for service analysis}

The only type of coupling that can be influenced during the analysis phase is the coupling between services by specifying which operations should be provided by which service. Thus, all other types of coupling\textsuperscript{540} need to be focussed during subsequent phases, since

\textsuperscript{540} Please refer to Erl (2007).
they typically comprise the detailed definition of the service contract and considerations concerning the realisation of the service.

D.e Statelessness

Software service

Services should not manage their own state and minimise the duration for which they hold state information; thus, services should be as stateless as possible.\(^{541}\) State information can be considered specific data sent by the service consumer to the service or specific data that is used to control the execution of the service. Different terms can be found in literature describing the state information. Erl distinguishes between three types of state information, namely session data, context data and business data.\(^{542}\) However, to minimise the consumption of resources, state management should be deferred when necessary. A database, for example, can be used as the means of state management deferral.\(^{543}\)

A service has a certain state when it is invoked since it has to process the information that has been used as the input parameters of a service. The longer a service holds this kind of information, the longer it is unavailable for other service requestors.\(^{544}\) Additionally, the more state information a service holds, the more complex is the exception handling. A stateless service does not need to know the state of any other services and does not hold the received information for longer than the invocation and execution. An invocation of a service’s operation should not depend on any stored information by the service.\(^{545}\) Thus, by invoking a service operation, all data that is needed for a correct execution of the operation should be included. This way, a service is not dependent on any other services or inputs that are needed for the execution, processing and returning of the result. By minimising the amount of data a service has to store and process to return the desired result and the time a service has to hold this information, a better reliability of the complete services that needs to be executed to return a certain result can be achieved. In addition, the scalability of a service should be enhanced to be able to interact with multiple client requests. However, a service will always be stateful to a certain extent, if its operations need certain input parameters for the execution.

\(^{545}\) Cp. Newcomer, Lomow (2005), p. 82.
Business service

A business service can only be as stateless as the underlying realisation. However, similar arguments as described for software services are also applicable to business services.

Applicability for service analysis

This design principle is not applicable for the service analysis phase, since it primarily focuses on the implementation of state management, which should be addressed during the realisation and implementation of the service.

D.f Cohesion

Software service

The idea about cohesion has emerged in the area of software engineering, since it refers to the concept of grouping elements based on their functional relatedness to perform a certain task.546 Thus, service cohesion can be analysed on two separate levels. On the one hand, the cohesion of the different operations exposed by a service’s interface can be analysed regarding their functional relatedness. On the other hand, the relatedness of a service’s different implementation elements can be analysed as well. Also PAPAZOGLOU AND VAN DEN HEUVEL refer to service cohesion as the strength of functional relatedness between operations in a service. They propose three levels of service cohesion:547

- Functional service cohesion: Functional cohesion enforces services to only perform one problem-related task, e.g. check product availability. Thus, all operations within a service should only be related to the purpose of existence of the service.

- Communicational service cohesion: Communicational cohesive operations within a service use the same set of input and output messages. The related activities encapsulated inside a service are hardly related to other activities in other services.

- Logical service cohesion: A logic cohesive service encapsulates operations that have logically similar functions, but are independent of one another (e.g. mode of payment). Logical cohesive services are therefore grouped by the means of accomplishing a certain task, but are disjunctive from one another (services present alternatives for the accomplishment of one task).

Additional levels of cohesiveness can be found in literature related to software engineering. However, functional relatedness is seen as the preferred and strongest level of cohesion. The functional relatedness can be interpreted as the interdependencies between operations or resources.

Business service

Cohesion is probably the most applicable design principle for business services, since it focuses on the grouping of related operations. The same considerations have to be made for business services.

Applicability for service analysis

This design principle is fully applicable in the service analysis phase in order to cluster the functionality that should be provided by the service. For service analysis, the perspective of the service’s interface is the more important point of view on cohesiveness and will therefore be in focus in the remains of this thesis. The operations within a service should be grouped related to a certain task, which in turn needs to be defined based on certain business requirements. Hence, the exemplified grouping of operations in the course of this thesis can only be seen as possible solutions.

D.g Discoverability

Software service

Services should provide means to be discovered by their service consumers. Since service consumers include other services and systems as well as humans, the service must also be understandable by all entities that might want to use that service. Thus, the service description needs to be as comprehensive as possible. Therefore, the service contract must list all necessary information to describe the purpose, functionality and additional requirements of the service. The more extensive the description of a service is, the better the discoverability of the service. If services are discoverable, a proliferation of redundant services will be prohibited. Extending the view from one service to a SOA, some kind of service repository needs to be established that can be used to discover services and their functionalities. Especially for software services, two types of discovery need to be distinguished. On the one hand, the design time discovery is conducted most often by humans, who assemble

services to compositions or check for already existing services to avoid building redundant
services. Thus, a registry is needed that provides certain documents about related services
that are interpretable by humans. On the other hand, the runtime discovery technologies are
used by other programs and services that issue dynamic discovery queries to a service reg-
istry that provides programmatic interfaces.552

Service discoverability is closely related to the visibility potential between service provider
and service consumer as described in Chapter 2.2. Both parties can only interact with each
other, if they can “find” each other. Hence, the discoverability on a service level related to
the description of the service has to be extensive.553

Business service

The same considerations for software services are applicable to business services. In order
to be used by external consumers or internal business entities, the business services have to
be made public. This can be done by publishing the description of a service in a service
registry that can be accessed and interpreted by the potential service consumers.

Applicability for service analysis

The focus of this design principle lays on the adequate specification of the service contract
as well as on the mechanisms to discover the services. Therefore, this principle is not ap-
licable during service analysis.

D.h Reusability

Software service

The principle of reusability has straightforward underlying concept: make the service use-
ful for more than one single purpose.554 Thus, services should be flexible and composable
in other situations or domains.555 This can be achieved by defining an agnostic functional
context and defining the encapsulated logic as generic as possible, so that the service is
not bound to one single usage scenario. Furthermore, the service contract should encom-
pass generic and extensible operations, so that they are not restricted to one specific type of
input parameters, for example.556

Two types of reusability can be identified, namely the actual reuse and the planned reuse. Three different kinds of reusability can be distinguished, when focussing on the planned reuse. First, the tactical reusability focuses on the specification of service operations for immediate requirements that must be fulfilled by the service. Second, the targeted reusability focuses on the implementation of service operations that have the highest potential of being reused under more than one scenario. Lastly, if the boundary of the service has been specified before and the requirements of the service has been analysed, a complete reusability of the service can be focussed. Hereby, the service provides a set of operations that address known requirements. Actual reuse on the other hand, can be analysed by identifying the amount of services or service consumers that were built to use the service and the frequency with which these service consumers have used the service.\textsuperscript{557} However, the better the reuse potential of one service the lower the long term costs, since redundant functionalities can be avoided that otherwise may result in increased costs of maintainability of the system. In the short term, costs might be avoided since already existing services might be reused and recontextualised based on project-specific requirements and the execution context of a service may change consequently.

\textit{Business service}

Based on the granularity of the business service, a business service might be designed with the reusability principle in mind. However, a coarse-grained business service describes what a business can offer to its environment. Therefore, the principle of reusability is not directly applicable to business services. Nonetheless, if business services are decomposed into finer-grained services, reusability may play a more important role.

\textit{Applicability for service analysis}

The planned reuse is focussed during the service analysis phase, since the principle can be applied to decide what operations should be provided by the service.

\textbf{D.i \quad Composability}

\textit{Software service}

This principle focuses on the design of a service, so that they can participate effectively in different service compositions. Hereby, it is irrelevant if the service itself is composed of other services or if it is an atomic service.\textsuperscript{558} A composed service may consist of several other services since it can represent any type of logic from various resources (including

other services). However, elemental services do not depend on any other services. If a service is used in two separate service composition the service is also reused.\textsuperscript{559} Hence, the service composability and reusability are closely related and the participation of a service in different composition is one form of reuse. Nonetheless, they focus on two different issues and objectives. A service can be composable, but not reusable. This is the case, if a service is used in one service composition, but not reused by more than one service consumer. On the other hand, a service can be reusable, but not composable. This situation may occur, if a service is used on different occasions by different service consumers. However, the service may be an elemental service that does not play a part in a service composition. The underlying idea of this principle can be found in software engineering as well, since most sort algorithms use the divide-and-conquer strategy, which breaks down a big problem in several smaller parts that can be solved separately to solve the bigger problem consequently. This strategy is adapted to support the separation of concerns focussing on the development of solution logic to solve a corresponding small problem that together solve a bigger problem.\textsuperscript{560} To achieve composable services, the execution environment must be highly efficient. In addition, the service contract needs to be flexible enough to support different data types.

\textit{Business service}

The differences between reusability and composability are primarily related to the runtime environment of a service. Since the specification of business services is technology-agnostic, this principle is not applicable for business services.

\textit{Applicability for service analysis}

Service composability is closely related to service reusability, which is applied during service analysis. Measures related specific to the service composability, have to be addressed during subsequent phases of service analysis, since they focus on the service contract and the runtime environment.

\textbf{D.j Other design principles}

\textit{Interoperability}

Since one of the proposed benefits of SOA is the integration of heterogeneous application systems,\textsuperscript{561} services should be designed for interoperability using standards.\textsuperscript{562} By using

\textsuperscript{559} Compare the service design principle „reusability“.
\textsuperscript{560} Cp. Erl \textsc{(2007)}, pp. 388-396.
\textsuperscript{561} Cp. Chapter 2.2.3.
standards, one could even enforce a seamlessly integration between different organisations. Interfaces should therefore be described in a standardised way. Services communicate via consistent protocols and rely on standardised data formats. The vision is to use openly available standards, that are not restricted or under control of any vendor or industry consortium. Since this thesis abstracts from any implementation-specific considerations, an analysis of the suitability for certain implementation standards is beyond the scope of this work. If interoperability is regarded as a requirement to enable services to communicate with each other, all stated design principles that have already been stated support or contribute to interoperability in a certain manner.

ERL concludes, “stating that services must be interoperable is just about as basic as stating that services must exist. [...] A fundamental goal [...] is for interoperability to become a natural by-product [...].” Hence, the interoperability is not regarded as an own principle in itself.

Service granularity

Service granularity refers to the scope of functionality that is exposed by a service to its environment. Different services may offer different levels of granularity, based on the business usage scenarios and requirements. The granularity of services has been discussed by several authors, who describe how to adapt the granularity based on business needs.

Fine-grained services do not provide a high-level of business value or usefulness, but are highly reusable based on their very specific task. On the other hand, they might not be loosely coupled, since they might be dependent on other services (logical sequence of services). Additionally, the administration and coordination costs increase with the proliferation of fine-grained services. It might also be harder to find adequate services. Fine-grained service may involve more implementation work and most often, more services that are technical are developed that lack a close relation to the actual business processes of an organisation. SNEED proposes to use as fine-grained services as possible, in order to keep development and testing costs as low as possible. The more comprehensive the interface of a service, the longer is the input and output parameters and the transmission time for mes-
sages. Furthermore, interfaces exposing only a few operations are easier to test and validate.\textsuperscript{570}

Coarse-grained services may expose functionalities that have a high value for the business. Since they expose more business logic than fine-grained services, they are loosely coupled in a sense that the encapsulated operations are highly cohesive. Thus, the message exchange and communication between coarse-grained services is less than between fine-grained services. On the other hand, coarse-grained services have typically more complicated interfaces and are not very reusable because they might fit in only one specific context.\textsuperscript{571} If applied in new situations or circumstances, coarse-grained services might not encompass the functionality needed or might even provide functionality that is not needed at all. Papazoglou and Van den Heuvel argue that it is preferable to create coarse-grained services that implement a complete business process. This way the service consumer gets direct access to a complete business service rather than being confronted with a large number of data entries for each different fine-grained service.\textsuperscript{572}

The ongoing discussion about the service granularity is directly related to the design principles and service types. Task services and entity services may be regarded as finer-grained services or basic services, whereas process services are more coarse-grained since they can encapsulate the logic of a business process by composing the related services.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{granularity_of_services.png}
\caption{Granularity of services}
\end{figure}

\textbf{D.k \hspace{1em} Interrelations between design principles}

The analysed service design principles are not disjunct and isolated since they influence each other. Some characteristics directly benefit from the support of other principles (e.g. service reusability and service composability) whiles other provide support for the realisa-
tion of other principles (e.g. service contracts and service autonomy). For example, to minimise coupling, one could comprise all relevant operations into one single service. However, this would contradict with the principle of cohesiveness, since operations are comprised within that one service that might not be functional related. Additionally, the principle of reusability is disregarded, since such a coarse-grained service would typically be not applicable in several business scenarios. The right trade-off between reusability, coupling and cohesiveness is highly dependent on organisation specific factors such as transaction volumes and usage of specific services. Therefore, the developer, who is responsible for the implementation of the service and its providing entity, has to decide about the adequate granularity specific to organisation’s requirements.

The interrelationships and interdependencies between different service design principles have been extensively analysed by ERL. He also describes how the design principles affect the different types of software services. Since an extensive description of the interrelationships goes beyond the scope of this thesis, one should keep in mind that focussing on one design principle, influences the characteristics of a service derived by applying the other design principles. As an example, one may analyse how the design principle of reusability is influenced by other design principles:

- Service autonomy provides an execution environment conductive to service reusability.
- Service statelessness maximises opportunities for service reusability.
- Service discoverability provides a medium for promoting service reusability.
- Service abstraction packages service in support of service reusability.
- Service coupling minimises cross-service dependencies in support of service reusability.
- Service reusability primarily enables service composability.

---

E  Detailed overview of the consolidated approach

Figure 5-5: Phases, activities and inputs/outputs for the derivation of business services
**Figure 5-6:** Phases, activities and inputs/outputs for the derivation of software services
Event-driven Process Chains and the Operation Model

The operational model visualises how the exposed operation(s) of a business service is internally realised by an organisation. For that purpose, a process model can be established using a modelling language. For this thesis, the process model is derived by using the syntax and semantic of the notion for Event-driven Process Chains (EPCs).577

The main elements of an EPC can be described as follows:578

![Graphical elements of an EPC](image)

Source: Adapted from Scheer, Thomas, Adam (2005), p. 128.

Figure 5-7: Graphical elements of an EPC

The three core nodes of an EPC are functions, events and connectors. Events represent a point in time, whereas functions consider a time-consuming perspective of the accomplished task. An EPC starts with an event, ends with an event and contains at least one function. An event cannot be the predecessor or the successor of another event. Similarly, an activity cannot be the predecessor or the successor of another activity. Each event or activity has only one incoming and one outgoing edge.579

To visualise functions within a process that represent and are decomposable to subprocesses, the following graphical element has been used.

![Decomposable Function](image)

Figure 5-8: Decomposable Function

---

577 For further details, please refer to Nüttgens, Rump (2002); Scheer, Thomas, Adam (2005).
Figure 5-9: The operation model for “quote processing”
G  Concluding Declaration

I hereby confirm that I have written my master’s thesis “A consolidated approach for service analysis” autonomously and without assistance. I have marked and referenced all parts, which I have taken directly from other authors, or which resemble the lines of thoughts of other authors and cited the sources.

Brisbane, den 4. March 2008

Ich versichere hiermit, dass ich meine Masterarbeit “A consolidated approach for service analysis” selbstständig und ohne fremde Hilfe angefertigt habe, und dass ich alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlehnenden Ausführungen meiner Arbeit besonders gekennzeichnet und die Quellen zitiert habe.