



TECHNISCHE
UNIVERSITÄT
DARMSTADT

GOVERNANCE *for*
SERVICE-ORIENTED ARCHITECTURES –
COMPONENT ANALYSIS
and DECISION SUPPORT *for*
PROCESS CONFORMANCE ASSESSMENT

Vom Fachbereich Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertation

von

Dipl.-Wirtsch.-Ing. Michael Niemann

geboren am 1. August 1978
in Tübingen, Baden-Württemberg

Vorsitz: Prof. Dr.-Ing. Hans Eveking
Referent: Prof. Dr.-Ing. Ralf Steinmetz
Korreferent: Prof. Dr. rer. nat. Paul Müller

Tag der Einreichung: 19. August 2011
Tag der Disputation: 14. November 2011

Hochschulkennziffer D17
Darmstadt 2012

Michael Niemann: *Governance for Service-oriented Architectures – Component Analysis and Decision Support for Process Conformance Assessment*, © 2012

Dieses Dokument wird bereitgestellt von tuprints, E-Publishing-Service der Technischen Universität Darmstadt.

<http://tuprints.ulb.tu-darmstadt.de>

tuprints@ulb.tu-darmstadt.de

Bitte zitieren Sie dieses Dokument als:

URN: [urn:nbn:de:tuda-tuprints-29411](https://nbn-resolving.org/urn:nbn:de:tuda-tuprints-29411)

URL: <http://tuprints.ulb.tu-darmstadt.de/2941>

Die Veröffentlichung steht unter folgender Creative Commons Lizenz:

Namensnennung – Keine kommerzielle Nutzung – Keine Bearbeitung 3.0 Deutschland



<http://creativecommons.org/licenses/by-nc-nd/3.0/de/>

Meiner kleinen Gingelschlöre ;)

ABSTRACT

TIMES characterised by globalisation, volatile markets, and increasing numbers of legal regulations, present companies and their IT systems with new challenges. Increasing flexibility requirements and the requisite continuous adaptation of internal, IT-based processes often exceed the IT departments' capabilities for timely reaction to environment changes and fast implementation of sustainable solutions. However, experts consider this very ability – adapting the business strategy as well as the IT system – an important competitive advantage.

In the last decade, the paradigm of Service-oriented Computing has emerged that addresses these challenges. Based on software *services*, Service-oriented Architectures (SOA) represent a new concept of efficiently mapping business processes and business goals to IT, adhering to the companies' process flexibility and adaptability requirements outlined above. Regarding IT system control and adherence to legal regulations, so far, in large companies, *IT Governance* proved itself as indispensable support. In recent years, *SOA Governance* has been recognised as an holistic discipline that is to realise this support for a large SOA system, while preserving its high degree of flexibility. So-called IT Governance Frameworks serve as a basis, formulating best practices and reference processes. For the assessment of adherence to these, experts certify IT systems, for example concerning process conformance. In the case of the latter, in order to ensure process conformance already in the early state of process design, efficient reference process retrieval techniques are beneficial. Currently, experts' notions of SOA Governance vary considerably so that SOA Governance Frameworks as well as analytical support for process conformance assessment are scarcely represented in both, research and industry.

In this context, this thesis contributes in the two topics of *SOA Governance* and *Process Model Similarity*. First, we elaborate, amongst others, a new operational model for SOA Governance. Second, supplementary, we develop a novel analysis technique for process conformance assessment.

Regarding the first contribution, we perform a structural analysis of approaches and notions of SOA Governance, authored in consulting industry, in academia, and by software vendors. As key result, we identify 10 central components of SOA Governance. Based on the insights of the analysis, we develop a comprehensive definition, a consolidated service life cycle, as well as a generic operational model for SOA Governance.

Regarding the second contribution of this thesis, complementing the operational model, we develop a technique for comparison of reference process models with models of realised processes, as well as for their retrieval (*related cluster pairs*). The comparison approach performs a structural decomposition based on similar process regions. The evaluation of this approach attests a high level of accuracy, which indicates an effective assistance to manual business process conformance assessment. Additionally, based on related cluster pairs, we introduce a novel metric for process model similarity. Evaluation results show clear performance improvements compared to related approaches. Beyond the scope of SOA Governance, this approach represents a general contribution to *Process Model Similarity* research.

KURZFASSUNG

IN VON Globalisierung, unbeständigen Märkten und einer steigenden Anzahl gesetzlicher Regulierungen geprägten Zeiten sehen sich Unternehmen und ihre IT-Systeme neuen Herausforderungen gegenüber. Steigende Flexibilitätsanforderungen und die notwendige ständige Anpassung interner IT-gestützter Prozesse übersteigen oft die Fähigkeiten des IT-Bereichs zur rechtzeitigen Reaktion auf Änderungen im Unternehmensumfeld und der schnellen Umsetzung von nachhaltigen Lösungen. Experten halten jedoch gerade diese Fähigkeit, welche die Anpassung sowohl der Geschäftsstrategie als auch des IT-Systems betreffen, für einen wichtigen Wettbewerbsvorteil.

Im letzten Jahrzehnt kam das Paradigma des Service-orientierten Computing auf, welches diese Herausforderungen angeht. Gestützt auf Software *services*, stellen Service-orientierte Architekturen (SOA) ein neues Konzept dar, effizient Geschäftsprozesse und Geschäftsziele auf die IT abzubilden und dabei den Prozessflexibilitäts- und Anpassbarkeitsanforderungen zu genügen. Zur Steuerung des IT-Systems sowie Einhaltung gesetzlicher Regulierungen hat sich, insbesondere in großen Unternehmen, *IT-Governance* bisher als unverzichtbare Unterstützung erwiesen. In den letzten Jahren wurde *SOA-Governance* als eine ganzheitliche Disziplin wahrgenommen, welche diese Unterstützung für ein großes SOA-System unter Bewahrung dessen hohen Grades an Flexibilität leisten soll. Grundlage sind hierbei sogenannte IT-Governance-Frameworks, welche bewährte Vorgehensweisen und Referenzprozesse formulieren. Zur Gewährleistung ihrer Einhaltung werden Zertifizierungen vorgenommen, z.B. bezüglich Prozesskonformität. Um diese Konformität bereits im Stadium des Prozessentwurfs zu gewährleisten, sind effiziente Techniken zur Auffindung relevanter Referenzprozesse zuträglich. Die Auffassungen von SOA-Governance seitens Experten gehen derzeit noch deutlich auseinander, so dass akzeptierte SOA-Governance-Frameworks sowie analytische Unterstützung für Konformitätsüberprüfungen in Forschung und Industrie wenig vertreten sind.

Die vorliegende Arbeit leistet hierzu Beiträge in den zwei Themengebieten *SOA-Governance* und *Process Model Similarity* (Prozessmodellähnlichkeit). Zunächst wird u.a. ein neues Betriebsmodell für SOA-Governance entwickelt. In Ergänzung wird zweitens eine neue Analysetechnik für die Beurteilung von Prozesskonformität entwickelt.

Als Teil des ersten Beitrags wird eine Strukturanalyse von Ansätzen und Auffassungen von SOA-Governance durchgeführt, welche in der Beratungsindustrie, an Hochschulen und von Softwareherstellern verfasst wurden. Hauptergebnis ist die Identifizierung von 10 zentralen Komponenten von SOA-Governance. Basierend auf den Analyseergebnissen werden ein allgemeines Betriebsmodell, ein konsolidierter Service-Lebenszyklus für SOA-Governance sowie eine umfassende Definition entwickelt.

Im zweiten Beitrag wird, in Ergänzung des Betriebsmodells, eine Technik zum Vergleich von Referenzprozessmodellen und Modellen realisierter Prozesse, sowie deren Auffindung entwickelt (*related cluster pairs*). Der Vergleich beruht auf struktureller Dekomposition ausgehend von ähnlichen Prozessregionen. Die durchgeführte Evaluation attestiert diesem Ansatz eine hohe Genauigkeit, welches auf eine effektive Arbeitsunterstützung der manuellen Prozesskonformitätsbewertung hinweist. Zusätzlich wird ein neues Maß für Prozessähnlichkeit entwickelt. Verglichen mit bestehenden Ansätzen zeigt

die Evaluation hier eine deutliche Leistungssteigerung. Über den originären Anwendungsbereich hinaus stellt dieser Ansatz einen generellen Beitrag zum Forschungsfeld der *Process Model Similarity* dar.

„Erfolg ist eine Chance, verpackt in harte Arbeit.“

– Gustav Knuth (1901-87)

DANKSAGUNGEN

HÄTTE ich mich während dieser Arbeit nicht ständig auf die Unterstützung lieber Menschen verlassen können, wäre sie nicht zu dieser Zeit und nicht in dieser Form entstanden.

In erster Linie gebührt meinem Erstbetreuer, Prof. Dr.-Ing. Ralf Steinmetz, ein herzliches Dankeschön, der mir die Möglichkeit zu dieser Arbeit gegeben hat und an seinem Fachgebiet ein professionelles Arbeitsumfeld mit Strukturen geschaffen hat, welche die jahrelange Arbeit an einem Forschungsthema und die Durchführung einer solchen Arbeit sehr gut unterstützen und fördern. Vielen Dank, lieber Ralf, dass Du mir diese Chance gegeben hast.

Zweitens möchte ich meinem Zweitgutachter, Prof. Dr. rer. nat. Paul Müller, danken. Ihre fachlichen Ratschläge und konstruktive Kritik sowie Ihre herzliche und offene Art des Gesprächs haben mich bei meinem Vorhaben in der entscheidenden Phase in die richtige Richtung vorangebracht.

Darüber hinaus möchte ich mich bei allen administrativ-technischen und wissenschaftlichen Mitarbeitern des Fachgebiets für die sehr gute Gruppenatmosphäre und Zusammenarbeit bedanken. Insbesondere möchte ich mich bei Euch, den Mitgliedern und Ehemaligen meiner Gruppe, Rainer Berbner, Nicolas Repp, Julian Eckert, Stefan Schulte, Lars Arne Turczyk, André Miede, Apostolos Papageorgiou, Dieter Schuller, Sebastian Zöller, Ulrich Lampe, Melanie Siebenhaar und Olga Wenge, herzlich für Eure Unterstützung in Form von stets konstruktiver Kritik, fruchtbaren Diskussionen, erfolgreicher Teamarbeit, sowie der von respektvollem gegenseitigen Umgang geprägten Gruppenatmosphäre bedanken, die zu einem sehr lockeren und produktiven Arbeitsumfeld geführt hat – und uns, wo nötig, erfolgreich gemeinsam an einem Strang hat ziehen lassen. Mein besonderer Dank gebührt Nicolas für die fruchtbaren Diskussionen zur Themenfindung und Stefan für seine positiv-kritische und umfassende Auseinandersetzung mit meinen Manuskripten. Weiter gebührt meinen zahlreichen Studenten Dank für ihre Unterstützung und ihren unermüdlichen Fleiß. Insbesondere möchte ich mich bei Melanie Siebenhaar, Kim David Hagedorn, Sascha Hombach und Marc Bachhuber bedanken. Darüber hinaus gilt mein herzlicher Dank all denjenigen, die Teile meiner Arbeit (oder gar alles) korrekturgelesen haben!

Vielen Dank meinen Eltern, Euch, Biggi und Manti, dafür, dass Ihr mir mein Studium und diese Arbeit überhaupt erst ermöglicht habt und für Eure ständige bedingungslose Unterstützung – sowie natürlich Euch, Ulrike, Mareike und Ursula!

In besonderer Weise möchte ich mich bei Euch, Elli und Luise, bedanken. Meinen allerherzlichsten Dank, Elli, für die Zeit die Du investiert hast, Deine konstruktive Kritik und Empathie, und Deine ständige Bereitschaft und Hingabe. Während der ganzen Zeit hast Du mich in sämtlichen Bereichen des (Zusammen-)Lebens selbstlos unterstützt, ob-

wohl wir während dieser Zeit unsere kleine Tochter bekamen, die unser Leben sowieso komplett umkrempelte!

Vielen Dank Dir, Luise, für die erfrischenden und aufbauenden Momente der Erheiterung durch Dein kleines goldiges Wesen. Vielen Dank für Deine wärmende Unterstützung während des Schreibens im Baby-Tragetuch am Schreibtisch – und dafür, Luise, dass Du mit dem Zähne-Bekommen bis nach der Abgabe dieser Arbeit gewartet hast. ;)

Darmstadt, im Frühjahr 2012

M.N.

CONTENTS

1	INTRODUCTION	1
1.1	Motivation	1
1.2	Contributions of this Thesis	3
1.3	Thesis Outline	6
2	BACKGROUND AND FUNDAMENTALS	7
2.1	Service-oriented Architectures	7
2.1.1	Service Characteristics	9
2.1.2	SOA Roles	11
2.1.3	SOA Challenges	12
2.2	Governance of IT Systems	15
2.2.1	IT Governance	16
2.2.2	SOA Governance	18
2.3	Business Processes	19
2.3.1	Business Processes and Workflows	19
2.3.2	Business Processes and Governance Processes	21
3	COMPONENT ANALYSIS OF SOA GOVERNANCE APPROACHES	25
3.1	Analysed Components	25
3.1.1	Governance Policies	26
3.1.2	Organisational Structure	29
3.1.3	Artefact Management	31
3.1.4	Roles and Responsibilities	32
3.1.5	Service Life Cycle	35
3.1.6	SOA Procedure Model	37
3.1.7	Strategic Aspects	39
3.1.8	Governance Processes (and Policy Enforcement)	41
3.1.9	Metrics	43
3.1.10	SOA Maturity Measurement	45
3.1.11	Further Aspects	46
3.2	A Definition for SOA Governance	47
3.3	Analysis Results – Summary and Conclusions	49
3.4	Conclusion	52
4	A SERVICE LIFE CYCLE AND AN OPERATIONAL MODEL FOR SOA GOVERNANCE	55
4.1	A Service Life Cycle for SOA Governance – Survey and Approach	55
4.1.1	Design Time	58
4.1.2	Runtime	60
4.1.3	Service Change or <i>Change Time</i>	62
4.1.4	Discussion and Conclusion	63
4.2	An Operational Model for SOA Governance	65
4.2.1	Best Practices and Categorised Policy Catalogue	66
4.2.2	Organisational Aspects and Governance Processes	67

4.2.3	Conformance Observation	68
4.2.4	Conclusion	70
4.3	Conclusion and Outlook	70
5	COMPARISON AND RETRIEVAL OF BUSINESS PROCESSES BASED ON RELATED CLUSTER PAIRS	75
5.1	Introduction	76
5.2	Related Work	77
5.2.1	Similarity Notions	77
5.2.2	Related Approaches	78
5.3	Basic Concepts and Definitions	82
5.3.1	Process Model Graphs and SESE Regions	82
5.3.2	Assignment Problem	85
5.3.3	Similarity	85
5.4	Node Label Similarity Measures	86
5.4.1	Word Preprocessing	86
5.4.2	String-based Node Label Similarity	87
5.4.3	Semantic Node Label Similarity	88
5.5	Related Cluster Pairs	90
5.5.1	Correspondences Determination (Step A)	91
5.5.2	Cluster Determination (Step B)	93
5.5.3	Determination of Related Cluster Pairs (Step C)	95
5.5.4	Computation of Process Model Similarity (Step D)	99
5.5.5	Computational Complexity and Limitations	100
5.5.6	Implementation of ProcSim.KOM	101
5.6	Evaluation	104
5.6.1	Process Model Comparison	104
5.6.2	Process Model Retrieval	108
5.6.3	Discussion	114
5.7	Conclusion	115
6	CONCLUSION AND OUTLOOK	119
6.1	Conclusions	119
6.2	Outlook and Future Work	122
	REFERENCES	125
	LIST OF FIGURES	147
	LIST OF TABLES	149
	NOMENCLATURE	151
A	FURTHER BASICS	153
A.1	Enterprise Architecture	153
A.2	Event-driven Process Chains (EPC)	154
B	DETAILED RESULTS OF THE COMPONENT ANALYSIS	157
B.1	Components – Selected Results in Detail	157

B.2	The SOA Forum Survey on SOA Governance	162
B.3	Definitions for SOA Governance – Details	162
B.4	Selected Approaches in Detail	166
B.5	Service Life Cycle Survey – Details	170
B.6	Components Analysis – Details	173
C	EVALUATION DETAILS	175
C.1	Preliminary Evaluation of Comparison Scenario	175
C.2	Test Case Details	176
C.3	Evaluation of the Comparison Scenario	179
C.4	Evaluation of the Retrieval Scenario	180
D	IMPLEMENTATION DETAILS	183
E	AUTHOR’S PUBLICATIONS	185
E.1	Main Publications	185
E.2	Further Publications	186
F	CURRICULUM VITÆ	189
G	ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG	193

INTRODUCTION

WITHIN this chapter, we discuss the motivation behind the thesis topics, explain the thesis contributions, and provide the general chapter organisation.

1.1 Motivation

Recent economic developments, such as globalisation and financial crises, as well as constantly changing market conditions and new competitive threats, present companies and their IT organisations with new challenges. The ability to react quickly and efficiently to environment changes while continuously adopting the business strategy to new conditions, is considered a crucial competitive factor today [71]. However, increasing flexibility requirements, such as the continuous adaptation of internal processes, often exceed the capabilities of IT departments and existing monolithic IT systems. In fact, only companies with the ability to exploit high IT system flexibility potentials are believed to prevail in the long term [12].

In the last decade, as a general means of dealing with these challenges, the paradigm of Service-oriented Computing (SOC) has emerged, laying the foundation for Service-oriented Architectures (SOA). According to a survey from 2009, flexibility increases are one of the major reasons for SOA implementations in companies [53]. By modelling business and IT functionality in appropriate scopes and encapsulating them in small software artefacts called *services*, SOAs provide the ability to flexibly compose applications and workflows. This way, business processes can be more efficiently mapped to the IT system and, compared to monolithic IT systems, changes cause manageable effort [55, 101, 143, 158]. Today, large organisations, in particular, rely on business processes that are supported by IT.

The role of IT, in recent decades, has been the duty to operate and support internal processes. Not only in times of financial crises, IT operation costs sum up to 80 to 90 percent of the complete IT budget [194]. In many cases, the remaining budget (supported by a corresponding business budget) is to be used for projects serving *compliance*, i.e., the adoption of legal requirements [159]. However, IT organisations are expected to address new business challenges in a proactive way [82]. With the prevailing perception of IT departments as merely being internal service providers and, additionally, small IT budgets, this is, however, hardly possible. Due to the abovementioned expectations based on the role of IT and the financial restrictions, limits for innovations are set low.

Since the second half of the last decade, when Weill and Ross [263] first proclaimed the relevance of *IT Governance* in this context, it has propagated a change to the role of IT within companies. From the IT Governance perspective, IT is considered a success factor, rather than an internal service provider of necessity [8, 24, 69]. IT Governance, in fact, considers IT a general foundation for – or, at least, one of the originators of – added value that makes a company's products competitive. Generally, IT Governance

is holistically targeted at keeping IT systems controllable, and at assuring *compliance* of internal processes with regulations such as norms or laws [87, 131]. Governance frameworks recommend guidelines and reference processes in order to steer and control IT systems and their organisation, and to assure the systems' compliance. A central goal is the improvement of business-IT alignment in companies, i.e., the improvement of mapping business processes to the underlying IT infrastructure, and *vice versa*. Given this perspective and the accompanying new role of IT, IT Governance endeavours to ensure that the challenges faced by IT departments when tackling innovative projects, such as proactively and flexibly adjusting to market changes, become manageable.

SOA systems, apart from providing a promising foundation for flexible IT infrastructures, expose special challenges. The system complexity generated by the increasing number of services imposes high demands on a company's ability to efficiently exploit its advantages. Current IT Governance approaches are not designed to address challenges of SOA systems such as cross-organisational cooperation based on services, service life cycle management, and the strong involvement of SOA in business organisation [27, 97, 164]. Hence, in recent years, diligent governance for SOA, *SOA Governance*, has been recognised as a major requirement for the successful adaptation and operation of an SOA, especially for large systems [17, 151]. Beyond IT Governance goals, the new discipline of SOA Governance claims to more effectively exploit the capabilities of SOA systems [138, 215, 248].

The need of a discipline SOA Governance, as well as its importance for the control and steering of SOA-based systems is indisputable among experts from academic research and industry. However, definitions of central aspects and identification of major components of SOA Governance vary widely and have received little contemplation in research efforts. Describing expectations, experiences, and requirements from their specific points of view, authors from IT consulting industry, from software vendors, and from academia maintain differing approaches to SOA Governance (e.g., [10, 198, 215]). However, a consistent and commonly accepted definition cannot be identified among the variety of different suggestions – none of those provided have, as yet, become widely accepted.

A further aspect to consider concerns the components, of which an SOA Governance approach is considered to be composed. Many authors neglect the heritage of IT Governance, i.e., the immediate link to IT and company strategy, rather considering, for example, the service life cycle [10, 271], or the integration of management software [25, 246, 261] as core governance aspects. Certainly, a subsidence slope (or tilt) concerning approaches and definitions of SOA Governance can be attested [161, 167].

For IT Governance, standardised frameworks have become accepted that provide reference processes, the so-called best practices (cf. [27, 86, 248]). In order to publicly document the realisation of norms (e.g., IT system security), or adherence to established control structures, companies have their IT systems certified by experts concerning these norms (e.g., ISO/IEC 17799) or frameworks (e.g., COBIT), respectively. These conformance assessments against reference processes are an important aspect of realising the control that governance approaches aim to provide.

However, performing the required conformance assessments is elaborate and resource-intensive even for experts. Though SOA Governance approaches mention corresponding assessment techniques (e.g., check list processing during service approval [233]), there has been little research concerning decision support systems that help companies to pre-

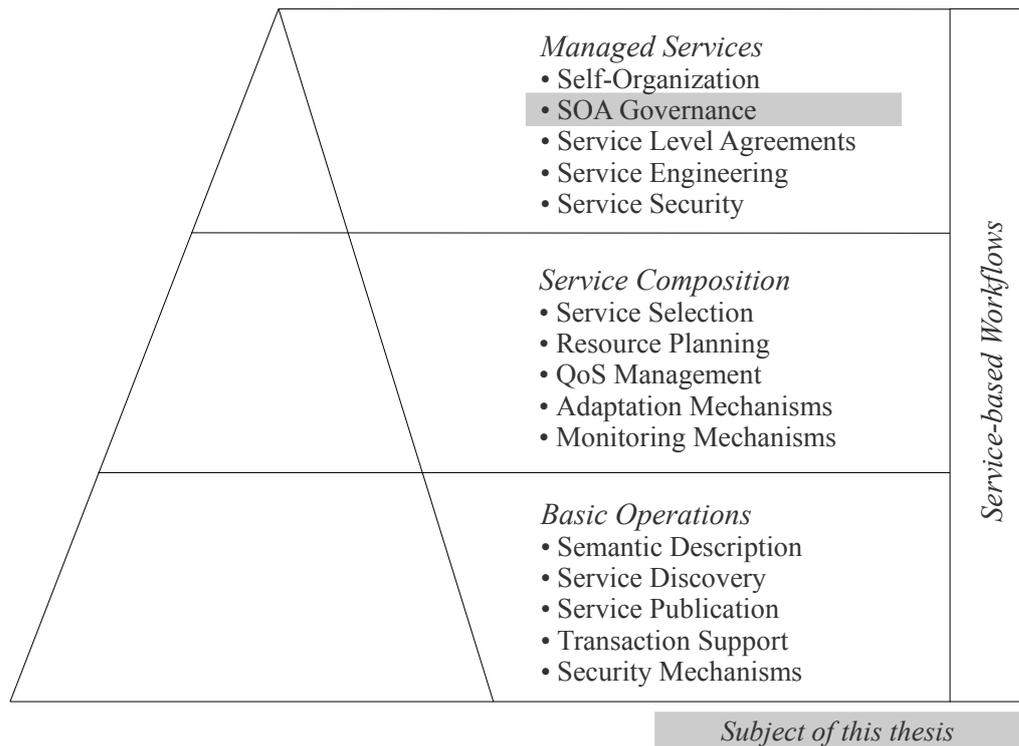


Figure 1.1: Research agenda for Service-oriented Computing
(adapted from [13, 186, 187])

pare those certifications efficiently. Specific techniques for the automation of *procedural* conformance assessment have not yet been intensively investigated in research either, for example, for techniques for comparison of reference processes with realised processes. A further challenge is the consideration of these regulations during the design of new processes. In order to ensure conformable processes, an efficient reference process retrieval approach for assistance in the design of new processes (authoring support) is important [197, 199]. The corresponding research problem, i.e., the efficient identification of relevant reference processes given a modelled flow of activities, is part of a common research field (business process retrieval) in Business Process Management (BPM) and has, apart from process authoring support, a variety of further applications (e.g., process model merging, avoidance of process variants, or process repository maintenance [48]).

1.2 Contributions of this Thesis

In 2006, Papazoglou et al. [186] proposed a general research agenda for SOC, consisting of three layers: managed services, service composition, and basic operations. According to this roadmap, the thesis at hand contributes to the research area of *SOA Governance*, or Governance for Service-oriented Architectures (cf. Fig. 1.1).

More specifically, this thesis provides contributions in two fields. As a broad foundation, a comprehensive investigation of the term and the discipline *SOA Governance* from several perspectives is performed. Building on the results, amongst others, a decision

support approach for process conformance assessment in the context of governance is developed. In addition to research in SOC, the developed approach covers research contributions in BPM, in particular in the area of business process retrieval. This work has been performed in the context of the large German government-funded research project Theseus/TEXO that developed a holistic concept and a comprehensive realisation of service marketplaces as part of the Internet of Services (IoS) [93, 96].

In detail, the contributions of this thesis cover the following.

- A) Fundamental analysis of the discipline SOA Governance
 - A1) Structural analysis of SOA Governance approaches, including the identification of major components and provision of a novel consolidating definition for this discipline
 - A2) Application of identified major components: development of a consolidated service life cycle and a generic operational model for SOA Governance, integrating the major components identified throughout the analysis
- B) Development of the novel concept *related cluster pairs* for the analysis of business process models and its application as decision support approach for process conformance assessment in the context of governance
 - B1) Development of an algorithm for the identification of differences of modelled business processes, based on both the structure of process models and string-based, semantic, and hybrid node label similarity metrics, as well as evaluation using an established collection of operational standard procedures
 - B2) Development of a novel process model similarity metric based on process model structure and several node label similarity metrics, and evaluation by comparison with a state-of-the-art metric, as well as with standard text-search engines

The contributions are outlined in detail in the following.

Although SOA Governance has been derived from IT Governance and adjusted to allow for the special challenges of SOA systems, understandings of this discipline are not uniform. In this thesis, we perform a comprehensive structural analysis of 22 major approaches to SOA Governance. The analysis considers recent perspectives from academia, software vendors, and the IT consulting industry. The key result of the analysis is the identification of 10 major components that are common to the majority of SOA Governance approaches. Based on the insights of the analysis, we provide a novel comprehensive definition for SOA Governance (contribution A1).

According to the general perception by experts, the discipline of governance is generally considered crucial for the successful operation of IT systems and, since business processes are largely operated by IT systems, for business success. However, for the introduction and operation of SOA Governance there are almost as many different notions as approaches. Many of these are biased by software vendor products. Further, life cycles for services as part of SOA Governance are vividly discussed in literature. Based on the analysis performed in A1 and an additional survey of service life cycles, we discuss and develop a consolidated service life cycle for SOA Governance. Also emerging from the results of the structural analysis above, we further provide an unbiased operational model for SOA Governance. The major advantage of these contributions is that

they are based on the analysis insights, i.e., they all allow for the major aspects of SOA Governance that are considered crucial by the expert authors (contribution A2).

In particular, our operational model for SOA Governance integrates a decision support approach to check conformance with regulations. The assessment of conformance is a central part of governance operation. An important regulation type is the enactment of reference processes (by so-called *governed* processes). Using a process modelling language, reference processes specify recommended, desired, or successfully proven activity flows, organised in reference process model repositories. As a decision support approach in the context of SOA Governance, and as part of our generic operational model, we suggest the novel concept of *related cluster pairs* to support (B1) the identification of relevant reference processes for a given flow of activities and (B2) the comparison of two process models and identification of their difference.

The decision support approach for the assessment of process conformance of business processes is based on our process model analysis concept called *related cluster pairs*. For the determination of correspondences between activities of process models, we make use of several string-based, semantic, and novel hybrid similarity measures. Exploiting these correspondences, the models are hierarchically decomposed into region pairs containing similar elements. A merging process aggregates sets of neighbouring similar regions in both models and forms larger similar regions. Based on these considerations, the technique identifies process model differences at a reasonable level of computational complexity. In order to test and demonstrate that it is an appropriate and valuable approach for process conformance checks, we use a test case of established standard procedures for evaluation (contribution B1).

The efficient and reliable identification of relevant reference process models in a process repository is a requirement for their inclusion in conformance checking procedures and for their consideration at process design time.

For this purpose, we develop a novel measure for process model similarity. Based on the determined regions, it can generally be applied for the retrieval of process models that are specified in a graph-based process notation. We realise this concept and both applications as a plug-in for an established process mining framework. We test and evidence the approach's practical applicability by using an established test case of operational standard procedures for the evaluation. Further, we compare it with related approaches and text search engines (contribution B2).

Concluding, this thesis provides a comprehensive investigation of the discipline SOA Governance, together with a novel process analysis approach for comparison and retrieval of business processes.

By performing a structural analysis of major Governance approaches, we identify common components of the typical SOA Governance approach. As application of the revealed insights, amongst others, we develop a generic operational model for SOA Governance that allows for all typical characteristics that have been identified.

Proceeding from process requirements formulated as reference processes (as part of the above mentioned model), we develop a process analysis approach in order to assess their achievement by realised processes. As central part of this approach, we present a novel process model similarity measure for the efficient comparison and retrieval of reference models. A comprehensive evaluation demonstrates its practical applicability as well as resulting improvements compared to a related process retrieval approach.

1.3 Thesis Outline

The remainder of this thesis is organised as follows.

Chapter 2, as a background, provides a fundamental overview of the basic concepts used throughout this thesis. We outline important terms such as Service-oriented Architectures, several types of governance, Compliance, and Business Processes. In particular, the challenges of SOA systems in conjunction with the discipline SOA Governance are discussed and compared with the “classical” IT Governance perspective and monolithic IT systems.

In Chapter 3, a structural components analysis of SOA Governance approaches is performed. 22 major approaches to this discipline authored by experts from academic research, software vendors, and IT consulting industry are investigated and structured. We discuss the resulting components in detail. Based on the results, a novel comprehensive definition for SOA Governance is elaborated that allows for the insights of the analysis.

In Chapter 4, major results from the structural analysis provide the basis for the development of a service life cycle, as well as of an operational model for SOA Governance. Based on an additional survey, a consolidated service life cycle for SOA Governance is elaborated. Both the service life cycle and the operational model regard the major aspects identified throughout the investigations. As one of the operational model’s major aspects, it integrates a decision support approach targeting the procedural assessment of SOA Governance operation.

In Chapter 5, we introduce the process model analysis concept of *related cluster pairs* that provides the basis for a decision support approach for process conformance assessment. We discuss related work from BPM and basic principles that are employed throughout the chapter. Further, the string-based, semantic, and hybrid node label similarity metrics, as well as special word preprocessing techniques used throughout the concept, are outlined. Based on related cluster pairs, we introduce an approach for business process comparison, and a novel process model similarity measure. Using structural process decomposition, process model differences are identified. We realise this concept and both applications as a plug-in for an established process mining framework. We test and evidence the practical applicability by performing several analytical evaluations, using an established test case of operational standard procedures. During the comprehensive evaluation of this approach, our results are compared with the results of an established text search engine, as well as with the results of state-of-the-art process model similarity measure from the field of BPM.

Chapter 6 concludes the thesis by summarising the contributions, outlining the conclusions, and providing an outlook of possible future work.

BACKGROUND AND FUNDAMENTALS

IN THIS thesis, we intensively investigate SOA Governance approaches and develop a decision support approach for conformance assessment of processes. This chapter gives an overview of the fundamental concepts used throughout this thesis.

After an introduction to Service-oriented Architectures and its characteristics in Section 2.1, we outline the development and relationships of governance approaches for IT systems in Section 2.2, and the notion of business processes and workflows in the context of governance in Section 2.3.

2.1 *Service-oriented Architectures*

In recent years, due to economic developments like globalisation and financial crises, as well as deregulation of markets, enterprises are increasingly forced to quickly react to changing environments and adapt their business processes continuously. In the last decades, IT landscapes within organisations have often grown heterogeneous and have developed towards a high complexity that is hardly manageable. A large amount of legacy systems, middleware platforms, programming languages, operating systems, and communication channels are the prevailing characteristics of such architectures [71]. According to Becker et al. [12], among international large companies, only those with a high IT flexibility and the ability to quickly adapt to new market conditions will survive in the long-term. In order to realise flexible business processes for changing business priorities, architectural support for the integration of legacy systems, as well as flexible mapping of business processes to IT systems (business-IT alignment) is required [119, 158].

In the last decades, however, due to changing business models, mergers, and acquisitions, many IT landscapes could not be realised as they were planned in advance, and rather organically grew into their current state over time. This usually resulted in a vertically organised architecture with a so-called pillar or silo structure. These are quite sophisticated and complex, and particularly suit the support of operational sequences in their domain [143]. A reason for these silos is the fact that many IT systems used to serve only a single department or business unit. Difficulties and even serious problems arise if this structure has to be modified significantly. Common side effects include data redundancy and multiple implementations of the same functionality in different places. This raised the well-known issue of integration, which has been challenging IT departments for decades [143, 158].

Building on Service-oriented Computing (SOC), *Service-oriented Architectures (SOA)* are often recommended to address the mentioned challenges. The term SOA has first been mentioned by Gartner analysts [224] in 1996. Since when the term was coined, SOA has evolved to become an established IT system (cf. recent surveys, e.g., [53]).

Generally, the SOA paradigm represents a high level concept for designing information architectures [119]. By the ANSI standard 1471-2000, the term *architecture*, in this context, is defined as follows.

<i>Business Benefits</i>	<i>Technical Benefits</i>
Increase of business agility	Efficient development
Reduced integration costs	Simplified maintenance
Better business alignment	Easier reuse
	Graceful evolution
	Incremental adaptation

Table 2.1: Potential business and technical benefits of an SOA [158]

“[An architecture is] the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution.” [173]

The SOA paradigm implements these requirements by defining the *service* as fundamental concept (cf. Sect. 2.1.1). This way, it provides the capability of both (i) integrating heterogeneous systems, as well as (ii) achieving a high agility of business processes and their underlying IT [101, 119, 158, 187]. This high agility of processes implies an intensification of communication between business and IT departments. Due to this integrative nature, benefits of the application of SOA can in fact be found both on the business and the technical side (cf. Tab. 2.1).

Within standard literature, several different definitions for SOA have been formulated. While some rather focus on technical characteristics [101, 119, 133, 143], others additionally consider business aspects [17, 55, 185].

Melzer et al. [143], for example, emphasise the usage of open standards, programming language independence, and the fact that parts of the system (applications or methods) may be incompatible without impact on its operation. Erl [55] expects of a “contemporary” SOA system to be easy to extend and be composed of independent services that are offered by different providers. For him, interoperability of services and their reuse factor are of particular importance. Erl generally considers SOA to be the abstraction layer between business logic and the deployed technology. The definition by Josuttis [101] explicitly considers definitions by further authors. As a result, though, the author emphasises the technical aspect of SOA: “SOA is an architectural paradigm for dealing with business processes distributed over a large landscape of existing and new heterogeneous systems that are under the control of different owners.”

For their definition, Bieberstein et al. discussed versions from both business and technical perspectives and related them to results of a survey of business executives [17]. We adopt their definition in this thesis, which dates from 2006 and hence is one of the early ones:

“A service-oriented architecture is a framework for integrating business processes and supporting IT infrastructure as secure, standardized components – services – that can be reused and combined to *address changing business priorities.*” [17]

Especially the last four words express high demands – and precisely address the motivation for SOA we gave during the introduction.

As the *service* is a common concept among distributed IT systems [218], the SOA paradigm shares commonalities with service-related technologies such as the Future Internet [59, 155], cloud computing [5, 31, 72], wireless sensor networks [105, 190], and also

peer-to-peer (P2P) systems [244]. For investigations of the SOA paradigm and specific problems such as service discovery [225, 226], general service monitoring [148, 202], QoS optimisation [51, 52], QoS optimisation in complex workflows [222, 223], process reliability [221], service management and governance [102, 147, 183], mobile services [182, 184], and even wireless sensor networks [191, 275], service execution platforms have been provided (such as WSQoSX [13, 14] and VENICE [77, 79, 115]). Investigations of service-orientation towards the integration with the former technologies are constantly being made (e.g., P2P [78, 80, 206] or Cloud Computing [121, 231]).

In the context of SOA, *Enterprise Architectures* (EA) are often mentioned. An EA consists of several smaller architectures, and represents the entire enterprise organisation [249]. Further, an EA captures the organising logic in technical choices and policies, while defining data and infrastructure as stable platform to support quickly changing applications [263]. The Open Group [249], who authored the most popular established EA framework TOGAF (The Open Group Architecture Framework), identifies three practice domains: *Business Architecture*, *Information Systems Architecture*, and *Technology Architecture*, where the second is broken down into *Information Architecture* and *Applications Architecture*. As a main objective of the development of an EA, they specify

“providing the fundamental technology and process structure for an IT strategy. This in turn makes IT a responsive asset for a successful modern business strategy.”¹

As EA, in the course of this thesis, we understand a generic reference model for enterprise organisation, in particular addressing business structure, technology, and information systems, and their complex interrelations. In this context, it can be claimed that the paradigm of SOA allows for all these aspects, and hence can represent an EA. According to experiences from practitioners, for adjustment reasons, SOA is often introduced in companies in a scope of an IT project (in contrast to IT system), often called the “SOA initiative” [1, 17, 27, 268]. In these cases it is unsure, whether the complete development to finally become the EA is finished. In this thesis, an SOA system is considered a potential candidate for an EA.

2.1.1 Service Characteristics

SOA emerged from the software engineering principle of *separation of concerns* [55]. The major concept to implement this principle is the *service*.

In order to define the architectural part of SOA, Channabasavaiah et al. [33, 34] make use of the following principles:

- All functions are defined *as services*. This includes technical, infrastructure functions as well as business functions.
- All services are *independent* and can be used without paying attention to the actual implementation.
- All services are *invokable*. Services can be accessed by an interface without any knowledge of its location.

¹ TOGAF 8 Web page, cf. <http://pubs.opengroup.org/architecture/togaf8-doc/arch/toc.html>, last accessed 2011-08-11

A service, according to Krafzig et al., is the “technological representation of business functionality”, and, technically, can be understood as a “remotely accessible, self-contained application module” [119]. In combination, Josuttis [101] defines

“A service is the IT realization of some self-contained business functionality.”

Using services as building blocks, business processes can be composed from them, abstracting from the underlying (monolithic) applications and allowing for compositions even across organisational boundaries.

As a consensus of most authors (e.g., [55, 101, 119, 143, 158]), the following are major characteristics of a service:

LOOSE COUPLING of services intends to minimise dependencies between services. This especially supports fault tolerance and flexibility. Services interact while still staying independent of one another. The minimisation of dependencies is also referred to as services being self-contained [101].

SERVICE REUSABILITY is a central and general goal of service-orientation. The specifications of design standards target at making any service reusable, independently of immediate reuse requirements. The goal is to reduce future development efforts when services are used in a different than the first intended context. A crucial factor when sizing service functionalities, is service granularity: while small service functionality can improve the reuse rate, it can decrease overall efficiency. The usage of open standards, supporting interoperability, is also considered a central aspect.

RETRIEVABILITY of services concerns their description and appropriate search functionalities. Services expose metadata artefacts that describe their interfaces and functionality using suited technology (e.g., WSDL). This is especially important in large systems, and when using more than one service repository.

SERVICE AUTONOMY refers to the service’s complete control within its execution scope. At the time of execution, the service has exclusive control of the used logic. Autonomy is an important aspect when grouping operations and distributing functionality on services, i.e., deciding on service granularity.

ABSTRACTION. Services realise the software engineering principle of *Information Hiding*. Acting as “black boxes”, services hide *interna*. They completely abstract from their realised inner logic such as implementation details or realised security measures. Further, there are no limitations concerning the implementation scope (i.e., the realisation of a simple technical functionality vs. the combination of functions provided by several IT systems).

STATELESSNESS of services refers to the minimisation of necessary storage of state information. This way, as a simple HTML website, services do not rely on keeping track of invocations and connection-depending data exchange. This characteristic in particular supports reusability and scalability.

SERVICE COMPOSABILITY assures that services can be combined to form composite services. It is a special form of service reuse and enables service orchestration, where a central instance (e.g., a service, a BPEL engine, a WfMS) controls the execution of

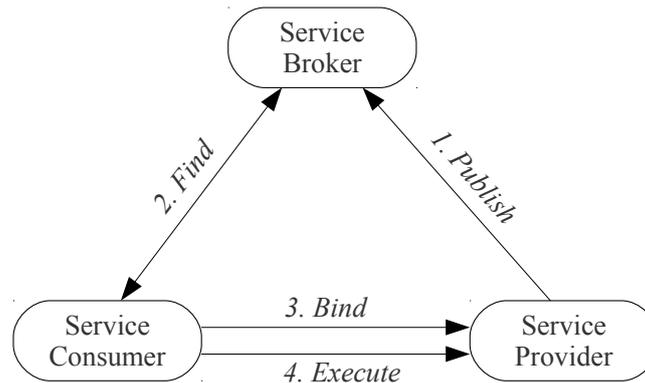


Figure 2.1: SOA roles (e.g. [65, 125, 143, 187])

several, combined services. In particular for service composability, service operations need to be standardised.

SERVICE CONTRACTS describe the interaction and the conditions of information exchange between one service and the invoking client or other services. The description covers the service interfaces, all service operations including all possible input and output messages, as well as all properties of the services and their operations. Using semantic information, services can be described particularly accurate [225].

These principles are considered the basis when targeting the ambitious goal of separating interfaces from their implementations [158]. While an SOA system is shaped by these principles, it is coined by its usage by at least three stakeholders, the typical *SOA roles*, which are introduced in the following.

2.1.2 SOA Roles

As immediate users of an SOA system, three stakeholder roles have established (e.g., [16, 55, 65, 83, 101, 119, 125, 143, 158, 187], cf. Fig. 2.1):

- the service broker,
- the service provider, and
- the service consumer.

The *service broker* is also called *service intermediary*. He manages a service catalogue containing published services and their descriptions. Services are described such that they can be retrieved successfully (e.g., using semantic Web services [225]). He provides interfaces for service publication (to service providers) and for service retrieval (to service consumers). As part of the SOC research agenda (cf. Fig. 1.1), at this stakeholder, several optimisation problems concerning service request handling and Quality of Service are object to investigation [51].

The *service provider* offers one or multiple services. Before provision, the role registers services with the service broker, who publishes or publicly announces it (“publish”

in Fig. 2.1). Using the provided service description, interested parties (i.e., service consumers) can search and retrieve the service. A service provider can be a human, a software agent, or organisation. In peer-to-peer (P2P) systems [244], for example, software agents provide and consume services for data exchange, for example, multimedia content [106, 193, 242, 243].

The *service consumer* is in need of one or more services. Given a functionality description, he uses the service broker's platform to search and find the service providing the required functionality ("find" in Fig. 2.1). Once a service is identified, the consumer is interested in integrating it in his operational environment ("bind" in Fig. 2.1). He starts negotiations with the provider concerning service usage conditions. Once a service level agreement (SLA) is mutually agreed on, the service consumer starts using it ("execute" in Fig. 2.1). A service provider can also be service consumer, and *vice versa*.

In case of service marketplaces in the context of the IoS (e.g., as envisioned by the German government-funded research project Theseus/TEXO² the classical number of three stakeholders is considered insufficient (e.g., [93, 172, 188]). In fact, additionally to these, further stakeholders are important such as the marketplace host, the service producer, and the service innovator.

In this scenario, the service provider is not obliged to implement services by its own and can outsource this to the *service producer*. Moreover, services do not need to be thought of by providers. In fact, the role of *service innovator* specialises in investigating market situations, creativity techniques and methods, and hence, the creation of innovative services (e.g., the Idea Ontology³ [208, 209]). The *marketplace host* is in charge of running the marketplace platform, managing one or more service repositories, organising search requests, establishing marketplace-wide standards concerning (cross-organisational) security [146], and coping with the internationalisation and automation of legal aspects [196]. Beyond this, the IoS bears many additional technical challenges, for example, consistent service description [171], distributed service monitoring [200, 205] and service contracting [239], and distributed service hosting [238].

2.1.3 SOA Challenges

Service-oriented systems promise to realise agile implementations of business structures that are able to flexibly adjust to changing environments. Prominent "promises" by software vendors in this context are, for example, increased code reuse, reduced integration expense, better security, greater business agility, and a shorter realisation time [267].

Not necessarily, however, does an SOA expose a lower complexity as the legacy systems it is to complement or capsule. When measuring system complexity by component count and their interrelations, SOA contributes to an *increasing* system complexity [216].

Thus, companies introducing and operating an SOA-based system must be aware of typical SOA-related challenges and need to address them (e.g., by governance approaches) in order to successfully realise the mentioned "promises". The major ones are:

- An increased number of software artefacts
- Unrealised service reuse

² <http://www.internet-of-services.com/> and <http://theseus-programm.de/en/>, last accessed 2011-08-10

³ <http://www.ideaontology.org/>, last accessed 2011-08-10

- Increasing maintenance cost

These are the fundamental new challenges and risks that the management of an SOA system has to cope with (cf. e.g., [97, 164, 167, 215, 216]).

INCREASED NUMBER OF SOFTWARE ARTEFACTS. Applications based on SOA usually consist of a large number of single and composite services. This means more flexibility on the one hand, while it requires more administrative efforts on the other. Services as the smallest parts of SOA systems provide the appropriate means to enable an enterprise architecture to flexibly adjust to changing business processes on the one hand. On the other, they implicitly increase the system complexity. Although SOA reduces the heterogeneity of enterprise architectures, heterogeneity is replaced by the multitude of services. When compared with monolithic systems, the *increasing number of software artefacts* is the most frequently mentioned SOA-related challenge. It is the basis for most other mentioned risks of such a system.

Without appropriate control structures, the SOA-inherent complexity can lead to structures whose maintenance might be similarly extensive as the one of the legacy application landscape that is to be replaced by SOA [103, 114, 215]⁴. The homogenisation and control of this emerging complexity is the central challenge of an SOA system.

As complexity increases, Schelp et al. [216] argue that this implies a lower degree of service reuse: the large service count causes low system lucidity, which causes a higher search effort causing an increasing number of cancelled search requests, decreasing the reuse rate due to relevant services that were not found. Operating more than one repository is considered a further indicator for this effect. Beyond this, it is more likely that business and technical rules or guidelines are violated [267].

Further, in order to realise the largest possible benefit in terms of business strategy, business and IT departments (operation and management) must elaborate precise agreements on the development and operation of services (“strategic alignment”) [97]. A larger service number demands the intensification of communication between business and IT departments.

Summarising, it is increasingly difficult to realise service reuse, to manage complexity, and, generally, to operate an efficient, and cost-effective service administration. All these issues must be addressed more carefully and diligently than in case of monolithic systems, and solved in a sustainable way.

UNREALISED SERVICE REUSE. Developing reusable services is considerably more challenging than the development of traditional software. Functionality must be encapsulated in a manner that services can be reused in other contexts. This produces initially higher costs that are usually compensated by realising reuse. Commonly, according reusable services are considered to be more than three times more expensive than services not designed reusable [215], and bear at least 50% higher development costs [138]. In case these reusability opportunities are not realised, as, for example, a service is coupled too tightly to the architecture of an application, or an existing service is not known to the developers of a new application, an

⁴ or, in other words, SOA becomes “a mess waiting to happen” [114]

SOA can be permanently more expensive than a monolithic system. The realisation and improvement of the service reuse rate depends on two major aspects: *service granularity* and *variant management*.

The decision on *service granularity* is an important aspect during service design and a requirement for service reusability. It is, basically, the compromise between the number of services and the functional range of a single service. The major challenge is to define functionality boundaries for services. Functionality requirements must be divided into parts that can best be realised by a single service. Largely, the relationship between service number and functional range actually varies from case to case [27, 103, 109, 164, 216]. Choosing a fine grained approach, a number of services can result that probably cannot be managed reasonably. Defining coarse grained services, these become comprehensive in function, similar to monolithic applications, whose reuse possibilities in new contexts are limited. Identifying the right granularity extremely depends on the context in the single case and often requires experienced service architects. Decisions on service granularity can be supported by service blueprints, service definition scope concepts that apply in the definition phase, and guidelines, that structure typical service tasks by defining task levels. Measures like these, individually aligned to specific SOA requirements, help with globally controlling and keeping service granularity in a manageable range [27, 103, 164] (cf. Sect. 3.1.7).

An advantage of service-oriented systems is the possibility of creating *service variants*. A service can be changed without the necessity to change all depending services. This allows the fast realisation of comprehensive functionality alterations. It is an elegant solution, avoiding among others the renegotiation of service usage terms (e.g., SLAs) in case the externally visible functionality has not changed. However, service variants reduce the overall service reuse rate [216]. The new service is only used by the new constellation that requested the change, while the old version is continuously used by prior configurations. In practice, normally, there are no efforts to completely replace the old service and retire it [70].

Summarising, the decision on service granularity bears the risk to choose disadvantageous service scopes, especially during SOA introduction or when experiences with SOA and services are yet scarce. The operation of variants of the same service in parallel has similar consequences: negative effects on the overall system complexity, on cost, and on the number of interrelations between services.

INCREASING MAINTENANCE COSTS. All these arguments have at least one common consequence. The increased number of services, the opportunity of service variants, the need for precise responsibility definition and increased communication activity between departments concerning service handling, and increasing system complexity are all originators for maintenance cost increase. According to Schelp et al. [216], increased search efforts, more complex service tests, and increased coordination efforts are hardly avoidable. Any SOA management approach must be ready to deal with this dilemma.

In fact, SOAs implicate a number of challenges for IT management [1, 3, 18, 97, 138, 164, 167, 215, 216] that often cause SOA introductions to fail [241]. Advantages of SOA are bound to new challenges that must be addressed by initiatives for the introduction of

this service-orientation paradigm. The discipline targeting this is called *SOA Governance* (cf. Sect. 2.2.2).

2.2 Governance of IT Systems

Generally, the term *governance*, as used in political governance, Corporate Governance, IT Governance, or SOA Governance, refers to the successful governing of organisations or projects. According to the Merriam-Webster Dictionary ⁵, it means

“... the way that a city, company, etc., is controlled by the people who run it”.

Basically, the term is borrowed from politics and has an analogous meaning in the context of IT systems. Empowered institutions interact with the purpose of regulating a large complex heterogeneous system, for example, a state. The main target is to keep it controllable by introducing and enforcing policies or laws (structures, rights, behavioural guidelines, standards, ...) – the system is *governed*. In the control of IT systems, parallels emerge with this concept. First, the objects to be regulated or governed are part of the IT system, i.e., the SOA. Second, the actual laws ensuring the compliance of the system map to *policies*. The third element, the observation and control of adherence to laws, i.e., the “police”, are compliance observation mechanisms.

At the beginning of this century, mismanagement in global companies such as *Enron* and *WorldCom* harmed shareholders interests. Their behaviour (and poor management) encouraged the establishment of *Corporate Governance*. A number of regulatory requirements, corresponding to legal or voluntary regulations, have their seeds in Corporate Governance. With legal regulations such as the Sarbanes Oxley Act, Basel II, or the German “Gesetz zur Kontrolle und Transparenz im Unternehmensbereich (KonTraG)” transparency rules and requirements for internal control systems have been accomplished in order to establish new confidence to company management.

Generally, Corporate Governance is the judicial and factual regulation framework for the leadership and controlling of an enterprise. For the operational requirements of a good enterprise leadership, four general resorts can be derived that governance rules are applied to [256]:

- Regulations for the determination of the superior business objectives,
- Regulations for the structures, processes, and people on corporate management level that are incorporated to achieve these objectives,
- Regulations for periodical evaluations of leadership activities, as well as
- Regulations for the proactive corporate communication.

According to the “Principles for Corporate Governance” by the Organisation for Economic Cooperation and Development (OECD) [179], Corporate Governance is defines as

“providing the structure for determining organisational objectives and monitoring performance to ensure that objectives are attained”.

⁵ <http://www.learnersdictionary.com/search/governance>, last accessed 2011-07-25

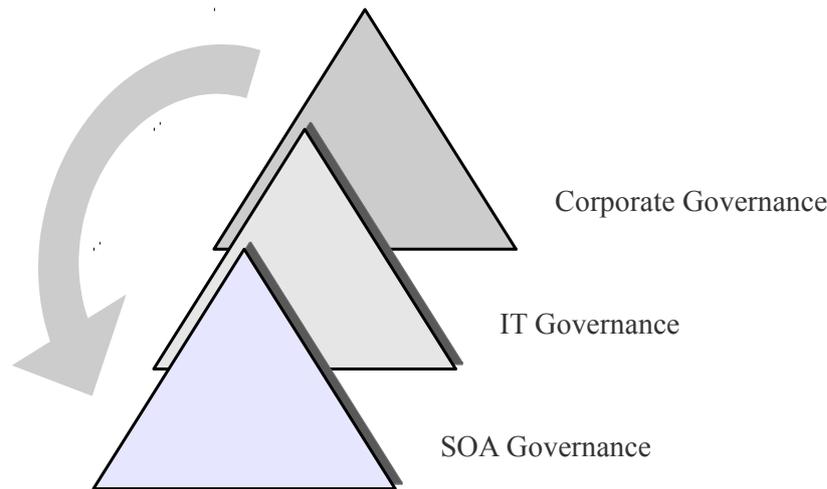


Figure 2.2: Embedding of SOA Governance in the enterprise context [215]

Many countries have their own regulation of Corporate Governance. In companies, it is mostly organised as supervisory board protecting shareholders' and other stakeholders' rights [263].

Corporate Governance represents the basis for *IT Governance*, which itself is the point of origin for *SOA Governance* (cf. Fig. 2.2). All these are outlined in the following subsections (Sect. 2.2.1 and 2.2.2, respectively).

2.2.1 *IT Governance*

The approach for re-establishing confidence to company management (e.g., transparency rules and requirements for internal control systems) by Corporate Governance has direct and indirect significant implications on the IT operation. This makes IT Governance a *pendent* of Corporate Governance [97].

One of the first definitions for IT Governance has been provided in 2004 by Weill and Ross [263]:

“IT governance: Specifying the decision rights and accountability framework to encourage desirable behavior in the use of IT” [263]

While this early definition has a focus on decision rights and behaviour, following the Information Systems Audit and Control Association (ISACA) and the IT Governance Institute (founded by ISACA), IT Governance is

“... the responsibility of executives and the board of directors, and consists of the leadership, organisational structures and processes that ensure that the enterprise's IT sustains and extends the organisation's strategies and objectives.” [86]

Additionally to ensuring conformance to necessary rules, companies expect from IT Governance cost reduction at IT services, productivity increase by more efficient processes, and the reduction of operational risks [39].

Knolmayer and Loosli [113] distinguish an *introversive* and an *extroversive* perspective of IT Governance. According to the extroversive perspective, IT Governance is a supporting instrument of Corporate Governance. Its main task is to advise IT management how to implement compliance requirements. The introversive perspective exceeds the pure alignment to the compliance specifications. The focus is on the economic optimisation of the performance of the IT system and the IT organisation, as well as on the required adjustment of decision, design, and implementation processes of IT management [113].

For the adoption and introduction of IT Governance practices, rules and standards, as well as frameworks of collected techniques and management structures that proved of value in business practice, have established. Among the most accepted ones are the *IT Infrastructure Library (ITIL)* [176], the control models of the *Committee of Sponsoring Organizations (COSO)* [174], and the *Control Objectives for Information and Related Technology (COBIT)* [86]. Beside these, there are numerous further frameworks, standards, and norms that support the achievement of the different IT Governance goals (e.g., ISO norms). Basically, each of these frameworks focuses on a different aspect of a company's IT. While ITIL, for example, mainly deals with IT process definition [176], the ISO 17799 standard primarily targets security management [84]. COBIT is a high level Governance and control framework, more tightly aligned with the business objectives of the organisation than with operational issues [86]. It has become a *de facto* standard for IT control globally, and its implementation and application increasingly gains interest among companies. For an overview of major frameworks as well as a list of regulations, refer to Johannsen and Goeken [97].

These frameworks provide an orientation for companies concerning the appropriateness of management techniques for the achievement of successful IT Governance (e.g., the formulation of IT strategy), as well as concerning helpful tools and methods to support its implementation. IT departments, having successfully introduced an IT Governance approach using these methods, are able to have themselves certified according to common standards (e.g., ISO/IEC 20000) [85]) to ensure that internal processes fulfil the requirements of a standard. ISO/IEC 20000-1:2005, for example, is a standard of the *International Organization for Standardization* that defines requirements of professional IT service management for service providers.

This way, IT management recommends the own IT department as reliable partner for the provision of services of qualitatively high value.

Compliance

The term *Compliance* is used in multiple contexts. Generally, compliance concerning *regulations* (regulatory approach) and *standardisation* (standardisation approach) can be distinguished (as outlined in the following). They are administrated by *compliance management* ([104], cf. Fig. 2.3):

“Compliance management is a broad term covering all activities and methods to ensure that a company follows all policies required by an external or internal regulation.” [110]

The *regulatory* approach targets the adherence to legal or enterprise-specific requirements (e.g., corporate rules and policies) [104]. It comprises the legal regulations with mandatory character (e.g., SOX, Basel II), also covering Corporate Governance (cf. Fig. 2.3). In this case,

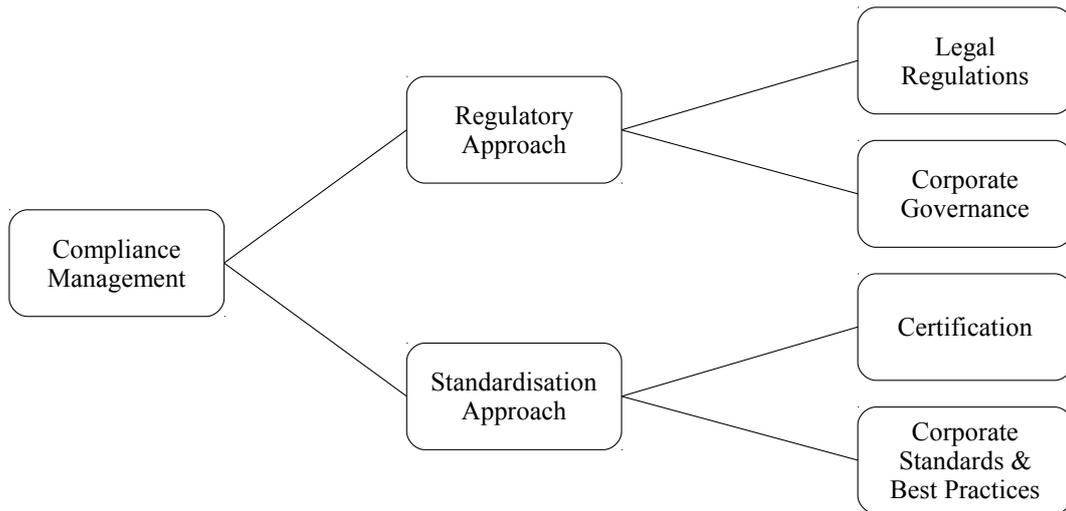


Figure 2.3: Elements of Compliance Management [104]

“[...] if a company follows all guidance defined in [...] a regulation document, the company is said to be in compliance with the given regulations.” [110]

This aspect of compliance aims at increasing transparency for shareholders and stakeholders. In the scope of Corporate Governance, it aims at generally renewing confidence to corporate management [60, 97].

The *standardisation* approach focuses on the certifications and the usage of best practices, where the latter refer to methods and procedures that have proven useful by experience [104]. In this context, compliance is defined as follows.

“Compliance essentially means ensuring that business processes, operations and practice are in accordance with a prescribed [...] set of norms.” [131]

Certification, in this context, refers to the approval of properties according to a requirements catalogue. It is normally performed by an auditor. The procedure of assigning a certification is called *audit*, referring to the assessment whether requirements have properly and effectively been implemented [60].

2.2.2 SOA Governance

Over the past years, Governance for SOAs (SOA Governance) has gained increasing importance. Even though governance is considered crucial to successful long-time operation and control of an SOA, due to the complexity and heterogeneity of SOA systems, a consensus concerning a uniform approach has not been achieved yet. The main task of SOA Governance is to define and introduce company-wide policies for the adoption and operation of an SOA, as well as to introduce mechanisms controlling their enforcement [57, 103, 109, 136, 215, 267]. It targets the achievement of predefined SOA goals. It elaborates guidelines and rules that need to be adopted and realised by the affected management processes. Up to now, there is a multitude of holistic approaches dealing with

SOA Governance – however, most of them have different scopes, use different methods, and address different goals.

SOA Governance topics strongly relate to IT Governance – hence SOA Governance approaches are often comparable to those in the area of IT Governance. As, scientifically, the topic of SOA Governance has not been extensively investigated yet, many software companies introduce their own definitions in whitepapers, often driven by own market interests [57, 109]. Concerning their perspectives on this discipline, Allen [3] attests a “narrow view”. This lead to a number of different approaches so far.

Mostly, SOA Governance is described as “little brother” of IT Governance. Often, an extension of existing IT Governance approaches is proposed that in particular allows for the specific requirements of an SOA [271]. In the discussion on the classification of SOA Governance in the scope of Corporate Governance, most authors argument that it is a *subset* [109, 261], *extension* [81, 271], or *specialisation* (cf. Fig. 2.2, [215]) of IT Governance. According to our notion, SOA Governance is a specialisation or subset, combined with an extension of IT Governance. Although SOA Governance addresses special SOA-related issues, such as service ownership or service deployment (*specialisation*), it is still a part of the IT in an enterprise (*subset*). Hence, IT Governance mechanisms are considered to apply to an SOA, as well [114, 136]. However, aspects like cross-organisational collaboration (service deployment) or the concept of service marketplaces have not yet been considered in IT Governance approaches (*extension*).

Basically, one of the objectives of this thesis is the analysis of existing approaches, as well as the analysis and clarification of the understanding of SOA Governance. The first identifies building blocks, a typical SOA Governance approach is composed of (cf. Chapter 3). A definition of SOA Governance is derived and outlined in Section 3.2.

2.3 Business Processes

SOA bears the opportunity to partly automate the execution of business processes and workflows. To elucidate the term *business process* and its relation to the term *workflow*, major definitions of these terms are discussed in the following.

2.3.1 Business Processes and Workflows

Weske [264] provides the following definition:

“A business process consists of a set of activities that are performed in coordination in an organizational and technical environment. These activities jointly realize a business goal. Each business process is enacted by a single organization, but it may interact with business processes performed by other organizations.” [264]

The definition points out that a business process consists of a *set of coordinated activities* (e.g., atomic tasks), and aims at achieving a *business goal*. These two major aspects are also part of the definition by Scheer [213], who considers a business process a related sequence of activities in an enterprise for the creation of goods and services. In the definition by Davenport [40], business processes are considered “a set of logically related tasks performed to achieve a defined business outcome”. Again, the constellation and goal are emphasised. The activity relationship is detailed by Davenport [41]:

“A process is thus a specific ordering of work activities across time and place, with a beginning, and end, and clearly identified inputs and outputs: *a structure for action.*” [41]

Österle [180], in a further definition, assigns tasks to organisational units in order to fulfil process management objectives. He extends the definition by introducing IT-based applications to support the execution of these tasks. A process is at the same time producer and consumer of work, while linking the business strategy with the information system [180]. This definition introduces resources (i.e., organisational units) as part of a business process, and specifies IT as a means of execution.

Generally, a business process can be implemented by organisational policies or by the use of a software system [264]. The part of a business process which is automated by the use of IT systems is called *workflow* [36]. According to the Workflow Management Coalition [36], a workflow is defined as “[...] the computerized facilitation or automation of a business process, in whole or part.” They further specify:

“A workflow is the automation of a business process, in whole or part, during which documents, information, or tasks are passed from one participant to another, according to a set of predefined rules.” [37]

The main difference between business processes and workflows, hence, is their execution context. The central concern within this thesis will be business processes.

The term *Business Process Management (BPM)* refers to the general handling of business processes in various aspects. It comprises

“... all concepts, methods, and techniques to support the design, administration, configuration, enactment, and analysis of business processes.” [264]

BPM is considered “a discipline and a technology” [175]. It aims at enhancing business agility, control, and accountability. It targets streamlining internal as well as external business processes, the elimination of redundancies, and automation increase [232]. In order to design, administrate, configure, enact, and analyse business processes, they must be appropriately represented. Generally, business processes are represented by *business process models*.

“A business process model consists of a set of activity models and execution constraints between them. A business process instance represents a concrete case in the operational business of a company, consisting of activity instances. Each process model acts as a blueprint for a set of business process instances, and each activity model acts as a blueprint for a set of activity instances.” [264]

According to this hierarchically structured and precise definition, a process instance refers to a concrete procedural situation in a company. A process model is the representation (“blueprint”) of a set of these instances. Both process models and process instances are detailed by activity models and activity instances, respectively. Optimally, a process model describes all permutations of a specific procedural situation. In other words, a business process model outlines the structure of a business process in the real world [124]. It determines all possible paths of the business process and defines the activity ordering [88, 124].

Business process models are described using business process notations, or business process description languages. The most common representatives are UML Activity Diagrams (UML 2.0) [178], Event-Driven Process Chains (EPC) [108, 111], the Business Process Modelling Notation (BPMN) [177], and Petri Nets [62]. All of these provide graphical notations, based on graphs. In this thesis, we propose a process analysis concept that can be applied to most graph-based process model representations such as EPC, UML Activity Diagrams, and BPMN.

A common research problem in BPM is the efficient retrieval of business process models, i.e., the efficient identification of relevant reference processes given a modelled flow of activities. This problem is tackled by the thesis at hand and has a variety of applications, such as process authoring support, process model merging, avoidance of process variants, or process repository maintenance (cf., e.g., [46, 48, 264]).

In the course of this thesis, we use business processes in the context of governance in particular.

2.3.2 Business Processes and Governance Processes

In BPM, generally, several *business processes levels* can be distinguished. From SOA Governance perspective, two types of processes are recognised. In this section, we discuss the assignment of these process types.

Weske [264] identifies five different process classification levels, ranging from implemented business processes (workflow level) to business strategy (business management level). Each level determines the next level in terms of a breakdown structure, and contributes to the realisation of the next higher level (cf. Fig. 2.4).

- *Business Strategy* refers to the determination of strategic business concepts, that form the basis for a long-term strategy ensuring sustainable market advantages (example given is “domain-dependent product cost leadership”).
- *Goals* represents the next hierarchical level, the breakdown of strategy into a goal hierarchy. In the governance (and company-level business management) context, these two process layers refer to the strategic management as part of Corporate Governance, as well as the setup of IT goals and strategic business-IT alignment at the level of IT Governance and SOA Governance [86, 248].
- *Organisational Business Processes* describe high-level procedures covering their inputs, outputs, expected results, and their relationships to other organisational business processes. For the description of processes, up to this level, simple *text notation* is used, supported by flow charts. Weske [264] characterises this description type as “informal or semiformal technique”. In the context of governance, IT Governance frameworks such as COBIT ([86]), and SOA Governance Frameworks such as the framework by The Open Group [248] map to this level. They textually describe best practices at a high level (“informal”), using structuring, “semiformal” aspects, for example, processes, artefacts, inputs, and roles.
- Several processes on the *Operational Business Processes* level are required to detail the usually coarse-grained organisational business processes. In operational business processes, activities and their relationships are described, excluding their im-

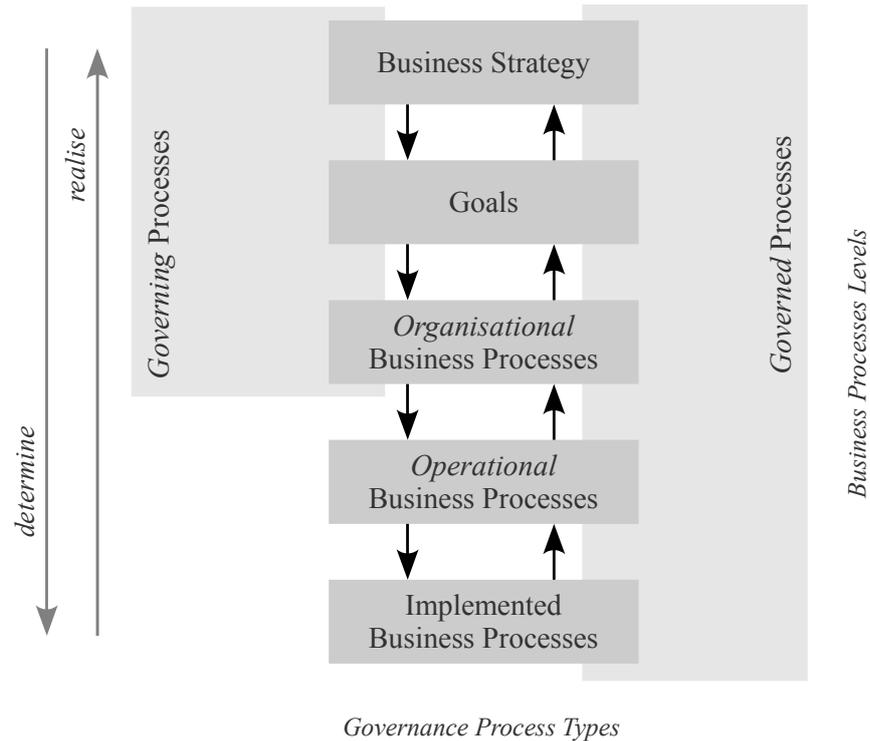


Figure 2.4: Levels of business processes (adapted from [264])

plementation details. Technically, these processes are specified by business process models (using, e.g., EPC or BPMN).

- Finally, *Implemented Business Processes* describe the specification of the technical execution context and organisational environment, in which they are executed [264].

In the context of SOA Governance, there is an additional perspective on these process levels (cf. Fig. 2.4). Orthogonally to the level by Weske, two general types of governance processes can be distinguished: *governing* processes and *governed* processes [17, 27, 248] (cf. Sect. 3.1.8).

GOVERNING PROCESSES. *Governing* processes are high-level processes that perform control and steering of the IT system, aiming at enacting governance structures. They are the primary means of a governance approach. Examples are processes for compliance, vitality, exceptions and appeals, dispensation and communications [27, 248]. Clearly, for these processes, the process goal rather than the detailed process procedure description is in focus.

GOVERNED PROCESSES. *Governed* processes are controlled, monitored, and assessed by the governance approach [17, 248]. They are subject to measurement and assessment concerning performance (by metrics) and adherence to policies. Concerning the latter, *metric collection points* and *policies and standards evaluation points* are inserted in their activity flow, in order to determine compliance or non-compliance [27].

Note that, due to their nature, on the last two levels (*Operational* and *Implemented Business Process* level), no *governing* processes are located. Governing processes are generic high level processes, that are usually described in text form enriched by simple flow charts.

Both, governing processes and governed processes are included and described by SOA Governance approaches, that are investigated later in this thesis (cf. Chap. 3), and the conformance check of governed processes is addressed by a central contribution of this thesis, a comparison and retrieval approach for processes (cf. Chap. 5).

Process Conformance

While *Compliance* (see above) defines the strict implementation of given laws or norms, the term *Conformance* has a different notion. According to Johannsen and Goeken [97], *SOA Conformance* refers to the check of the enterprise that plans to implement SOA concerning its general adherence to requirements. Conformance is understood as a part of Compliance with the second part being *Performance*. Concerning business processes, exactly the same distinction can be made. While *Performance* of processes measures the outcome, *Process Conformance* measures adherence to guidelines. In the course of this thesis, we investigate *Process Conformance*.

The determination of so-called *Process Compliance* for given process models and regulations is an active field of research. According to Rinderle-Ma et al. [211], Process Compliance is understood as assuring that process execution adheres to norms that are relevant to the company. Approaches for Process Compliance use logic-based modelling of regulations for compliance validation, in contrast to considering reference processes (e.g., [66]). As example, this includes the check whether task B occurs before task A using the modelled requirement "B must occur before A".

The techniques used in this thesis for automated conformance checks do not use formal logic-based modelled regulations. SOA Governance frameworks rather provide holistic reference models than regulations with small scope. I.e., normally, regulations are provided as reference models. Hence, in the case of governance, process models are checked for conformance by comparing them to reference process models. Further, in the governance context, the final assessment is always left to a human expert.

Summarising, we use a different approach from Process Compliance called *Process Conformance*.

COMPONENT ANALYSIS OF SOA GOVERNANCE APPROACHES

ACCORDING TO A survey conducted among companies that use SOA as enterprise architecture, 79% of the respondents consider it a negative risk to put services into production that are not effectively *governed*. Beyond that, 88% of the companies find their current SOA Governance approach insufficient – only 12% implemented a sufficient approach according to their own assessment (cf. App. B.2). This draws the picture of an extreme disaccord: although companies are aware of the high risk of a governance lack, they have not implemented sufficient mechanisms to address this risk. The need for appropriate governance approaches is high – and companies are obviously aware of the increasing risk.

In recent years, a number of models and frameworks for SOA Governance from different perspectives have been proposed. While most of them address similar goals, they proceed from diverging challenges and definitions and propose varying techniques and differing combinations of them to reach these goals. Only few of them show a level of comprehensiveness that could be sufficient for becoming an accepted standard approach. In this chapter, we perform an analysis that identifies frequently used concepts in SOA Governance, as well as aspects that are considered most important by a majority of authors. We elicitate criteria, compare them, and outline their perspectives, objectives, and proposed methods.

In this chapter, we investigate 22 approaches that claim to provide SOA Governance (cf. Section 3.1). Based on these insights, we provide a definition for SOA Governance in Section 3.2. Final conclusions are discussed in Section 3.3.

3.1 *Analysed Components*

Motivated by this lack of clarity, we investigate and compare 22 SOA Governance approaches, developed at companies and research institutions.

The investigated approaches have been proposed by three groups of authors. Scientifically published, academic approaches cover reviewed publications such as journal articles, conference papers, as well as books and book chapters. Many governance approaches have been made available by software manufacturers, published as company whitepapers that target governance for an SOA system aligned with proprietary software products (e.g., SOA infrastructure). The third group is formed by authors from the IT consulting industry that published their expertise in whitepapers based on achieved experience. During our analysis, 10 similar criteria, in the following also called components, have been identified. They are detailed in this chapter.

Generally, the approaches expose different levels of quality: often, the recommendation of components is not explained by the experts. The approaches, whose recommendation of a respective criterion is backed by arguments and critically discussed, are considered a *founded recommendation*: in this case, the suggested components and its purpose

have been consistently explained, concrete suggestions are made or examples are given – in contrast to the mere mention of a criteria. In fact, some approaches are characterised by a more *legère* view of the topic. These approaches, when introducing a component, regarding a particular aspect, expose a lack of clear instantiation, explanation, level of detail, or specification. Generally, in these cases, the approach lacks precision concerning that particular component. It is recommended as being valuable and useful without explanation. These cases are considered as *proposals* of the given component, meaning “partially specified”, or “mentioned”.

The overall results are summarised in Figure 3.1. Per criterion, the aggregated number of mentions is indicated. Additionally, the diagram shows the absolute amount of criteria mentions per author group and quality aspect, where 22 is the maximum. *Founded recommendations* are distinguished from *proposals*, as well as the different author groups from one another. Detailed results are outlined in Table 3.1. Rows list approaches and the columns show the criteria. The approaches are grouped according to the perspectives: academic publications like books and journal articles, publications from software vendors, and those from the IT consulting industry. For each criterion and approach, it is indicated whether and in what quality the criterion is considered. *Founded recommendations* are marked with a full dot (●) and *proposals* are marked with a circle (○) in the table. A hyphen (–) indicates that the concept is not integrated in the corresponding approach. Further details of the results are provided in Appendix B.6.

In the following, we introduce and discuss the 10 major components that were identified (in Sect. 3.1.1 to Sect. 3.1.10) and summarise further aspects (in Sect. 3.1.11).

3.1.1 Governance Policies

A major aspect of SOA Governance are considered *governance policies*, organised in policy catalogues. 21 out of 22 approaches mention this aspect, 13 of them provide detailed recommendations. It is the most frequently integrated component of SOA Governance in the investigated approaches (cf. Tab. 3.1).

By almost all approaches, governance policies are informally defined as ‘means to define what is *right*’. Generally, governance policies represent guidelines, conventions, rules, and best practices that support the controllable and efficient operation of the SOA system. They are often applied in the administration of a service life cycle (cf. Sect. 3.1.5), or within an SOA procedure model (cf. Sect. 3.1.6).

Generally, governance policies are considered distinct from *service performance-related policies* as described by standards such as WS-Policy or WS-Re2Policy [56, 204, 257]. Main aspects of governance policies are their application to *roles*, *service design and operation*, and *service documentation*. Some approaches, however, leave the specified policies unclassified. Concerning policy handling, procedures for policy exception handling, as well as recognition of too restrictive policies are suggested. We summarise these aspects in the following.

Policies are usually used to regulate service interactions and operation (*service runtime policies*), design (*service design time policies*), or their description (*service description policies*). According to Bernhardt and Seese [15], service-related policies are always derived from governance policies.

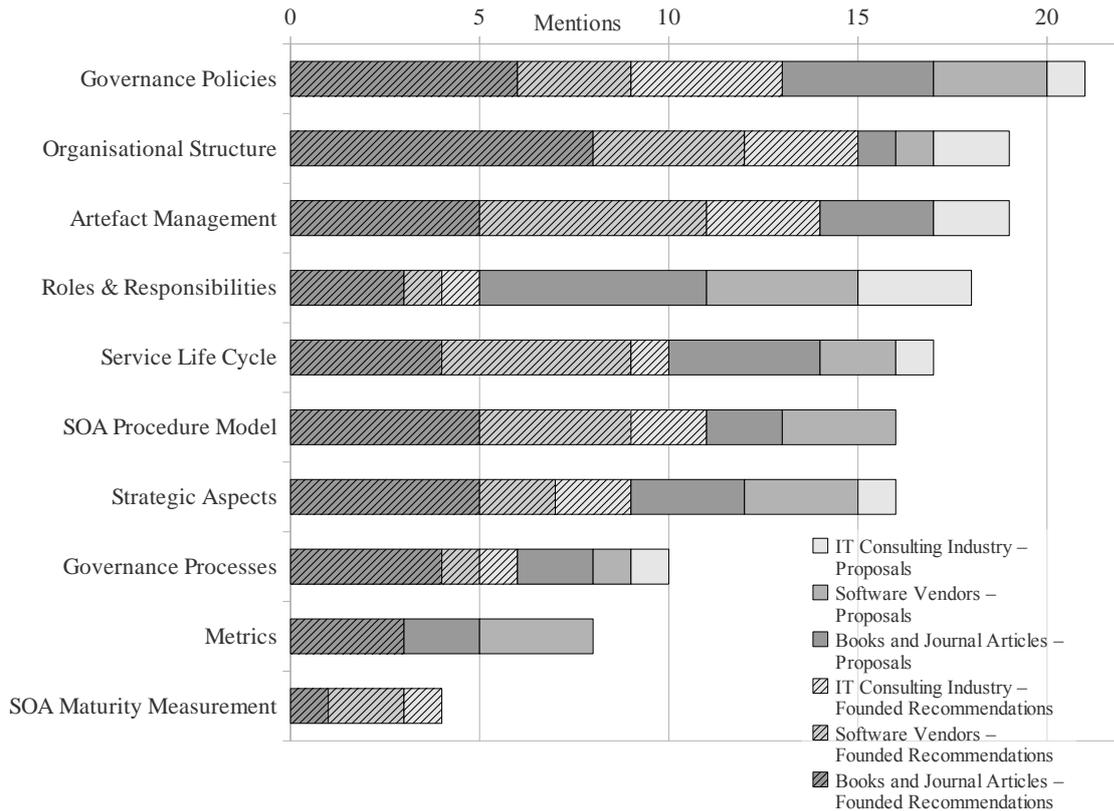


Figure 3.1: Results of the component analysis – number of mentions per component, quality level, and author group

SERVICE DESIGN TIME POLICIES typically regulate the manner of service design and development, providing technical standards and architectural requirements, e.g., the regulation of design patterns that are used during service development [1, 10, 20, 27, 74, 138, 217, 233, 248, 265, 267]. Often, design time policies intentionally limit the choice of technical standards and interface formats. Interoperability assures the flawless cooperation of services. For example, interfaces are often considered to provide secure and encrypted communication [1, 195]. For service operation, authors recommend to use accepted standards (e.g., SOAP, WSDL, REST) during design and implementation, rather than proprietary solutions of few software manufacturers. The goal is to ensure a flexible platform and interoperability of the services (e.g., deployment of services on technically different platforms). Design time policies are introduced by the majority of approaches [1, 27, 74, 75, 116, 138, 195, 198, 210, 215, 248].

SERVICE RUNTIME POLICIES target service operation. They comprise security regulations (e.g., encryption), access control (e.g., usage of authentication methods, availability, and performance of services), precise event logging, and correct billing. Two runtime policy types are distinguished: role authorisations and service operation (or monitoring) policies. An important aspect when enforcing policies at service runtime is their formalisation in a machine-understandable format such as

Table 3.1: Detailed results of the component analysis

	Governance Policies	Organisational Structure	Artefact Management	Roles and Responsibilities	Service Life Cycle	SOA Procedure Model	Strategic Aspects	Governance Processes	Metrics	SOA Maturity Measurement
<i>Legend</i>										
•	founded recommendation									
○	proposal									
–	not integrated									
<i>Page</i>	26	29	31	32	35	37	39	41	43	45
<i>Section</i>	3.1.1	3.1.2	3.1.3	3.1.4	3.1.5	3.1.6	3.1.7	3.1.8	3.1.9	3.1.10
<i>Books and Journal Articles</i>										
Schepers et al. [217]	•	○	○	○	○	•	•	–	–	•
Bernhardt and Seese [15]	•	•	○	–	•	•	–	•	•	–
Derler and Weinreich [43]	○	–	•	○	•	–	–	–	–	–
Kohnke et al. [116]	•	•	–	○	○	–	•	○	○	–
Bieberstein et al. [17, 18]	○	•	•	•	–	•	•	•	–	–
Marks and Bell [138]	•	•	•	○	○	•	•	•	•	–
Brown et al. [27]	•	•	•	•	•	•	•	•	•	–
Schelp and Stutz [215]	○	•	–	○	–	○	○	–	○	–
Rieger and Bruns [210]	•	•	•	•	○	–	○	–	–	–
Josuttis [101]	○	•	○	○	•	○	○	○	–	–
<i>Software Vendors</i>										
HP, Systinet [25, 74, 75, 246]	•	•	•	–	•	○	○	–	–	–
webMethods, Inc. [139, 261]	○	–	•	○	○	○	–	○	–	–
Software AG [233, 234]	•	○	•	○	•	•	○	–	○	•
BEA Systems, Inc. [10]	○	–	•	–	•	○	–	–	–	–
Oracle [1]	•	•	•	○	○	•	•	–	–	•
IBM [26, 81, 140, 151, 154, 271]	–	•	•	○	•	•	○	–	○	–
The Open Group [248]	○	•	–	•	•	•	•	•	○	–
<i>IT Consulting Industry</i>										
Everware-CBDI Allen [3]	•	•	•	○	–	–	•	•	–	•
BearingPoint [198]	•	○	○	○	○	•	○	–	–	–
ZapThink [20]	•	○	•	•	–	–	•	○	–	–
Windley [265, 266, 267]	•	•	○	○	•	•	–	–	–	–
Berlecon [195]	○	•	•	–	–	–	–	–	–	–

WS-Policy, WSPL, or WSDL 2.0. In this context, Marks and Bell [138] recommend not to link runtime policies tightly to services, as policies can also be dependent on the service consumer. In this case, the policies to be applied should be agreed upon

by service contract negotiations [138]. Runtime policies are considered a common policy type [1, 3, 15, 17, 43, 74, 75, 101, 138, 198, 210, 215, 217, 233, 248, 261].

SERVICE DOCUMENTATION POLICIES provide the aspects and the scope of service documentation, and assure that all relevant information is included in a service's description (such as, e.g., the documentation from technical, functional, and business viewpoints). Bernhardt and Seese [15] recommend to set up a service taxonomy to simplify discovery. Allen [3] recommends to adjust the scope of documentation to the maturity level of the SOA initiative and the degree of service reuse that is aimed at. The explicit regulation of the description of services targets at proper service retrieval and use, as well as the realisation of service reuse in different contexts [1, 3, 15, 265].

ROLE-RELATED POLICIES refer to employee roles and their definitions, and target personal behaviour as well as behaviour of organisational entities. Authors emphasise that the success of these policies depends on their precise and clear formulation, proper communication of the added value, and employees being made attentive to them [15, 74, 75, 138, 266]. Further, adjusting the behaviour to regulations should not be an additional burden. Windley [266] points out, however, that there is the danger to limit creative and productive developers in their work by policies that are too strongly and inappropriately regulating. Frequent violation against few policies should also be considered an indication of this type of inappropriateness [265].

Defining policies anew for each service means additional effort in the service development process. Some authors recommend to manage, administrate, and reuse policies like services in a specific policy life cycle that includes policy revision and policy reuse [25, 75, 198, 265, 267].

In special cases, granting non-conformance to policies is considered reasonable. Marks and Bell [138] advise to install a formal process for policy exception granting. They point out that it should include escalation procedures including higher instances (e.g., a management board) for the case of conflicts. For example, for services that are deployed with admittedly little opportunities of being reused or that can reach the required performance only when using other than the required standards, an exception grant can be applied for [20, 138, 265]. Excessive exploitation of this process, however, can be an indication for too rigid or inappropriate design policies [267].

As a consensus of all authors, policies are considered mighty instruments that combine various application aspects. They represent the most important and quite complex aspect of SOA Governance. Application aspects are roles-related, service design and operation-related, and, explicitly, service documentation-related policies. The latter are to ensure useful retrieval processes that are performed by, e.g., service requesters. Important aspects of policy handling are policy exception regulation and recognition of inappropriate (too restrictive) regulations.

3.1.2 *Organisational Structure*

Due to the changed conditions of SOA systems compared to other IT systems, the majority of authors considers to adjust organisational structures. Changes in the structure of

the organisation are mentioned by 19 out of 22 approaches, whereas almost 70% provide founded recommendations (more than for *governance policies*, cf. Fig. 3.1). The components *organisational structure* and *governance policies* can hence be considered equally important (cf. also Tab. 3.1). The approaches outline and introduce new boards, councils, and institutions for special accountabilities around SOA.

Typical competencies of boards are of *executive* or *consulting* nature, are concerned with the specification of *requirements* for implemented services as well as applications, or the *development* direction of the SOA project. Almost all of the approaches propose the establishment of three main organisational entities: the *SOA Governance Board*, the *SOA Board*, and the *SOA Centre of Excellence*.

- The *SOA Governance Board* mainly deals with the *design and configuration of governance* itself. The board is composed of representatives and specialists from all IT and business domains that are relevant to the SOA system. In order to ensure enough authority and influence, it is considered crucial that business managers and decision makers are board members. Rane and Lomow [198] propose the involvement of a dedicated “sponsor” from the CEO board, managing the SOA initiative. The board’s main tasks are the control and supervision of the current status of the SOA system, the development of governance methods such as improving processes, the establishment of new policies, and monitoring of their enactment and performance. Accountabilities cover project prioritisation, funding, the iterative transition to SOA, and the management of the SOA system. Depending on the organisation form (distributed vs. centralised), it either possesses the exclusive decision rights concerning processes and policies itself, or can delegate them to subordinate councils [e.g., 3, 15, 25, 27, 74, 75, 138, 198, 233, 248, 266]. This board is also called *SOA Governance Council* [27, 198], or *SOA Leadership Team* [138].
- The *strategic leadership* for the SOA system is borne by the *SOA Board*. The identification of business goals, their prioritisation, and the control and decision upon SOA goals and an SOA strategy are its central competencies. Usually, it cooperates with experts from business departments and business process designers in order to analyse business processes, formalise them, and prioritise them for the realisation by services. It manages the service portfolio. In case no SOA Board is established, the SOA Governance Board is considered to be in charge of these strategic SOA tasks [1, 3, 15, 17, 20, 27, 74, 116, 138, 195, 198, 210, 215, 217, 233, 248, 267].
- The establishment of a *SOA Centre of Excellence (SOA CoE)* is recommended by almost all approaches. The main purpose of the CoE is *bundling competencies* concerning SOA implementation and operation. Expertise is created by the implementation and management of SOA pilot projects, which is redistributed by specification of best practices and by involvement in further projects. In some cases, it is considered to be in charge of policy development, the specification of technical standards as well as reference architectures, i.e., to additionally bear directive competence [1, 3, 15, 17, 20, 27, 75, 101, 116, 138, 151, 195, 215]. In some cases, the SOA CoE is considered to be a consulting board. In case the SOA CoE is assigned these competencies, the SOA Governance Boards’ competencies are limited to determination of principles that the SOA CoE can derive detailed policies and technical standards from [26, 198, 210, 217, 233, 234, 248, 267].

Often additional boards are specified (e.g., the service development team, the applications development team, IT operations). An according table is provided in Appendix B.1.1.

All approaches that give *founded recommendations* concerning organisational changes (15 out of 22), recommend setting up an SOA CoE. Apparently, this institution has convinced in theory (academic approaches, seven mentions) as well as in practice (eight mentions from IT consulting and software vendors). It can be considered a crucial organisational institution for the operation of an SOA system.

The *SOA Governance Board* and *SOA Board* are often defined in a mutually excluding way, i.e., approaches defining the first one omit the latter (e.g., [3, 195, 198]). This indicates that the respective competencies, SOA Governance and strategic leadership, are often considered to be replaceable, or even congruent. However, strategic leadership is commonly regarded one component of a governance approach. The classification above shows the lack of holistic understanding of this discipline, even on the part of the providers of SOA Governance.

Concluding, the presented organisational entities are the three most frequently integrated ones. Competencies, however, are not very clearly attributable. Especially concerning the question how decision and consulting competencies are to be distributed among the entities, the approaches give different recommendations. The majority, however, agrees on the SOA CoE bundling many of the discussed competencies – in some cases even all of them.

In contrast to organisational entities that could also be named *group roles*, the precise definition of (single) roles and responsibilities has been a major aspect of SOA Governance approaches (cf. Sect. 3.1.4).

3.1.3 *Artefact Management*

19 out of 22 approaches name software support or artefact management a central building block of SOA Governance. Most of them come from the software industry (cf. Tab. 3.1 and Fig. 3.1).

During the development process of an SOA system, many artefacts are created, for example, *services*, *service meta data*, *service descriptions*, *interface descriptions*, and *message format specifications*. Services in operation are bounded by *policies* and *service contracts*. Further meta data are SOA Governance artefacts such as *roadmaps*, *process descriptions*, and *reference architectures* [248]. All these artefacts have to be managed and made available to the participants of an SOA system. For their administration, mainly two terms have been mentioned within the approaches, *service registries* and *service repositories*. Both represent software systems that manage SOA artefacts and make them available to the involved parties. Generally, service registries provide the technical view, while the service management perspective is represented by service repositories [101]. In SOA Governance approaches, however, these terms are not clearly distinguished from one another.

Half of the approaches that mention a *service registry*, define it as system for publishing, storage, and retrieval of services [1, 15, 195, 198, 210, 217, 233]. A *service repository* mainly is considered a tool to manage meta data, e.g., technical service descriptions, service contracts, development documentation, overview of service versions, and guidelines associated with the service [101]. Most of the approaches see the registry as complementary to the repository, whereas the registry provides the service while the repository stores

the corresponding meta data (as “meta data repository”). However, four approaches assign technical aspects like service retrieval and publication to the service repository [20, 43, 215, 267]. They also define the storage of general service information such as meta data to be a concern of the service repository. Obviously, the notions of service repository and service registry have not yet established among all SOA Governance authors.

Some authors state that splitting up registry and repository into two different software systems and hence splitting up the storage of service-relevant information, is not useful [75, 138, 233]. Most software solutions integrate all these functions in uniform systems.

Further systems concerning SOA Governance are *Web service management systems* [1, 265, 267]. They serve service operation monitoring by automatically supervising policy conformance, monitoring and controlling service status and interactions. Usually, these systems are realised as plug-ins for enterprise service buses. Bernhardt and Seese [15] name them “Policy and Contract Management” systems, Marks and Bell [138] “Policy Engines”. Additionally, Afshar [1] recommends the operation of “Business Process Management Suites” that integrate business processes into service management, providing modelling process functionalities.

Some authors recommend additional techniques for artefact and meta data handling. Schelp and Stutz [215] and Derler and Weinreich [43] propose the modelling of artefacts including, e.g., policies, descriptions, and processes, and their relationships as *meta model*. Allen [3] suggests to establish an owner for each of these artefacts, i.e., to introduce an additional responsibility and role *artefact owner*. Similarly, Afshar [1] and Bloomberg [20] recommend the introduction of *data owners* or *data stewarts* that are in charge of quality management and unification of utilised company data.

Concluding, the approaches suggest the operation of a service registry or service repository, and of a Web service management system. As main function of a service registry, most authors refer to publishing and discovering services, while the service repository is considered to serve as meta data storage. However, none of the approaches recommends to operate both of these. Nevertheless, it is obvious that the understandings of service registry and repository in terms of functionalities diverge among SOA Governance authors, i.e., from the governance perspective. Some authors recommend to organise and structure artefacts types and their relationships in a meta model to clarify interdependencies as well as the establishment of additional data and artefact related roles and responsibilities.

3.1.4 Roles and Responsibilities

Almost 80% of the approaches mention the adjustment of roles and responsibilities for the operation of an SOA system. In total, 17 approaches mention it, where nine come from academic environments, and eight from the practitioners’ domains (cf. also Fig. 3.1 and Tab. 3.1).

By all these authors, implementing and operating an enterprise architecture as SOA is considered to have impact on the organisational structure of the entire company. Besides the introduction of new organisational entities (cf. Organisational Structure, p. 29), this covers the definition of new roles and accountabilities. In order to assign clear and non-overlapping definitions of competencies, a solid concept for roles and accountabilities is commonly considered to be advantageous for all involved persons and the operation of

the SOA system.

In the following, the five most frequently mentioned roles are outlined, as well as a number of suggested personnel administration methods.

- The *SOA Governance Lead* is accountable for the holistic planning and monitoring of governance methods. Important tasks are the strategic direction and alignment of the governance efforts as well as the combination and interaction of governance methods [3, 17, 27, 248]. In most approaches, this competency is defined as a group role and is assigned to a board consisting of different stakeholders, e.g., the SOA Governance Board.
- The duty of the *SOA Architect* is to plan the holistic service architecture. Generally, this role is considered a mediator between business and technology. It is in charge of resolving all technical issues concerning the infrastructure [17, 27], as well as of the management of service composition, the service portfolio, and compliance with standards [248]. Rieger and Bruns [210] divide this role into two parts: the *business* SOA architect and the *technical* SOA architect. While the first is in charge of requirements analyses, is experienced in business process modelling, and communicates the SOA vision throughout the enterprise, the technical part makes architecture-related decisions, performs component modelling, and is responsible for service interfaces and documentation [210]. The enacting person is mostly part of the SOA Governance Board [3, 17, 27, 138, 210, 233, 248, 261].
- The role of the *Business Process Designer* investigates and outlines the company's business processes and formalises them in a common process description language. The role is in charge of the creation of reusable automated business processes, process testing, and investigates all service orchestration possibilities. It is a general expert in technical aspects of modelling [17, 26]. The business process designer realises the processes identified by the SOA Architect as a composition of atomic services [210]. These processes are then subject to the execution by single or composite services. The role is also called *Process Flow Designer*, *Process Developer*, or *Process Modeller* [17, 18, 20, 27, 210, 233, 248].
- The *Service Designer* works together with the SOA Architect, is in charge of message exchange modelling, and creates service designs. The role finalises services to make them meet QoS demands, and to ensure the usefulness of their interface designs. Further, the service designer supports the SOA Architect in the determination of the right service granularity [1, 17, 20, 27, 210, 233, 248].
- Generally, the *Service Owner* is the contact partner for a particular service, concerning specific informations, change or extension queries, or issues of any kind concerning the service. It is responsible for a service during the complete service life cycle. The role can be enacted by a person, a department, or organisation. The service owner is expected to have comprehensive knowledge concerning the specific service and general experience in handling services. Almost all authors emphasise the importance of explicitly defined *service competencies*
The task of a governance approach is to create and provide a catalogue that identifies a service owner for each service. As checkpoints, before service operation (e.g., as part of service testing), service checks are to cover whether each service has an owner. [1, 3, 27, 43, 75, 138, 198, 215, 217, 233, 248, 271]

RACI Chart

Functions

Activities	CEO	CFO	Business Executive	CIO	Business Senior Management	Head Operations	Chief Architect	Head Development	Head IT Administration	PMO	Compliance, Audit, Risk and Security
Determine risk management alignment (e.g., assess risk).	A	R/A	C	C	R/A	I					I
Understand relevant strategic business objectives.		C	C	R/A	C	C					I
Understand relevant business process objectives.				C	C	R/A					I
Identify internal IT objectives, and establish risk context.					R/A		C	C	C		I
Identify events associated with objectives (some events are business-oriented [business is A]; some are IT-oriented [IT is A, business is C]).	I			A/C	A	R	R	R	R		C
Assess risk associated with events.				A/C	A	R	R	R	R		C
Evaluate and select risk responses.	I	I	A	A/C	A	R	R	R	R		C
Prioritise and plan control activities.	C	C	A	A	R	R	C	C	C		C
Approve and ensure funding for risk action plans.		A	A		R	I	I	I	I		I
Maintain and monitor a risk action plan.	A	C	I	R	R	C	C	C	C	C	R

A RACI chart identifies who is Responsible, Accountable, Consulted and/or Informed.

Figure 3.2: Example for a RACI chart [86]

Some more roles have been pointed out, e.g., the *Tester* and the *Administrator*. For further information refer to Appendix B.1.2. Further, in the context of SOA Governance, the involvement of *management representatives* is considered to be of utmost importance (cf. Sect. 3.1.7, *Management Commitment*).

A currently widely accepted and established method for competency assignment is the so-called *RACI chart*. An example from COBIT [86] is provided in Figure 3.2. Many of the examined approaches, as well as standard IT Governance frameworks like COBIT and ITIL integrate them [27, 86, 176, 198, 215, 217]. RACI charts are competency tables for a given process. Rows show the process' activities and columns indicate roles (cf. Fig. 3.2). For each pair of activity and role, a character *R*, *A*, *C*, or *I* indicates the type of the role's involvement in the activity. *R* indicates *responsible*, i.e., the executing, operative task, while *A* defines *accountability*. The *A*-role is accountable for success or failure of the given activity, e.g., in financial terms. The specification of an accountable person is mandatory for each task; this person delegates the work to the *R*-role, and must approve the results. *C* means *consulted*, i.e., the designated role is to be asked for advice, and *I* indicates *informed*, i.e., a role that must be informed about the course, the result, or similar actions taken in the context of the activity [17, 103, 138]. Authors at Everware-CBDI propose a similar approach named *RAEW Analysis (responsibility, authority, expertise, and work)* [198].

In order to ensure that persons meet the requirements of a given role, Brown et al. [27] recommend the *Capability Assessment Method*. It provides tables listing the required technical and organisational skills per role that are defined by the SOA Governance approach. Using these lists, suitable candidates are identified. Identified gaps in the employees' skills can be closed by specific training or the employment of new persons. A further central aspect is the SOA-based employee training [1, 27]. Bieberstein et al. [17] propose to establish an *SOA Education Plan* in parallel in order to identify the gaps mentioned above and arrange specific required trainings. Some authors propose the creation of motivation by incentive systems to increase acceptance among the employees and mo-

tivate them to adjust behaviour. They state, that the new IT architecture demands special employee behaviour and consider *impact on behaviour* a central issue of SOA Governance. They see the necessity to influence the employees' behaviour regarding the new system, for example, by specific trainings or incentive mechanisms. These methods directly affect employees dealing with development, operation, and maintenance of services in the IT departments, as well as responsible persons in the business departments in charge of the alignment of business processes and services [18, 101, 116, 263, 266]. Marks and Bell [138] additionally apply a metrics model to influence the employees' behaviour. Integrated in their approach, the metrics model creates the SOA behavioural interaction model, "which defines the expectations for the collective behaviour of the SOA overall" [138].

Concluding, by the majority of approaches, five employee roles are pointed out that are specific to the operation of an SOA system. Further, in the context of SOA Governance, an important aspect is the targeted impact on behaviour. Obviously, the IT Governance goal to "achieve desirable behaviour in the use of IT" – as stated in the first definition of IT Governance by Weill and Ross [263] – is an important goal for SOA Governance as well. This clearly shows the existing relations between these two governance disciplines.

The roles and the personnel management methods in the context of SOA Governance suggested here can be regarded as central components of the discipline SOA Governance.

3.1.5 Service Life Cycle

According to the analysis performed above, service life cycle management (SLCM) is a central aspect of SOA Governance. More than 75% of the approaches mention a service life cycle to be an integral part of SOA Governance (cf. Tab. 3.1). The majority of approaches emphasising the service life cycle are from the author group of *software vendors* (cf. Fig. 3.1).

Life cycle models, in general, are widely used additives for design, development, operation, and maintenance of software (e.g., [236]). As a purpose of SOA Governance, the design, implementation, operation, and version management of services can be improved by comprehensive and reasonable regulations in service life cycles [25, 139, 267]. Their planning and implementation is part of SOA Governance. The notions of definition and distribution of activities in life cycle phases vary in wide ranges [1, 10, 15, 27, 43, 75, 101, 116, 138, 139, 198, 210, 217, 233, 248, 267].

Services are common software artefacts – however, their functionality scope is small and they appear in multiplicity. Services are delivered to the customer and installed, i.e., deployed, in a higher number of environments and contexts compared to common (monolithic) software. SLCM targets the steering, direction, and control of every service in life cycle phases. The goal is to ensure manageability and conformity, in spite of large service numbers. To tackle these challenges, service life cycle approaches are often divided into three super-phases: design time, runtime, and change time (e.g., [25, 68, 139, 246], cf. Fig. 3.3).

SERVICE DESIGN TIME. The phases up to deployment are commonly referred to as *service design time*. The following aspects are considered important by the majority of approaches. The *categorisation* of services using subcategories determined by taxonomies [15, 138, 198, 233] or service domains [20, 217] is considered a major

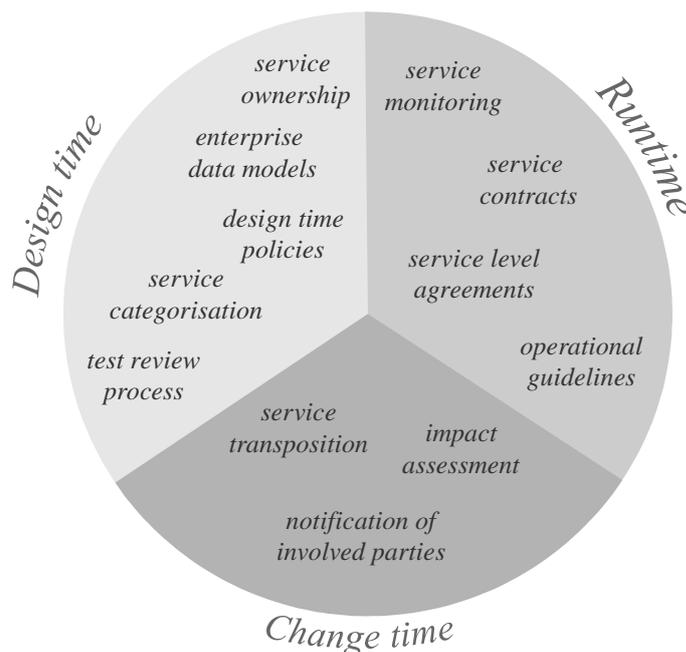


Figure 3.3: Overview of important aspects of SLCs

aspect. Enacting *service ownership* is regarded a fundamental aspect of the life cycle and included by almost all approaches [1, 3, 27, 43, 75, 138, 198, 215, 217, 233, 248]. Design time *policies* determine the manner services are designed, e.g., the technical standards, architectural requirements, or design patterns, ensuring interoperability by interface regulations. They are specified by 13 approaches [1, 20, 27, 74, 138, 195, 198, 210, 215, 217, 233, 248, 267] (cf. Sect. 3.1.1). In this respect, reference architectures as an exemplary implementation are considered especially useful [1, 198, 248, 265]. So-called *enterprise data models* define a uniform perspective on the organisation's data (e.g., types, structure, locations). These are considered mandatory for consistent service data integration [17, 27, 75, 138]. Concerning services, a *test review process* is recommended, where design, source code, and meta data are checked against the corresponding guidelines [15, 74, 75, 138].

SERVICE RUNTIME. The usage phase is referred to as *service runtime*. *Operational guidelines* are considered a central element during the runtime (cf. Sect. 3.1.1). Further, by concise management of *service contracts* and *service level agreements (SLA)*, the *service reuse factor* can be increased [1, 3, 15, 27, 43, 74, 75, 116, 138, 195, 198, 217, 233, 248, 265]. The detection of undesired service invocations [248], the determination of the service reuse rate [27], and service profitability calculation [195, 233] are important tasks of *service monitoring* during runtime [1, 15, 17, 20, 27, 74, 75, 116, 138, 195, 233, 248].

SERVICE CHANGE TIME. Some authors additionally define *change time* that is generally not considered to be a chronological phase, but occurs individually for each service (e.g., [25, 68, 139, 246]). In this phase, service changes are performed in a way

that is compatible with all involved roles such as other services, persons, and applications. Mainly three procedures form this phase: *notification of involved parties*, the *service transposition* procedure, and an *impact assessment*. Amongst others, as part of governance policies, these steps are applied in order to ensure cost-neutral, transparent, and flexible service changes. However, foundations for a smooth service change can be laid in previous phases. For example, interface design guidelines can minimise visibility of the performed changes, aiming to keep the service description consistent [1, 15, 43, 74, 75, 138, 195, 198, 217, 248].

Using life cycles, many artefacts beyond services can be controlled. Exceeding guidelines, applications composed from services, as well as business processes can be controlled by life cycles (as proposed by [1]). Also, readjustments of SOA goals to changed business requirements, or frequent transposition of the SOA Governance model are performed using corresponding life cycles (e.g., [27, 81], cf. Sect. 3.1.6). Using life cycles is a powerful instrument of control, i.e., a powerful instrument of governance.

3.1.6 SOA Procedure Model

Besides the management and effective administration of governance methods, the strategy and procedure of adopting and introducing an SOA system, i.e., an *SOA Procedure Model*, is considered a crucial part of an SOA Governance approach. What is summarised and referred to as *SOA Procedure Model* in this analysis are many different variations of procedures for regulated SOA introduction and operation that are called, e.g., “SOA Life Cycle”, “SOA Governance Roadmap”, or “SOA Adoption Model” by the respective approaches. Of the 22 approaches, 16 point out the importance of a procedure model, most of them from the software industry and academia (cf. Tab. 3.1 and Fig. 3.1).

Generally, SOA Procedure Models act as a global guideline for the future development of the SOA system. They designate and communicate planned future developments of an SOA system and describe the phases from plan to realisation. A major goal is to minimise risks during the introduction of the SOA system [203].

In the following, we outline their differences and commonalities and provide a consolidated overview (cf. Fig. 3.4). The precise descriptions of the approaches are provided in Section B.1.3. Typically, the proposed procedure models are divided into four iterative phases.

- (1) As a general rule, the *first phase* covers the determination of abstract goals for the SOA system, the SOA vision, as well as constraints [1, 27, 81, 248]. These are directly derived from strategic enterprise goals. The initial assessment of the IT and business environment is performed, and a consensus of preliminary governance strategy and the scope of the SOA program are formulated [198, 217, 267].
- (2) In a *second phase*, the SOA goals are refined to form a specific strategy, and organisational entities are set up (e.g., the SOA CoE), role models are introduced and service ownership is defined [27, 138, 198, 217]. The system’s *status quo* is determined, and a comparison with reference architectures is performed [27, 248]. A chronological realisation plan of these methods, i.e., an SOA Governance roadmap or transition plan is determined [17, 27, 138, 248]. First guidelines are defined and an initial management process [1, 17, 27, 198], service life cycle policies, governance processes,

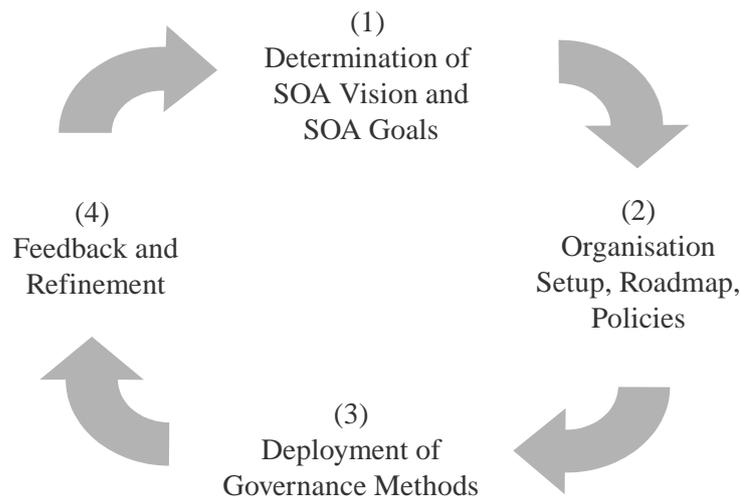


Figure 3.4: Consolidated phases of the proposed SOA Procedure Models

service owners, and funding models are set up [138]. Tools are selected and governance processes are developed [1, 198]. Further, the potential reuse of existing systems is investigated [233]. Schepers et al. [217] and Afshar [1] recommend for this step the consideration of the maturity level of the current SOA, in order not to strain, e.g. a pilot project unnecessarily with bureaucracy.

- (3) As a *third phase*, building on these SOA goals and strategic requirements, specific governance methods are derived. These focus on the arrangement of processes in the service life cycle and the strategic alignment of the SOA system, design and operational policies, and the selection of suitable technical support. The service portfolio management, and the service life cycle are set up [217]. Policy enforcement, messaging platforms, service registries and metadata repositories, technical standards, service operation policies, and security solutions are implemented [138, 265], also called “Interoperability Framework” [267]. The planned system is deployed, i.e., the roadmap implemented [27, 248], the defined roles, competencies and decision rights are transferred to involved persons [17, 27, 233], and general governance mechanisms are installed [1]. The accountable board starts monitoring and controlling the service life cycle as well as policies [198].
- (4) In the *fourth* and last consolidated phase, based on metrics (e.g., service reuse, errors in service operation) and feedback of involved persons, the governance methods and their interplay are refined. Quality assurance and regular audits are performed [1, 27]. Its overall effectiveness is measured, generating recommendations for improvement [198]. Based on the results, policies are refined, and processes improved [1]. Important aspects, according to Windley [265], are the monitoring of policy effectiveness and acceptance. Schepers et al. [217] recommend “service enforcement” and service level management in this step. Services are integrated into applications, and new services are formed by assembly, not development [233]. According suggestions by Brown et al. [27], Schepers et al. [217], and The Open Group [248] are

defined as reiterating SOA life cycles. This way, the feedback by metrics and by involved roles triggers a new life cycle (process model) iteration. In the next iteration, according to Afshar [1], the next level of maturity is reached.

Three further approaches do not specify phases: according to Schelp and Stutz [215], the SOA roadmap is considered to be the implementation plan for the SOA strategy. It serves as transparent proof for the benefit of the implementation. An iterative SOA life cycle serves the introduction of new SOA processes and the adjustment of existing architecture processes. At webMethods, authors define an SOA Governance life cycle that is similar to the service life cycle. It consists of design time, runtime, and change time governance [261]. At BEA Systems, SOA Governance is defined as service life cycle governance [10].

3.1.7 *Strategic Aspects*

The conception of a strategic plan as well as business-IT alignment are considered a further central element of SOA Governance by the experts. 16 out of 22 approaches refer to strategic alignment, the majority with concrete suggestions. All author groups equally put a strong emphasis on this point (cf. Tab. 3.1). Four aspects of strategic alignment considered most important are outlined in the following: formalisation of SOA goals, identification and prioritisation of services, adequate financing of service development, and SOA commitment of the management.

Formalisation of SOA Goals

In order to have a clear view of business goals and to include them into the strategic planning process, it is important to elicitate and document these goals. The documentation serves as input for the determination of SOA goals, aligned with the business goals. This type of documentation is called *Enterprise Principles* [20] or *Strategic Business Plan* [27] (cf. also [3, 20, 27, 195, 210, 215]). Many approaches demand the concrete formulation and documentation of strategic SOA goals as an abstract *SOA vision* (e.g., [17, 138, 248]). Some authors recommend to monitor the goals using predefined metrics or KPIs [15, 138]. Overall, it is important that SOA goals are aligned to company and business goals. SOA goals can be juxtaposed with their motivating business goals in a *Business-IT Alignment Table*. Additionally, for each SOA goal, the performance of a risk analysis is recommended [3]. In case business goals change, SOA goals must be adjusted. This is covered by iterative procedure models for SOA Governance, which some approaches suggest (for details cf. Sect. 3.1.6). In most of these, in each cycle reiteration, SOA goals are aligned anew with business goals. Formalisation of SOA goals is emphasised by the majority of approaches [1, 3, 17, 20, 27, 74, 75, 138, 215, 217, 233, 248].

Identification and Prioritisation of Services

Once SOA goals are determined, services are to be identified and prioritised. Two different approaches are suggested for this step, a bottom-up and a top-down procedure. Using the *bottom-up procedure*, software architects or the involved persons from business departments design the service interaction, before they are submitted to a board in charge of the strategic SOA overview (e.g., the SOA Board). In order to avoid the cases where a similar service already exists, the implementation of a function as service is not

profitable, no reuse can be anticipated, or technical policies are not met, the board makes the final service selection. In case of the *top-down* variant, business analysts or business process designers analyse business processes, divide them into subprocesses and specify them in a language like BPEL. Some of the identified activities then become blueprints for services. The interaction is defined by the chain of activities in the business process [1, 17, 27, 43, 151, 215, 217]. According to Schepers et al. [217], the latter variant is more complex than the first, but can be more profitable in some cases. The latter variant, however, is proposed more often [1, 17, 215]. Some propose both methods and demand a pragmatic decision depending on which one better fits the SOA strategy and current maturity [17, 43, 217].

A further important issue when identifying services is *service granularity*. Depending on the respective context, services must be designed large enough in functionality, in order to encapsulate a reasonable process part and avoid unnecessary communication on the one hand. On the other hand, service functionality should be designed small enough to be reusable in different contexts. As part of governance regulations, appropriate granularities are to be defined [27, 103, 164]. Experiences made by experts such as *service designers* are a crucial criterion here.

For the administration of existing and planned services, usually, service portfolios are set up. They provide a complete overview of all services, and support the planning and prioritisation for the service implementation. Service identification and management of service candidates is often performed as part of service portfolio management [1, 27, 198].

Service identification and prioritisation is commonly considered a crucial aspect of SOA Governance [1, 3, 15, 17, 27, 43, 116, 138, 195, 198, 210, 215, 217, 248].

Adequately Financing Service Development

Usually, services are deployed across divisional or department borders and have the potential to turn profitable after multiple usage. Hence, for financing an SOA system, the benefit of financing models that charge departments on project base is limited. Adjusted accounting models distribute the additional cost among all business departments alike, or the cost are borne by a central SOA budget. It is recommended to account services internally department-wise, according to the benefit provided, e.g., by their usage. Issues to be decided upon from case to case cover the distribution of the initial development cost (cost until a service is profitable), and the determination of the council or role that bears the risk for unprofitable services.

The majority of authors agrees on the fact that traditional accounting models on project level are not profitable and useful in the context of SOA. It is considered a central part of SOA Governance to determine adequate financing models [1, 3, 15, 17, 27, 74, 75, 116, 138, 198, 210, 215, 217, 233].

Management Commitment

Usually, *management representatives* are not directly involved in IT system management. In the context of SOA Governance, however, their involvement is considered to be of utmost importance. Financing of SOA Governance usually exceeds project limits, organisational entities are to be changed or new ones created, new roles and responsibilities are to be established (e.g., in the area of business process identification and examination), and governance methods often are not accepted *per se* by employees. For these reasons,

the authority of a management representative and the acceptance of the system by the management are considered generally required for successful SOA Governance (cf. also *SOA Governance Board*, p. 30).

Bloomberg [20] defines the crucial factor of management commitment for the effectiveness of IT Governance as follows:

“[...] the effectiveness of IT Governance in the enterprise depends on the successful cascading of strategy and goals down into the organization, and the traceability of project-specific activities in support of those organizational goals back up to the executive level.”

In fact, an important aspect is the willingness of the company to holistically adopt to and accept the SOA paradigm. SOA systems draw the majority of their added value from the decrease of levels of abstraction between business processes and their support by IT systems. This means, the members of business departments that are involved in business processes must agree to be “made a part” of IT processes. It is important that they identify and document their business processes (maybe for the first time). Further, an SOA system makes higher demands on, for example, accounting than the operation and development of monolithic applications, or the involvement of both business and IT departments in service development. Thus, it is important that the new conditions are met successfully [1, 18, 20, 27, 138, 198, 215, 248].

The acceptance of these changes are considered crucial success conditions for SOA systems, as these depend on the support and commitment of the company’s management level.

3.1.8 Governance Processes (and Policy Enforcement)

10 out of 22 approaches formulate governance processes and policy enforcement to be crucial aspects of SOA Governance, four of them are founded recommendations from academia. *Governance processes* are the actual implementation of governance. They define the business and IT-internal processes that are required to operate an IT system from the perspective of governance. They provide the activities and accountabilities for the operation of an SOA system on a meta level. By the term *policy enforcement* used by the approaches, mechanisms for automated policy conformance checks are summarised. They target the monitoring of adherence to policies and their operational enactment and are integrated in processes. In the following, we provide a short overview per approach.

The Open Group SOA Governance Framework [248] distinguishes *governing processes* from *governed processes*. The first ones are part of the governance approach (the “SOA Governance Reference Model”), defining three continually performed processes: *compliance*, *dispensation*, and *communication*. *Governed processes* are those that are being controlled, monitored, and measured, i.e., that are subject to governance (e.g., testing, design, deployment of services).

A similar classification is chosen by authors at IBM [17, 27]. Governance processes define strategic business and IT planning and steering (e.g., strategy development, portfolio management, innovation management). Complementary to four *governed processes* (*service strategy*, *service design*, *service transition*, and *service operation*), they define four *governance processes*: the *compliance process*, the *vitality process*, the *exception and appeals process*, and the *communications process* [27]. Similarly, Bieberstein et al. [17] define the

governance processes *business component identification and prioritisation process, business exception fallback process*, and four further architecture-related process types.

Marks and Bell [138] and authors at WebMethods [261] distinguish *design time* and *runtime governance* processes. As a third group, the latter propose *change time governance* processes, while Marks and Bell add *publishing and discovery governance* processes. All of these, similar to IT Governance processes, are used to instantiate control of the system, i.e., regulate the operational processes. Accordingly, in terms of The Open Group [248], these describe *governing* processes.

Bernhardt and Seese [15] define two types of governance processes: *policy-related* and *approval, review, and reporting* processes. While the first ones address and regulate the definition, propagation, violation, and adoption of policies, the second type of processes deals with approval (policy enforcement), review (service design), and reporting (feedback from organisational units, interventions).

As a third area of action besides *organisational structures* and *employees*, Kohnke et al. [116] define *processes*. These cover strategic alignment, business process management, service management, and service controlling. Two of these are governance processes: *strategic alignment* (consisting of business and IT goal alignment, requirements management, project portfolio management), and *service controlling* (service level management, service accounting, SOA compliance).

As one of four views¹ on SOA Governance defined by Allen [3], the *process view*, distinguishes two process levels: *management* and “*day-to-day*” processes. The management level is aligned to the organisation requirements and defines policy, organisation, process, and infrastructure requirements. The detailed policy definition and enforcement, in contrast, is the duty of the “*day-to-day*” SOA Governance activity and spread across different disciplines according to the policy type (e.g., architectural policies are defined by the “SOA Architecture and Design” discipline).

Authors at ZapThink [20] define the *architecture* and *IT Governance* functions in the context of SOA. There are five processes defined as duty of these functions: the architecture review and approval process, architecture exception and escalation process, the architecture maintenance process, the architecture communication process, and the architecture compliance review process. They lay a strong focus on the architecture of the system.

Further important processes are policy management, enforcement mechanisms, communications, change management, architecture review processes, SOA maintenance and many more (cf. [3, 17, 138, 215, 261]).

Many approaches mention the category *processes* as a central point of their approach. However, the classification types vary from governing vs. governed processes [17, 27, 248], runtime vs. design time governance [138, 261], policy-related vs. review-related [15], processes vs. organisational structures vs. employees [116], processes vs. organisation vs. infrastructure vs. maturity [3], and architecture review processes [20]. The classification that is mentioned most frequently is *governing vs. governed processes*. *Governing* processes cope with performing and realising governance methods and structures. They serve as means for the governance approach. *Governed* processes are subject to governance. They represent activities such as service development, process management, and service operation. Further, all authors agree that concise definition and structuring of governance processes is crucial to the successful operation of an SOA system.

¹ The views are: *organisational, maturity, infrastructure*, and *process* views (cf. [3]).

An often mentioned controlling technique is the definition of *check points*, or *policy enforcement points* [3, 27, 198, 233, 248]. At important points of governance processes, these points of control are defined, where, e.g., the status of an artefact is checked against defined requirements. Check points are universally applicable (they can, e.g., include architectural reviews, code reviews, services, service meta data checks, or checks of behavioural policies). In particular, they are considered useful in case policies cannot be checked automatically and work results must be analysed by humans. This often also covers the involvement of organisational entities, such as the SOA CoE (cf. Sect. 3.1.2).

Beyond the definition of appropriate operation guidelines, it is important to provide means and methods for policy enforcement. For this purpose, mostly policies are provided using machine processable and automatically enforceable formats (e.g., WS-Policy, WSPL, WSDL 2.0, or WS2RePolicy [138, 204]). Additionally, corresponding software tools are to be provided (e.g., proprietary “Policy Management”, or “Web Service Management” systems) [3, 27, 198, 233, 248].

Authors at WebMethods [261] propose policy enforcement points for automation: compliance checks are performed at the service registry or a proxy when invoking services. Marks and Bell [138] additionally propose the ESB for this task.

Concerning policy enforcement (as part of governing processes), all approaches propose control points that are part of or reside in (cyclic) governance processes. Techniques or concrete examples for automated policy enforcement (other than manual revision of artefacts) are provided by none of the approaches.

3.1.9 Metrics

More than a third of the proposed approaches mention a *metrics system* as an important building block for SOA Governance. Metrics, in general, are defined along with goals and make processes and parameters of the SOA system more transparent. The measurement of goals, combined with a corresponding management structure, supports the judgement on the effectiveness of the adoption of SOA. For the implementation of SOA Governance in a company, usually a set of goals are defined that is striven to be achieved (cf. Sect. 3.1.6). Metrics, in general, report on the performance of the SOA system as a whole, by measuring the goals set by the governance initiative [15, 27]. The following gives an overview of the purposes of metrics in SOA Governance defined by the respective authors.

According to Marks and Bell [138], “SOA metrics put a steering wheel on your SOA”. Metrics provide *performance monitoring* including, e.g., service behaviour, enabling technology, consumers and providers, and human participants. Due to numerous bad experiences that were made without metrics, they recommend considering metrics from the beginning of the SOA planning processes. They distinguish *business, process, and SLA and performance* metrics (e.g., ROI – Return On Investment), and *SOA Governance* metrics (e.g., conformance reporting, policy breaches, and developer exception reporting), *service reuse* metrics, and metrics for *service design enforcement*. While the last address the monitoring of *enterprise-wide design best practices* in order to avoid one-time design principles, reuse metrics service the enforcement of reusing existing and proven services, in contrast to novel development or the reuse of rogue services. Together with policies, the metrics model forms the *behavioural model*, that targets employee behaviour [138] (cf. Section 3.1.4, Roles and Responsibilities).

According to Brown et al. [27], one of the key responsibilities of SOA Governance is the incorporation of measurements of general SOA goals, such as the alignment of IT initiatives to business goals, in order to foster reusability on the one hand, and to decrease the total cost of ownership (TCO) on the other hand. The authors outline two ways of defining metrics when implementing an SOA Governance approach. The bottom-up method starts with an operating model (e.g., “operational excellence”, “product leadership”), and proceeds via the definitions of business goals, SOA governance goals, SOA governance metrics and measurement. The top-down approach defines metrics proceeding from the SOA Governance goals. The authors propose a list of 12 goals and metrics as a starting point for the adoption of an SOA system. These include, among others, service reuse, service TCO, and business IT-alignment through SOA. Brown et al. [26] and Holley et al. [81] consider measurement a key to effective governance. Unmonitored processes will soon be unable to meet business objectives. Monitoring in governance supports the prediction of increasing needs, as well as determining the TCO of services.

Bernhardt and Seese [15] define metrics for *services*, *service operations*, and *projects*. Service metrics report on the number of existing services, services being developed, proposed, published, consumed, changed, deprecated, and retired. They measure consumer numbers per service and service version. The latter is especially important in the context of service changes. Service operation metrics support the general visibility and the identification of improvement potential in service operation. They report on numbers of service interaction (correct vs. faulty) as well as reasons (e.g., protocol errors, size and correctness of input data). Further, the authors describe performance KPIs for identification of infrastructure bottlenecks and service design revisions, as well as service level and security violations (being service operation metrics). Finally, project-related metrics provide insights concerning project success, failures, and the corresponding reasons.

Common service metrics such as “degree of service reusage”, “service availability”, and “service downtime” are integrated by Kohnke et al. [116]. According to the authors, as objective of SOA Governance, “SOA Compliance” addresses the requirements for internal finance controlling and reporting systems. Starting with the adoption of SOA, for example, identity and access management is to be extended from human-computer interface supervision to computer-computer interface supervision [116].

Authors of the SOA Governance Framework by The Open Group [248] emphasise the importance of monitoring the SOA system, as adjustment of SOA Governance methods becomes increasingly difficult without information on IT system performance. They provide further exemplary metrics, such as “percentage of development trouble tickets caused by inadequate design”. However, they do not assign these metrics to specific IT or governance goals.

Using the term “Control SOA”, Schelp and Stutz [215] shortly subsume the continuous assurance of agreed upon performance and functionality specifications (e.g., SLAs). “SOA Quality Management” deals with the optimisation and continuous improvement of these service levels. “SOA Risk and Security Management” addresses potential flaws and violations of these specifications. The authors point out that, when further developing the overall architecture, it is important to check to what extent the IT goals (e.g., high service reuse) are allowed for.

According to authors at Software AG [233], the according organisational entity (e.g., the “Governance Team”) establishes SOA metrics to measure SOA effectiveness (e.g., service reuse). They suggest sufficient tool support for SOA metrics, and recommend to re-

ward corresponding organisations or employees for good rankings in metrics, especially in service development.

Summarising, improving the assessment of achievement of SOA goals by the definition of a metrics system is considered an important aspect of SOA Governance by all authors mentioning this issue. Most of the authors especially emphasise the management of service operation, service statistics, project performance, and the relationship to employee behaviour to be important in the context of metrics for SOA Governance. Further, the measurement of service reuse seems to be an important aspect.

3.1.10 *SOA Maturity Measurement*

According to Windley, implementations of governance that are not adjusted to the scope and maturity of an SOA system cannot display their full effect: either they exercise too few control, or they limit the involved persons by an overdose of regulation in their freedom of action and possess a demotivating effect [265, 267]. Governance methods and procedures are to be planned proactively, in order to keep up with the development of the SOA system and enable controlled growth. Documentation of the planned development as a roadmap is an often proposed method to keep track of the state and development direction of the SOA system (cf. Sect. 3.1.6). In order to assess the current maturity of an SOA system, SOA maturity models have proven useful [1, 3, 97, 217, 233].

A number of maturity models for Service-oriented Architectures (SMM) have been proposed in the last years [229, 237, 240]. Generally, SMMs assess an SOA system in terms of its maturity [97]. They define several levels, each specifying goals and metrics that determine and verify the current progress and the system's status. In combination, these requirements are called *maturity levels* of an SOA implementation. An SMM yields a profound indication whether an organisation is ready to introduce SOA, whether its SOA implementation needs improvement to meet minimum criteria, and what additional requirements are to be met to achieve a certain maturity level. In an SOA Governance model, the maturity assessment continuously delivers feedback to the corresponding organisational entities, where decision on the next steps in control of the SOA systems are made (e.g., abolishment and enactment of policies).

Most maturity models for SOA are based on the Capability Maturity Model Integration (CMMI) by the Software Engineering Institute [247]. Johannsen and Goeken [97] adopt the basic structure of a CMMI with its five common levels of maturity and expand the model by analysing the maturity along three characteristics: technology, processes, and organisation. For each maturity level and each characteristic a profile is available with criteria to be fulfilled on that specific level. The underlying assumption – that technology has to reach a higher level of maturity earlier than processes and organisation – reflects the fact that a given technology requires certain management processes, roles and responsibilities and often causes organisational changes [97].

Concerning the structural analysis, four examined approaches integrate SMMs into their SOA Governance approach [1, 3, 217, 233]. Schepers et al. [217] include an SMM to align the governance efforts with the current level of maturity in order to avoid unnecessary bureaucracy and the imposition of excessive governance procedures. They adapt the Service Integration Maturity Model (SIMM) [6, 7] (which is in turn an implementation of the CMMI [247]) to their SOA Governance life cycle. Authors at Software AG [233] integrate a simple maturity model into their approach that is based on the CMMI and

recommend a “step by step” introduction of SOA. Afshar [1] instrumentalises a maturity model for the adoption, as well as, for the continuous further development of the SOA. As a “SOA Life Cycle”, he defines six steps that are performed per maturity level and, once accomplished, guide to the next level of maturity. This approach explicitly combines a procedure model for the SOA system with a maturity model for use in practice [1]. Allen [3] proposes an SOA Governance Maturity Roadmap for governance capability definition, and risk identification. Along typical maturity levels, he aligns development steps of the SOA system along with governance capabilities that may “reasonably be attained stage by stage”. By applying a per-risk structure, this method directly aims at avoiding identified risks in a roadmap manner.

Further approaches make suggestions in the direction of maturity assessment, however, without naming it [15, 17, 27, 74, 75, 138, 248]. The assessment of processes against given quality criteria gives insights concerning the overall quality and maturity of realisation. Additionally, it is recommended to perform risk assessments of the SOA project and its subprojects. Once areas of high risk are identified, they can be adequately tackled and monitored by governance methods in order to minimise them [17, 27, 198, 215]. Further, metrics are recommended for monitoring the added value of the SOA in order to identify problems and counter them timely [15, 75, 138] (cf. Sect. 3.1.9). Marks and Bell [138] recognise governance as the chance, to constantly keep a holistic overview of the SOA system.

Overall, SOA maturity is explicitly considered in four out of 22 approaches, where three mentions come from the practitioner’s domain, and one from academic work. As SMMs are already widespread and well-known instruments of SOA Governance, it seems astonishing that the integration of maturity models into SOA Governance is proposed by a minority of authors. Obviously, only few authors recognise the benefits of maturity measurement in the context of SOA Governance. However, several additional authors proclaim SMM-related methods. So it might be a lack of awareness that causes the little assignment of maturity models to governance.

3.1.11 *Further Aspects*

In this section, interesting aspects are discussed that are mentioned marginally by the approaches.

DELIMITATION AGAINST MANAGEMENT. A legitimate question in the context of SOA Governance is, whether the control of SOA projects, of the people involved, and the developed services should be a management duty, and thus the introduction of SOA Governance as a new discipline is obsolete.

Schelp and Stutz [215] warn that short-term objectives of projects are often in conflict with the long-term goals and the value maintenance of an SOA initiative. So when in doubt, project managers make pragmatic choices in order to, for example, meet deadlines or budgets. In this case, superordinate goals such as reliable service quality and service reusability, are neglected for the short term. This means that management itself must also be subject to some scrutiny.

Weill and Ross [263] state:

“IT Governance is not about making specific IT decisions – management does that – but rather determines who systematically makes and contributes to those decisions.”

SOA Governance is considered a framework to support decisions that ensure the long-term value, or a “super-management” [241]. If in doubt, decisions made based on SOA Governance for the short term might not be the best ones.

SERVICE INFRASTRUCTURES. An SOA implementation typically has several layers, from the corporate data sources up to the interfaces that interact with the user. According to Afshar [1], this offers an alternative to the prioritisation of business processes, and allows to quickly achieve a high level of reuse. Services are implemented on a low technical level, such as data services encapsulating databases. These services provide abstract functions and are reusable in the context of many applications. Further, the company develops a system of hierarchically arranged infrastructure services in this manner, so that for new applications only few services must be implemented at higher levels of abstraction. This way, applications can be composed quickly and efficiently and the overall reuse rate quickly increases.

CULTURAL ASPECTS AND COMMUNICATION. Many measures and policies of SOA Governance aim at the conduct of persons and their participation in value-added processes. Since involved employees will not support governance measures against their own setting in the long term, Schepers et al. [217] point out that a change in individuals’ attitudes is necessary. Therefore, in the long run, SOA Governance will also affect corporate culture. Here, a parallel optimisation of corporate and SOA Governance, as proposed by the Software AG [233], can be beneficial. Brown et al. [27], Kohnke et al. [116], and Windley [267] recommend bonus systems, that financially reward SOA Governance compliant behaviour.

The comprehensive communication of SOA Governance measures in this context can also support a rethinking of the involved parties. Special communication measures are core aspects of some approaches [1, 74, 75, 266]. Communication, here, must work in two directions. On the one hand, measures and their purpose are communicated to the parties concerned. On the other hand, their opinion must be directly incorporated in the planning process of SOA Governance.

When referring to SOC as “Service-oriented Culture”, as done by Marks and Bell [138], extensive communication and motivation of employees is therefore one decisive factor. Cultural aspects and communication can be considered definite key aspects of SOA Governance.

Following the discussion of the 10 components of SOA Governance (cf. Sect. 3.1.1 to 3.1.9), in the following, we discuss definitions that have been proposed by the mentioned approaches, and suggest a comprehensive definition.

3.2 *A Definition for SOA Governance*

SOA Governance is not a consistently defined discipline. The approaches give several definitions of SOA Governance that diverge in coverage of aspects and focus. Among recent definitions, we identified major dimensions that are frequently used. Based on

these, we formulate a comprising definition. For a detailed overview refer to Appendix B.3.

The adjustment of establishment of new *organisational structures* is considered a central element of SOA Governance. Organisational structures and roles complement processes by the specification of formal accountabilities and competencies. Some definitions focus on the organisational embodiment of SOA Governance [3, 15, 17, 57, 138, 210].

A further aspect is the reference to the general *achievements of goals* such as IT goals and business goals (e.g., [3, 15, 138]). The definitions by Schelp and Stutz [215], Rane and Lomow [198], and Bieberstein et al. [17] emphasise the alignment of SOA Governance with company objectives. Further definitions include the improvement of conditions “allowing an SOA to grow” [109] and “ensure SOA success” [1]. Most frequently, the approaches focus on the integration and definition of special *processes*. More than 70% of the definitions consider specific processes a central important component of SOA Governance (e.g., [1, 116, 138, 267]). This refers to organisational processes [3, 15] as well as to production processes [233]. *Policies* are a further important asset of SOA Governance definition. They specify, e.g., mandatory company-wide guidelines for service development and operation [25, 210]. Some approaches define SOA Governance as an extension or enhancement of IT Governance. Among these authors, however, it is unclear, *in what way* SOA Governance is integrated in the context of IT Governance. Historically, SOA Governance can be seen as related to IT Governance, or even considered the application of IT Governance on SOA systems. Bloomberg [20] defines “[...] it is how IT Governance should operate within an organization that has adopted SOA as their primary approach to EA.” Authors at The Open Group [248] state: “SOA Governance should be viewed as the application of Corporate Governance, IT Governance and EA Governance to Service Oriented Architecture.” As also propagated by IT Governance, an SOA system needs to be strategically well-aligned to business goals in order to generate added value (e.g., by service reuse). Directly involving business departments, SOA clearly exceeds the boundaries of IT departments. Concluding, the origin of SOA Governance might be IT Governance – however, SOA Governance clearly exceeds the applications domains of IT Governance.

Based on the previous considerations, we provide a formulation for the integration of all aspects that obviously are covered by SOA Governance. We define the term as follows (cf. also Niemann et al. [161, 167]).

Definition 1 (SOA Governance). *“SOA Governance is a management discipline that provides processes, policies, metrics, supporting systems and tools, and defines organisational structures and decision rights in order to successfully control and reliably operate an SOA system. An SOA Governance approach – as extension of IT Governance – is tailored to the special challenges and abilities of an SOA system in order to support the achievement of IT, business, and corporate objectives, as well as the compliance to regulations and standards.”*

According to our definition, a governance approach focuses on the smooth adoption and successful operation of an SOA in a company. It provides guidelines and mechanisms to ensure the integrity of an SOA and its adaptability to business and administration processes. Governance tools (such as artefact management systems, cf. Sect. 3.1.3) support the monitoring and control of services concerning their alignment to business processes, security issues, conformance with policies. All of these procedures are supported by a policy catalogue storing best practices that are continuously supplemented.

Besides supporting the achievement of IT goals and the realisation of business-IT alignment, an important objective of SOA Governance is to achieve adherence to legal and normative regulations and corporate standards by the SOA system (e.g., security standards, (inter-)national legal regulations, ISO norms, and corporate guidelines). A central important goal is the realisation of the additional added value an SOA can provide compared with monolithic enterprise architectures, for example in form of financial advantages and increased agility of the company's business and IT processes.

Our definition is founded on the insights of the comprehensive analysis of SOA Governance approaches, the results of which are outlined in Section 3.3. It considers all aspects of prior definitions of this discipline due to a concise investigation of definitions (cf. Sect. B.3). Compared to existing definitions, the clear advantage of our comprising definition is this validity.

3.3 Analysis Results – Summary and Conclusions

In this chapter, we compared the structure and core aspects of several approaches that first structured SOA Governance. We investigated 22 approaches that were published in 35 different publications by three author groups: journal articles and books, contributions by software vendors, and published experiences from IT consulting. As a result, 10 major components have been identified, that most of the authors make use of to compose their approaches. The detailed results have been described in Sections 3.1.1 through 3.1.11. The first version of this analysis has been published in Niemann et al. [161, 162], subsequent versions in Niemann et al. [164, 166, 167] and Janiesch and Niemann [91].

Throughout the approaches, different structures and types of presentation have been used. Clearly, the majority of approaches is dominated by a *legère*, informal presentation language. Statements such as “SOA Metrics put a steering wheel on your SOA” [138] do not explain much and can only be understood in clearly outlined broad contexts. These types of explications are used by practitioners as well as authors of books and journal articles. Further, as part of those explanations, often non-self-explaining terms (e.g., “SOA Repository”) are utilised without further explanation. Most authors seem to avoid clear language. For these reason, two levels of quality had to be used for the analysis: *founded recommendations* and *proposals* (cf. Sect. 3.1).

The approaches do not usually adhere to the consistent criteria structures used throughout the presented analysis. Most approaches use as a main criterion either organisational means, SOA goals, or governance guidelines. In most cases, one important aspect is selected, and other (equally important) ones are presented in a cross-sectional way. Among most approaches, the obviously inherently multidimensional nature of this area is simplified and reduced to few structuring criteria in most cases. However, no reasons are provided for the selected and presented structure; the choices of the main criteria seem arbitrary.

The analysis results are summarised in Figure 3.5. It outlines the relative number of mentions of the components for all three author groups (considering both quality levels, proposals and founded recommendations). For example, a value of 50% for component *A* shows that 50% of the approaches in this author group have pointed out component *A* to be important (i.e., has been at least classified as *proposal*). For further result details, refer to Appendix B.6.

As a general result, the four most important components of approaches to SOA Governance are considered the support by *Governance Policies*, the setting up of new *Organisational Structures*, introduction of new *Roles and Responsibilities*, as well as software support for *Artefact Management*. None of the approaches neglects three out of these four components, and only one approach does integrate less than three of these four components. 68% of all approaches integrate at least two of these four as *founded recommendations*. In fact, a majority of approaches (12 out of 22) integrates all four of them.

GOVERNANCE POLICIES. According to our structural analysis, catalogues of *Governance Policies* containing guidelines, conventions, as well as best practices are the most frequently integrated components of SOA Governance. To the *policy*, there seems to be no adequate alternative mechanism with comparable abilities. As a consensus of all authors, policies are considered mighty instruments that combine various application aspects. Applications are *roles-related*, *service design and operation-related*, and, explicitly, *service documentation-related* policies. Important aspects of policy handling are policy life cycle management, policy exception regulation, and recognition of inappropriate regulations, such as too restrictive policies.

ORGANISATIONAL STRUCTURE. Changes in *Organisational Structure* and the introduction of new SOA-specific *Roles and Responsibilities* are considered crucial by almost 70% of the examined approaches. As one of three discussed councils, all of the authors agree on establishing an SOA CoE that bundles many of the discussed competencies (e.g., concerning SOA implementation and operation) – in some cases even all of them. Apparently, this institution has convinced in theory (academic approaches, seven mentions) as well as in practice (eight mentions from IT consulting and software vendors). It can be considered a generally required organisational institution for the operation of an SOA system.

The two further councils, the *SOA Governance Board* and the *SOA Board* are often defined in a mutually excluding way, i.e., approaches defining the first one omit the latter (e.g., [3, 195, 198]). The focus of the respective competencies, however, stays the same. This shows, that those competencies, SOA Governance and strategic leadership, are often considered to be replaceable. By the majority of approaches, however, strategic leadership is commonly regarded an integral part of SOA Governance. This shows a lack of holistic understanding of this discipline, even on the part of the “providers” of SOA Governance.

ROLES AND RESPONSIBILITIES. By the majority of approaches, five employee roles are pointed out that are specific to the operation of an SOA system. Further, in the context of SOA Governance, an important aspect is the targeted *impact on behaviour*. It is considered a necessity to influence the employees’ behaviour in favour of the new system, for example by trainings or provision of incentives. Obviously, the classical IT Governance goal to “achieve desirable behaviour in the use of IT” [263] is an important goal for SOA Governance as well. A further important aspect is the more intensive active involvement of employees in business departments into specific planning activities of business services and processes. According to the majority of authors, an SOA system can only be successful by comprehensive communication and cooperation between the departments, as well as, by the abolishment of static organisational boundaries between business and IT departments.

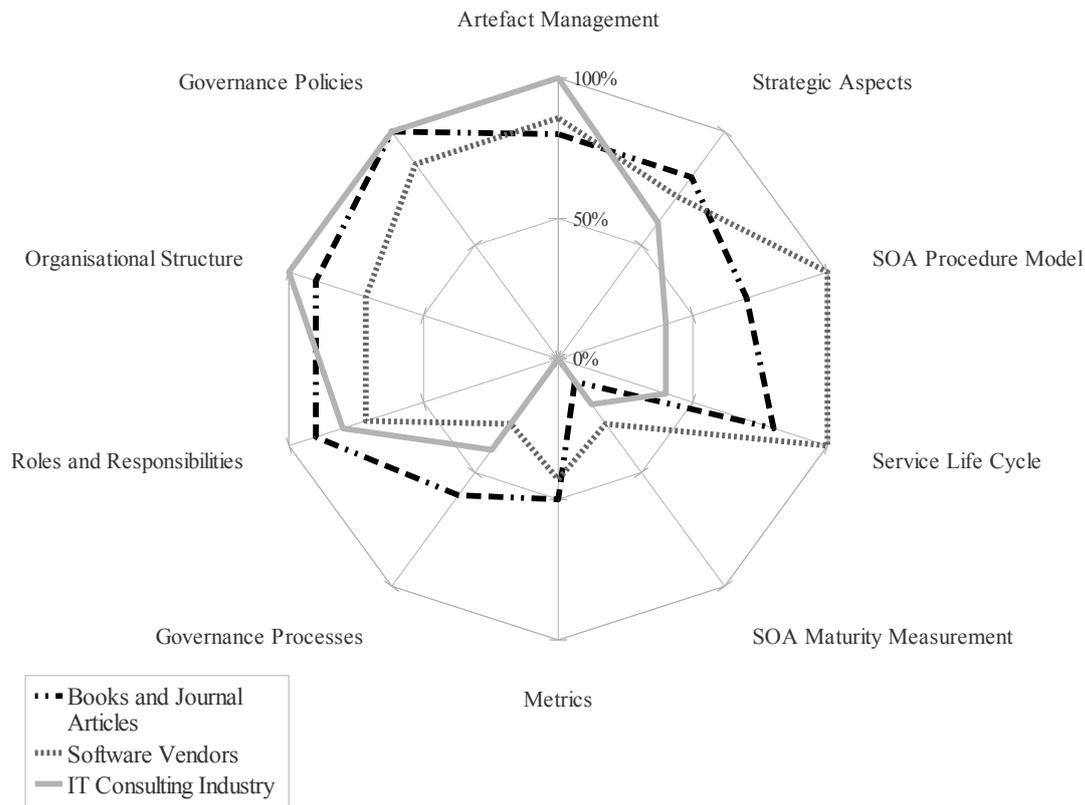


Figure 3.5: Integration of components per author group and quality level

ARTEFACT MANAGEMENT. Concerning the *administration of meta data and artefacts* (e.g., service descriptions, governance policies), the approaches suggest the operation of a service registry or service repository, as well as a Web service management system. Astonishingly, the understandings of service registry and repository in terms of functionalities diverge. As main function of a service registry, a majority of authors, however, refers to publishing and discovery of services, while the service repository is mainly considered to serve as meta data storage. Further, it is recommended to structure all kinds of artefacts in a meta model to clarify definitions and relationships. The establishment of additional data- and artefact-related roles and responsibilities is advised as well. As to the question, what meta data is to be stored for respective services, however, the authors make diverging recommendations.

Generally, for the components *SOA Procedure Model*, *Service Life Cycle*, and *Metrics*, the diagram shows the largest deviations between the number of mentions per author group. Interestingly, the first two are the strengths of the software vendors' proposals. Further, considerations concerning *Metrics* and *SOA Maturity Measurement* were integrated by less than 50% of all 22 approaches. Concerning *SOA Maturity Measurement*, however, most authors do not name it, but implicitly suggest techniques of maturity measurement (cf. Sect. 3.1.10). Overall, the diagram underlines the different strengths, focal points, and opinions of the respective author groups.

Books and journal articles provide the largest amount of approaches considered in the analysis (ca. 45%). 80% of the components (eight out of 10) have been integrated by 60% or more of the approaches proposed in this author group. As least mentioned components, *Metrics* are included by still 50% of the approaches, and maturity consideration by 10%. However, the latter has been considered by a clear minority of authors overall. Summarising, authors of journal articles and books show a good coverage of the components in average (cf. Fig. 3.5).

Software vendors provide ca. 32% of the approaches considered in the analysis. In particular, the components *SOA Procedure Model* and *Service Life Cycle* are emphasised by this author group. Each further component is included by at least 35% of all software vendors' approaches. Apart from three components (*Governance Processes*, *Metrics*, and *SOA Maturity Measurement*), all are integrated by ca. 75% of the approaches. The least considered component *SOA Maturity Measurement* is still integrated by at least 35% of the software vendors' approaches. Approaches from this domain provide, in comparison, the most balanced coverage of components. This might be due to the fact that they bear the best relation to both theory and practice of SOA Governance.

Circa 23% of the considered approaches originate from the IT consulting author group. These contributions show a clear unilateral focus on *Artefact Management*, *Governance Policies*, *Organisational Structure*, and *Roles and Responsibilities*. Besides *Strategic Aspects*, all further components are mentioned by clearly less than 50% of the IT consulting approaches. Especially the aspects *Metrics* and *SOA Maturity Measurement* are poorly mentioned. Authors from this group show the smallest coverage of components. This might be due to the fact that they provide the smallest number of approaches here. These, however, show a clear agreement on important components.

3.4 Conclusion

As major result of the analysis, we identified the four most important components of SOA Governance: *Governance Policies*, *Organisational Structures*, *Roles and Responsibilities*, and *Artefact Management*. Generally, concerning organisational issues, a *SOA Centre of Excellence* as the central coordinating organisational institution is considered necessary in unison by the experts. Further, it is considered to be of central importance to influence employee behaviour in favour of the system. In this respect, the classical IT Governance goal "achieve desirable behaviour in the use of IT" [263] proves valid for SOA Governance in particular.

The analysis covered approaches by three author groups. These different perspectives on SOA Governance all emphasise different aspects. Scientifically published approaches integrate eight out of ten components on average, representing rather holistic approaches. Software vendors clearly emphasise technology-oriented components (e.g., *SOA Procedure Model* and *Service Life Cycle*), and expose the most balanced coverage of components. This indicates a good relation to the practice of SOA Governance. Approaches from IT consulting industry show an unilateral focus on a small number of components: *Artefact Management*, *Governance Policies*, and the two organisational issues *Roles and Responsibilities* as well as *Organisational Structure*.

The building block *SOA Maturity Measurement* is among the least frequently integrated components. However, we observed that most approaches integrate aspects of maturity

measurement without mentioning it explicitly. This reveals a lack of awareness regarding maturity measurement of SOA systems among the expert authors.

Concluding, the authors agree in unison on the necessity of SOA Governance (the “SOA Governance imperative”). Based on common characteristics of SOA systems and the emerging challenges, the installation and operation of governance approaches for SOA are considered essential. In this context, the authors emphasise in particular the management and unification of SOA-inherent heterogeneity and complexity on the one hand, and on the other hand the regulation and exploitation of new capabilities such as cross-organisational service deployment.

The analysis showed that most approaches are characterised by a “tunnel perspective” (as attested by Allen [3]), limiting the focus on selected issues. Only few of them expose holistic perspectives. In contrast, the majority of authors agree that a *holistic* governance approach is crucial for SOA Governance. Clearly, this shows a lack of awareness of the scope of SOA Governance. In the introduction of this Chapter, we referred to a survey that revealed a big disaccord in perception of SOA Governance by companies that operate an SOA system (“requestors” of SOA Governance). Now, our structural analysis shows a big gap between understanding and action in this domain – from the opposite perspective: experts that propose SOA Governance approaches (“providers” of SOA Governance). In combination, as a general result of our analysis, a general lack – or at least a definite disaccord – of understandings of the term *SOA Governance* becomes apparent, and reveals an area of general improvement needs, in particular concerning its understanding and the discussion of its definition, as well as its realisation approaches and components.

Additionally, we exploited the revealed insights and, addressing the above mentioned needs, developed a novel comprehensive definition for SOA Governance in this chapter.

A SERVICE LIFE CYCLE AND AN OPERATIONAL MODEL FOR SOA GOVERNANCE

DEFINING a life cycle for services is a powerful method to control the design, implementation, operation, and version management, i.e., the “life” of services. As the analysis of SOA Governance approaches in Chapter 3 revealed, in unison, all authors consider *service life cycles* powerful, central instruments of SOA Governance. By formalising it, the process of development and operation of services can be efficiently controlled and governance measures such as checkpoints can be placed and applied in an effective way. Looking at the details, however, reveals dissonances concerning definition and integration of the defined phases (e.g., superphases like design time, runtime, change time and subphases like design, deploy), as well as the cyclic behaviour of a life cycle. In this chapter, we develop a consolidated service life cycle that allows for the particularities of eleven different proposals for service life cycles.

Further, *procedure models* combine many of the typical instruments of SOA Governance and define an institutionalised way (also called “SOA Governance Roadmap”) to support the controlled introduction, operation, and development of an SOA system in a company. According to our analysis, they rank on rank 4 concerning the authors’ *founded recommendations* (cf. Fig. 3.1). However, none of the investigated procedure models is aligned to the main components of SOA Governance (cf. Sect. 3.1.6). In this chapter, we present an *operational model* for SOA Governance that has been developed based on the results of the analysis performed in Chapter 3, integrating the main aspects of SOA Governance.

In the remainder of this chapter, we present (i) a consolidated service life cycle for application in SOA Governance approaches, based on an analysis of service life cycles (cf. Sect. 4.1), and (ii) a generic operational model for SOA Governance (cf. Sect. 4.2). We draw final conclusions in Section 4.3.

4.1 A Service Life Cycle for SOA Governance – Survey and Approach

A variety of different approaches regarding Service Life Cycles (SLC) have been developed and used by academia and software companies. Very often, the distinction between design time, runtime, and change time is made – each of them covering a number of different life cycle phases. In this section, we outline and compare existing life cycle approaches and challenge the purpose of the distinctions made between these three ‘times’. Based on this discussion, we present our SLC approach that omits the change time aspect and, nevertheless, maps all SLCs discussed in related work.

Services are part of the *SOA System*. It represents the IT system to be controlled and consists of SOA processes, such as service production, operation, maintenance, including the corresponding business processes. The technical backbone is a central part, representing the actual architecture including registries, repositories, and the enterprise service bus (ESB). A SLC, generally, is part of the organisation of the SOA System.

In general, life cycle processes originate from the software engineering discipline. Software engineering processes define any software's life cycle from requirements analysis to implementation, operation and maintenance. The main purpose is to ensure a structured software development process in a well-defined and cost effective way [132, 236]. The typical life cycle (e.g., in the "waterfall model" [236]) consists of these phases: requirement analysis and definition, software design, implementation and testing, integration, and operation and maintenance [89, 132, 228, 236]. It covers (cf. Fig. 4.1):

- Phase 1 - *Requirements analysis and definition* outlines the software system's purpose, constraints and goals in cooperation with the system users. These aspects compose the system specification.
- Phase 2 - *Software design* distinguishes between hardware and software requirements and defines a general architecture. It covers the identification and description of software components and their relationships.
- Phase 3 - *Implementation and testing* comprises the realisation of the defined design. It involves testing to ensure that the software meets the agreed upon specifications.
- Phase 4 - *Integration* includes system tests to ensure that interaction with other software components works faultlessly. Furthermore within this phase, the software is delivered to the customer.
- Phase 5 - *Operation and maintenance* is usually the longest phase. It includes software usage and monitoring, as well as error-correction processes. Should it become necessary to implement new requirements, a change process is triggered in this phase.

These phases are designed to form a loop. As soon as changes in functionality or requirements are registered, the phase *operation and maintenance* is interrupted and the life cycle continues with the first phase, requirements analysis, or any other phase of the life cycle that can directly or better address the occurred problem [236] (cf. Fig. 4.1).

Service Life Cycle Management (SLCM) targets the steering, direction, and control of all services of the system in all life cycle phases. The goal is to assure manageability and conformity in spite of large service numbers. Important aspects, for example, are change management procedures, service deployment, service granularity, and consumer integration. The views on the division of activities in phases of the life cycle differ widely in approach (cf. Sect. 3.1.5).

Life cycle models are widespread means of structuring the development and operation of services. Services as part of an SOA system are also software artefacts, however, their functionality scope is rather small and they appear in multiplicity. Obviously, however, different conditions are relevant when defining a life cycle for services. Services are delivered to the customer and deployed in different environments and contexts compared to common (monolithic) software. SLCs differ from common software engineering life cycles [25, 139, 267].

SLC approaches are often divided into three superphases: *design time*, *runtime*, and *change time* [68, 139, 246]. The life cycle phases up to deployment are commonly referred to as design time. The usage phase is referred as runtime, followed by the change time, which addresses service change and revision (cf. [25, 68, 139, 246]).

In SLCs, a general method of control is the definition of *checkpoints* or *policy enforcement points* [233, 248]. At dedicated points during the life cycle, the current work products

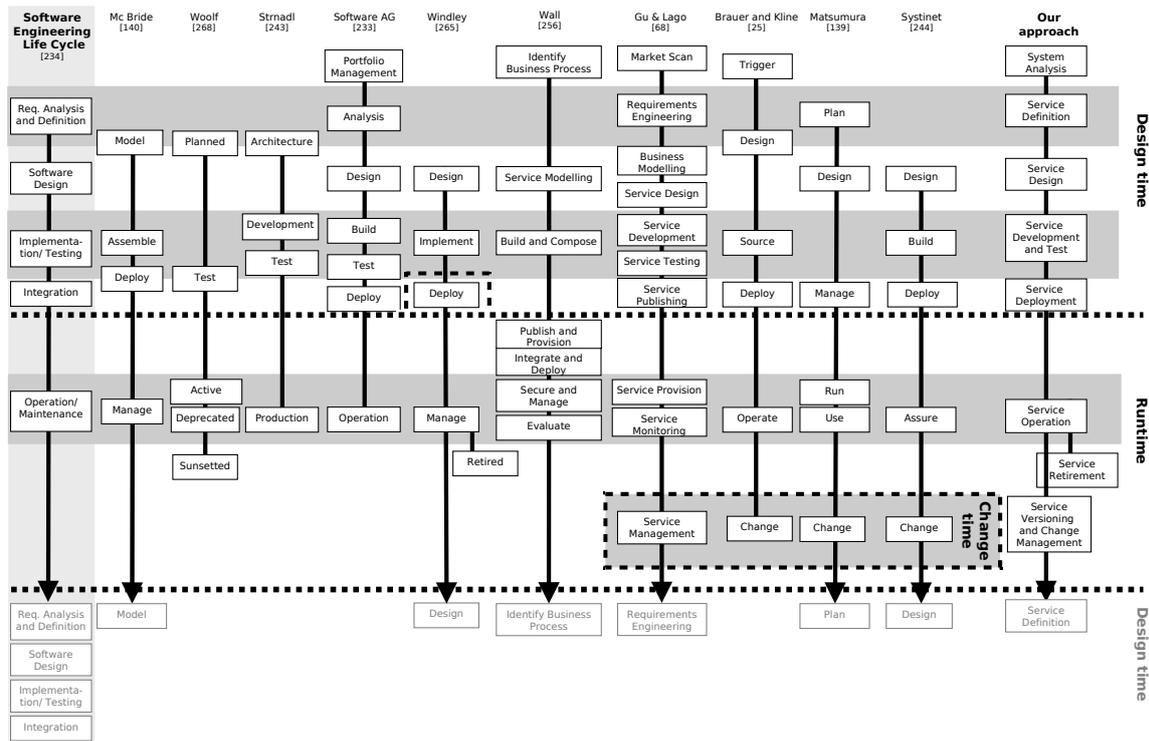


Figure 4.1: Survey on Service Life Cycles (cf. Niemann et al. [163], Fig. B.5)

are checked concerning the fulfilment of predefined requirements. If positive, the life cycle of this specific artefact can continue. Examples for checkpoints are reviews of an architecture integration, service source code, service metadata, or the check concerning the conformance to guidelines [248]. In governance approaches, usually, life cycles are combined with checkpoints, defined as formalised processes. This also involves roles, groups, or boards like the SOA CoE [3, 27, 198, 233, 248].

We conducted a survey of ten suggestions for SLCs envisioned by industry and academia. In the following, we introduce and discuss these approaches. We investigate whether and how the super-phases design time, runtime, and change time are addressed. Further, we performed a matching of all presented life cycles to the classic software engineering life cycle model. The results are outlined in Figure 4.1. The white and grey lanes mark the phases that correspond to one of the software engineering life cycle phases. State-intersection could not completely be avoided; however, it has been minimised. Dashed lines indicate the super-phases design time, runtime, and change time.

Based on the findings of this survey and the approaches presented in Section 3.1, we propose a consolidated SLC model in this section in order to unify related efforts. We present a SLC consisting of two super-phases (cf. Fig. 4.2). Each service traverses the three common phases design, run-, and change time. Each life cycle phase consists of activities, alternating with checkpoints (e.g., for reviews). Overall, the SLC approach consists of eight phases: system analysis, service definition, service design, service development and test, service deployment, service operation, service versioning and change management, and service retirement. Excluding the last three, all phases are part of design time. Service operation belongs to runtime, while versioning and retirement are part of change time (cf. Fig. 4.1).

In the following, the particularities of the phases design time, runtime, as well as service change, and their relationships are outlined against the background of governance.

4.1.1 *Design Time*

Within the field of software engineering, design time covers the first four of five phases in the classical software engineering life cycle: *requirements analysis*, *software design*, *implementation/testing*, and *integration* (cf. Fig. 4.1, [236]).

According to Gu and Lago [68], design time covers the service design, satisfying the defined functional and non-functional requirements, refined service interfaces, and the style of interaction between services and their clients (asynchronous or synchronous invocation). Wall [258] emphasises service categorisation, modelling methodology, as well as service building and composing concepts as essential parts of design time. According to Software AG [235], design time comprises requirements analysis concerning the services' behaviour, performance, Quality of Service (QoS), and security. HP proposes a "Trigger" phase as a pre-design phase [25]. Most approaches treat deployment as a part of design time. Authors at Oracle, however, include deployment in runtime [258, 259]. Windley [267], additionally, defines a "deploy time".

All remaining approaches define three activities in design time: design, development, deployment or test. These represent a subset of the software engineering activities in design time that can be mapped [25, 139, 140, 245, 246, 271]. Summarising, design time covers all activities of the software construction process that take place before and including service deployment.

Our consolidated approach comprises five phases for design time (cf. Figs. 4.1 and 4.2).

SYSTEM ANALYSIS AND SERVICE DEFINITION. The entry phases are *system analysis* and *service definition*. Fundamental decisions concerning the system or a particular service and its scope are made, respectively. Services are identified, their granularity is determined, and they are prioritised. These are the main differences compared with the software engineering life cycle. When introducing a service life cycle, existing services are identified, for example, by service inventory projects [1, 201].

SERVICE DESIGN. Once the service definition is accomplished, the *design* phase starts, in which basic software design procedures concerning, e.g., interface definitions and design principles such as communication standards and code reuse are applied. Important guidelines during this activity are *reference architectures* and *design guidelines* (cf. Sect. 3.1.1). At the checkpoint *design review* following the design phase, the adherence to these regulations is verified.

SERVICE DEVELOPMENT AND TEST. The subsequent phase *service development and test* basically covers the implementation and service testing. During *service design* and *development*, especially architectural requirements and technical standards are important (*reference architectures*). Further, aspects such as the responsibility for implemented services or the scope of documentation are to be clarified. For the latter, explicit *documentation guidelines* are defined (cf. Sect. 3.1.1). In *service tests*, the conformance with functional and non-functional requirements is to be checked.

The last step during the development of a service are comprehensive *service tests* concerning functional aspects (e.g., correctness), as well as non-functional aspects

(e.g., stability, security, performance). According to Derler and Weinreich [43], a test can be performed in three steps. After passing tests in an insular *test zone*, the service is tested in the *integration zone* using productive data. Having passed these tests, the service is deployed and put into operation. In this context, also the service design and source code (against the design guidelines) and the created metadata (e.g., the functional description), are subject to a review process [74, 75, 138]. *Test guidelines* are recommended that prescribe what criteria a service and its metadata are to be checked against [1, 15, 43, 233, 248].

Important regulations are development and documentation guidelines. At the *Code* and *Metadata Review* checkpoints, according adherence is checked. Finally, at the checkpoint *tests*, service testing is approved.

SERVICE DEPLOYMENT. Once the development of a service is accomplished and the tests have been passed, it is usually commissioned. In service systems, we consider a distinctive *deployment* phase that requires much more attention than in component development or *integration* in software engineering. In addition to activities typically performed in the corresponding *integration* phase in the software engineering cycle, this phase covers the installation on a server, which implies the delivery to the service host, registration in service registries or repositories, conformity checks to governance policies, final service tests, and the resulting final approvals (e.g., description and interface checks). This implies the transmission and installation of policies, documentation, and meta data of the service. Furthermore, when lacking corresponding guidelines and measures, there is the danger of a service being used in a closed SOA system without having been registered at a service registry or repository. These “rogue services” can have various negative consequences, such as incompleteness and inconsistencies of service registries, as well as duplicate implementation and provisioning of services [114, 217]. It is important to address this risk early, i.e., during design time.

Further, important aspects of design time are the management of service dependencies and reference architectures. Supporting service change management, it is considered useful to model *service dependencies* with each other as well as with further software libraries. It is important to preserve and assure the ability to determine what services and other software artefacts are affected, if services need to be changed. It is not sufficient to consider only dependencies between business processes, as many approaches propose [43, 138]. In fact, as a top down approach, it can be determined what services to change if processes change. Service-to-service relationship needs to be included. This risk is to be addressed early, during service design.

Reference architectures are considered useful concerning design and architectural guidelines. They provide a variety of technical standards and their combinations that can be used to implement services that can be deployed within the organisation. Usually, these standards are integrated in an exemplary implementation. It is important that the reference architecture is adequately and comprehensibly documented and developers are to be trained in their implementation [1]. A special kind of reference architectures are so-called *Enterprise Data Models*, defining a uniform perspective on the organisation’s data (e.g., types, structure, locations). According to some authors [17, 27, 75, 138], the adherence to a data model should also be mandatory for services in order to achieve complete and consistent service integration with the data processed by the SOA system

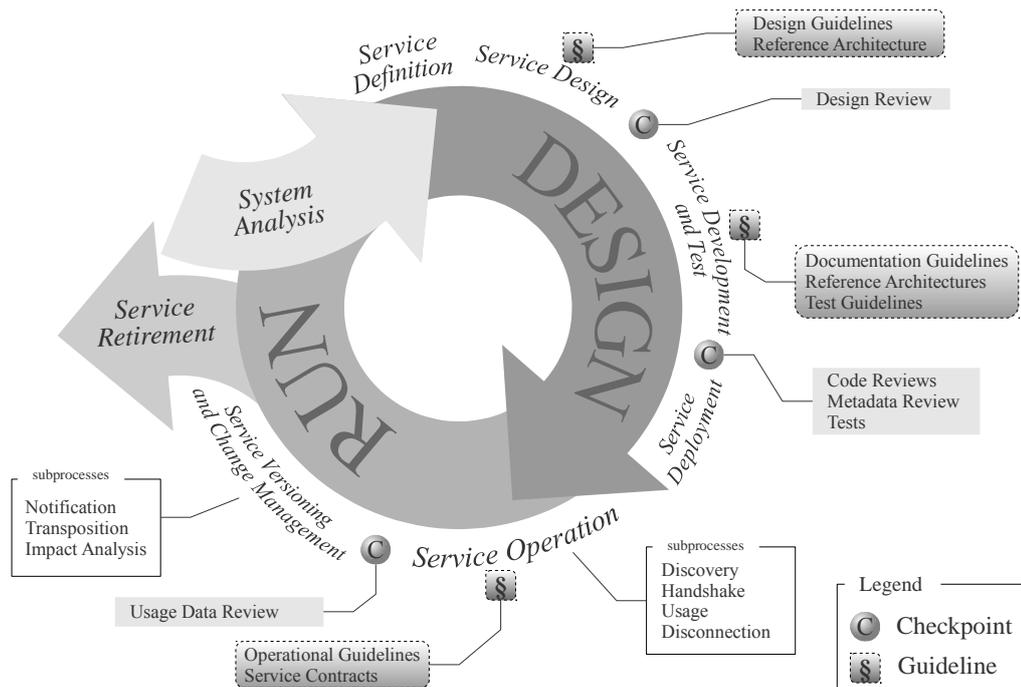


Figure 4.2: Consolidated Service Life Cycle

[1, 198, 248, 265]. The need of adherence to reference architectures is usually expressed as governance policy.

Summarising, concerning its sub-activities, design time is the largest of the super-phases. Concerning service deployment, two authors deviate in their definitions, lacking a discussion of the reasons. Based on the analysis of SOA Governance, we added important guidelines and checkpoints. Overall, apart from unspecific activity naming (“Model”, “Architecture”, “Trigger”), the activities of the software engineering life cycle are adequately adapted by the majority of the approaches.

4.1.2 Runtime

According to Sommerville [236], runtime covers phase 5 of the classic software engineering life cycle, *operation and maintenance* (cf. Fig. 4.1). This includes software usage, as well as maintenance and change management, and monitoring. In our consolidated life cycle, runtimes covers *Service Operation* and *Service Versioning and Change Management*. The first one is detailed in this section, while the latter is discussed in the next one, Section 4.1.3.

Service operation is usually controlled and regulated by *operational guidelines* (runtime policies) [210]. During operation, a service is controlled by these guidelines and service contracts between service consumer and provider (or intermediary), i.e., consumer and producer of data. The definition, control and check of adequate guidelines and service contracts lies within the competence of SOA Governance (cf. Sect. 3.1.1).

A service contract is the complete specification of a service “between one service provider and one service consumer” [101]. According to Erl [56], it establishes the terms of engagement, and comprises technical constraints and requirements, and any semantic information needed for its usage. More concretely, a service contract consists of a

WSDL definition, an XML schema, a WS-Policy description, and a corresponding SLA [56]. Concerning The Open Group [248], service contracts are to be checked before service commissioning, e.g., concerning adequate security levels (e.g., encrypted communication). By defining and using service contracts, the trust in services as well as the service reuse factor within an SOA can be increased. For this reason, it is a central duty of governance approaches to carefully consider this aspect in the SLC for governance [1, 3, 15, 27, 138, 198, 217, 233, 265].

As a part of Service Operation, service monitoring and their transactions during operation is a common procedure that is used to control service execution. Service characteristics that are monitored include frequency of invocation, the types of service invocations, response time, and availability, as well as any occurring problems [68]. The central purpose is the monitoring of conditions granted by the SLA [55, 101]. From the perspective of governance, this in particular covers the detection of undesirable, not authorised invocations [248], the retrieval of information that can be used for service improvements [138], the detection of errors and exceptions during service operation [17], as well as, the recording of the degree of reuse, and the profitability calculation per service [27, 195, 233]. Monitoring is considered to be an important aspect of service runtime, although it is mostly included in the service execution phase [139, 235, 259].

A governance programme is also in charge of the installation of technical support such as service management or service monitoring solutions. Mandatory guidelines regulate the logging and recording of runtime data. Further, processes for statistical exploitation of these data are to be established by the governance approach, defining data analysis and the way of generating improvement suggestions [1, 15, 17, 20, 27, 74, 75, 116, 138, 195, 233, 248]. A general check concerning these data is performed at the checkpoint *usage data review* (cf. Fig. 4.1).

In order to emphasise the customer's involvement during the usage phase, Gu and Lago [68], Matsumura [139], Systinet [246], and Windley [267] additionally present a consumer SLC, which consists of the four steps *discover*, *bind*, *use*, and *disconnect*. *Discover* describes a service, capable of fulfilling the customer's demand, being discovered within the service provider's registry. *Bind* involves contract management by the provider and customer, as well as the SLA handshake. The *use* phase describes the time until disconnection in which the service is actually used by the customer. If necessary, the customer life cycle can be reiterated, starting at the *discover* state [267]. Matsumura [139] defines the consumer SLC to regulate transactions between service provider and service consumer. These interactions include broker transactions, data transformations, message queuing, and security handshakes. Authors at Systinet [246] additionally define the phase *monitor*. In this phase, details concerning the quality of service performance are collected and reported to the provider. Woolf [271] distinguishes the states *active*, *deprecated* and *sunsetted*. The remaining approaches [25, 140, 245] define no more than one phase for runtime named *operate*, *production*, or *manage*, respectively (cf. Fig. 4.1).

Concluding, the understanding of runtime basically covers service operation including service monitoring. A consumer life cycle is also part of this phase, covering activities and phases involving the service consumer starting in the service operation phase. Operational guidelines and service contracts are the main given conditions. A further part of runtime is service versioning, change, and retirement – which is outlined in the next section, as it is often made part of “change time”.

4.1.3 *Service Change or Change Time*

In software engineering, a change time is not explicitly considered. However, considerations for system improvements or error correction are included [236]. Concerning SLCs, the importance of the service change procedure for the reliable operation of the SOA system is beyond dispute. However, realisation approaches vary.

All authors that define an explicit “change time” insert it as additional phase following the classical operation phase. For example, Gu and Lago [68] explicitly define a change time phase (cf. Fig. 4.1). The “Trigger” activity they mention is business requirements change which directly affects functionality adjustments. They argue that the ability to change services is required to meet user expectations and stay competitive. Nevertheless, they add that a life cycle reiteration is potentially necessary at this point in time. According to authors at Systinet [246], change time covers service version management and service adaptation to newly encountered security requirements. Following change time, a service restarts processing the life cycle. Matsumura [139] emphasises service version upgrades, operational change of services, service decommissioning, and service deprecation as central aspects of change time. In order to plan and design the outlined operational change, a subsequent reiteration of the cycle is required. According to authors at HP [25], change time is a continuous adaptation process following the runtime phase. Change time involves service delivery, service delivery management, and service version management. A reiteration of the proposed life cycle is not considered. Authors at Software AG [233] consider change time a crucial component of the SLC. However, they do not explicitly integrate it. Definitions of change time vary considerably. Given that only a small number (four out of 11) of approaches consider change time, this shows that there exist no similar understandings of this phase at all.

There are different ways to control procedures concerning service change for a governance approach. The definition of guidelines regulating the correct behaviour during shutting-down and service versioning [198], and the definition of formal processes for this SLC phase [217] are two complementing alternatives [1, 15, 43, 74, 75, 138, 195, 198, 217, 248].

According to the analysis performed above (cf. Sect. 3.1), SLCM is one of the most prominent aspects of SOA Governance. An important part of SLCM are change management procedures. In case a service is to be changed, this normally affects a large number of further artefacts, composite services, or applications. According to the survey, change time covers the management of service decommissioning, versioning, and retirement – including handling of functional changes, covering, i.e., service adoption, business process change, service portfolio change, service versioning, technical requirements change, and security issues. In our opinion, it organises the proper transition to the service definition phase in case a change is requested.

As soon as a need for service change has been identified, either a new service version is needed – or an additional service. In the latter case, a new SLC iteration is started targeting the generation of a new service. In the first case, the service development process is forked. The current service version continues being operated and is marked *deprecated*. A second instance is gradually taken off operation and reiterates the SLC for improvement. It passes the complete design time phase, similar to the development of a new service. As soon as this instance has passed the service deployment phase, the old deprecated version is gradually replaced by the improved one. The original service traverses

to the phase *retirement*. This process of preparing and performing the reiteration of the life cycle maps the change time described in the approaches above.

The relevance of definition of a “change time” in SLCs can be discussed indeed. In our opinion, as it includes a cycle reiteration in any case, and obviously in the opinion of seven out of eleven approaches, this procedure cannot explicitly be described as change phase or time. Change time is generally not considered to be a chronological phase as the previous two. It is, hence, not completely correct to call it “change time”, although it is located “at the border” of runtime. Concluding, change time, actually, is another name for life cycle reiteration.

In our consolidated approach, service change is handled by the phases *service versioning* and *service retirement* during runtime phase (cf. Fig. 4.1). The service versioning procedure can be divided into three major activities: *notification of the involved parties*, a *transposition* phase, and *impact assessment*.

All parties making use of a service to be changed are notified. This information is generally gathered during service commissioning and operation, i.e., there is no additional information required here. The availability of all required information is an important requirement for flexible service change. Generally, the notification is the duty of the service owner. It is important that all addressed parties reply to the notification, conforming potential implications or SLA changes. Following, the parties that perform the changes, are granted a time period for general *service transposition* and interface updates, as well as testing of the new service. The new service iterates along the life cycle phases. Afterwards, the old and the new version of this service are operated in parallel; the old one is marked as *deprecated*. It is in particular important that service registries and repositories support version management, and support that service changes are performed in compatible manner concerning all interacting roles (such as other services, persons, applications). For this, concise documentation during the prior phases is necessary (e.g., documentation of service dependencies at service design time), and documentation is accordingly checked (in case, e.g, if existing new services are entered in the registry).

Subsequently, an impact assessment of the changes is performed. Potential impact as well as risks are analysed for all services affected by the changes. Additionally, the impact on the complete system is estimated. Finally, the controlled shutting-down of the old services is performed (*service retirement*), and the new one’s operation can be approved.

4.1.4 Discussion and Conclusion

SLC is a central issue in SOA Governance (cf. Sect. 3.1.5). Life cycles and their management are crucial for the success of SOA Governance approaches. SLCs are adjusted to the specific needs, perspectives, and notions of a company’s IT. There are, for example, almost as many different life cycles as proposals [1, 10, 25, 233, 271]. As these various different definitions show, there is obviously no standard or generic life cycle that fits all perspectives.

We investigated eleven different approaches of SLCs. Approaches vary on a large scale in structure, definitions, and differences concerning important aspects such as change time consideration or cyclic behaviour. All approaches contain particularities that reflect experience in addressing the specific needs of an SOA system.

As reaction to these deficiencies, we propose a consolidated life cycle approach (cf. Fig. 4.1). The consolidated approach that integrates all hitherto existing findings and

allows for the particular requirements of an SOA system. The life cycle model is congruent to the classic software engineering life cycle and maps to all investigated approaches. It details most of the shorter cycles such as those by McBride [140], Windley [267], and Systinet [246]. As defined above, it covers the approaches by Woolf [271], Software AG [235], Brauer and Kline [25], and Strnadl [245], and extends them by the ability to reiterate the cycle. It summarises the longer cycles defined by Wall [258], Gu and Lago [68], and Matsumura [139] by consolidating detailed sub-phases.

In particular, our consolidated model maps the life cycles that define an explicit change phase or change time. It addresses the necessity for change in the service operation phase and the reiteration of the life cycle. In this respect, all phases in the change time box (cf. Fig. 4.1) become part of the phase *service operation* – the remainder of the change activities are covered by reiterating the SLC.

Seven out of eleven approaches we reviewed include neither a change phase nor change time. Even the classic software engineering approach does not involve it, although change of software components or systems has always been an important topic in software engineering. The original software engineering life cycle covers software change and versioning, but does not define or require a change time phase in the corresponding life cycle. Change of software artefacts is classically considered a subprocess of *operation and maintenance* and triggers a reiteration of the cycle. Every iteration of the life cycle after the first one represents a software change process. Basically, change time stands for and addresses the need to manage change activities that are required upon change requests originating from necessary functional changes. It implies addressing the change request in order to solve the occurred problem. As soon as the service itself is concerned, its requirements, definition, and design are reanalysed – in order to finally accomplish code change with implies testing. Obviously, this is equivalent to reiteration of the SLC, as defined by many of the approaches presented. In fact, Boehm [21] describes a spiral software development process that reiterates one cycle while addressing different evolving issues in each iteration. The approach by Brauer and Kline [25] almost implements this by specifying a change loop. Given that the operation and maintenance phase of the classic software engineering life cycle includes change handling, the definition of a change phase or time seems unwarranted. It is assumed that it can safely be replaced by a cycle reiteration. Additionally to Matsumura [139], Systinet [246], and Gu and Lago [68], this also exceeds the approaches by Software AG [235], Strnadl [245], Woolf [271], and Brauer and Kline [25], whose life cycle do not reflect a cyclic behaviour. After sifting through and analysing all these arguments, we cannot identify a clear motivation for a change time in a SLC. We systematically disproved all arguments or advantages of such a super-phase. The imperative conclusion is the absence of the need to define and integrate a change time into a SLC.

Concluding, more than 60% of the approaches do not define “change time”. In fact, they consider software change procedures as part of runtime and design time. Another 60%, however, do not require a SLC to have cyclic behaviour. Overall, the investigation proves our perspective that SLC are a very specific characteristic of a system. Further, life cycles for the management of services, as for the management of software in general, are not in need of an explicit “change time”. A generic definition that is omni-applicable is hardly definable. A well-founded, consolidated approach, as proposed in this section, is the closest to a recommendation we can get.

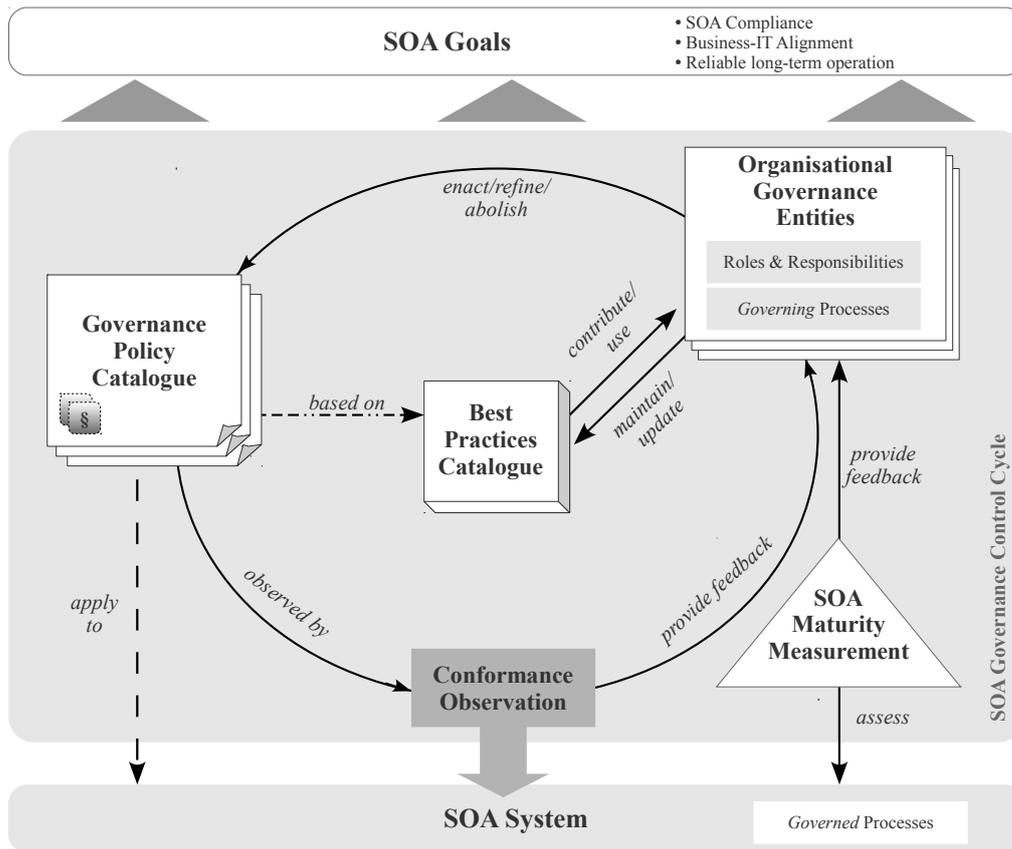


Figure 4.3: Operational Model for SOA Governance (cf. Niemann et al. [167])

4.2 An Operational Model for SOA Governance

Despite the fact that SOA Governance is considered a crucial element of SOA projects, so far there is no consensus about its actual structure, definitions, or comprised elements. In Section 3.1, we addressed these challenges and identified several commonly used concepts and gave a definition. Considering and integrating the major essential elements of SOA Governance, in this section, we propose an operational model for SOA Governance that provides means to deal with the growing complexity of SOA systems. We consolidated the common components (or building blocks) according to their actual tasks and composed them to best fulfil their primary purposes. The operational model is outlined in Figure 4.3. It was first published in [161], enhanced in [164] and [167], and refined for the version outlined here.

As our model integrates a control cycle, it can be regarded as refinement of procedure models (cf. Sect. 3.1.6). *Artefact management* (cf. Sect. 3.1.3) is considered an important aspect throughout the model (catalogue systems for policies and best practices). As major building blocks, we integrate *SOA goals* (cf. Sect. 3.1.7) and *metrics* (cf. Sect. 3.1.9), *policies and best practices* (cf. Sect. 3.1.1), *organisational entities and responsibilities* (cf. Sect. 3.1.2 and 3.1.4), *governance processes* (cf. Sect. 3.1.8), and *maturity measurement* (cf. Sect. 3.1.10). In the following, we outline each component in detail (cf. Fig. 4.3).

The *SOA system* represents the IT system to be controlled. All its technical and non-technical elements are subject to the governance control. In governance terms, this comprises the *governed* processes (cf. Sect. 3.1.8).

The purpose of the approach is to assure the achievement of goals for the SOA system. In general, these goals are derived from the overall IT goals (in IT Governance), which are specialised business goals (in Corporate Governance). Overall goals are SOA Compliance, Business-IT Alignment and reliable long-term operation. They determine the necessary actions of the underlying control structures. SOA Compliance refers to the adherence of the system to legal, normative (technical) and internal regulations. Compliance with legal specifications is mandatory (e.g., Sarbanes Oxley Act), while compliance with ISO norms or standard frameworks is often striven for by companies for competitiveness reasons. Internal regulations, e.g., the enforcement of company security directives are also tasks for governance. A further goal, Business-IT Alignment, refers to the achievement of the best possible integration and adoption of IT processes into the business environment and is crucial to the success of an SOA system [3, 215]. Reliable long term operation is a goal that results from due diligence management of an SOA. Summarising, the overall governance goal is to provide the achievements of these SOA goals in the long term.

The creation of a *metrics system* is a central issue. It is common to align metrics with specific goals and to assess the achievement of these goals. Goals are usually arranged in several nesting levels. Low level goals are defined by governance policies, such as the implementation of interfaces or the adherence to a standard. These goals are part of higher level goals like “conformity of service design” and “general standards conformance”, respectively. Metrics refer to the activities or processes that are regulated in order to contribute to the achievement of goals. They are dynamic, i.e., subject to change, in case goals or policies change. The measured results provide feedback regarding the degree of adherence to a given policy, or achievement of a goal. There is a number of metric types: business, process, performance, SLA, and SOA conformance metrics (cf. [86, 138]). Each of these corresponds to a specific type of policies.

This goal-metric constellation is a typical asset of maturity measurement (cf. Section 3.1.10). Thus, one of the six components of our model is *SOA maturity measurement*. Its main task is to provide feedback to the responsible council, which it derives from the overall assessment of the SOA system. It targets in particular the assessment of adaptation and the operation of an SOA in a company (cf. also [1, 97]).

4.2.1 *Best Practices and Categorised Policy Catalogue*

All policies defined for SOA Governance are based on *best practices*, i.e., prior experience with SOA systems or recommendations provided by experts. The best practices catalogue is a fundamental component of the governance model and contributes massively to the ability of SOA Governance to constantly improve an SOA system. It stores external and internal experiences in form of policies and ratings, and records their success as well as failure in the SOA system. In the case of policy change or abolishment, the respective recordings are added as negative experiences. It is always consulted when new policies are created or changed. (cf. Fig. 4.3)

This collection of best practices represents the foundation for a further core element of the governance model, the structured *catalogue of governance policies* (cf. Fig. 4.3). It

comprises the major policies currently valid for an SOA system. It is the task of the corresponding organisational entity to define, adjust, and abolish policies.

The catalogue is divided into several governance areas. We identify nine major governance areas to be regulated by SOA policies. The six primary aspects are *Architecture*, *Technology Infrastructure*, *Organisation*, *Service Asset Management*, *Information and Data* and *Project Execution* (cf. [1, 138]). We distinguish three cross-sectional areas: *Security*, *Service Operation*, and *Finance* (cf. [167]). As a third perspective, each of these domains has two aspects. The *company-internal* aspects comprise all the general policies concerning a company's SOA system. The *standards conformance* covers all aspects of adherence to accepted standards. This targets the design and operation of services, but also project execution (e.g., ISO/DIN norms) or SOA organisation.

The policy category *Architecture* includes reference architectures, architectural assessment mechanisms, application guidelines as well as architectural blueprints. The area *Technology Infrastructure* comprises aspects concerning the strategic SOA platform and governance platform, the migration of legacy systems as well as design and implementation of infrastructure services. *Organisation* covers all aspects dealing with human resources and organisational structures. Incentives for employees, the definition of roles and responsibilities, and the installing of SOA trainings are also part of these considerations, in addition to the definition of service and process owners (cf. Sect. 3.1.2). The governance area *Service Asset Management* manages SOA project portfolios, service portfolios and legacy portfolios for the strategic planning of an SOA and support for project management. *Information and Data* determines the rules for data ownership, data service architecture, data formats and standards, formalisation of the description of data requirements in SLAs as well as data quality. Project-related issues are covered in the area *Project Execution*: project selection, competence alignment, as well as the formalisation of the life cycle control of business processes and policies.

All these areas specify *Security*-related policies. They comprise data and communication security, systems security, as well as secure authentication and authorisation mechanisms (cf. [146]). The category *Service Operation* comprises all operational aspects, such as capacity planning, operational models for cross-department deployment and enforcement of SLAs, for example. It comprises cross-company cooperation, which covers legal, technical, and organisational issues for SOA-based cooperation between companies, e.g., operating applications that consist of services from several enterprises. In particular, it considers service ownership issues. *Finance* covers all financial issues, such as the funding of business and technical services, hardware and software infrastructure, as well as accounting models, such as funding models, usage feeds, or end-to-end funding.

Summarising, this governance policy category system organises the policies required in a governance approach for SOA systems by type. All proposed policy types can be classified into these categories.

4.2.2 *Organisational Aspects and Governance Processes*

As one central component of SOA Governance, specific organisational entities bear responsibility for the reliable operation, regulation, and control of the SOA system (councils like the *SOA Board*, cf. Sect. 3.1.2). Depending on the organisational composition of a company, the single organisational entities can be structured in a hierarchical or in a coordinating manner, e.g., in the case of territorially structured company branches

(cf. [57, 109]). They consist of representatives from every organisational layer. Ideally, they are manned with members from every section and department in the company, i.e., representatives of the upper management, lines of business as well as from the IT department, e.g., software developers (also cf. [103, 109]). A crucial requirement for effective work is the full support of the company's upper management (cf. Sect. 3.1.7). The council is in most cases a new organisational institution.

The main task of these institutions is the definition, adjustment, and abolishment of governance policies, as well as their enforcement. For the definition and design of new policies, a best practices catalogue is consulted in order to benefit from previous experience. Policies are created at all stages of the control cycle and all SOA maturity levels. Successfully deployed and applied policies are stored in the best practices catalogue (cf. Fig. 4.3). The concrete definition of competencies varies with companies and purposes of the SOA system (cf. also Sect. 3.1.2). Along with the introduction of new organisational governance entities come new roles and accountabilities that are implemented or realised by those entities. More than 70% of the examined approaches provide comprehensive lists of new SOA-related roles, introducing new accountabilities and decision rights [17, 27, 210, 248] (cf. also survey results in Tab. 3.1).

Concerning processes, generally, two types of governance processes can be distinguished – *governing* processes and *governed* processes (cf. Sect. 3.1.8). When building or introducing a governance approach, as a first step, governance processes and structures are to be established [1, 3, 138, 261]. The new SOA Boards are the actors in the overall governance processes. The control cycle in Fig. 4.3 is one of them. Governance processes constitute the foundation for control structures and provide techniques, methods, and decision rights in order to design, form, monitor, and control operational process structures. The continuous consistent regulation of the SOA system is guaranteed by well-formed and reliable control structures. As initial task of governance, processes on the operational level are set up, changed, and adjusted to fit the control structures.

The main part of the operational model constitutes the control cycle (cf. Fig. 4.3). The councils define and abolish governance policies. In this respect, they act like a republic parliament. During this procedure, the best practices catalogue is continuously maintained, i.e., enhanced and adjusted, respectively. The enacted policies apply to the SOA, and the components *conformance observation* and *maturity measurement* provide feedback to the SOA Boards. While the latter component assesses the system from a general and strategic point of view, the first component performs detailed compliance checks, e.g., concerning security issues or process compliance. Based on their feedback, the councils decide on further steps. Thus, the feedback cycle that emerges between the council and the policy catalogue, the SOA Maturity Measurement component and the conformance observation component assures the adherence to policies and enables the monitoring of policy effectiveness (cf. Fig. 4.3).

4.2.3 Conformance Observation

According to a survey concerning policy enforcement, 84% of interviewed companies perform manual design, code, and artefact reviews. 55% rely on manual pre-registration checks, and only 13% perform automated pre-registration checks (cf. App. B.2). Concerning the techniques, automated business activity monitoring and data consolidation techniques, or check lists are recommended (e.g., [233, 261]).

Regulating a system without effective enforcement and control mechanisms is hardly possible. As one central element of governance, we define the *conformance observation* component for the monitoring and enforcement of system conformance. It consists of various perspectives of conformance concerning, for example, processes, behaviour, services, projects. In all these fields, it is important for the successful operation of the SOA system that regulations (i.e., policies) are adhered to. Automated or manual mechanisms check adherence to the given regulations and enable the continuous conformance observation of the SOA system. The goal is, however, to increase the degree of automation.

A general aspect of conformance observation is the time component. Depending on the system and its peculiarities, compliance checks can be scheduled in different ways.

EX-POST VS EX-ANTE ANALYSIS. As part of the governance approach, the time of checks is to be decided upon. Checks can be performed after the occurrence of an issue (ex-post). Alternatively, in order to identify potential future violations, the analysis can be performed beforehand, i.e. proactively (ex-ante).

FREQUENCY. Conformance checks can be scheduled individually, or performed upon request. Alternatively, strict time intervals can be defined for the checks. This decision depends upon the management, as well as available resources and deployed techniques for conformance checks.

Automation of *conformance observation* is covered by few SOA Governance approaches. Approaches often provide mere proposals for basic support. Precisely, as conformance observation techniques, *automated check list processing* [233, 261], combined with checks at proxies or intermediaries (realised by, e.g., an ESB) [138, 261] have been proposed on the one hand. On the other, *business activity monitoring* (BAM) [261] linked with automated data consolidation in order to compute warning levels has been suggested. Most of the approaches, however, neglect automated conformance checks at all.

Beyond check list processing and BAM, we identify the following two major techniques that can strongly support the operation of SOA Governance concerning policy enforcement.

INFERRING ON MODELLED STRUCTURED DATA. Governance models often show more or less clear structures (cf. the frameworks by The Open Group [248] and COBIT [86]). These structures can be formalised using appropriate data modelling techniques. As additional application to exploiting the advantages of a meta model, implicit knowledge can be derived by inferencing on this data. This can be used for automating the answering to expert questions, as well as for decision support based on expert knowledge (cf. [165]).

For example, the modelling of governance frameworks as meta models or as ontologies specified in the Web Ontology Language (OWL) is a currently increasingly recognised area of research. We provide a proof-of-concept implementation for meta models in Janiesch et al. [96], and develop an Ontology-based approach in Niemann [160], Niemann et al. [165, 168]. A few further approaches exist in this direction [9, 64, 90, 94]. In particular for SOA systems, there are few such approaches yet.

COMPARISON AND RETRIEVAL OF BUSINESS REFERENCE PROCESSES. Managing internal processes and their adherence to reference processes is a central duty of confor-

mance observation. Reference processes are organised as best practices and policies in corresponding catalogues or repositories. The duty of governance councils is the design, as well as the enforcement of these modelled reference processes.

In BPM, a common research challenge is the effective retrieval of process models that are stored in central process model repositories. As enforcement technique in the context of SOA Governance, we suggest the enhancement of these techniques in order to (i) identify relevant reference processes for a given flow of activities and (ii) perform automated comparison of two process models to identify their difference. These techniques have the potential to automate many aspects of the enforcement of process-related governance policies. In Chapter 5, we present a novel technique for retrieval and comparison of business reference process models.

4.2.4 Conclusion

The presented operational model has been developed based on the results of a structural analysis of existing approaches. As central part, it defines a control cycle, covering the basic process of policy design, enactment, enforcement, and abolishment, supported by best practices, enforcement methods, and maturity assessment. Hence, it integrates the main aspects of SOA Governance and includes their perspectives and capabilities.

It is one of the first models to explicitly address and discuss the issue of conformance observation. By software vendors, this issue is addressed only in case their products support a type of enforcement (e.g. checks by the service registry). In literature as well as in industry whitepapers these issues are often omitted, supporting the “narrow view” on SOA Governance attested by Allen [3]. As part of our model, we make two specific suggestions for techniques to automate conformance observation. We address one of them, *comparison and retrieval of business reference processes*, in detail in Chapter 5.

Overall, the outlined generic operational model represents a fundamental approach to structure and define the operation of SOA Governance. The model allows for perspectives of existing approaches, integrates them, and additionally considers the concept of conformance observation. This general perspective aims at supporting future considerations, as well as the design of similar approaches.

4.3 Conclusion and Outlook

In this chapter, we defined a consolidated SLC based on survey results, as well as a generic operational model for SOA Governance. Both of them are based on the insights of the structural analysis performed in Chapter 3.

SLCs are a central aspect of SOA Governance and their management is crucial for the success of SOA Governance approaches (cf. Sect. 3.1.5). They are adjusted to the specific needs, perspectives, and notions of a company’s IT. However, there are as many different life cycles as proposals – there is obviously no standard or generic life cycle that fits all perspectives. We introduced a consolidated SLC that allows for all particularities of the approaches investigated in the survey (cf. Sect. 4.1) as well as of those identified by the structural analysis (cf. Chap. 3). The inclusion of a “change time” in a SLC is dispensable. Although often specified as an extra “change time” phase, service change can be safely performed within the operation phase and the reiteration of the cycle.

Based on our analysis in Chapter 3, identifying the common elements of SOA Governance, we developed an operational model for SOA Governance that comprises all main concepts and components of the examined approaches. The model consolidates perspectives and techniques of existing approaches, introduces a control cycle, and enhances it by the concept of conformance observation. Further, we defined categories for governance policy domains that structure common policy types in catalogues, structuring the major aspects of SOA Governance that are subject to regulation.

A clear deficit has been identified concerning policy enforcement or conformance observation. While policy enforcement mechanisms as extension to policy frameworks are integrated by a few approaches (e.g., WebMethods [261]), the automation of conformance checks in the context of SOA Governance has not been intensively addressed by the recent approaches that have been investigated. The least approaches make proposals concerning any technical support for governance activities. Among those approaches that propose corresponding techniques, almost none exceed the processing of a check list. As part of this model, we make two specific suggestions for techniques to automate conformance observation: *inferencing on modelled structured data*, and *comparison and retrieval of business reference processes*.

Overall, the proposed operational model achieves a generic perspective on SOA Governance, supporting future considerations and design of similar approaches. Backed by the analysis results, all concepts of the compared approaches map to our operational model. A consistent definition, consistent structuring by components, and a consistent role and accountabilities model all integrated to form a generic operational model, as we proposed it, have, to the best of our knowledge, not yet been proposed.

Outlook

Analyses of SOA Governance approaches show that a governance approach has a major impact on the success of SOA projects (cf. [141]). In this respect, the missing consensus on a generic approach for SOA Governance is not a relieving factor. It remains to be seen whether the increasing number of propositions of standardised frameworks for SOA Governance leads to a general basic agreement.

Many projects introducing a company-wide SOA system have failed (cf. [73]). The vice president of the technology consultant Burton Group stating that “SOA is dead”¹, indicates that SOA initiatives do not play the same role as in the past. According to her, the future development is towards *services*. This shows that SOA Governance keeps being subject to change – in particular against the background of a “larger service world”, e.g., service-based company cooperation and marketplaces in the Internet of Services (IoS). SOA Governance apparently evolves towards the discipline “Service Governance”. This development will also cause frameworks to change or continuously adapt.

Service Governance is the foundation for governance for the IoS, i.e., service marketplace governance [92, 95]. Compared to an SOA system, a marketplace bears much more complexity and the same time has the potential to yield more benefits. While, basically, the stakeholder roles in an SOA sum up to three, a variety of additional roles are needed in the IoS (e.g., service innovator, service producer, service aggregator, marketplace host, cf. Sect. 2.1.2). As soon as service marketplaces are shared across country borders, new

¹ cf. <http://apsblog.burtongroup.com/2009/01/soa-is-dead-long-live-services.html> (last access: July 4, 2011)

challenges emerge. When *ad hoc* service usage is to be realised, the investigation of legal consequences in case of SLA or contract breaches, strongly depending on local laws and legislation, is a still enduring challenge and research problem. Further, the efficient management of semantic service description (marketplace-wide unification), service composition, access management, delegation of service monitoring (centralised vs. decentralised approach), service pricing, marketplace-wide service development guidelines, and many more aspects have to be considered, as the recent German government-funded research project *Theseus/TEXO* revealed and addressed [19, 91, 92, 95, 96]. Most important, additional legal regulations apply in this scenario and different challenges arise, for example, automated service provision across borders of legislative authorities. All these factors have impact on the design and implementation of future service governance approaches.

A further field of research in terms of governance is *Cloud Computing* [5, 31, 72]. Most recently, it gained much momentum in both research and the software industry as a paradigm that utilises virtualisation technologies in order to maximise infrastructure flexibility in and between companies. The heterogeneity and complexity of cloud-based systems make high demands on efficient steering and management. Additionally, cloud computing is based on a heterogeneous, often not consistently organised environment. Many challenges of cloud governance can be coped with using means from IT Governance, such as compliance, legal issues and standardisation. For cloud characteristics such as the *ad hoc* provision of complex services, for example, infrastructure as a service (IaaS), the cloud-wide management and unification of service monitoring delegation, access management, development standards, and new legal regulations are new challenges that need experienced governance structures that can be adopted.

The largest area of potential scientific achievements is located in the field of conformance observation. As discussed above, there is a multitude of potential mechanisms and techniques to address the problem of automating the check of a system's adherence to regulations or policies. Though a small number of SOA Governance models, especially among those authored by software vendors, propose ideas or techniques for this problem, the author is not aware of specific automated techniques in this context that exceed check list processing.

Future developments might include a formalisation of approaches to SOA Governance, for example, using structured data approaches such as meta models or ontologies. The modelling of governance structures as meta models or ontologies bears potential concerning the automated support of the operational aspect of governance. Our work concerning the design of governance meta models (cf. [96]), and OWL ontologies for standard IT Governance frameworks such as COBIT 4.1 (Niemann et al. [168]) addresses an currently increasingly recognised area of research. Further work in this area shows that the modelling of structured governance data and its exploitation by various knowledge-inferencing techniques is currently considered a promising area of research [64, 90, 94].

A further important part of conformance observation is the management of internal processes and their adherence to reference processes. The duty of corresponding organisational entities (in the operational model) is the design of conform processes, as well as the assessment of the conformance of existing processes. In BPM, a common research challenge is the effective retrieval of process models, stored in a central process model repository. As enforcement technique in the context of SOA Governance, we suggest the enhancement of these techniques in order to (i) identify relevant reference processes for

a given flow of activities and (ii) compare two process models and identify their difference. Both techniques support the operational application of SOA Governance, as well as the enforcement of process-related governance policies. In Chapter 5, we present a novel approach (for retrieval and comparison of business reference process models) that tackles these challenges.

COMPARISON AND RETRIEVAL OF BUSINESS PROCESSES BASED ON RELATED CLUSTER PAIRS

AS REVEALED by the results of the analysis outlined above, the specification of guidelines and rules – governance policies – can be considered a central aspect of SOA Governance (cf. Sect. 3.3). However, governance approaches can only unveil their full impact if adequate enforcement mechanisms for policies are specified and established, the so-called conformance observation (cf. Sect. 4.2.3). One important aspect of conformance observation are *governed* processes (cf. Sect. 3.1.8). A central challenge is the facilitation of conformance assessment for these modelled processes, which is especially important, when handling a huge number of large processes.

Conformance assessment has two timing *foci*. Once a new process is to be set up at design time, efficient process model retrieval techniques have to assure that the right reference processes are considered within the design of the new process and therefore, provide authoring support (*ex-ante* perspective). Alternatively, processes can be checked for conformance frequently after their putting into service (*ex-post* perspective, cf. Sect. 4.2.3). In this scenario, efficient automated comparison techniques can decrease manual work effort.

In this chapter, we develop a decision support approach for process conformance assessment. As a basis, we propose the concept of *related cluster pairs* for the two scenarios process retrieval (*ex-ante*) and efficient process comparison (*ex-post*). Generally, our technique disaggregates process models into model fragments that are related by similarity. Further, we compute node assignments to identify the differences. We use related cluster pairs to calculate similarity of two process models for usage in the retrieval scenario. We compare our technique to related work and, as a baseline, to text search engines. For the evaluation of both scenarios, we use the so-called “SAP process reference model” (SPRM, [107]), a collection of ca. 600 process models that is frequently used throughout the research community.

The remainder of this chapter is structured as follows. After the introduction in Section 5.1, Section 5.2 discusses related work. Basic concepts, i.e., definitions for the graph-based representation of process models and definitions for utilised process model processing techniques, are introduced in Section 5.3. In Section 5.4, we outline the used similarity measures and their scope. We present the concept of related cluster pairs in Section 5.5, define a novel notion of process model similarity, and explain details of its computation. The evaluation results are presented and discussed in Section 5.6. For our process comparison approach, we perform cross-validation experiments using an annotated test case (cf. Sect. 5.6.1). The performance of the process model similarity metric of our process retrieval approach is evaluated using a second test case that has been annotated by process model experts (cf. Sect. 5.6.2). Section 5.7 concludes the chapter.

5.1 Introduction

Today, especially large companies increasingly organise and specify their procedural knowledge as process models. The challenges that arise for the management of large process model repositories are manifold. Today, immense amounts of process models and lacks in specification conformity cause isolated storage of different versions or variants of process models, models with overlapping application scopes, and with different granularity levels [50, 197]. Efficient management of reference process repositories that can cope with these increasing flexibility challenges requires extended and improved functionality of process model repositories. Common challenges in BPM research cover, e.g., the realisation of process reuse, efficient storage of large process model numbers, successful process comparison and retrieval for various purposes, operation of effective variant and version management, and the implementation of reference processes and conformance management [66, 135, 197, 211]. However, the efficiency concerning the exploitation of procedural knowledge contained in a company's process models still bears improvement potential. According to a survey on available process model repository implementations [272], efficient, i.e., computationally cheap approaches, have been neglected. Further, generally, tool support addressing these challenges is yet scarce [199]. In the last years, improvements for process model comparison and retrieval have been introduced. Contributions cover, for example, detailed investigations of various similarity measures [48, 254], variant identification based on structural decomposition [120], measures based on graph matching algorithms (e.g., [46, 142]), semantic approaches (e.g., [54, 118]), and retrieval approaches based on case-based reasoning [134, 150] (cf. Sect. 5.2).

In both scenarios, process model comparison and retrieval, governed processes as well as the company's reference processes are represented as process models. The latter are usually contained in a process repository.

The scenario *process model comparison* targets the comparison of a governed process model and a reference process model. Concerning the company's guidelines, the governed process must adhere to regulations (e.g., the inclusion of mandatory activities, or ordering of activities). Proceeding from two process models, an employee role's (e.g., process engineer) task is to check conformance with an corresponding reference process. Further, once introduced, the adherence to the reference process may be diminished (e.g., by undocumented changes, or merging with new processes or process fragments). In these cases, differences must be identified in retrospect. Especially when dealing with large models, this can be a costly and time consuming procedure.

The second scenario, *process model retrieval*, supports various aspects during process design. Once a new process is designed in a department, it is important that relevant reference processes are identified, and are integrated into the process at design time. Further, efficient process retrieval is a common challenge for the handling of large process repositories, the realisation of avoiding process variants, and, finally, for increasing process reuse.

Based on the concept of *related cluster pairs*, our approach performs the comparison of two process models, i.e., of the realised process and the reference process, by their structural decomposition based on similar process regions. As a first step, similar nodes are assigned to each other, using a novel hybrid similarity measure (i.e., a combination of string-based and semantic measures). Using a control flow analysis technique, the pro-

cess models are hierarchically decomposed into area pairs containing similar elements. By recursively decomposing the process models into corresponding elements, we can identify related as well as isolated process model fragments as support for manual adjustment of the given process model, i.e., the improvement of conformance with the reference model. By identifying all largest-possible corresponding process regions and those without counterpart, the difference between two process models can be specified. Hence, our approach represents a divide-and-conquer approach that reduces the problem space when searching differences. In particular, more complex techniques in terms of computational complexity can be applied to the resulting process parts.

Overall, the approach computes node assignments using various similarity measures for the identification of differences. We provide a respective implementation that provides an interactive user interface and visualises the results (cf. Sect. 5.5.6).

5.2 Related Work

When comparing process models (for search or mutual comparison), a multitude of properties can be tackled with a multitude of similarity notions. This section provides a discussion of existing similarity notions, as well as an outline of scientific works that are related to our approach.

5.2.1 Similarity Notions

For process models, different similarity notions, such as the notions of node label similarity, structural similarity, and behavioural similarity can be distinguished. A well structured overview is provided in [47] and [48]. The approach at hand focuses on node label similarity, using both string-based and semantic measures. Semantic measures that can be employed for label comparison are among others introduced in [54, 58, 128]. Cohen et al. [38] provide an overview of string-based label measures.

Generally, two types of process model similarity can be distinguished: the absolute and the relative notion. The latter, *relative* notion, is based on similarity values indicating relative matchings. It is employed in the further course of this contribution. *Absolute* notions of equivalence determine exact behavioural matches and are computed using, for example, bisimulation [192, 255], trace equivalence [76], or workflow inheritance [251]. The notion of bisimulation, e.g., assesses the ability of one process model to simulate the other. The result of these techniques, however, is mostly a boolean answer, which is not useful for process comparison in all scenarios.

All approaches proceed according to the same schema:

- (1) Computation of node assignments (or identification of similar process model elements): an assignment matrix for each considered node and edge type (e.g., activities, gateways) of the process model is computed.
- (2) Investigation of process relations: this is performed by considering, for example, the *structure*, or the *behaviour* of the process models.

So, when comparing process models, a *general* prerequisite for the computation of process model similarity is a node assignment matrix.

Generally, different types of similarity are distinguished: string-based, semantic, structural, and behavioural similarity. While string-based and semantic similarity measures

are used for the determination of node assignments, approaches to process model similarity can be divided into those that use *structural*, and those using *behavioural* similarity notions. The first ones are outlined in Section 5.4, and the latter are detailed in the following.

Structural process similarity measures the way an entity or component is composed, e.g., the sequence or hierarchy of its elements. Concerning business processes, structure refers to the arrangement of process elements. Existing approaches are based on the amount of structural components occurring in both models [117, 252].

A further approach is to use graph-edit distance ([61] e.g., as done by [46, 120, 126, 150]). Basically, the graph-edit distance approach compares two graphs by determining the cheapest sequence of basic graph-edit operations, i.e., deletion, insertion, or substitution of nodes and edges, to transform one graph into the other. Each operation is assigned costs that can be varied (mostly 1). The result is a minimum cost-edit path, as well as the respective transformation costs. Normalised, i.e., in relation to the overall node and edge count, the latter serve as an indicator for the similarity of two given graphs, and can be used as similarity function (cf., e.g., [48]). However, the graph-edit distance problem is NP-complete. A computation in reasonable execution time concerning complexity is only feasible for small graphs [29, 157]. According to [47], a further drawback is that, although they are relevant, indirect relations of activities via inserted or deleted process gateways are not considered, and that, in fact, structurally similar process models may be different in behaviour [42].

Process models use a description language to describe in a semi-formalised way, how a process, given a number of activities and gateways, can behave. I.e., for each process model, there is a variety of possible process execution traces, called the *process behaviour*. The notion of behavioural process similarity compares process behaviours concerning causality, exclusion, and concurrence of corresponding elements and measures the degree of similar behaviour. Usually, for behaviour, some process elements are considered more relevant than others [144, 252]. As examples, de Medeiros et al. [42] compare given behaviours to a typical behaviour contained in event logs, computing a similarity value for process models, which is relative to the event log. Dijkman et al. [48] compute behavioural similarity for event-driven process chains (EPC) using a distance vector space based on so-called causal footprints. Causal footprints are approximations of a process' behaviour and have, amongst others, the advantage of a small computational effort.

As governance targets management processes rather than operative processes, its reference processes mostly reside at a high level of abstraction. As a consequence, comparison focuses on correspondences of activities (node similarity), and their ordering (structure), rather than on execution traces or behaviour. The latter are rather important for more fine grained processes (cf. Fig. 2.4).

5.2.2 Related Approaches

We identified several criteria to compare existing related approaches. An overview is provided in Table 5.1. In the following, we explain the criteria, before outlining the relevant related approaches in detail.

The computation of *node assignments* is a principal first step when comparing process models that represents the basis for all similarity considerations (cf. Sect. 5.2.1). It implies a $n : m$ comparison procedure, and the actual correct assignment of activity labels (nodes)

that have been described by different human experts is not a trivial problem. However, some approaches (marked by ‘–’ in Tab. 5.1) still rely on the external provision of node assignments.

Five criteria indicate the utilisation of different similarity notions. While structural process similarity has almost been deployed by all approaches, only one approach except ours, for example, explicitly uses a hybrid similarity measure, i.e., the combination of several similarity notions.

Computation of *process model differences* indicates whether the approach compares processes in terms of identifying their differences. Some quantifications of similarity do not specify process differences. Further criteria are the computation of *change suggestions*, *(semi-) automated merging*, and *result visualisation*. In addition to our approach, only one of the presented approaches has performed an *evaluation*, making it comparable to other works.

Andrews et al. [4] present an approach for visual graph matching. The prototype analyses similarities of graphs, suggests a merged graph, and visualises the results. It provides means to the user for final visual assessment and manual node assignment in order to adjust the results. The user can manually edit the resulting graph by, e.g., replacing labels or changing a node’s position. The approach assumes the manual input of node similarities, i.e., does not automatically determine node assignments.

Dijkman [45] presents an approach to determine process model differences provided as EPC. A detailed computation of differences is conducted, whereas the type of a difference using the difference topology introduced in [44] (e.g., different roles, skipped activity, refined activity) as well as the exact position of the differences is determined. For the computation, formal semantics are utilised. The approach has exponential computational complexity and thus requires repeated scoping of the process models. Manual node assignments are required as a first step.

Dijkman et al. [48] propose and compare three different similarity metrics for the problem of retrieving process models in a given repository. In detail, they utilise label matching similarity to determine the similarity of node labels, structural similarity, and behavioural similarity to consider the node labels and causal relationships of activities (nodes) in the model, respectively. The label similarity covers string-based and semantic node similarity, neglecting their combination. For the semantic node similarity, they consider word synonyms [47, 254]. The approaches determine a final similarity score, but do not indicate where the similarities and differences are located within the process models.

Ehrig et al. [54] propose a (semi-) automatic approach to detect similar elements in process models. The authors compute similarities based on semantic information. More precisely, they make use of the so-called Pr/T net ontology, which represents a description of Petri net elements based on OWL-DL introduced in [118]. The comparison considers semantic and string-based node similarity measures, which are combined to a similarity measure for concept instances. The concept instances are compared to each other and the similarity values are aggregated to an overall similarity of the two process models. However, node similarities and process differences are not explicitly stated.

Küster et al. [120] introduce an approach for process model comparison that compares different versions of a process model in the absence of a change log. The authors use single-entry-single-exit (SESE) fragments and assume externally provided node correspondences. After the computation of differences, change suggestions are derived. The

Table 5.1: Overview of related work in BPM

Approaches	Objects	Node assignments	String-based node similarity	Semantic node similarity	Hybrid node similarity	Behavioural process similarity	Structural process similarity	Process model differences	Change suggestions	(Semi-) automated merging	Result visualisation	Evaluation
Andrews et al. [4]	process models	-	-	-	-	-	-	•	-	•	•	-
Dijkman [44, 45]	EPCs	-	-	-	-	-	•	•	-	-	-	-
Dijkman et al. [47, 48, 254]	process models	•	•	•	-	•	•	-	-	-	-	•
Ehrig et al. [54, 118]	process models	•	•	•	•	-	-	-	-	-	-	-
Küster et al. [120]	process models	-	•	-	-	-	•	•	•	•	•	-
Li et al. [126, 127]	process models	-	-	-	-	-	•	•	•	-	•	-
Lu and Sadiq [130]	process models	•	-	-	-	-	•	-	-	-	-	-
Madhusudan et al. [134]	process models	•	•	-	-	-	•	•	-	-	-	-
Melnik et al. [142]	data struct./ graphs	•	•	-	-	-	-	-	-	-	-	-
Minor et al. [150]	process models	•	-	-	-	-	•	•	-	-	-	-
Our approach	process models	•	•	•	•	-	•	•	-	-	•	•

differences and change operations are then grouped and associated to the affected SESE fragments. Based on the gathered information, an hierarchical change log is composed that can be applied to resolve all or parts of the differences in order to obtain a consolidated model. This approach primarily considers different versions of the same process. It has not been analytically evaluated.

Li et al. [126] introduce an approach for a Process-Aware Information System that identifies a generic process reference model for a given set of variants. Based on a so-called aggregated order matrix, they determine activities (nodes of the process graph) to be clustered as blocks. Referring to the blocks, they investigate the behavioural similarity (ordering) of activities. The algorithm has been assessed using simulation on more than 7000 process models [127]. The activity assignment matrix is an external requirement.

Lu and Sadiq [130] propose a search approach for retrieving process variants, based on the identification of process model fragments. Given a process query, they use three basic structural similarity measures (equality, subsumption, and implication). They consider a number of features per process model and define a similarity functions for each of them. The authors do not evaluate their approach.

Madhusudan et al. [134] propose a case-based reasoning approach for workflow modelling and design support. Their approach, called “similarity flooding for workflow”, is based on the similarity flooding method by Melnik et al. [142]. It consists of a structural process similarity for retrieval, and is based on initial similarity values. The authors state that these values are based on “Natural Language Processing and string-matching techniques”. They do not provide an evaluation of their approach.

A further graph matching algorithm is presented by Melnik et al. [142], which determines a mapping between the corresponding nodes of two given graphs. Their approach is applicable in different scenarios with diverse data structures, such as matching of two data schemas in data warehousing applications or comparison of process model graphs.

Before the actual comparison of the graphs, the two data structures are converted into directed labeled graphs. In the next step, the so-called *similarity flooding* is performed. This step represents an iterative fixpoint computation to determine the set of similar nodes. It is based on the assumption that two nodes are more likely to be similar, if their adjacent nodes are similar. For the determination of node similarities a simple string-based node similarity is used. Hence, neither semantic node similarity nor structural or behavioural similarity of the graphs is considered in their approach. The computation results in a mapping between corresponding nodes, whereas differences are not determined.

Minor et al. [150] propose an index-based retrieval approach for workflows. The approach targets the search among past or changed workflows to assist authoring of recent workflow instances. They apply structural similarity using graph-edit distance. Unfortunately, the approach has not been evaluated in terms of information retrieval measures (precision or recall).

Further approaches concerning process model retrieval (e.g., [112, 152]) focus on the design and application of specific process query languages and the exploitation of process metamodel data specifying basic process entities and their relationships. They are not further regarded here, as the approaches' core functionality are different from the one at hand.

Approximately 40% of the outlined approaches require node assignments as input [4, 45, 120, 126], although their determination is not a trivial problem. Semantic node label similarity is only considered by two further approaches ([48, 54]). However, none uses advanced word preprocessing exceeding stemming (e.g., lemmatising as used by our approach). Most of the authors consider structural process similarity, for example, graph-edit-distance, which has a high computational complexity. Few ones apply hybrid node similarity measures, or perform an evaluation that enables the comparison with other approaches.

While Küster et al. [120] use a similar approach for structural investigation to the one presented here, they assume the *ex-ante* provision of node assignments. In most cases, this does not represent a realistic scenario. Dijkman et al. [48] use similar basic notions of node similarity. However, they do not use hybrid similarity notions and apply different structural investigations.

Generally, for process models, the notions of node label similarity, structural similarity, and behavioural similarity are distinguished. Van der Aalst et al. [252] provide an approach for comparing a process model with a set of event logs based on a behavioural similarity measure. Further approaches consider behavioural similarity for process models [48] and state charts or finite state machines [156, 270]. The problem of correct node assignments, however, represents a fundamental requirement for fully automated process comparison and retrieval approaches, which in turn consider, for example, structural or behavioural process characteristics. Further, it can directly affect the results of these relying approaches, if improved. For this purpose, amongst others, we investigate the integration of semantic node label similarity measures.

Mendling et al. [145] provide interesting related fundamental research in this respect, although their intention is slightly different from the one at hand. They classify verbs occurring in node labels of the process models in the SAP reference model using two established verb classification schemes. Based on the most frequently occurring verb classes, they propose a set of icons to support process modelling practice. Performing a general classification of node labels using the two schemes, their approach achieves a coverage

of 0.68 and 0.44, respectively. As a general problem, they identify missing specificity of node labels (e.g., by frequent usage of the verb “to process” in labels). The authors do not aim at investigating the quality of the classification nor do they provide details of the approach. Differing from their approach, we use (amongst others) a technique from natural language processing (NLP), *lemmatizing*, and test it on annotated test sets aiming at improving current classification result quality.

The approach at hand combines a variety of label-based similarity approaches to tackle this problem. We apply string-based and semantic similarity measures, and by their combination create hybrid similarity measures. It is the general focus to improve the exploitation of label meanings using considerate word preprocessing and reduction techniques in order to improve assignment quality. Additionally, we simultaneously consider node label similarity and structural characteristics of process models. In contrast to Ehrig et al. [54], we do not demand explicit semantic description (e.g., using ontologies), but rely on and exploit the meaning of words in the context of their label(s). As a core part of our approach, we compute *word* and *node label similarities* for process model comparison and retrieval, where a *node label* is a sequence of *words*. Based on these, we determine node assignments, region differences, similarities of process model fragments, and the similarity of process models.

Generally, our concept of related cluster pairs targets exploiting label meanings using considerate word preprocessing techniques from natural language processing. Additionally, we introduce the simultaneous consideration of node label similarity and structural aspects of the process model. These aspects have not been investigated in related work so far.

5.3 Basic Concepts and Definitions

In this section, we introduce fundamental definitions and terms that are used by our approach. In particular, we introduce the utilised notion of process model graphs and the so-called SESE regions (cf. Sect. 5.3.1), outline the assignment problem (cf. Sect. 5.3.2), and define a general similarity function (cf. Sect. 5.3.3).

5.3.1 Process Model Graphs and SESE Regions

Throughout our approach, we consider process models as graphs we call *process model graphs* (PMG, cf. Fig. 5.1).

Definition 2 (Process Model Graph). *Let G be a graph $G = (V, E)$, Λ be a set of labels and Θ be a set of types. A process model graph P is a directed, weakly connected graph defined as tuple $P = (V, E, \lambda, \tau, \alpha)$, where:*

- V is a finite set of nodes,
- $E \subseteq (V \times V)$ is a finite set of edges,
- λ is a labelling function: $\lambda : (V \cup E) \rightarrow \Lambda$ that assigns labels to nodes and edges,
- $\tau : (V \cup E) \rightarrow \Theta$ assigns types to nodes and edges, and
- $\alpha : (V \cup E) \rightarrow (A \rightarrow \Lambda)$ assigns key value pairs to nodes and edges, where A is a set of names (keys) that are assigned labels (values).

In particular, the sets A, Θ and Λ all include ϵ (the NULL element).

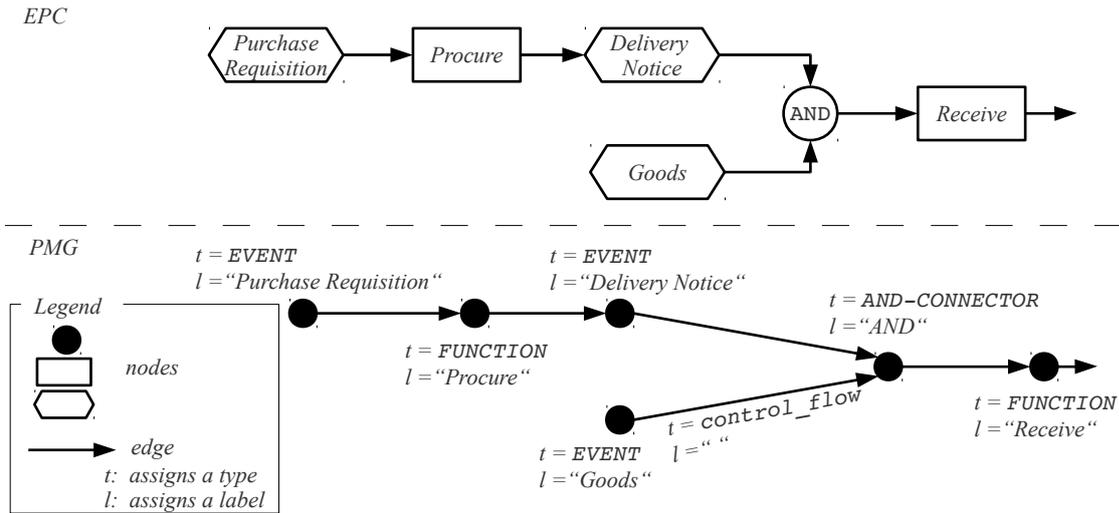


Figure 5.1: A process model graph (PMG) representing an EPC

This definition (adopted from [48]) for process model graphs is applicable to describe a variety of graph-based process description languages, including EPC.

Let e be the number of edges $e = |E|$ and $v = |V|$ be the number of nodes. As many nodes in a process model have but one incoming and one outgoing edge, and process models are normally far from being fully connected, i.e., we can assume $e \ll v^2 - v$. (A fully connected directed graph with n nodes has $n(n - 1)$ edges.) This is important for complexity estimations (cf. Sect. 5.5.5).

This definition for process model graphs can be used to describe a variety of graph-based process description languages, for example, Event-driven Process Chains (EPC) [108, 111].

Definition 3 (Event-driven Process Chains). Let $P_{EPC} = (V, E, \lambda, \tau, \alpha)$ be a PMG. We define

- $\Theta = \{XOR, OR, AND, function, event\}$
- $\lambda : V \rightarrow \Lambda$ assigns labels to nodes,
- $\tau : V \rightarrow \Theta$ assigns the types to nodes,
- $\alpha : \emptyset \rightarrow \emptyset$ is not used.

According to [111], we add the following constraints that further specialise this PMG:

- Each function and each event have at most one incoming and one outgoing edge.
- Each connector (XOR, OR, AND) has at most one outgoing (join connector) or one incoming edge (split connector).
- Functions and events alternate along the control flow.
- The OR-split and the XOR-split connectors must be preceded by a function.
- There should be no cycle of control flow that consist of connectors only.

An example is shown in Figure 5.1. Further details on EPC are outlined in Section A.2. For a mapping to the Business Process Modelling Notation (BPMN), for example, the definition of α is made use of to express lanes, specialisation and generalisation of process steps (subprocesses and superprocesses). A detailed mapping is provided by Dijkman et al. [48].

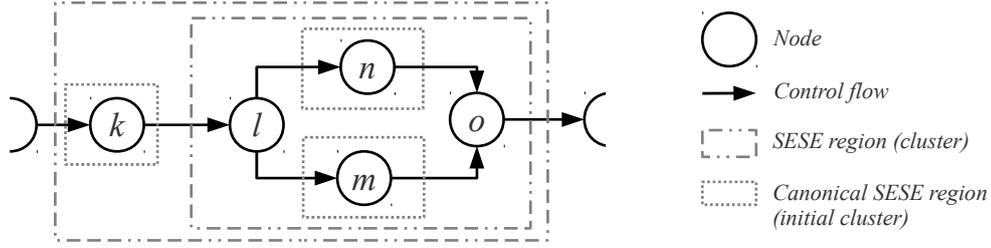


Figure 5.2: SESE regions

For the structural investigation of process model graphs, we use the concept of *Single-Entry Single-Exit (SESE) regions* from control flow analysis. Control flow analysis plays a major role in, e.g., compiler optimisation for testing the legality of transformed loops and detecting run-time errors [98]. They were introduced for the construction of program trees in compiler analysis [98, 99, 100], and have also been applied in an approach that analyses variants of process models [120]. Originally, the authors proceeded from a graph with exactly one start and one end node, respectively. The non-mathematical definition the authors provide is based on this fact. We transfer this definition to business process models that can have more than one start and end node.

Definition 4 (SESE Region). Let $G = (V, E)$ be a directed graph, defined by a finite set of nodes V and a finite set of edges E . A SESE region is defined as a connected, directed subgraph $R = (V', E')$ of G with $V' \subset V$ and $E' \subset E$. The following conditions hold:

- $e^{in}, e^{out} \in E'$ with $e^{in} \neq e^{out}$ (single incoming and outgoing edges)
- $\forall v \in V' : \nexists e = (v, w) \in E'$ with $e \neq e^{out}$ and $w \in V$ and
- $\forall s \in V' : \nexists e = (r, s) \in E'$ with $e \neq e^{in}$ and $r \in V$.
- $|\{(r, s) | (r, s) \in E' \wedge s \notin V'\}| = 1 \quad \wedge \quad (r, s) = e^{out}$
- $|\{(t, u) | (t, u) \in E' \wedge t \notin V'\}| = 1 \quad \wedge \quad (t, u) = e^{in}$

A SESE region can be denoted in graph notation $R' = (V', E')$ or as edge pair (e^{in}, e^{out}) . Generally, SESE regions are either node disjoint or nested. (Definition adopted from [100])

Intuitively, a SESE region represents an area within a graph that has a distinct entry and a distinct exit edge (cf. Fig. 5.2). Nodes inside can only be reached from outside by passing the entry edge and nodes outside can only be reached from inside by passing the exit edge. Besides the distinct entry and exit edges, the nodes inside the area possess no other connections to the rest of the graph. It follows that a single node with a single entry and exit edge represents a trivial SESE region itself. Trivial SESE regions are called *canonical SESE regions*. Generally, canonical SESE regions represent a unique and node disjunctive decomposition of a process model graph.

SESE regions meet the condition of transitivity [100]. Given two SESE regions S_1, S_2 and a PMG $(V, E, \lambda, \tau, \alpha)$ with $S_1 = (a, b)$, $S_2 = (b, c)$, and $a, b, c \in E$, their union represents a new SESE region S_3 :

$$(a, b) \cup (b, c) = (a, c) = S_3 \quad (\text{transitivity}) \quad (5.1)$$

In our context, canonical SESE regions, i.e., those containing a single node, are referred to as *initial clusters* (cf. Sect. 5.5.2).

5.3.2 Assignment Problem

A major class of problems in graph theory and combinatorial optimisation are matching problems. In these problem scenarios, an optimal pairwise assignment of the elements of two sets has to be found, while satisfying further conditions.

The *linear sum assignment problem* [2, 30] can be formulated as follows. The assignment is represented by an assignment matrix X with matrix elements x_{ij} and $i, j \in \{1, \dots, n\}$. C is a cost matrix with elements $c_{ij} \geq 0$ and $c_{ij} \in \mathbb{R}$ that defines the cost for all potential assignments. Element j has assignment cost c_{ij} for assignment to element i (and *vice versa*). Consider two element sets U and V with $|U| = |V| = n$. The variable x_{ij} equals to 1, if element $i \in U$ is assigned to element $j \in V$, and to 0 otherwise. Given n elements i and n elements j , their best possible assignment with respect to cost is wanted. Additionally, the following constraints must hold [30, 49]:

- (1) Each element i is assigned to exactly one element j and each element j is assigned to exactly one element i .
- (2) The resulting assignment x_{ij} is the one with the minimum cost of all possible assignments meeting the first condition.

As result, the overall assignment matrix X represents the optimal assignments in terms of cost. Formally, this corresponds to the objective function

$$\text{Minimise } F(x) = \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \quad (5.2)$$

with the constraints

$$\sum_{j=1}^n x_{ij} = 1 \quad \forall i \in \{1, \dots, n\} \quad \text{and} \quad \sum_{i=1}^n x_{ij} = 1 \quad \forall j \in \{1, \dots, n\} \quad (5.3)$$

as well as

$$c_{ij} \geq 0, x_{ij} \in \{0, 1\} \quad \forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, n\} \quad (5.4)$$

Note that the assignment and cost matrices X and C are quadratic.

A number of algorithms to solve the assignment problem have been proposed. The so-called Hungarian algorithm solves it in $O(n^3)$ time [2, 30, 149].

5.3.3 Similarity

In the course of this work, we determine the similarity of node label words, node labels, and process models, respectively. We make use of several similarity functions.

Definition 5 (Similarity Function). *A similarity function sim measures the degree of similarity between two objects a and b [207]. The function is defined as*

$$\text{sim}(a, b) : U^2 \rightarrow [0, 1]$$

where U^2 represents the universe of the object descriptions. We consider 1 to indicate high similarity, 0 dissimilarity, and all other values partial similarity. Further, the reflexivity axiom [207, 273] holds: $\forall a \in U : \text{sim}(a, a) = 1$ (Reflexivity).

If the following axiom holds, a similarity function is called symmetric.

$$\forall a, b \in U : \text{sim}(a, b) = \text{sim}(b, a) \text{ (Symmetry)} \quad (5.5)$$

Note that not all similarity functions are symmetric.

In order to determine significant similarity, we use similarity functions sim_i in combination with thresholds $t_i \in [0, 1] \forall i \in \mathbb{N}$. For (a, b) to be significantly similar, we demand for the measure sim_i : $\text{sim}_i(a, b) \geq t_i$.

Further, we need to flexibly combine similarity metrics. We define the similarity function combination \oplus_{w_1, w_2} :

$$\text{sim}_1(a, b) \oplus_{w_1, w_2} \text{sim}_2(a, b) := w_1 \text{sim}_1(a, b) + w_2 \text{sim}_2(a, b), \quad (5.6)$$

where $w_1, w_2 \in \mathbb{R}$, $w_1, w_2 \in [0, 1]$ and $w_1 + w_2 = 1$.

5.4 Node Label Similarity Measures

Large numbers of metrics for string-based (e.g., [38]) as well as semantic similarity considerations (e.g. [28]) exist that are used in a number of different applications areas (e.g., semantic service discovery [225, 226], process similarity [48, 169], and E-learning [32, 220]).

Generally, several objects of similarity considerations (words, node labels, and process models), as well as general types of similarity (string-based, semantic, structural, and behavioural similarity), are distinguished. While structural and behavioural similarity notions are applied to compare process models (cf. Sect. 5.2.1), word and node labels are compared using string-based and semantic similarity measures, respectively. Node assignments, determined based on node label similarity, are always a requirement for performing investigation of process models concerning structural or behavioural characteristics. In case of semantic similarity, for most measures, node labels must be decomposed into single words before semantic similarity can be applied, i.e., an aggregation of similarity values is necessary. String-based similarity measures can be applied to all objects, as they do not distinguish between single words and labels (word sequences). For this, process models are reduced to a list of node labels (which is also useful for the usage of a text search engine, cf. Sect. 5.6.2).

As a core part of our approach, we compute *word* and *node label similarities* for process model comparison and retrieval, where a node label is considered to be a sequence of words. Based on these, we determine node assignments, region differences, and similarities of process models.

In this section, we introduce general word preprocessing techniques (cf. Sect. 5.4.1) and all string-based (cf. Sect. 5.4.2) and semantic (cf. Sect. 5.4.3) similarity measures we use throughout our approach.

5.4.1 Word Preprocessing

In all cases, we perform preprocessing of words. Generally, we transform labels to lower case, remove punctuation (., - ; /) as well as single-character-words, and perform stop word removal. Stop words are frequent function words (such as “the”, “as”, “in”, “over”, “by”, ...). We decided, however, not to remove words like “with”, “without”, “needs”, “not” or similar ones that are part of common stop word lists, as these can influence and

essentially coin the meaning of the activity label they are part of. The intention is to avoid turning node labels with conflictive meaning using similar words into sentences with similar meaning, e.g., “Capacity is available” and “Capacity is not available”¹. This is a common problem in NLP that must be coped with.

When applying string-based measures, we optionally use Porter’s stemming algorithm [189]. Stemming is a rule-based method to reduce similar words to a simple stem. For the words “orders”, “warehousing”, “adjusting”, and “receive”, the results of Porter’s algorithm are “order”, “wareh”, “adjust”, and “receiv”, respectively. Further, we consider the usage of a more advanced method than stemming for word reduction, *lemmatising*, useful. The lemma of a word is its actual base form. This reduction techniques preserves the word meaning. Lemmatising, however, requires and depends on the *part-of-speech* (POS) of a given word. For example, “purchasing” can be lemmatised to the verb “purchase” or the noun “purchasing”, depending on the context. If these information are given, words can be automatically reduced to their base form, such as “are” to “be”, “communities” to “community”, and “warehousing” to “warehouse”. We use the lemmatising approach introduced and realised by Schmid (1994) [219].

5.4.2 String-based Node Label Similarity

String-based metrics calculate the similarity of given words or sentences based on their syntactic elements (characters) without regarding their meaning. In the following, we introduce the measures we use. For each one, we provide an example using nodes from the examplary process model pair provided in Figure 5.4.

A common string-based similarity measure is *Levenshtein’s string-edit-distance*. For two strings, or node labels L_1 and L_2 , it computes insert and deletion operations required to transform L_1 into L_2 . Each operation is assigned a cost of 1. The sum is the result of the metric $lev(L_1, L_2)$ [123]. We apply the metric sim^{Lev} defined as

$$\text{sim}^{\text{Lev}}(L_1, L_2) = 1 - \frac{lev(L_1, L_2)}{\max(|L_1|, |L_2|)}, \quad (5.7)$$

where $|L_1|$ designates the length of string L_1 in terms of characters. The strength of this measure is the identification of the same characters for words occurring in the same sequence. Simple plurals, word variants, or verb conjugations are generally well recognised by this measure. Referring to the example in Figure 5.4 (cf. p. 92), the similarity value for the labels “transfer to warehouse” (node g) and “warehousing” (node y) is 0.33 (using stop word removal).

In order to address its deficiencies, e.g., in case one of the two labels to be compared contains long additional words, or when processing alternating words (e.g., “transferring to warehouse” and “warehouse transfer”, where $lev = 0.11$), we use the Jaccard distance measure sim^{Jac} as *set* measure. A set is assumed to be a sequence of words, divided by a common separator. A set A of length $|A| = n$ is supposed to consist of n words referenced as A_i . Especially, the union $A \cup B$ does not contain identical words. sim^{Jac} divides the number of identical words by the number of all (differing) words [38, 137]:

$$\text{sim}^{\text{Jac}}(L_1, L_2) = \frac{|L_1 \cap L_2|}{|L_1 \cup L_2|}. \quad (5.8)$$

¹ This is the case in process model ID 1In_b7s7 in the SAP reference model.

For example, the labels “transferring to warehouse” and “warehouse transfer” are assigned a similarity value of 0.25, using stop word removal and lemmatising even a value of 1.0. Referring to Figure 5.4, the similarity $\text{sim}^{\text{Jac}}(f, x)$ equals to $\frac{2}{3}$.

As this measure proceeds from word level, simple word variants are not understood as the same word. The labels “transfer to warehouse” and “warehousing”, e.g., are not recognised as similar. Word preprocessing or the combination with other measures normally remedy this deficiency.

5.4.3 Semantic Node Label Similarity

String-based similarity measures for node comparison, however, can lead to significantly wrong similarity interpretations. For example, comparing “goods” to “good”, as well as “procurement” to “purchasing” leads to undesired results. Automatically revealing relationships in these cases requires the consultation of word corpora or lexicons. These are mostly represented as graphs based on word senses like, e.g., the WordNet corpus [58]², or Wikipedia³. Semantic measures that can be used for label comparison are amongst others introduced in [54, 58, 128]. As semantic measures generally compute the similarity for two words, word similarity values need to be aggregated on node label level. For this, we use additional aggregating functions.

As semantic similarity indicators, we use three measures: a word synonym and a word distance measure both based on WordNet, as well as the “distributional semantic first order” based on Wikipedia. Again, we provide examples referring to the process model pair provided in Figure 5.4.

WordNet contains the most frequent English words, organised in synsets. A synset is a collection of synonymous words. As a word can have more than one meaning, each word can be contained in several synsets. So, using WordNet, we can systematically derive a set of words that share any of the meanings of a given word. By $\text{sim}_a^{\text{ws}}(w_1, w_2)$, we calculate the similarity of two words w_1 and w_2 based on their WordNet synsets:

$$\text{sim}_a^{\text{ws}}(w_1, w_2) = \begin{cases} 1 & \text{if } w_1, w_2 \text{ are identical words} \\ a & \text{if } \exists \text{ synset } S \text{ with } w_1, w_2 \in S \\ 0 & \text{otherwise} \end{cases} \quad (5.9)$$

The measure assigns two identical words the value 1.0, while the similarity of synonymous words is indicated by the parameter a (with $a \in \mathbb{R}$ and $0 < a < 1$). However, this definition does not provide the aggregation of word similarities to node label similarities. In preliminary experiments, we identified two scenarios with acceptable results, depending on the aggregation function and the parameter a .

In the first one, we choose $a = 0.75$ and aggregate by taking the mean value of the word similarity values. This measure has also been specified in [254]:

$$\text{sim}^{\text{Smean}}(L_1, L_2) = \frac{|L_1 \cap L_2| + \sum_{(v,w) \in L_1 \setminus L_2 \times L_2 \setminus L_1} \text{sim}_{0.75}^{\text{ws}}(v, w)}{\max(|L_1|, |L_2|)} \quad (5.10)$$

² <http://wordnet.princeton.edu>

³ <http://wikipedia.org>

In the second scenario, we use the *Monge Elkan similarity metric* sim^{ME} as aggregating function. It is defined as

$$\text{sim}^{ME}(A, B, \text{sim}^M) = \frac{1}{|A|} \sum_{i=1}^{|A|} \max_{j=1}^{|B|} \text{sim}^M(A_i, B_j), \quad (5.11)$$

where sim^M can be any metric that operates on words [153]. This metric also uses the dissection of labels into words as outlined prior to Equation 5.8. As above, the metric distinguishes atomic strings and subfields. An atomic string is either a single word or an abbreviation, while a subfield is a part of a field (or word set) containing atomic strings, i.e., an atomic string represents the smallest, possible subfield. In order to determine the degree of similarity between two given fields A and B , a subfield w_i of the first field A is compared to every subfield of the second field B . The measure returns the mean of the *maxima* of all w_j . For computation of the similarity, the measure wraps the measure $\text{sim}^M(A_i, B_j)$.

For the second synonym measure, we choose the Monge Elkan measure as the aggregating function and, based on the results of preliminary experiments, define $a = 0.5$:

$$\text{sim}^{SME}(L_1, L_2) = \text{sim}^{ME}(L_1, L_2, \text{sim}_{0.5}^{WS}) \quad (5.12)$$

As an example, consider the two node labels “client order processing” and “handling of customer order”. A result of $\frac{1.0+0.75+0.0+0.0+0.0}{3} = 0.583$ is yielded using $\text{sim}^{S_{\text{mean}}}$, and using sim^{SME} it is $\frac{0.5+1.0+0.0}{3} = 0.5$ (in both cases using stop word removal). In the process model pairs given in Figure 5.4, for example, the nodes g and y are assigned a similarity value of 0.50 by both metrics (using stop word removal and lemmatising). Again, slight word variants such as plural forms, are not recognised as identical words unless word preprocessing is performed.

However, for example, the relationship between “purchasing” and “procure” is not identified by these measures. Terms that are not synonyms but still relate, such as hypernyms or hyponyms, are not considered. Therefore, we use a further measure, the word distance. Based on WordNet, it refers to the length of the shortest path between two words w_1 and w_2 in the WordNet taxonomy ($\delta^{\text{WordNet}}(w_1, w_2)$). In order to process label pairs, we define

$$\Delta^{\text{WN}}(L_1, L_2) = \frac{1}{|L_1 \setminus L_2|} \sum_{w_1 \in L_1 \setminus L_2} \min \left\{ \delta^{\text{WordNet}}(w_1, w_2) \mid w_2 \in L_2 \setminus L_1 \right\}. \quad (5.13)$$

The measure removes identical words from both labels. We consider all possible POS and word relations (e.g., hyponym, hypernym). The metric calculates the mean of the minimum values of the distances per word in L_1 and all words in L_2 . We define the inverse WordNet distance as

$$\text{sim}^{\text{WND}}(L_1, L_2) = \frac{1}{\Delta^{\text{WN}}(L_1, L_2)}. \quad (5.14)$$

A strength of this measure is the recognition of definite word relationships that are not identified as being synonyms. The shortest path between the words “procure” (node b in Figure 5.4a) and “purchasing” (node n in Figure 5.4b), e.g., is 4, resulting in a similarity value of 0.25.

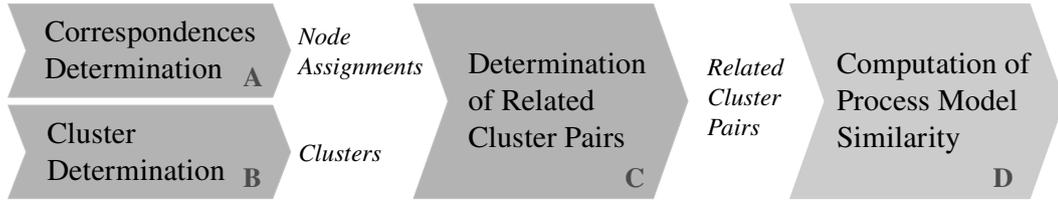


Figure 5.3: Computation of Related Cluster Pairs

WordNet is a collection of selected word meanings, whose relationships are professionally maintained. Thus, its coverage of word meanings is limited. As a valuable resource for word meanings, Wikipedia, in contrast, is constantly updated by thousands of volunteers. Word relationships are not explicitly defined in Wikipedia. Therefore, we compute the corpus-based similarity of words using the so-called “distributional semantic first order similarity” measure by Lin [128]. Two given words w_1 and w_2 are compared by their dependency triples using a context window size of ± 3 words. Moving the window over the corpus results in a set of dependency triples for a given word w_x . A dependency triple is of the form (w_x, r, w') , where w_x represents the word whose context is examined, w' is a word occurring in the context of w_x , and r refers to the relationship between w_x and w' (i.e., the relative position of w' with respect to w_x) [128].

$$\text{sim}^{\text{Lin}}(w_1, w_2) = \frac{\sum_{(r,w')} (w_1, *r, *w') + (w_2, *r, *w')}{\sum_{(r,w')} (w_1, *, *) + \sum_{(r,w')} (w_2, *, *)} \quad (5.15)$$

The measure is based on the assumption that the similarity between two words can be expressed as the amount of information contained within the dependency triples which are common to both words, divided by the amount of information contained in all the dependency triples of w_1 and w_2 that match the pattern $(w_1, *, *)$ and $(w_2, *, *)$, where $*$ is a wildcard for r and w' , respectively. For application on node labels we define sim^{SFO} :

$$\text{sim}^{\text{SFO}}(L_1, L_2) = \text{sim}^{\text{ME}}(L_1, L_2, \text{sim}^{\text{Lin}}) \quad (5.16)$$

The metric identifies word similarities exceeding synonym relationship, based on the linked context in the Wikipedia corpus.

Using this measure, for example, the node labels “determine level of significance” and “assess relevance” have been correctly matched in our experiments, although only two out of six words are synonyms. Further, the example provided in Figures 5.4 and 5.5 has been computed using this measure using a threshold of $t = 0.02$. The resulting similarity values are provided by Table 5.2.

5.5 Related Cluster Pairs – A Novel Concept for Process Model Retrieval and Comparison

Related cluster pairs represent mutually assigned (as well as unassigned) subgraphs of process models containing pairwise assigned nodes. Their computation consists of three major steps: (A) *Correspondences Determination*, (B) *Cluster Determination*, and (C) *Determination of Related Cluster Pairs*, as well as an additional fourth step (D) *Computation*

Algorithm 1: Computation of Related Cluster Pairs

Input: PMG $P_1 = (V_1, E_1, \lambda_1, \tau_1, \alpha_1), P_2 = (V_2, E_2, \lambda_2, \tau_2, \alpha_2)$, sim, threshold t
Data: cluster sets C_1, C_2, U ; set of related cluster pairs R ; similarity matrices M^i
Output: A set of maximal related cluster pairs R and unassigned clusters U (with similarity values)

```

1 COMPUTERELATEDCLUSTERPAIRS( $P_1, P_2$ ) begin
2   for all node types  $y \in \Theta$  do
3      $M^y \leftarrow$  COMPUTESIMILARITYMATRIX( $y, V_1, V_2, \lambda_1, \lambda_2, \text{sim}, t$ )
4     APPLYHUNGARIANALGORITHM( $M^y$ )
5   end
6    $C_1 \leftarrow$  COMPUTEINITIALCLUSTERS( $P_1$ ) /* gen. tree of initial clusters */
7    $C_2 \leftarrow$  COMPUTEINITIALCLUSTERS( $P_2$ )
8    $R \leftarrow$  MERGEHIERARCHICALRELATEDCLUSTERS( $\{C_1, C_2\}$ )
9    $R \leftarrow$  MERGENEIGHBOURINGRELATEDCLUSTERS( $R$ )
10  DETERMINECLUSTERSIMILARITYVALUES( $R$ )
11   $U \leftarrow$  MERGEHIERARCHICALUNASSIGNEDCLUSTERS( $\{C_1, C_2\}$ )
12   $U \leftarrow$  MERGENEIGHBOURINGUNASSIGNEDCLUSTERS( $U$ )
13  DETERMINECLUSTERSIMILARITYVALUES( $U$ )
14 end

```

of *Process Model Similarity* that covers the computation of the process model similarity (cf. Sect. 5.5.1, 5.5.2, 5.5.3, and 5.5.4). Figure 5.3 provides an overview. The process comparison and retrieval approaches outlined in this thesis are based on the concept of *related cluster pairs* as introduced in Niemann et al. [169, 170] and [230]. In the further course, we outline these steps along an exemplary process model pair provided in the Figures 5.4 (setup) and 5.5 (result). In Section 5.5.5, we discuss the computational complexity and limitations, while details of the implementation are provided in Section 5.5.6. The general data flow and procedure of the realisation of the steps *A* through *C* is shown in Algorithm 1.

5.5.1 Correspondences Determination (Step A)

In step *A*, node correspondences are determined. We apply a number of measures covering word similarity and node label similarity, as well as word preprocessing techniques (cf. Section 5.4). In order to achieve optimal unique node assignments (in terms of calculated similarity), the resulting matrix is optimised by solving the assignment problem. Hence, this step consists of two inner steps: *A1) determine potential match candidates per node*, (cf. Alg. 1, Line 3) and *A2) solve assignment problem* (cf. Alg. 1, Line 3).

As part of *A1*, per node type, all nodes of one graph are compared to all nodes of the other graph. We compute similarity values for node pairs using the introduced word preprocessing techniques (cf. Section 5.4.1), the measures outlined in Sections 5.4.2 and 5.4.3, as well as combinations of them (hybrid measures). The measures calculate a value for a given label pair $(\lambda(n_1), \lambda(n_2))$ for nodes n_1 and n_2 . Generally, we consider two nodes similar concerning a metric sim_i , if the similarity value of their label pair exceeds a threshold t_i : $\text{sim}_i(\lambda(n_1), \lambda(n_2)) \geq t_i$. In this case, we call node n_1 correspondent of node n_2 . For the hybrid measures, we use the weighting defined in Equation 5.6.

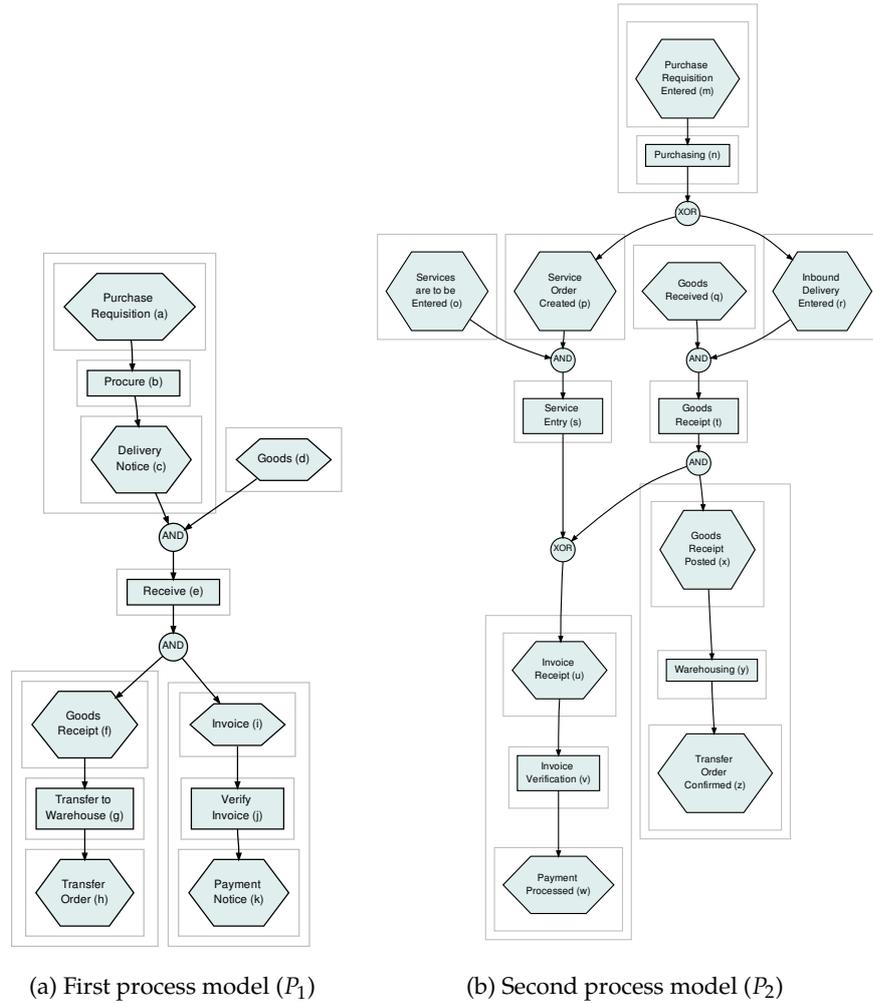


Figure 5.4: Example pair of process models: visualisation of all determined clusters (as grey rectangles)

Node pairs with a similarity value above a threshold t are added to a match list which is specified for each single node. We learn the thresholds for the measures by cross-validation (cf. Section 5.6.1.1). All other nodes are added to a list of unassigned nodes. After computation, every node's match list contains all potentially corresponding nodes of the other process model graph in terms of similarity. Combining these match lists results in an $m \times n$ -assignment matrix (with m and n indicating the respective node counts).

Optionally, in this step, the neighbourhood of nodes is considered: if a node pair shares similar preceding and successive nodes, their similarity score can be increased by a constant factor, e.g., 1.3. For more details refer to Appendix D.

As an example, Table 5.2 provides the similarity values and node assignments determined for the process model pair shown in Figure 5.4. For this example, we used the measure sim^{SFO} (cf. Equation 5.16). Nodes from P_2 that do not appear in the lists have been assigned a similarity value below the threshold (here $t = 0.02$) for the respective combination. Underlined node IDs indicate that the corresponding node from P_2 has been assigned to the respective node from P_1 .

In step (A2) *solve assignment problem*, the node assignment matrix is optimised by solving the assignment problem. We use it for maximising the overall sum of similarity values of selected node assignments, i.e., the sum of the similarity values s_{ij} needs to be *maximised*. Hence, cost can be redefined as $1 - s_{ij}$, or the target function can be maximised [30]. In our adaptation of the assignment problem (cf. Eq. 5.2), we chose the latter variant (cf. Model 2). X is the matrix of binary decision variables x_{ij} that represent the node assignments. We further assume, without loss of generality, that the larger process model contains n nodes, i.e., $n \geq m$, and that s_{ij} do not exceed the range of similarity values (cf. Eq. 5.19). The sum of the similarity values s_{ij} of the chosen assignments is maximised (cf. Eq. 5.17), satisfying the condition that all nodes of the smaller process model are assigned to one node of the other model (cf. Eq. 5.18). Further, when comparing two process models based on nodes, node counts most probably differ. As the original assignment problem deals with square assignment matrices, we modified the problem model to fit rectangular matrices according to the proposal by Bourgeois and Lassalle [22, 23]. This way, the result of the algorithm – additionally to the optimal assignments concerning similarity values – indicates all nodes that are not assigned a correspondent in the other process model. To allow unassigned nodes in the smaller model, we ignore all assignments between nodes i and j that have been made based on $s_{ij} = 0$, a similarity value of zero. According to Bourgeois and Lassalle [22], the mean execution time of this algorithm is $O(cn^2m)$, where c is a constant (and $n \geq m$).

5.5.2 Cluster Determination (Step B)

In parallel to step A, in step B, we consider structural process fragments, i.e., SESE regions and clusters. SESE regions are subgraphs of a PMG having but one incoming and one outgoing edge. They were originally introduced for the construction of program trees in compiler analysis [99, 100]. The basis for related cluster pairs are the so-called *clusters* that are, intuitively, typed SESE regions. We define a *cluster* as follows.

Table 5.2: Similarity values and node assignments for process models graphs P_1 and P_2 using the measure sim^{SFO} (cf. Figure 5.4)

$n \in V_1$	List of match candidates from V_2						
a	<u>m</u> : 0.6806	w: 0.0288	r: 0.0268	q: 0.0242	o: 0.0232	z: 0.0211	
b	<u>n</u> : 0.0332						
c	<u>r</u> : 0.3402	w: 0.0337	z: 0.0276	o: 0.0215			
d	<u>q</u> : 0.5040	x: 0.3352					
e	<u>t</u> : 0.0208						
f	<u>x</u> : 0.6685	u: 0.5155	q: 0.5168	w: 0.0246			
g	<u>y</u> : 0.5065						
h	<u>z</u> : 0.6701	p: 0.3422	w: 0.0254	r: 0.0226	m: 0.0221		
i	<u>u</u> : 0.5155						
j	<u>y</u> : 0.5121	t: 0.0216					
k	<u>w</u> : 0.5030	u: 0.0360	q: 0.0258	r: 0.0257	z: 0.0238	m: 0.0231	x: 0.0208

Model 2: Adapted Linear Sum Assignment Problem

Objective Function

$$\text{Maximise } \sum_{i=1}^n \sum_{j=1}^m s_{ij} x_{ij} \quad (5.17)$$

Constraints

$$\sum_{i=1}^n x_{ij} = 1 \quad \forall j \in \{1, \dots, m\} \quad (5.18)$$

$$\begin{aligned} n &\geq m \\ 0 &\leq s_{ij} \leq 1 \quad \forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, m\} \\ x_{ij} &\in \{0, 1\} \quad \forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, m\} \end{aligned} \quad (5.19)$$

Definition 6 (Cluster). Let $P = (V_P, E_P, \lambda_P, \tau_P, \alpha_P)$ be a PMG and Θ a set of cluster types. A cluster L in P is a connected subgraph $(V, E, \lambda, \tau, \alpha, t)$ such that

- with nodes $V \subseteq V_P$ and edges $E \subseteq E_P$
- $S = (V, E)$ is a SESE region (cf. Def. 4):
 $|\{(u, v) \mid (u, v) \in E_P \wedge v \notin V\}| = 1 \quad \wedge \quad |\{(u, v) \mid (u, v) \in E_P \wedge u \notin V\}| = 1$
- $\lambda = \lambda_P, \tau = \tau_P$ and $\alpha = \alpha_P$ are functions as in Def. 2
- the function $t : \{L\} \rightarrow \Theta$ assigns a type to the cluster

The set of cluster types Θ covers clusters that contain *node or cluster sequences* (cluster 1 in Fig. 5.6), *split/join constructs of nodes or clusters* (cluster 2), *single nodes* (cluster 3), and *node or cluster loops*. Trivially, each node that is not a gateway represents a cluster. Each cluster can consist of further, nested, clusters (i.e, graph nodes or SESE regions), referred to as *cluster elements* in the following. For example, cluster 1 of PMG 1 in Figure 5.6 consists of cluster 2 and two more cluster elements.

In the algorithm, for each process model, the function `COMPUTEINITIALCLUSTERS` computes the initial cluster sets L_1 and L_2 , taking the PMG P_1 and P_2 as input, respectively (cf. Alg. 1, Lines 5 and 6).

Two clusters can form a cluster pair, if they are assigned a corresponding similarity value. Similarity values for cluster pairs are based on node similarity. We define the similarity of clusters L_1 and L_2 as follows:

$$\text{sim}^{\text{RCP}}(L_1, L_2) = \frac{1}{|V_1|} \sum_{i=1}^{|V_1|} \text{sim}^{\text{Node}}(v_i, \cdot), \quad (5.20)$$

where $|V_1|$ is the number of nodes contained in both clusters L_1 and L_2 and $\text{sim}^{\text{Node}}(v_i, \cdot)$ refers to the similarity value of node v_i and its counterpart.

Figure 5.4 shows all clusters that have been identified for an exemplary process model pair. Clusters are shown as grey rectangles and are either nested or disjoint. In step C, these structural information are combined with the node similarity values computed for each node pair.

5.5.3 Determination of Related Cluster Pairs (Step C)

Generally, a *related cluster pair* consists of two PMG regions (clusters) containing mutually assigned elements. Based on a combined structural, string-based, and semantic assessment of two different PMGs, our approach hierarchically decomposes the PMGs in a divide-and-conquer manner into nested regions. The computation of related cluster pairs identifies the largest-possible unassigned as well as assigned regions of process models. Information gathered by these *spots of similarity* prove valuable for the identification of process model changes and determination of process model similarity.

Based on the determined node assignments and identified clusters, in step C, the *related cluster pairs* are determined. For their computation, the information about node similarity and process regions are combined.

Definition 7 (Related Cluster Pair). Let L_1 and L_2 be clusters $L_1 = (V_1, E_1, \lambda_1, \tau_1, \alpha_1, t_1)$ and $L_2 = (V_2, E_2, \lambda_2, \tau_2, \alpha_2, t_2)$, and $V_Q \subseteq (V_1 \times V_2)$ a set of nodes. A related cluster pair Q is defined as $Q = (V_Q, \text{sim}^{\text{Node}}, t)$, where sim^{Node} is a node similarity function and t a similarity threshold $t \in \mathbb{R}$ with $0 \leq t \leq 1$.

For all $(x, y) \in V_Q$, the following conditions hold:

- $\text{sim}^{\text{Node}}(\lambda_1(x), \lambda_2(y)) \geq t$ (Similarity of nodes)
- $\nexists (v, w) \in V_Q : v = x \vee w = y$ (Unique node assignments)
- $\tau_1(x) = \tau_2(y)$ (Equality of node types)
- $t_{L_1}(L_1) \sim^{\text{ctSim}} t_{L_2}(L_2)$, (Similarity of cluster types)

where \sim^{ctSim} is a binary relation specifying the similarity of the cluster types defined in Θ (cf. Definition 6).

Initial related cluster pairs consist of two smallest possible subgraphs (initial clusters) in different models, each consisting of at least one node (e.g., “Cluster 12” in Figure 5.5). In order to form the largest possible related clusters pairs, clusters are merged in a hierarchical as well as sequential manner.

The aggregation of neighbouring clusters creates sets of largest possible related clusters. The result of the computation of related cluster pairs contained in the process model pair from Figure 5.4 are shown in Figure 5.5. Referring to P_1 in Figure 5.5a, cluster 11 has been constructed from three one-node-clusters. The size of three nodes could be realised, as three similar nodes were found in P_2 (Fig. 5.5b). For clusters 3 and 4 in P_1 (cf. Fig. 5.5b, cf. also Fig. 5.4a), for example, this is not the case.

Smallest possible related cluster pairs consist of two smallest possible subgraphs (SESE regions) in different models, each consisting of one node (cf. “A” in Fig. 5.6). The nodes in turn correspond in terms of node similarity. Nodes without correspondent reside in *unassigned* clusters, which do not have counterparts in the respective other process model (e.g., B and H’ in Fig. 5.6).

Assigned and unassigned nodes are distinguished by a threshold t of the similarity measure sim^{Node} . As sim^{Node} , any of the node label similarity measures outlined above or combinations of them can be used. Gateways are not explicitly regarded by similarity measures, while node types in fact have to be the same to be considered part of a related cluster pair (cf. Fig. 5.6 and Def. 7).

Clusters are either hierarchically arranged or sequentially arranged (cf. Sect. 5.5.3). The first comprise nested clusters, while the second cover neighbouring ones.

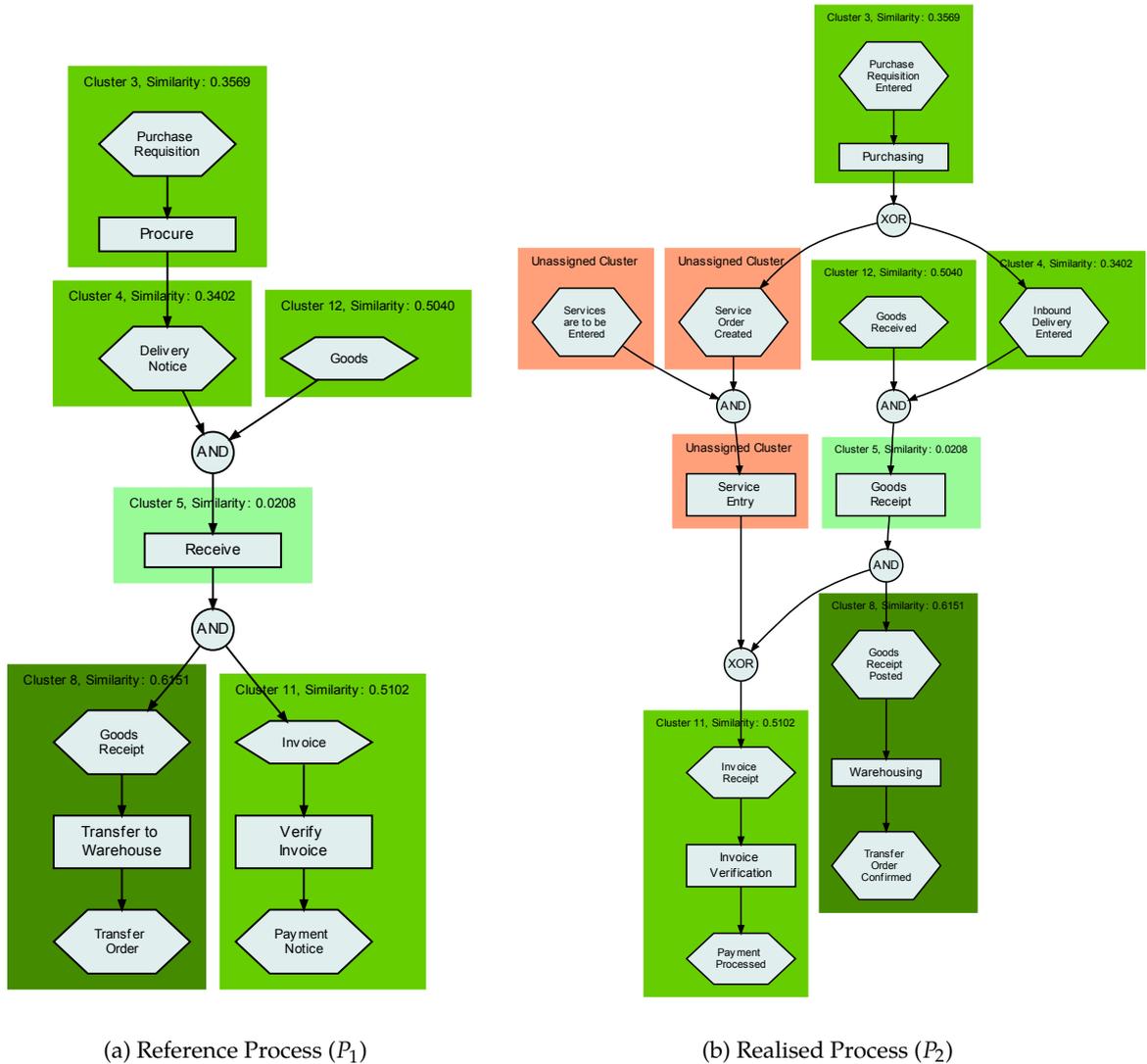


Figure 5.5: Identification of related cluster pairs – visualised computation results. This example has been computed using the metric sim^{SFO} (Eq. 5.16).

Cluster Merging

Each node represents an initial cluster. Thus, each corresponding pair of nodes represents a pair of corresponding initial (related) clusters and each unknown node constitutes an initial (unrelated) cluster.

As part of step C, simultaneously in both graphs, nested and adjacent clusters are merged. Subsequently, adjacent initial related clusters pairs and unrelated clusters are simultaneously enlarged to form largest possible clusters. While the latter are merged per model, the merging of the first ones is model spanning. The cluster merging step consists of 3 sub-steps: (CM1) merge similar hierarchical clusters, (CM2) merge similar neighbouring cluster sequences, and (CM3) calculate related cluster similarities. Aggregating neighbouring clusters creates sets of largest possible related clusters, each containing node sequences. Exemplary visualised computation results are provided in Figure 5.5.

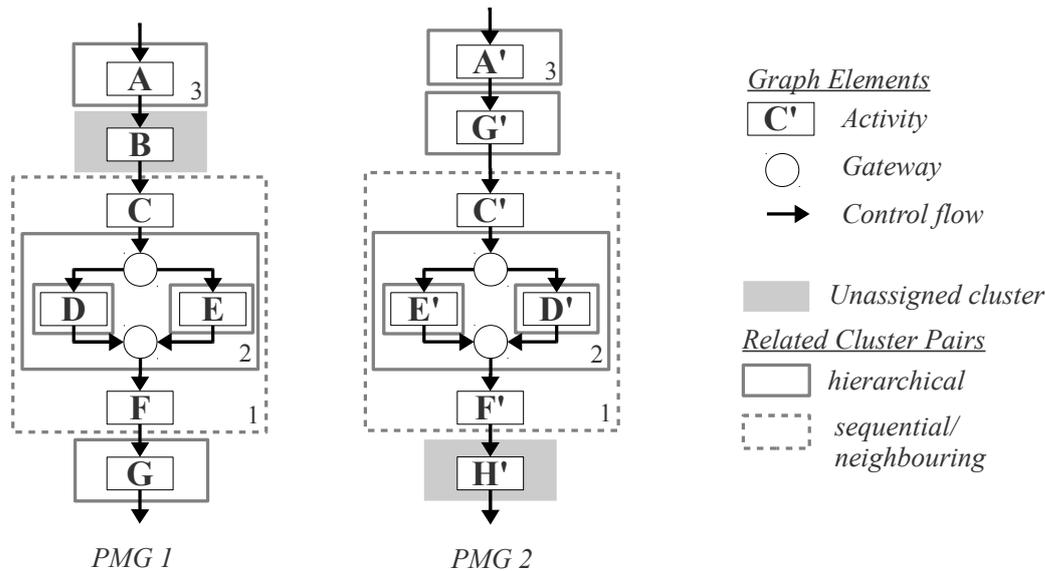


Figure 5.6: Related cluster pairs

Cluster merging generally adheres to two conditions. The first condition demands from an adjacent node B to node A in model P_1 that its corresponding node B' in model P_2 is adjacent (on the same side) to A' , which is the correspondent to A . The second condition is that the resulting node groups must in turn be (related) clusters.

Clusters must be similar in structure, i.e., contain activities that appear in the same order, to be related cluster pairs. Two clusters can only be merged if they cover smaller clusters with the same set of nodes, or initial clusters.

MERGE SIMILAR HIERARCHICAL CLUSTERS (CM1). In this step, the parent regions of similar initial clusters are analysed concerning similarity. Depending on activity order and similarity, it is checked whether larger related parent clusters can be created by merging all respective children (cf. Fig. 5.6). In Algorithm 1 (Line 7), the function `MERGEHIERARCHICALRELATEDCLUSTERS(C_1, C_2)` generates the related cluster pairs set R .

MERGE SIMILAR NEIGHBOURING CLUSTER SEQUENCES (CM2). In this step, sequences of related clusters are identified and summarised to form larger corresponding clusters, if possible. (cf. Fig. 5.6). This process stops if clusters cannot be enlarged any more, e.g., due to adjacent unassigned nodes or unrelated clusters. The function $R \leftarrow \text{MERGENEIGHBOURINGRELATEDCLUSTERS}(R)$ from Algorithm 1 (Line 8) is shown in detail in Algorithm 3. The procedure makes use of auxiliary functions:

- `MERGECLUSTERS(S)` merges two sets of similar cluster sequences S to form related cluster pair. (Line 8)
- `GETLARGESTCLUSTERSEQUENCE(p)` tests for two semantically similar cluster sequences ending with p . (Line 12)

The function `MERGENEIGHBOURINGRELATEDCLUSTERS` (shown by Alg. 3) receives the set of similar cluster pairs P determined so far as input parameter. As long as there are still sequences of clusters left that can be merged, the algorithm iterates over

Algorithm 3: MERGE_{NEIGHBOURINGRELATEDCLUSTERS}

Input: $P = \{(r_1, s_1), (r_2, s_2), \dots, (r_n, s_n)\}$, n pairs of clusters r_i and s_i
Output: $P = \{(r_1, s_1), (r_2, s_2), \dots, (r_m, s_m)\}$, m pairs of *merged* clusters r_i and s_i

```

1 MERGENEIGHBOURINGRELATEDCLUSTERS( $P$ ) begin
2    $c \leftarrow \text{TRUE}, s \leftarrow \text{FALSE}$ 
3   while  $c$  do
4     if  $s$  then
5       for all  $m \in mClust$  do
6          $P \leftarrow P \setminus \{m\}$ 
7       end
8        $P \leftarrow P \cup \text{MERGECLUSTERS}(mClust)$  /* merges two cluster sets */
9     end
10     $c \leftarrow \text{FALSE}, mClust \leftarrow \emptyset$ 
11    for all  $p \in P$  do
12       $mClust \leftarrow \text{GETLARGESTCLUSTERSEQUENCE}(p)$  /* ending with  $p$  */
13      if  $mClust \neq \emptyset$  then
14         $c \leftarrow \text{TRUE}, s \leftarrow \text{TRUE}$ 
15        break
16      end
17    end
18    return  $P$ 
19  end
20 end

```

all similar cluster pairs $p \in P$ (Line 11) and tests the process models for two corresponding sequences ending in the two clusters of p . If such a corresponding sequence $mClust$ is detected (Line 13), $s = \text{TRUE}$ and the iteration stops for updating the set of similar cluster pairs P . Then, the corresponding cluster pairs within each sequence are deleted from the set of similar cluster pairs P (Line 6). Subsequently, they are aggregated to a single cluster resulting in a cluster pair by the function `MERGECLUSTERS` (Line 8). The result is added to the set of similar cluster pairs P . In the case that no further cluster sequences are detected, the algorithm stops and returns P , which now contains m pairs of merged clusters (Line 18).

CALCULATE RELATED CLUSTER SIMILARITIES (CM3). After the determination of the largest corresponding related clusters possible, their degree of similarity is calculated, based on the arithmetic mean of the similarity values of all nodes within the related clusters (cf. Eq. 5.20 and Alg. 1, Line 9).

For the handling of unassigned clusters, the substeps *CM1-CM3* are performed in an analogous way (cf. Alg. 1, Lines 10-12). As a difference, the merging of clusters does not adhere to similarity restrictions and the computation of cluster similarity values is trivial.

Using this technique to decompose process models, we are able to identify differences between the models, to determine the similarity of cluster subgraphs, and to infer an

overall similarity notion for process models (cf. Section 5.5.4). The computation and results visualisation (cf. Fig. 5.5) has been realised as a plug-in for ProM⁴.

5.5.4 Computation of Process Model Similarity (Step D)

Based on the information gathered throughout the computation of related cluster pairs, we define a novel process model similarity measure \cdot . The related cluster pairs used for the similarity computation in the example (PMGs P_1 and P_2) are shown in Figure 5.5.

Let N be the set of all nodes of P_1 and P_2 , i.e., $N = V_{P_1} \cup V_{P_2}$. Based on the identified related cluster pairs, we classify all nodes in N into two disjoint node sets A and U , i.e., $A \cup U = N$ and $A \cap U = \emptyset$. The set of nodes with *assigned* node correspondents is denoted as

$$A = \{n \in N \mid \text{sim}^{\text{Node}}(n, \cdot) \geq t\} \quad (5.21)$$

For sim^{Node} and t we refer to Definition 7, while we define $\text{sim}^{\text{Node}}(n, \cdot)$ as the similarity value for node n , calculated by sim^{Node} for the assignment to its correspondent. Analogously, U is the set of all *unassigned* nodes from N , i.e., all nodes residing in unrelated clusters: $U = N \setminus A$. For a related cluster pair consisting of clusters L_1 (in P_1) and L_2 (in P_2), we refer to the number of nodes contained in cluster L_1 ($n \in V_{L_1}$) with $|V_{L_1}|$ (cf. Def. 7). Further,

$$A = \bigcup_{i=1}^{m_{\max}} A_i \quad \text{and} \quad A_k = \{n \mid n \in V_L \wedge |V_L| = k\} \quad (5.22)$$

where $i, k \in \mathbb{N}$ and m_{\max} indicates the size of the largest related cluster pairs of the current comparison, and V_L is the node set of a cluster L . Each node $n \in A_k$ is contained in a related cluster L with node size $|V_L| = k$. Accordingly, for example, A_3 refers to the node set that is part of related cluster pairs with size 3.

Based on these definitions, we define the *node-based similarity* of two process models P_1 and P_2 as follows:

$$\text{sim}_{\text{PM}}^{\text{Node}}(P_1, P_2, \text{sim}^{\text{Node}}) = \frac{\sum_{n_i \in A_1} \text{sim}^{\text{Node}}(n_i, \cdot) + \sum_{m=2}^{m_{\max}} (1 + q(m)) \sum_{n_j \in A_m} \text{sim}^{\text{Node}}(n_j, \cdot)}{|A| + |U| + \sum_{m=2}^{m_{\max}} q(m) \cdot |A_m|} \quad (5.23)$$

As weighting function $q : \mathbb{N} \rightarrow \mathbb{N}$, we use $q(m) = m + 1$, where m is the index of A , i.e., the number of nodes the current cluster contains. While smallest clusters (with size 1) are generally weighted with 1, all other clusters are weighted according to their node size. The larger a cluster is, the higher are the weights for the contained node similarity values; q specifies the intensity of the weighting. The overall similarity measure represents a dynamic weighted average based on q . Our measure hence emphasises large similar node sequences.

As an example, let P_1 be the model shown in Figure 5.5a and P_2 be the one outlined by Figure 5.5b. We use the similarity measure $\text{sim}^{\text{Node}} = \text{sim}^{\text{SFO}}$, i.e., the distributional semantic first order based on Wikipedia (cf. Eq. 5.16) and the threshold $t = 0.02$ (that has been learned using cross validation). The result is shown in Figure 5.5. Note that

⁴ cf. <http://www.promtools.org/prom5/>

the measure does not consider gateways of EPCs, the numbers of nodes hence refer to functions and events only. While their total number is $|N| = 25$, we see that $|A| = 22$ nodes have correspondents and $|U| = 3$ do not. With, obviously, $m_{\max} = 3$, we retrieve $|A_1| = 6$, $|A_2| = 4$, and $|A_3| = 12$ (cf. Eq. 5.22). With $q(m) = m + 1$, it yields an overall similarity value for the pair (P_1, P_2) of

$$\text{sim}_{\text{PM}}^{\text{Node}}(P_1, P_2, \text{sim}^{\text{SFO}}) = \frac{1.73 + (1 + 3) \cdot 1.4276 + (1 + 4) \cdot 6.7518}{22 + 3 + 3 \cdot 4 + 4 \cdot 12} = \frac{41.1994}{85} \approx 0.4847$$

Additionally to this qualitative variant, we define a quantitative variant that is based on the determined clusters and assumes $\text{sim}^{\text{Node}}(n_i, \cdot) = 1$ for all node pairs. We call it *cluster-based similarity*:

$$\text{sim}_{\text{PM}}^{\text{Cluster}}(P_1, P_2) = \frac{|A_1| + \sum_{m=2}^{m_{\max}} |A_m| \cdot (1 + q(m))}{|A| + |U| + \sum_{m=2}^{m_{\max}} q(m) \cdot |A_m|} \quad (5.24)$$

This measure considers the node similarities only for the creation of related clusters. Once clusters and node assignments are identified, these are considered in an absolute way; the node similarity values are neglected from this point on.

As an example for both of these process similarity metrics, we consider the two process model outlined in Figure 5.5. The cluster-based variant ignores the similarity values for the corresponding node pairs:

$$\text{sim}_{\text{PM}}^{\text{Cluster}}(P_1, P_2) = \frac{6 + 4 \cdot (1 + 3) + 12 \cdot (1 + 4)}{22 + 3 + 3 \cdot 4 + 4 \cdot 12} = \frac{82}{85} \approx 0.9647$$

Generally, the two related cluster pairs containing 3 nodes in each model are strongly emphasised, weighted five times higher than single assigned nodes. In the example, "Cluster 5" with a very low similarity value and the 3 unassigned clusters decrease the overall similarity value (cf. Fig. 5.5b).

5.5.5 Computational Complexity and Limitations

Generally, rather than improving computational efficiency, the primary intention of this approach is to improve node assignment quality. Supporting this, as also experienced in [48], the computation of string-based metrics is "very fast". Additionally, in our implementation, the runtime of dictionary lookups for the semantic measures has been significantly improved by implementing simple cache mechanisms. For example, the runtime for the computation of the related cluster pairs in the process model pair shown in Figures 5.4 and 5.5 took 217 ms. While the runtime of a specific computation heavily depends on the available computational power and the efficiency of the implementation, the *O*-Notation provides general comparability.

We investigate the computational complexity of the computation of related cluster pairs, proceeding from two PMGs, $P_1 = (V_1, E_1, \dots)$ and $P_2 = (V_2, E_2, \dots)$. We define $n = |V_1|$ and $m = |V_2|$ as number of nodes, and $g = |E_1|$ and $h = |E_2|$ as edge counts, respectively. Without loss of generality, we assume $n \geq m$.

Concerning the computational complexity of this approach, the calculation of the correspondences matrix is $O(mn)$ for n and m nodes of the two models, respectively. The implementation solves the assignment problem, whose processing costs $O(n^2m)$ [2, 30].

The complexity of the computation of the initial clusters (SESE regions) for a graph with e edges corresponds to $O(e)$ [100]. In our case, this is $O(g + h)$. As graphs of business process models are typically weakly connected, we can assume $g \ll n^2 - n$ as well as $h \ll m^2 - m$ (cf. Sect. 5.3.1). This leads to

$$O(g + h) \ll O(n^2 - n + m^2 + m) \leq O(n^2 - n + n^2 - n) = O(2n^2 - 2n) \approx O(n^2).$$

It follows that the computation of initial clusters costs $O(g + h) \ll O(n^2)$.

Summarising, the worst case computational complexity of our approach, $O(cn^2m) \leq O(n^3)$, is polynomial and depends on the complexity of solving the assignment problem, which is a requirement for the computation of node mappings.

Concerning limitations, process behaviour has, so far, not explicitly been considered. We include structural investigations that, e.g., demand the same order of activities in clusters and distinguish cluster types. However, gateway semantics, for example, are not considered so far, although their inclusion is thinkable. On word level, we experience common NLP challenges such as coping with stopwords such as “not”. Further, often words refer to meanings that are too specific such that they are not covered by word *lexica* (WordNet or Wikipedia). For process models including many gateways compared to the node count, the determined related cluster pairs tend to be of small sizes (scattered node assignments), which affects the results of the process similarity measure. In this case, the process similarity considerations for the retrieval scenario are reduced to pure node label similarity, neglecting parts of structural information. As countermeasure, the weighting function $q(m)$ can be set to an extended linear (e.g., $q(m) = cm + d$) or polynomial variant (e.g., $q(m) = cm^d$), in order to increase the emphasis on clusters of size 2 (assuming there are very few clusters that are larger).

5.5.6 Implementation of ProcSim.KOM

In order to realise the desired decision support that can be provided by our concept, *related cluster pairs*, we implemented the outlined approach as the plug-in ProcSim.KOM for the process mining framework ProM⁵ [253]. The implementation realises the computation of the discussed concept and provides an interactive user interface with additional functionalities. In this section, we shortly outline the architecture as well as the user interface.

Architecture

We implemented a proof-of-concept application, ProcSim.KOM, which realises the computation of related cluster pairs and its application for process comparison and process retrieval. We provide an overview of the general architecture of ProcSim.KOM using the FMC notation⁶ in Figure 5.7.

ProcSim.KOM uses the process model processing routines from ProM 5.2. Via the plug-in, the user invokes the *Analysis Panel*, which coordinates the user interaction. As part of its user interface, all above introduced metrics can be selected and combined. Their weights and the threshold is customisable. The default settings equal to those that yielded

⁵ The Process Mining Framework ProM, version 5.2, <http://www.promtools.org/prom5/>, last accessed 2011-08-08

⁶ Fundamental Modeling Concepts (FMC), <http://www.fmc-modeling.org/>, last accessed 2011-08-11

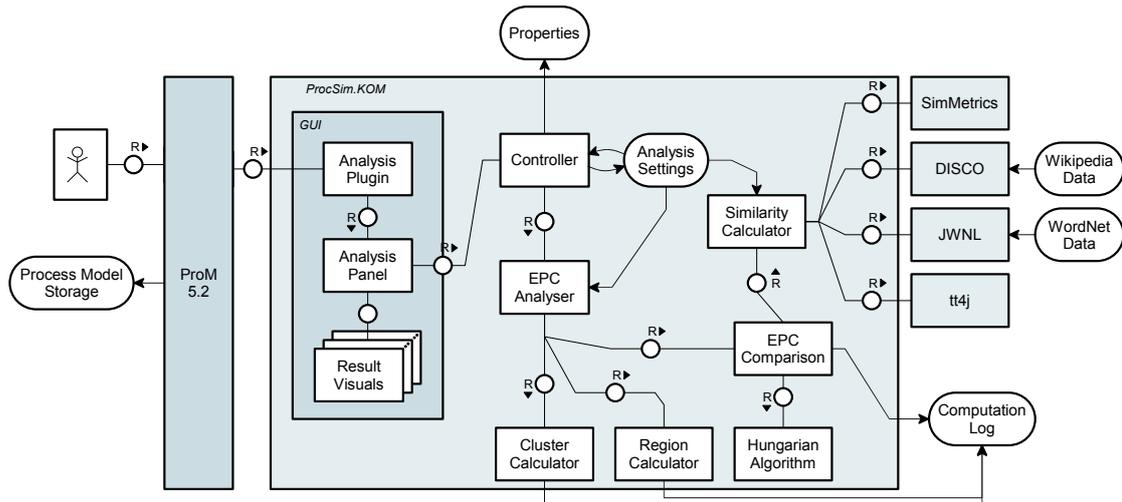


Figure 5.7: ProcSim.KOM: Architecture

the best results during our evaluation. The *Analysis Panel* invokes the computation process via the *Controller*.

Basically, the *Controller* component coordinates the computation. It reads general settings (such as the location of the Wikipedia and WordNet data) from *Properties* and stores user- and computation-specific information in *Analysis Settings* (such as metric configuration and user-specified weights). Subsequently, the *Controller* invokes the *EPC Analyser* component that performs the actual comparison in three steps. First, the *EPC Comparison* compares the nodes of the two given process models by performing all similarity computations. Several external libraries (see below) are used to determine similarity values, given the specified metrics, for all node combinations. Further, using the Hungarian algorithm, the final node assignment matrix is computed. *EPC Comparison* and all further components write their results into the *Computation Log*, realised by a log file. Second, the *Region Calculator* computes SESE regions. Third, based on these regions and the computed similarity information, the clusters and, particularly, the related cluster pairs, are identified and merged (Step D).

The *Result Visuals* visualise the results. For each process model pair, the comparison results are illustrated in a new tab (cf. Fig. 5.9).

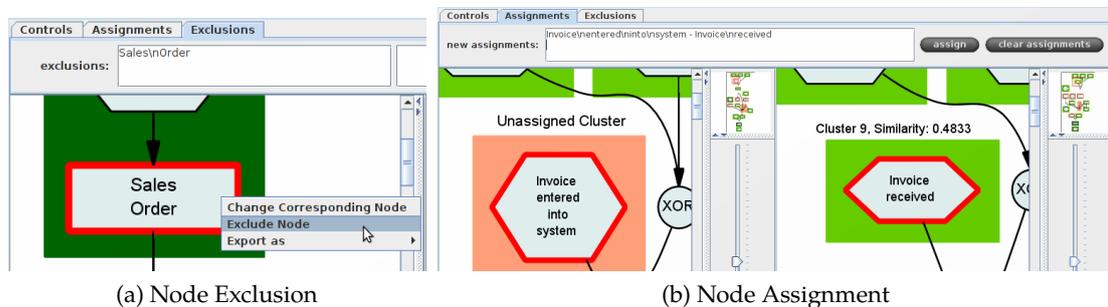


Figure 5.8: ProcSim.KOM: Manual result corrections

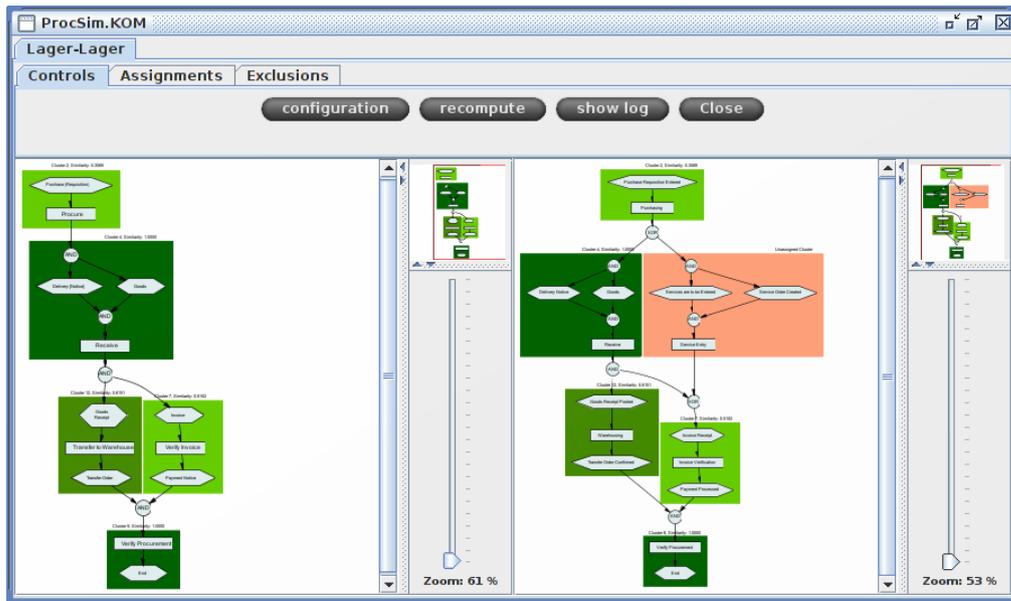


Figure 5.9: ProcSim.KOM: Result visualisation

ProcSim.KOM uses external libraries. *SimMetrics* is a collection of similarity metrics [35]. *DISCO* is a framework for access to Wikipedia word relations⁷. It also integrates some similarity measures, such as the Lin measure. The Java WordNet Library (JWNL)⁸ provides access to WordNet. The metrics based on WordNet we used are realised by ProcSim.KOM. The Java library *tt4j*⁹ is a wrapper for the tree tagger and lemmatising framework written in Perl by Schmid [219].

Result Visualisation and User Interface

The last step of the computation performs the visualisation of similarities and differences of the two models. All clusters and related cluster pairs are visualised in the interactive graphical user interface (GUI).

The process model visualisation colours the all outlined clusters. It chooses the colour from a set of shades of green according to the similarity value. A light green, for example, indicates weak similarity. Unassigned clusters generally appear in red. This assures a good user intuitiveness and aims at increasing user acceptance (cf. Fig. 5.5). Additionally to the visualisation, all results are logged in detail, and the GUI provides the functionality, to export a displayed process model in the graph description language *dot*¹⁰.

Once the computation has been performed, an interactive interface allows to adjust the results. The user can make additional manual node assignments, delete computed assignments, and trigger a new computation of the model comparison (cf. Fig. 5.8a). Additionally, the weights of the applied metrics, and the similarity metric combination can be adjusted.

⁷ <http://www.linguatools.de/disco/disco.html>, last accessed 2011-08-18

⁸ Java WordNet Library (JWNL), available at <http://sourceforge.net/projects/jwordnet/>, last accessed 2011-08-05

⁹ Tree Tagger for Java (tt4j), available at <http://code.google.com/p/tt4j/>, last accessed 2011-08-05

¹⁰ Graph Visualisation Software, cf. <http://www.graphviz.org/>, last accessed 2011-08-11

In summary, the user interface provides the actual decision support for the process conformance assessment. By the colouring and the ability to directly influence node assignments, the GUI provides user-friendly and intuitive access to the provided functionality.

5.6 Evaluation

A central aspect of every governance approach is the process conformance assessment, i.e., the management of internal processes and their adherence to reference processes. The duty of corresponding organisational entities (in the operational model) is the design of conform processes, as well as the assessment of the conformance of existing processes. Based on our concept of *related cluster pairs*, we realised two approaches for

- (i) the identification of relevant reference processes for a given flow of activities supporting the design of conform processes (*process model retrieval*) and
- (ii) the comparison of two process models and identification of their difference supporting process conformance assessment (*process model comparison*).

In both cases, we make use of the “SAP R/3 Reference Processes” (SPRM) as test case [107]. The SPRM consists of 604 process models in the EPC process notation. EPC are generally composed of functions, events, and connectors (cf. Def. 3). The process collection organises the processes of 28 business domains (e.g., *Sales and Distribution*, *Plant Maintenance*, or *Financial Accounting*). The number of nodes per process models including (and excluding) connectors range from 3 (3) to 78 (130). On average, each process model contains 15.5 (20.7) nodes. They describe established standard procedures that have been deployed in practice, and have supported the customisation of the SAP Enterprise Resource Planning system. For further information on the test case refer to Appendix C.2.

For similarity computation, we make use of all string-based and semantic similarity measures that have been outlined in Sections 5.4.2 and 5.4.3, respectively. Further, we use several (hybrid) combinations of these measures. Using $\oplus_{w_{\text{Sem}}, w_{\text{Str}}}$ on the string-based measure sim_{Str} and the semantic measure sim_{Sem} (acc. to Eq. 5.6), we define a hybrid measure sim_H is defined as follows:

$$\text{sim}_H(n_1, n_2) = \text{sim}_{\text{Sem}}(n_1, n_2) \oplus_{w_{\text{Sem}}, w_{\text{Str}}} \text{sim}_{\text{Str}}(n_1, n_2), \quad (5.25)$$

where $w_{\text{Sem}} + w_{\text{Str}} = 1$ with $0 \leq w_{\text{Sem}}, w_{\text{Str}} \leq 1$. Two nodes are considered similar, iff $\text{sim}(n_1, n_2) \geq t$.

For the two scenarios, we performed distinct, comprehensive evaluations. The results are outlined in the following Sections 5.6.1 and 5.6.2.

5.6.1 Process Model Comparison

In the first scenario, we assess the approach’s quality of identifying (correct) node assignments. Given two process models in the same process notation, the problem is to identify relevant process element mappings. Generally, it can be assumed that not all process models have been designed by the same expert. In this context, the correct mutual assignment of nodes, proceeding from the activity labels, is not a trivial problem, and has been tackled by most of the related approaches (cf. Sect. 5.2.2).

For the solution of this problem, we apply related cluster pairs that identify similar process activities, as well as similar and different regions these reside in.

5.6.1.1 Evaluation Setup

We evaluated the approach using the 604 processes from the SPRM. We randomly selected 48 processes from the collection. For the evaluation, we modified each of them by applying modifications, forming 48 pairs of process models. We applied the following two modification types:

LABEL CHANGES (LC). Node labels of the original process model are rephrased without changing the meaning for a human reader. For example, “Need for correction of benefits plans” has been changed to “Requirement for rectification of benefits plans”¹¹. This variation simulates different process model authors and serves to test the approach concerning its label matching ability, especially concerning semantically similar activity description.

LABEL AND STRUCTURE CHANGES (LS). Additionally to label changes, changes of the structure are performed. Nodes (functions and events) have been randomly deleted or new ones have been inserted without breaking the EPC modelling rules (cf. Definition 3). Nodes are reordered by randomly swapping two functions and/or two events, respectively. Additionally to challenging the ability to match semantically similar activities, this variation intends to simulate a real world scenario by further alienating process models.

We perform these modifications in three modification intensities $r1$, $r3$, and $r6$. While in $r1$, 10% of the nodes (not considering gateway nodes) of a process model are subject to modification, in $r3$ and $r6$ we change 30% and 60%, respectively. This results in 6 evaluation classes (3 intensities and 2 types of modification). In $LCr3$, for example, 30% of the nodes’ labels are modified. In the class $LSr6$, 60% of the nodes are subject to label changes, and another 60% are affected by structural and node label changes.

In order to produce unbiased results, we perform a three-fold cross-validation for each measure [227, 269]. Each fold contains 16 process pairs. The training set consists of 32 process model pairs (2 folds), and we test the learned configuration on the remaining set of 16. During cross-validation, the algorithm learns the parameters w_{Str} and the similarity threshold t . As baseline for our consideration, we use the Levenshtein distance (cf. Eq. 5.7).

In order to assess the quality of the matching approach, we make use of *match accuracy* for automated matching tasks [142]. This metric evaluates the quality of matching algorithms that require human quality assessment. The metric is based on the effort a user needs to convert an automatically created matching result

$$S = \{(s_1, t_1), \dots, (s_n, t_n)\}$$

suggested by the matching algorithm (called *prediction*) into the correct result

$$P = \{(p_1, r_1), \dots, (p_m, r_m)\}$$

(called *gold standard*), where s, t, r, p are nodes, and n the number of predicted matches and m the number of correct matches. The required effort to transform the result S into

¹¹ performed in the process model with ID 1Ar_m86y

the intended result P is measured in terms of adjustment operations (additions and deletions) proceeding from S . It is assumed that both operations require the same effort. The match accuracy is defined as follows [142]:

$$\text{Acc}_{\text{Match}} = 1 - \frac{(n - c) + (m - c)}{m} = \frac{c}{m} \left(2 - \frac{n}{c} \right), \quad (5.26)$$

where $c = |S \cap P|$ is the number of correct suggestions, $(n - c)$ is the number of false positives (to be deleted from S), and $(m - c)$ the number of false negatives (to be added to S). If the matching process would be performed manually, m add operations would be required.

In this context, the *recall* of a measure ($\frac{c}{m}$) is the ratio of correctly predicted matches and the number of all matches. *Precision* ($\frac{c}{n}$) corresponds to the amount of correctly predicted matches in the set of predicted matches. A way to consider both qualities is the traditional F_β measure that measures a test's quality considering both values precision and recall. It is calculated as the dynamic harmonic mean of precision and recall [137] and defined as

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}} \quad (5.27)$$

Throughout our evaluation, we use the frequently used F_1 measure with $\beta = 1$ that equally weights precision and recall:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5.28)$$

5.6.1.2 Evaluation Results

We performed a cross-validation evaluation on the described test data for each of the following single measures:

- the Levenshtein string-edit distance sim^{Lev} (baseline),
- the Jaccard coefficient sim^{Jac} ,
- the synonym measures $\text{sim}^{\text{Smean}}$ and sim^{SME} ,
- the WordNet word distance sim^{WND} , and
- the semantic first order measure sim^{SFO} based on Wikipedia.

The results for the whole data set are averaged over the three accuracy values resulting from the cross-validation folds (cf. Fig. 5.10 and Tab. 5.3).

The results show that the baseline, the Levenshtein distance at 85.33% accuracy, is outperformed by all of the other measures (cf. Figure 5.10 and Table 5.3). Among the string-based measures, the Jaccard coefficient sim^{Jac} performs best with an accuracy of almost 87%. The WordNet distance does not perform much better than the baseline. In contrast, the synonym functions based on WordNet reveal clearly better results than the string-based measures. Best performing among the semantic measures, and far better than the Jaccard measure, is the synonym measure sim^{SME} using the synonym constant 0.5 and the Monge Elkan measure as aggregating function (91.62% accuracy). The other

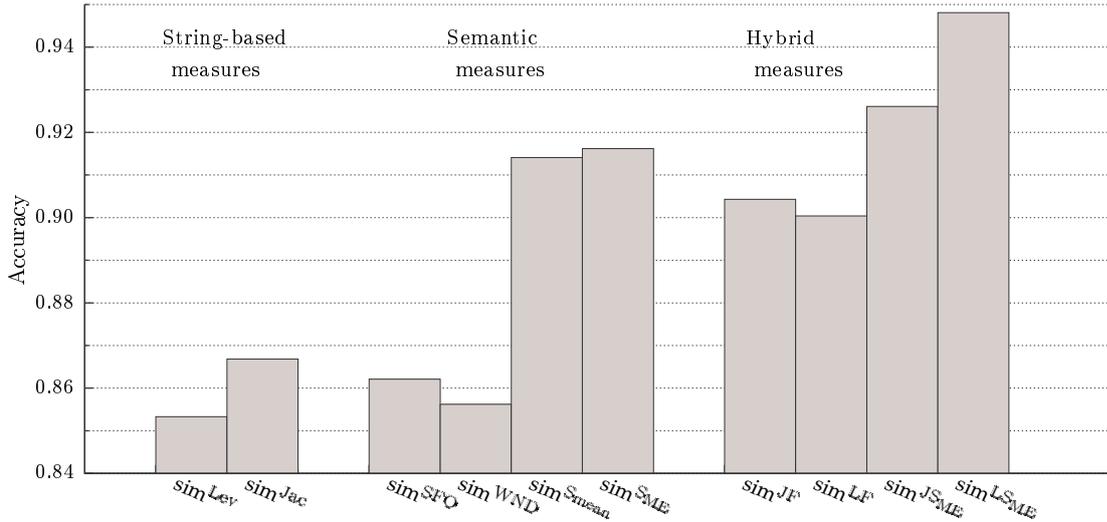


Figure 5.10: Process model comparison – cross validation results (accuracy per metric)

synonym measure $\text{sim}^{\text{Smean}}$ follows closely at 91.42% accuracy. However, it is not as stable as sim^{SME} concerning the similarity threshold (cf. Table 5.3).

As shown in Figure 5.10 there is a massive gap between the results of the string-based measures, the Wikipedia-based semantic first order measure and the WordNet distance on one side, and the synonym-based measures on the other side. The synonym functions (sim^{SME} and $\text{sim}^{\text{Smean}}$, also based on WordNet) reveal far better results. The gap indicates an improvement of almost 6% (ca. 5 percentage points). This is also reflected by the corresponding F_1 measure results (cf. App. C.1).

In order to achieve further improvements, we combine string-based and semantic metrics, that performed best in these experiments. Accordingly, using the weighted combination $\oplus_{w_{\text{Str}}, w_{\text{Sem}}}$ (cf. Eq. 5.6), we propose the following hybrid metrics (cf. Tab. 5.3).

$$\text{sim}^{\text{LSME}} = \text{sim}^{\text{Lev}} \oplus_{w_{\text{Str}}, w_{\text{Sem}}} \text{sim}^{\text{SME}} \quad (5.29)$$

$$\text{sim}^{\text{JSME}} = \text{sim}^{\text{Jac}} \oplus_{w_{\text{Str}}, w_{\text{Sem}}} \text{sim}^{\text{SME}} \quad (5.30)$$

$$\text{sim}^{\text{LF}} = \text{sim}^{\text{Lev}} \oplus_{w_{\text{Str}}, w_{\text{Sem}}} \text{sim}^{\text{SFO}} \quad (5.31)$$

$$\text{sim}^{\text{JF}} = \text{sim}^{\text{Jac}} \oplus_{w_{\text{Str}}, w_{\text{Sem}}} \text{sim}^{\text{SFO}} \quad (5.32)$$

Clearly, the metric sim^{LSME} outperforms all other measures at an accuracy value of almost 95% on unseen data as average of three folds. Second ranks sim^{JSME} with 92.61%. The combination of Jaccard coefficient and $\text{sim}^{\text{Smean}}$ achieves 91.41% (cf. Tab. 5.3). This ranking is confirmed by the F_1 measure values for these metrics (cf. App. C.1).

As part of the results of sim^{LSME} , concerning the modification type *LC*, the average accuracy (of the 24 contained pairs) is at 97%, while for *LS* it equals to 93%. For the modification classes *LSr1*, *LSr3*, and *LSr6*, for example, the average accuracy values are at 97%, 95%, and 87%, respectively. With increasing modification intensity, the accuracy decreases. However, increasing the count of changed nodes by 50% only results in a loss of 10 percentage points in terms of accuracy.

The threshold and combination weight values that were learned for each metric during the evaluation are outlined in Table 5.3. Apart from the mean values (across folds), the standard deviation indicates the stability of the measure. For half of the single measures,

Table 5.3: Cross validation results for process model comparison

<i>Measure</i>	<i>Id.</i>	<i>Eq.</i>	<i>Accuracy</i> [%]	F_1 [%]	<i>Mean values for</i>	
					<i>Threshold</i>	<i>Weight w_{Str}</i>
<i>Single string-based measures</i>						
Levenshtein distance	sim^{Lev}	(5.7)	85.33	92.93	0.10 (± 0.0000)	–
Jaccard coefficient	sim^{Jac}	(5.8)	86.68	93.23	0.10 (± 0.0000)	–
<i>Single semantic measures</i>						
Synonym (mean)	$\text{sim}^{\text{Smean}}$	(5.10)	91.41	95.76	0.25 (± 0.0613)	–
Synonym (ME)	sim^{SME}	(5.12)	91.62	95.88	0.21 (± 0.0000)	–
WordNet distance	sim^{WND}	(5.14)	85.62	93.03	0.05 (± 0.0471)	–
Semantic first order	sim^{SFO}	(5.16)	86.21	93.35	0.02 (± 0.0125)	–
<i>Combined measures</i>						
Levensht./ Synonym (ME)	sim^{LSME}	(5.29)	94.81	97.48	0.18 (± 0.0000)	0.10 (± 0.0000)
Jaccard/ Synonym (ME)	sim^{JSME}	(5.30)	92.61	96.41	0.05 (± 0.0000)	0.40 (± 0.0000)
Levensht./ Sem. first order	sim^{LF}	(5.31)	90.04	95.19	0.16 (± 0.0283)	0.23 (± 0.0943)
Jaccard/ Sem. first order	sim^{JF}	(5.32)	90.43	95.38	0.02 (± 0.0047)	0.43 (± 0.0624)

Standard deviation provided in parentheses; w_{Str} indicates the weight of the string-based part measure

the threshold and the weight have been determined unambiguously (standard deviation of 0.0). For the hybrid measures, two show no deviation at all, and two have light deviations. In particular, all measures that performed best (sim^{JSME} , sim^{LSME} , sim^{SME}), excel concerning parameter stability. For this reason, these metrics have been selected for the evaluation of the retrieval approach (i.e., our process model similarity measure) in the following section.

Concluding, the evaluation shows that the measures provide very promising results concerning work assistance compared to manual comparison. The accuracy values of the best metrics (sim^{JSME} and sim^{LSME}) are clearly beyond 90%, the F_1 measure values clearly greater than 95%. Our hybrid measures clearly improve the results achieved by the respective single measures. The top measures (sim^{JSME} and sim^{LSME}) provide reliable results, independent on their training data. Further, the hybrid measures we specified do not only reveal the best results, but also are the most stable measures on unseen data.

In the context of process conformance assessment, our approach provides very good work support. When assessing the conformance of a process using our approach, on average 90% of the correct activity assignments are identified, 95% of all retrieved node assignments are correct, and 95% of all correct node assignments are found.

5.6.2 Process Model Retrieval

The second scenario aims at the identification of relevant reference processes for a given flow of activities. It targets the support of the design of conform processes by providing authoring support during process design time.

Specifically, it addresses the process model retrieval problem. In the context of this thesis, this is defined as follows. Let $q_j \in Q$ be a query process model, where Q is the set of searches performed, and $D = \{d_1, \dots, d_n\}$ a collection of process models arranged in a process model repository. The problem is to identify all process models $d_i \in D$ that are similar to q_j and rank them according to their similarity. This is a typical problem of BPM and has a variety of application scenarios (cf. Sect. 2.3.1).

We apply the related cluster pairs approach for solving this problem, combining node label similarity with structural characteristics to calculate process model similarity. For the selection of measures, we use information concerning performance and parametrisation learned during the first scenario's evaluation.

5.6.2.1 Evaluation Setup

We performed the evaluation using the same set of 604 process models as described above. In particular, we used the same test case as used in [48], consisting of 100 randomly selected process models that form the process repository and 10 query models (taken out of the 100). Pairwise, two query models have been changed according to 5 modification types.

- Two models (1+2) have not been changed at all.
- Models 3 and 4 have been subject to label change. Labels are changed without changing the meaning for a human reader, challenging the label matching ability.
- Two further models have been reduced to approximately 50%-subgraphs of their own (5+6) as a structural variation.
- In two models (7+8), the connectors of the original model were randomly changed to different connectors, affecting at most the model's behaviour.
- The last two models have been modified by randomly swapping of function pairs and event pairs (9+10) to create structural and behavioural changes.

The 100 process models of the test case (the *process pool*) have been annotated by process model experts. For each process model contained in the collection, process model experts have been asked to assess its similarity. For every process model, each process model's relevance has been described on a scale from 1 to 7. This resulted in 1000 annotations. Model pairs assigned a 5 or above are considered similar or, in terms of retrieval, relevant. This way, a list of relevant models has been identified for each of the ten query models. For further details of the design of the test case, refer to [48].

The models contained in the collection of 604 process models do not only specify disjunct procedures; in fact, there are many overlapping procedure descriptions, for example, for the *procurement* process, there are seven descriptions. For each modified process model, the test case contains more than a single relevant process. In fact, for each of the ten query models, process experts identified at least 5 relevant models and 11 on average (cf. Tab. C.3). In the course of this chapter, we refer to this process pool as $D = \{d_1, \dots, d_{100}\}$ with index i .

As a baseline, we apply two standard text search engines on the ten query models. For this, all event and function labels of all 110 models (10 query models and the process pool) are extracted into a text file each. For the search, we use the Indri Search Engine¹², as well as one of the standard search engines used in research, Apache Lucene¹³. Further, we compare our results to the state-of-the-art, i.e., results of the *Label Similarity* metric proposed in [48]. Further, we marginally consider the *Structural Similarity* and *Behavioural Similarity* by the same authors. A comparison of these and related cluster pairs

¹² available from <http://www.lemurproject.org/>, last accessed 2011-08-09.

¹³ <http://lucene.apache.org/>, last accessed 2011-08-14

is interesting, as our approach also considers structural aspects. The comparison with their *Behaviour Similarity* is just for information purposes.

For each measure, we perform one search experiment per query model. We measure the performance of these experiments in terms of *average precision*, *R-Precision*, as well as *first-n-Precision* for $n = 5$ and $n = 10$ [137]. The average precision characterises the quality of a single search query. For one search query, it equals the *mean average precision* (MAP). For their calculation we use the following metric. Q is the set of searches that are performed. For each query $q_j \in Q$, the set of relevant documents is defined as $\{d_1, \dots, d_{m_j}\}$, and R_{jk} as the ranked list of retrieval results from the best result until and including document d_k . MAP is defined as follows [137]:

$$\text{MAP}(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} \text{Precision}(R_{jk}) \quad (5.33)$$

R-Precision is the precision of a query calculated after the first R documents have been found, where R is the number of relevant documents for this query. So R varies with the query. Further, at the *R-Precision* of a query, the precision value equals the recall value, i.e., it indicates the same as the *break-even-point* of precision and recall. *First-n-Precision* provides the query precision after a cut-off after the n^{th} predicted document [137].

In preliminary experiments we found, that using stemming as word preprocessing did not improve results in all cases. Especially for semantic measures, improvement by stemming mostly fails, as dictionaries mostly cannot cope with stemmed strings. This is why we did not employ Porter's stemming algorithm for the metrics sim^{LSME} and sim^{JSME} .

During the evaluation, we investigate two measures: sim^{LSME} and sim^{JSME} . We use the weights for the contained string-based measure and the thresholds as learned during the cross-validation evaluation (cf. Sect. 5.6.1). They are indicated in Table 5.3. We use both the *node-based* and the *cluster-based* process model similarity (cf. Eq. 5.23 and 5.24, resp.) with each of the two measures. The first is defined as

$$\text{sim}_{\text{PM}}^{\text{Node}}(q_j, d_i, \text{sim}^{\text{LSME}}), \quad (5.34)$$

where q_j refers to the ten query models in Q with $|Q| = 10$, and d_i iterates over the $n = 100$ models of the process pool D . For the cluster based approach we use the following metric:

$$\text{sim}_{\text{PM}}^{\text{Cluster}}(q_j, d_i, \text{sim}^{\text{LSME}}) \quad (5.35)$$

The metric sim^{JSME} is used analogously.

5.6.2.2 Evaluation Results

The results of the retrieval evaluation are shown in Figures 5.11 and 5.12, and in Table 5.4. For each query, the results of the two search engines, the related *Label Similarity*, and our three best performing metrics are outlined. We compare them in terms of average precision (cf. Fig. 5.11).

In 90% of the queries, our result clearly outperforms the baseline as well as the *Label Similarity* approach proposed by related work. Further, in 70% of the cases, our approach outperforms the Lucene text search, and in also 90% it yields better results as the second baseline, the Indri text search.

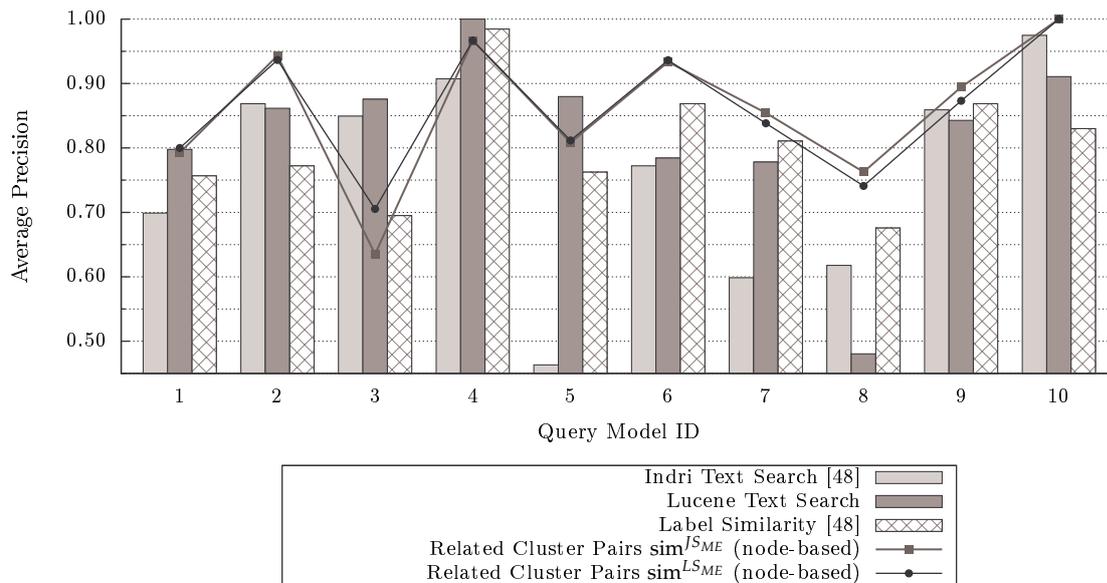


Figure 5.11: Average precision per query model

For process model 3, sim^{LSME} performs better than the label similarity approach. Both search engines, however, perform better. In case of query 4, the performance of sim^{LSME} is slightly worse than label matching. However, a definite improvement concerning Indri is obvious. In case of query 4 and 5, Lucene outperforms all other specialised approaches. For all remaining queries, our approaches represent a clear improvement compared to all both text search approaches and related work.

In seven cases, all our measures outperform the Label Similarity measure. In particular, with the model queries 1, 2, 5, 6, 8, and 10, our approach shows a clear improvement of ca. 5 percentage points in each case. For model 2, the gap represents an improvement of 21%. For models 3, 7 and 9, our measures clearly yield improvements, while for model 4, our best measure ranks 1.8 percentage points behind related work's results. Regarding the search engines, we clearly outperform the Indri text search in all cases except query model 3. The results show that Lucene text search performs better than Indri text search in six cases, for model 5 even with a gap of 30 percentage points. In turn, Indri's advance on Lucene in the other cases does not exceed 15 percentage points. Our measures outperform Lucene for seven models. In case of model 8, our measure even achieves a 59% improvement. For models 3, 4, and 5, the Lucene text search achieved the best results, also outperforming related work. However, these query models expose particularities. Query model 4 exposes the smallest number relevant models among all queries (5 out of 100) (cf. Tab. C.3). While Lucene reported all five of them first, our approaches ranked the first four correctly. Query model 3 is a similar case, where 7 out of 100 models are relevant. In these cases, the impact of one not correctly reported document is higher in comparison to other queries, due to the very small number of relevant documents.

Summarising, a definite improvement compared to the Indri text search engine is obvious. Further, in the majority of cases, our approach achieves better results than both Lucene text search, and related work.

Table 5.4: Overall results of process retrieval metrics

<i>Metric</i>	<i>MAP</i>	<i>Mean R-precision</i>	<i>First-20-precision</i>	<i>First-10-precision</i>	<i>First-5-precision</i>
sim^{LSME} (node-based)	0.8609	0.7774	0.4550	0.7900	0.9600
sim^{JSME} (node-based)	0.8591	0.7708	0.4600	0.7900	0.9400
Label Similarity [48]	0.8000	–	–	0.7900	–
Lucene Search Engine	0.8211	0.7491	0.4450	0.6800	0.9400
Indri Text Search [48]	0.7611	–	–	0.7000	–
Structural Similarity [48]	0.8300	–	–	0.7800	–
Behavioural Sim. [48]	0.8000	–	–	0.7400	–

In addition to the average precision above, we considered further performance measures: MAP, the mean-R-precision, and first-n-precisions. Further, besides the six investigated measures above, we considered the results of Structural Similarity and Behavioural Similarity as proposed in [48] that have also been tested using the same test case (cf. Tab. 5.4). Note that the latter often operate at a higher computational complexity. For details on structural and behavioural similarity, refer to Section 5.2.1.

Referring to the MAP values, our approaches outperform all others, including the text search engines, structural and behavioural considerations. The node-based measures sim^{LSME} and sim^{JSME} show very little difference in performance, while the cluster-based measure sim^{LSME} cannot yield improvements. In the following, we only refer to the node-based approaches. Compared to Label Similarity, our measure sim^{LSME} achieves an improvement of 7.6% (6.09 percentage points), compared to Lucene and Indri, the advances represent improvements of 4.8% and 13.1%, respectively. Note that also Lucene outperforms Label Similarity by 2.6%. While the improvement of our measure compared to Behavioural Similarity is also at 7.6%, it achieves still an 3.7% improvement compared with Structural Similarity.

The results for the mean R-precision show that at a position at the ranked result list where all relevant process models could have been found (at R), by our measure sim^{JSME} more than two third of the correct models have been retrieved on average, which is a promising result. For example, for sim^{LSME} this means having reported 77,74% of all relevant documents, 77,74% of them are relevant. These results demonstrate a good general performance. For related work results, unfortunately, the authors have not provided the mean R-Precision.

Regarding the first-10-precision, all approaches perform comparably, reporting 79% on average of the relevant documents among the first 10 results. In terms of the first-5-precision, our approach sim^{LSME} outperforms Lucene Text Search. However, if the number of relevant documents is low, say 5, the *first-n-precision* metric (at $n = 10$) is limited to indicate low precision values¹⁴ – in that case to a maximum of $5/10 = 0.5$. This is not the case for the R-precision measure. The results show that, at a point where all relevant process models could have been found for one query, more than two third of the models have been identified by our approaches in average, which is a promising result.

In summary, according to standard performance measures from information retrieval, our approach clearly outperforms all related measures, including the text search baselines and structural and behavioural investigations. Our measures, especially sim^{LSME} ,

¹⁴ The overall minimum number of relevant process models per query in this test case is 5 (for query model it is 4).

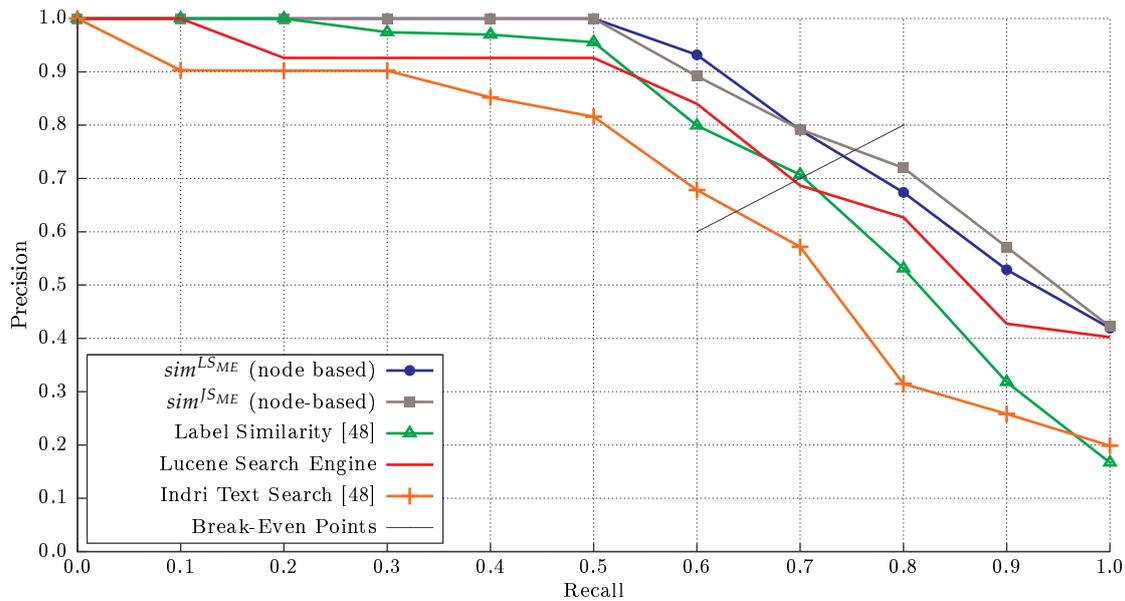


Figure 5.12: Interpolated precision-recall curve for process model retrieval metrics

achieve an almost 8% improvement compared with related work, and more than 5% improvement compared with the best text search engine.

We summarise the overall results of this evaluation in an interpolated precision-recall curve (cf. [137], cf. Fig. 5.12).¹⁵ The values for the Indri text search, as well as the Label Similarity have been taken from Dijkman et al. [48]. Overall, the node-based measures sim^{SME} as well as sim^{LSME} perform best. In particular for the latter, sim^{LSME} , at a recall level of 0.5 (i.e., having reported 50% of the relevant process models as part of the result list) 100% of the listed models are relevant. After having reported more than the half of relevant documents (at 0.6 recall), the precision for sim^{LSME} is still clearly beyond 90%. For a search engine, this indicates a very good result. Compared to Label Similarity, at the recall level of 0.6, the advance of our best metric refers to a performance increase of 16.6%. While the Indri text search is clearly outperformed by all tested metrics, the related work approach overall performs comparably to Lucene. Regarding the break even points, a general ranking can be identified. Our metrics clearly perform best, while Label Similarity and Lucene rank second. All metrics perform better than the Indri text search engine.

Concluding, our approach outperforms the current related work as well as established search engines. This indicates that the additional computation complexity of more sophisticated search approaches is rewarded.

Concerning related work, we show that an efficient combination of hybrid node label similarity measures with structural qualities of a process model can outperform state-of-the-art algorithms. We combined a configured WordNet-based measure (semantic consideration) with standard string-based measures such as the Levenshtein distance and the Jaccard measure, respectively. These combinations have not yet been applied in the context of the process retrieval problem so far, yet they yield clear performance improvements. Further, using related cluster pairs, our approach integrates the structural

¹⁵ The x-axis corresponds to the recall intervals [0.00, 0.05), [0.05, 0.15), ..., [0.95, 1.00].

investigation of process models. Our approach especially considers the ordering of similar node groups, exceeding the focus of pure node label consideration. In particular, it keeps computational complexity at a manageable level, while related approaches most apply the established graph-edit-distance method, which is costly in terms of computational complexity.

Concerning related work, we assume that word preprocessing is a major reason for the improvement. Lemmatising has, to the best of our knowledge, not been utilised in related approaches. In preliminary experiments we found, that using stemming did not improve results in all cases. Especially for semantic measures, improvement by stemming mostly fails, as dictionaries mostly cannot cope with stemmed strings. This is why we did not employ Porter's stemming algorithm for the measures sim^{LSME} and sim^{JSME} . Further, we carefully revised the stop word list. In traditional stop word lists, "needs", "omitted", or "part" and similar ones are considered function words. However, for processing of operational process models, we found that many of these can have high impact on a label's overall meaning, and in particular improve results, especially when using semantic similarity. In fact, we used lemmatising for all measures. For semantic similarity measures, reducing labels and words to a meaningful form is an unconditional technique for preserving information and meaning. It ensures reasonable lookups and obviously improves the overall performance.

As future work, the label meanings per node can be investigated in more depth. Current *community mining* approaches from natural language processing, for example, identify the basic notion of blog or forum entries in the internet. Thus, valuable information for the automated processing of these information, especially for the extraction of opinions or notions (positive expression vs. negative expression) can be retrieved and made processable to further improve the matching process. (cf., e.g., [129, 181]). This method can be used to analyse the intention of a node's meaning, i.e., to tackle the common problem of comparing labels that refer to opposite meanings and differ in one word only, e.g., a negation. Provided the information on the notion type could strongly improve the quality result of node comparison.

5.6.3 Discussion

Generally, as the selected test case is completely independent from the work at hand, we meet the requirement of "fair testing" as we do not make use of an artificially created test case that meets the requirements of the presented approach [274]. Further, to the best of our knowledge, we provide one of the few quantitative evaluations in this area whose results can be directly compared to other approaches as the test case is available. However, even though the evaluation approach and test case used were suitable to perform the evaluation of the related cluster pair approach, further evaluations could be performed to assess its performance in other contexts. In particular, the approach could be evaluated on a test case other than the SAP reference model, i.e., a test case consisting of more heterogeneous models – process models covered by the used reference model are aligned concerning terminology and modelling style.

Concerning taxonomy, our approach covers techniques that can be used to tackle different modelling taxonomies. As considerations of semantic node similarity, apart from the best performing synonym measures, we investigate a variety of semantic word relations (from used language via Wikipedia and from a linguistic viewpoint via WordNet)

and apply them together with a well-approved and established word reduction technique from the NLP domain, lemmatising. It avoids loss of word meaning and yields results that can always be further processed (in contrast to, e.g., stemming).

The utilised string-based measures are well-known reliable measures that achieve good results in a variety of application areas. As we could show, hybrid measures (i.e., efficient combination of string-based and semantic measures) clearly yield improvements compared to the respective single measures' results. From our viewpoint, an impairment of their results when applied to heterogeneous process models is improbable.

Concerning modelling style, as mentioned before, in case of scattered node assignments, our measure of process model similarity is reduced to a function of node similarities, which still yields good results (cf. Section 5.6.1.2). In this case, the weighting function can be adjusted (as outlined in Section 5.5.5). As our approach does primarily target the determination of correct node assignments, we consider changes in modelling terminology to have a higher impact on the results than different modelling styles.

Summarising, we use a variety of techniques to address different modelling terminologies. Given these details of our approach, the achievement of comparable or better results is thinkable. Unfortunately, appropriate test cases (especially for the heterogeneous case) are hardly available in the research community so far.

5.7 Conclusion

In this chapter, we introduced a novel concept for the analysis of process models called *related cluster pairs*. The approach combines similarity computation of process model elements with a structural investigation. We applied both string-based similarity and semantic similarity metrics and combined them to form more efficient measures. Using correspondences based on similarities, process models are hierarchically decomposed into region pairs containing similar elements. A merging process aggregates sets of neighbouring regions with similar content in both models and forms larger regions. Identified regions, so-called clusters, are both semantically and structurally similar. Related cluster pairs are applied in two different scenarios, process model *comparison* and process model *retrieval*.

In the first scenario, *process model comparison*, we focused on identifying the differences of two process models. We applied a number of standard node label metrics and proposed new hybrid ones, considering string-based similarity, as well as semantic similarity based on word meanings using Wikipedia and WordNet. We developed reasonable hybrid combinations of these metric types, and evaluated their performance in an elaborate scenario.

For this, we randomly selected 48 process models from an established test case in the BPM research community, the "SAP reference process model". We changed all of them according to two modification types while documenting changes, resulting in 48 process pairs. As evaluation, we performed cross-validation for string-based, semantic, and hybrid metrics. Within these experiments, we learned similarity thresholds, as well as the combination weights in case of hybrid metrics. As results, the majority of metrics, combined with the related cluster pair concept, show accuracy values of above 90%, which appears to be a very promising assistance to pure manual work. Concerning the learned parameters (similarity thresholds and combination weights), all metrics are stable in terms of standard deviation.

The best performing metrics are semantic measures, outperforming the string-based ones. However, the results of the semantic measures could still be improved by our proposed hybrid metrics. The best performing metric is a combination of a WordNet and string-edit distance metric proposed by us. It performs at 95% accuracy and exposes an F_1 score of 97.5%. As these results were achieved based on a test case that consists of established standard procedures, these results emphasise and prove the practical applicability of our approach.

Further, we defined a novel similarity notion for process models based on related cluster pairs. In the second scenario, we applied our process model similarity metric using hybrid node similarity metrics to the *retrieval of process models*. As test case, we used a set of 100 process models and ten process queries, annotated by BPM experts and based on the SAP reference process model. This data set has also been used by related process retrieval approaches.

Using the most successful and promising metrics pointed out by the first scenario, we performed 10 process model queries for each of the selected metrics. We compared our results with related work and a text search engine as baseline. In terms of mean average precision and further information retrieval metrics, our metrics outperformed both the text search engine, as well as state-of-the-art process metrics from related work.

In the retrieval and comparison scenarios, additionally to stop word removal, we considered special word preprocessing techniques such as lemmatising. We assume that these techniques partly caused the realised improvement. In related work, this technique has not been applied so far. Further, we assume that the lack of combination of structural process properties and hybrid measures of process nodes similarity in related work has lead to these results. Related cluster pairs, in fact, combine exactly these considerations.

Summarising, major benefits of our approach are the *performance*, its *extensibility* due to the decomposition potential, the *scalability*, the *similarity considerations*, and the *visualisation and manual result correction*.

PERFORMANCE. In the BPM community, this approach is one of the few approaches that actually performs an evaluation – making it comparable to other approaches, such as [47, 48]. According to numerous evaluation experiments we performed, both for comparison and retrieval of process models, the approach yields very good results (cf. above). It outperforms state-of-the-art approaches and two established text search engines.

EXTENSIBILITY. As clusters are either disjoint or nested, by identifying largest possible related clusters, every graph-based process model can be recursively decomposed into smaller process model fragments. Using this divide-and-conquer approach, the problem can be reduced to a size that can be processed by approaches that are more complex in terms of computation. Resulting process fragments can be subject to, for example, approaches focussing on process behaviour (cf. Sect. 5.2.1).

SCALABILITY. According to a 2009 survey, almost all current process retrieval approaches expose execution times that are not acceptable for usage within modern process model repositories [272]. The computation of related cluster pairs has polynomial complexity. On modern processors, even for large process models, this is an acceptable worst case execution time.

SIMILARITY CONSIDERATIONS. The concept at hand is one of the first approaches for process model comparison and retrieval to combine metrics, i.e., string-based and semantic metrics, in hybrid measures. We demonstrated that one particular combination outperforms all others. As our approach designs similarity measures in a modular way, similarity measures can be flexibly and freely substituted and combined, defined by the user.

VISUALISATION AND MANUAL RESULT CORRECTION. In addition to log files, we realised a graphical output. Visualising different and corresponding elements, clusters, and similarity values in colour shades is a valuable user support, extends the range of potential users, and increases intuitiveness (cf. Fig. 5.9). Additionally, we included the functionality to manually adjust results and start new computations. This aims at supporting manual conformance checks of processes in the context of governance in the best possible way.

Process analysts in companies, in the run up to a certification (of IT governance frameworks) or official process assessment, need to be trained concerning reference models. This type of process assessment often is a time and cost intensive task and requires corresponding expertise. Even internal pre-assessments become resource intensive tasks.

In this chapter, we presented an approach for matching process models that supports decision making when aiming at achieving regulations made by reference models. We demonstrated that our approach provides results of high quality at a reasonable computational complexity, and outperforms a text search engine, as well as state-of-the-art algorithms. We provide a prototype that realises the approaches with an interactive user interface, visualising the results. Concluding, we provide a well-performing decision support for process conformance observation in governance contexts.

As general conclusions, we provide an approach that excels both in BPM research as well as in research of governance approaches. While being useful in conformance observation of governance processes, it can also compete with BPM state-of-the-art business process retrieval algorithms.

In future work, the related cluster pairs concept could be further enhanced. So far, it does only marginally consider gateways in process model graphs, or further structural or behavioural aspects of business process model graphs. In reference process checking scenarios, e.g., where a *reference process* specifies an OR connector, valid implementations by the realised process model might be XOR, OR, or AND. Additionally, similarly to existing related work (e.g., [47]) the context of connectors should be investigated for similarity computation, mutually comparing levels of nodes, residing prior and after the connector, respectively.

Further, exploiting the interactive user interface, user reactions to computations can be made use of. Relating to the node label meaning, machine learning approaches can be employed to further improve the used node similarity metrics in terms of parametrisation.

Beyond this, the scope of business process graphs can be extended from the representation of activities to the inclusion of roles and responsibilities (e.g., annotated lanes), artefacts (e.g., data flow), and further process properties. A reasonable process notation to proceed from might be enhanced EPC (eEPC, [108]).

CONCLUSION AND OUTLOOK

CHANGING market conditions and increasing legal regulations impose high demands on the flexibility of IT structures in companies. In recent years, service-oriented systems have proven to be efficient IT systems which with to address these flexibility requirements in companies. In addition to these advantages, with the multitude of services, SOA systems introduce new challenges for companies. Having emerged from IT Governance, in the last decades, the discipline of SOA Governance addresses these issues. By best practice frameworks and reference processes, it provides methods to establish efficient control mechanisms for SOA systems, and means to ensure conformance with legal regulations and corporate standards. Certification procedures, assuring and attesting that an SOA system adheres to these requirements, however, are resource-intensive. While the design of frameworks and approaches to SOA Governance has been an active research field recently, the area of decision support techniques for conformance assessment of processes in governance contexts has not been intensively investigated.

The thesis at hand provides several contributions concerning the investigation of SOA Governance in general, as well as in the area of process conformance assessment in the context of SOA Governance in particular. The considerations made throughout this work have lead to a multitude of findings and insights. In this chapter, we collocate the main contributions and provide an outlook on future work.

6.1 Conclusions

As a foundation of our work, we performed a structural analysis of 22 major SOA Governance approaches. We discussed understandings of SOA Governance and developed a comprehensive definition. For the application of the insights, we developed a consolidated service life cycle that allows for the identified particularities of SOA Governance. Further, using the results, we developed a generic operational model for SOA Governance. It provides a generally applicable approach for the operation of SOA Governance in companies, while integrating all major components identified during the structural analysis. In particular, it integrates automated conformance checks of internal processes.

As a general result, the analysis shows that opinions and notions regarding the typical SOA Governance approach diverge. Approaches expose differing perspectives spanning from SOA Governance defined as “Service Life Cycle Management” to very comprehensive but unclearly structured approaches. However, all authors agree in unison on the necessity of SOA Governance for the successful operation of an SOA system – also called the “SOA Governance imperative”. In particular, the analysis revealed that there are 10 components of SOA Governance approaches that are most frequently considered by experts. These components have not yet been identified by related approaches and can be considered the crucial building blocks of SOA Governance.

As the four most important components of SOA Governance, we identified: *Governance Policies, Organisational Structures, Roles and Responsibilities, and Artefact Management*. As

regulation means in a governance approach, authors see few alternatives to policies. Generally, concerning organisational issues, an SOA Centre of Excellence as the central coordinating organisational institution is considered necessary in unison by the experts. Further, it is considered to be of central importance to influence employee behaviour in favour of the system. In this respect, the classical IT Governance goal “achieve desirable behaviour in the use of IT” proves valid for SOA Governance in particular. The support by artefact management solutions such as registries or service management systems is a further crucial component of SOA Governance.

The analysis covered approaches by three author groups. These different perspectives on SOA Governance all emphasise different aspects. Scientifically published approaches integrate eight out of ten components on average, representing rather holistic approaches. Software vendors clearly emphasise technology-oriented components (e.g., *SOA Procedure Model* and *Service Life Cycle*), and expose the most balanced coverage of components. This indicates a good relation to the practice of SOA Governance. Approaches from IT consulting industry show an unilateral focus on a small number of components, relying exactly on the four most important components. They expose the smallest coverage of components.

The building block *SOA Maturity Measurement* is among the least frequently integrated components. However, we observed that most approaches integrate aspects of maturity measurement without mentioning it explicitly. This reveals a lack of awareness regarding maturity measurement of SOA systems among the expert authors.

The analysis showed that most approaches are characterised by a “tunnel perspective”, limiting the focus on selected issues. Only few of them expose a holistic perspective. In contrast, the majority of authors agree that a *holistic* governance approach is crucial for SOA Governance. Clearly, this shows a lack of awareness of the scope of SOA Governance. In Chapter 3, we referred to a survey that revealed a big disaccord in perception of SOA Governance by companies that operate an SOA system (“requestors” of SOA Governance). In summary, our component analysis shows a big gap between understanding and action in this domain – this time from the opposite perspective, i.e., experts that propose SOA Governance approaches (“providers” of SOA Governance). In combination, a general lack – or at least a definite disaccord – of understandings of the term *SOA Governance* becomes apparent, and reveals a large area of general improvement needs, concerning in particular its understanding and the discussion of its definition, as well as its realisation approaches and components. Addressing these improvement needs, after the investigation of current definitions, we exploited the insights to develop and provide a new comprehensive definition for SOA Governance. Further, we consolidated the components and findings and provided a novel operational model that considers all aspects of SOA Governance that were identified throughout our analysis.

The second part of this thesis’ contributions targets a decision support approach for process conformance assessment. Reference processes are organised and managed in company-wide reference process repositories. The duty of corresponding organisational entities (in our operational model) is the design of conformable processes, as well as the assessment of the conformance of existing processes. As decision support technique for process conformance assessment in the context of SOA Governance, and as part of our generic operational model, we introduced the novel concept of *related cluster pairs* to realise (i) the identification of relevant reference processes for a given flow of activities, supporting the design of conform processes (*process model retrieval*), and (ii) the comparison

of two process models and identification of their difference, supporting process conformance assessment (*process model comparison*). The concept combines similarity computation of process model elements with a structural investigation. For the determination of correspondences between process model elements (activities), we utilise several string-based, semantic, and novel hybrid similarity measures. Using these correspondences based on similarity, the models are hierarchically decomposed into region pairs that contain similar elements. A merging process aggregates sets of neighbouring regions with similar content in both models and forms larger regions. The identified pairs of regions, so-called *related cluster pairs*, are both, semantically and structurally similar. Based on them, the difference of two process models can be indicated (*process model comparison*). Further, we developed a novel process similarity measure based on related cluster pairs that is used to compute similarity scores for pairs of process models, realising a *process model retrieval* approach.

The major benefits of our approach can be summarised along *performance*, *extensibility*, *scalability*, *similarity considerations*, and *visualisation and manual result correction*, as outlined in the following.

PERFORMANCE. According to numerous evaluation experiments we performed both for comparison and retrieval of process models, our concept yields very promising results. With the majority of similarity metrics, our process comparison approach using the concept of related cluster pair shows accuracy values of beyond 90%. This indicates a good assistance to manual work. The best performing metric is a new one proposed by us. It is a parametrised combination of a WordNet metric and a string-edit distance metric. In cross-validation evaluation experiments, it performed at 94.8% accuracy and exposed an F_1 score of 97.5%. As these results were achieved based on a test case that consists of process models for established standard procedures, they emphasise and prove the practical applicability of our approach. As part of the evaluation, we compared our results to a state-of-the-art approach for process model retrieval, as well as to text search engines. Our approach outperforms both of them.

EXTENSIBILITY. A further advantage of our approach is its *extensibility*. Exploiting its divide-and-conquer-characteristic, graph-based process models can be recursively decomposed into smaller process model fragments. Thus, process comparison problems can be reduced to a size that can be addressed by approaches that are more complex in terms of computational complexity. Resulting process fragments can be investigated, for example, using the graph-edit-distance approach. The decomposition ability makes the related cluster pairs concept flexibly combinable with further process model analysis techniques.

SCALABILITY. According to a 2009 survey, current process retrieval approaches need execution times that are not acceptable for usage within modern process model repositories [272]. The computation of our related cluster pairs concept exposes polynomial computational complexity. On modern processors, even for large process models, this is an acceptable worst case execution time.

SIMILARITY CONSIDERATIONS. We applied a number of standard node label metrics as well as proposed additional ones such as string-based similarity and semantic similarity based on word meanings using Wikipedia and WordNet. The concept

at hand is one of the first approaches in BPM to combine these metrics, i.e., to apply both string-based similarity and semantic similarity metrics in combination, resulting in more efficient hybrid measures. We assume that one reason for the performance improvements of our concept is the lack of combination of structural consideration and hybrid label similarity metrics in related work. The concept of related cluster pairs, in fact, combines these considerations.

VISUALISATION AND MANUAL RESULT CORRECTION. In addition to log files, we realised a graphical output. Visualising different and corresponding elements, clusters and similarity values in different colour shades is a valuable user support, extends the range of potential users, and increases intuitiveness. Additionally, we included the functionality to manually adjust results and start recomputations. This aims at supporting manual conformance assessment of processes in the context of governance in the best possible way.

Concluding, we performed a comprehensive investigation of SOA Governance approaches that revealed a lack of awareness of the scope of SOA Governance among expert authors. Based on these analysis results, amongst others, we provided a generic operational model for SOA Governance, that considers all major components that have been identified during the investigations. As conformance observation aspect of this model, we developed an analytical decision support approach for the process conformance assessment. We demonstrated by evaluation that it yields high quality results at a reasonable computational complexity, and outperforms text search engines, as well as related approaches. To the best of our knowledge, there is no comparable approach for process conformance assessment in the context of SOA Governance.

Overall, we provided contributions in both BPM research, as well as in research of governance approaches. While being useful in conformance observation of governance processes, the concept of related cluster pairs can also compete with BPM state-of-the-art algorithms for business process retrieval.

6.2 Outlook and Future Work

We identified three general fields of future work proceeding from the results of this thesis: (i) development support for emerging types of governance for, e.g., service marketplaces or cloud computing, (ii) the application and enhancement of BPM techniques for conformance enforcement in the context of governance, and, last but not least, (iii) the modelling and applications of models of specified governance frameworks using adequate data structures that represent modelled knowledge, for example, ontologies.

As observed in recent years, SOA Governance is evolving towards *Service Governance* (cf. Sect. 4.3). This discipline rather considers the single services, independent of their context, than services that are all collocated at one company's system. Further, mostly the addressee is a service marketplace host rather than a company's IT management. Governance for services extends SOA Governance by considerations like cross-organisational cooperation based on services, service management at service marketplaces, and, extending the functionality scope of services, aspects of *Cloud Computing* (see below).

Service Governance is the foundation for governance for the Internet of Services (IoS) or service marketplace governance. Compared to an SOA system, a marketplace bears much more complexity and at the same time has the potential to yield more benefits.

While, basically, the stakeholder roles in an SOA sum up to three, a variety of additional roles are needed in the IoS (e.g., service innovator, service producer, service aggregator, marketplace host, cf. Sect. 2.1.2). As soon as service marketplaces are shared across country borders, further new challenges emerge. When *ad hoc* service usage is to be realised, the investigation of legal consequences in case of SLA or contract breaches, strongly depending on local laws and legislation, is a still enduring challenge and research problem. Further, the efficient management of semantic service description (marketplace-wide unification), service composition, delegation of service monitoring (centralised vs. decentralised approach), consistent service pricing, marketplace-wide service development guidelines, and many more aspects have to be considered, as the recent German government-funded research project *Theseus/TEXO* revealed and addressed. Most important, additional legal regulations apply in this scenario and different challenges arise, for example, automated service provision across borders of legislative authorities. All these factors have a strong impact on the design and implementation of approaches for Service Governance.

A further field of research in terms of governance is *Cloud Computing*. Most recently, it gained much momentum in both research and the software industry as a paradigm that utilises virtualisation technologies in order to maximise infrastructure flexibility in and between companies. The heterogeneity and complexity of cloud-based systems impose high demands on efficient steering and management. Additionally, cloud computing is based on heterogeneous, often not consistently organised environments. Many challenges of cloud governance can be coped with using means from IT Governance, such as compliance, legal issues and standardisation. For cloud characteristics such as the *ad hoc* provision of complex services, for example, infrastructure as a service (IaaS), the cloud-wide management and unification of service monitoring delegation, access management, development standards, and new legal regulations are new challenges, needing experienced governance structures that can be adopted.

The 10 building blocks revealed by our work represent a basis for the future development of governance approaches for service-oriented systems in general. The consideration of the special role of service life cycle management or the involvement of the business departments, for example, are especially important in service contexts, compared to the IT Governance perception for monolithic IT systems. The results of this thesis can be a good starting point for the development of consistent governance approaches for service marketplaces as well as Cloud Computing.

In the second field of future work, techniques from BPM have been improved and applied for conformance enforcement in the context of governance. Especially, techniques from the fields of business process retrieval and similarity as well as process compliance can be used to tackle typical governance problems of assessing process conformance.

In this thesis, we provided a decision support approach from the first field. As future work, our approach, the *related cluster pairs* concept, can be further enhanced. So far, it marginally considers gateways in process model graphs, which can be extended to cover further behavioural aspects of business process model graphs. For example, in reference process checking scenarios, where a *reference* process specifies an OR connector, valid implementations by the *realised* process model might be either XOR, OR, or AND connectors. Additionally, the context of connectors can be integrated into similarity considerations, mutually comparing levels of nodes, residing before and after the connector, respectively. Beyond this, the scope of business process graphs and corresponding com-

parison techniques can be extended from the representation of activities to the inclusion of roles and responsibilities (e.g., annotated lanes), artefacts (e.g., data flow), and further governance-related process properties.

A promising second field called *Process Compliance* is experiencing increasing regard in the last years (cf. Sect. 2.3.2). Generally, it assures that process execution adheres to regulations using formal modelling techniques. Approaches for Process Compliance use logic-based modelling of single regulations for validation rather than considering reference processes. Applied on processes in the context of governance of IT systems, it proceeds from formally modelled process policies. As a requirement, policies and all involved artefacts such as documents and roles must also be formally modelled. Once this elaborate basis for the formal description of policies is provided, techniques from Process Compliance are a promising approach to address process conformance assessment in governance contexts.

A third field of future work we identified is the structured modelling of governance frameworks. The modelling of governance structures as meta models or ontologies bears a large potential regarding the automated support of the operational aspect of governance. Our work concerning the design of governance meta models [96] and ontologies for standard IT Governance frameworks [168] addresses an increasingly active area of research. Further work in this area shows that the modelling of structured governance data and its exploitation by various knowledge-inferencing techniques is currently considered a promising area of research [64, 90, 94].

REFERENCES

- [1] Mohamad Afshar. SOA Governance: Framework and Best Practices. Oracle Whitepaper, May 2007. URL <http://www.oracle.com/us/technologies/soa/oracle-soa-governance-best-practice-066427.pdf>. last accessed 2011-08-03.
- [2] Dimitris Alevras. Assignment and Matching. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 106–108. Springer-Verlag New York, Inc., 2009.
- [3] Paul Allen. SOA Governance: Challenge or Opportunity? *CBDJ Journal*, 4:20–31, 2008.
- [4] Keith Andrews, Martin Wohlfahrt, and Gerhard Wurzinger. Visual Graph Comparison. In *Proceedings of the 13th International Conference on Information Visualisation (IV 2009)*, pages 62–67, Washington, DC, USA, July 2009.
- [5] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A View of Cloud Computing. *Communications of the ACM*, 53(4):50–58, April 2010.
- [6] Ali Arsanjani and Kerrie Holley. Increase flexibility with the Service Integration Maturity Model (SIMM) – Maturity, adoption, and transformation to SOA, 2005. URL <http://www.ibm.com/developerworks/webservices/library/ws-soa-simm/>. last accessed 2011-08-03.
- [7] Ali Arsanjani and Kerrie Holley. The Service Integration Maturity Model: Achieving Flexibility in the Transformation to SOA. In *Proceedings of the International Conference on Services Computing (SCC 2006)*, page 515, Chicago, IL, USA, September 2006.
- [8] David Avison, Jill Jones, Philip Powell, and David Wilson. Using and Validating the Strategic Alignment Model. *Journal of Strategic Information Systems*, 13(3):223–246, 2004.
- [9] Jens Bartenschlager and Matthias Goeken. Designing Artifacts of IT Strategy for Achieving Business/IT Alignment. In *Proceedings of the Fifth Americas Conference on Information Systems (AMCIS 2009)*, San Francisco, CA, USA, August 2009.
- [10] BEA Systems, Inc. Service Lifecycle Governance. BEA Whitepaper, 2006. URL http://wp.bitpipe.com/resource/org_928437302_173/Service_Lifecycle_9941_edp.pdf. last accessed 2011-08-03.
- [11] Jörg Becker and Reinhard Schütte. *Handelsinformationssysteme*. Verlag Moderne Industrie, Landsberg, Germany, second edition, 2004.
- [12] Jörg Becker, Michael Rosemann, and Martin Kugeler. *Process Management. A Guide for the Design of Business Processes*. Springer-Verlag New York, Inc., 2003.

- [13] Rainer Berbner. *Dienstgüteunterstützung für Service-orientierte Workflows*. Books on Demand GmbH, Norderstedt, PhD thesis, Technische Universität Darmstadt, Germany, January 2008.
- [14] Rainer Berbner, Oliver Heckmann, and Ralf Steinmetz. An Architecture for a QoS driven composition of Web Service based Workflows. In *Networking and Electronic Commerce Research Conference (NAEC 2005)*, Riva Del Garda, Italy, October 2005.
- [15] Jan Bernhardt and Detlef Seese. A Conceptual Framework for the Governance of Service-Oriented Architectures. In *Service-Oriented Computing – ICSOC 2008 Workshops*, volume 5472 of *Lecture Notes in Computer Science*, pages 327–338. Springer-Verlag, Berlin Heidelberg, 2009.
- [16] Martin Bichler and Kwei-Jay Lin. Service-Oriented Computing. *IEEE Computer*, 39(3):99–101, March 2006.
- [17] Norbert Bieberstein, Sanjay Bose, Marc Fiammante, Keith Jones, and Rawn Shah. *Service-Oriented Architecture (SOA) Compass - Business Value, Planning, and Enterprise Roadmap*. IBM Press, November 2005.
- [18] Norbert Bieberstein, Sanjay Bose, Lance Walker, and Angela Lynch. Impact of Service-oriented Architecture on Enterprise Systems, Organizational Structures, and Individuals. *IBM Systems Journal*, 44(4):691–708, 2005.
- [19] Marc Billeb, Achim Schäfer, Mourad Abbou, Michael Niemann, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. Einfluss regulatorischer Anforderungen im Internet of Services. *IT-Governance, Zeitschrift des ISACA Germany Chapter e.V.*, 6:8–13, October 2009.
- [20] Jason Bloomberg. SOA Governance: IT Governance in the Context of Service Orientation. ZapThink, LLC Whitepaper, 2004. URL http://www.logiclibrary.com/resources/zapthink_wp.php. last accessed 2011-08-03.
- [21] Barry W. Boehm. A Spiral Model of Software Development and Enhancement. *IEEE Computer*, 21(5):61–72, 1988.
- [22] François Bourgeois and Jean-Claude Lassalle. An Extension of the Munkres Algorithm for the Assignment Problem to Rectangular Matrices. *Communications of the ACM*, 14(12):802–804, 1971.
- [23] François Bourgeois and Jean-Claude Lassalle. Algorithm 415: Algorithm for the Assignment Problem (Rectangular Matrices). *Communications of the ACM*, 14(12):805–806, 1971.
- [24] Koen Brand and Harry Boonen. *IT Governance based on COBIT 4.1*. Van Haren Publishing, Zaltbommel, The Netherlands, third edition, 2007.
- [25] Ben Brauer and Sean Kline. SOA Governance: A Key Ingredient of the Adaptive Enterprise. HP and Systinet Whitepaper, February 2005. URL <http://www.zdnet.de/whitepaper/60141901/soa-governance-a-key-ingredient-of-the-adaptive-enterprise.htm>. last accessed 2011-08-03.

- [26] William Brown, Garry Moore, and William Tegan. SOA governance – IBM’s approach. *IBM Whitepaper*, August 2006. URL <http://www-07.ibm.com/sg/do/downloads/soa/archive/week1-2/RAW10953-USEN-00.pdf>. last accessed 2011-08-03.
- [27] William Brown, Robert G. Laird, Clive Gee, and Tilak Mitra. *SOA Governance: Achieving and Sustaining Business and IT Agility*. IBM Press, December 2008.
- [28] Alexander Budanitsky and Graeme Hirst. Evaluating WordNet-based Measures of Lexical Semantic Relatedness. *Computational Linguistics*, 32:13–47, March 2006.
- [29] Horst Bunke, Peter J. Dickinson, Andreas Humm, Christophe Irniger, and Miro Kraetzl. Graph Sequence Visualisation and its Application to Computer Network Monitoring and Abnormal Event Detection. In Abraham Kandel, Horst Bunke, and Mark Last, editors, *Applied Graph Theory in Computer Vision and Pattern Recognition*, volume 52 of *Studies in Computational Intelligence*, pages 227–245. Springer-Verlag, Berlin Heidelberg, 2007.
- [30] Rainer Burkard, Mauro Dell’Amico, and Silvano Martello. *Assignment Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.
- [31] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. *Future Generation Computer Systems*, 25(6):599–616, June 2009.
- [32] Doreen Böhnstedt. *Semantisches Tagging zur Verwaltung von webbasierten Lernressourcen*. PhD thesis, Technische Universität Darmstadt, Germany, June 2011.
- [33] Kishore Channabasavaiah, Edward Jr. Tuggle, and Kerrie Holley. Migrating to a service-oriented architecture (Part 1). IBM developerWorks, December 2003. URL <http://www.ibm.com/developerworks/library/ws-migratesoa/>. last accessed 2011-08-08.
- [34] Kishore Channabasavaiah, Edward Jr. Tuggle, and Kerrie Holley. Migrating to a service-oriented architecture (Part 2). IBM developerWorks, December 2003. URL <http://www.ibm.com/developerworks/library/ws-migratesoa2/>. last accessed 2011-08-08.
- [35] Sam Chapman and Fabio Ciravegna. SimMetrics - Similarity Metric Library. University of Sheffield, UK. URL <http://www.aktors.org/technologies/simmetrics/index.html>. last accessed 2011-08-12.
- [36] Workflow Management Coalition. The Workflow Reference Model, January 1995. URL <http://www.wfmc.org/standards/docs/tc003v11.pdf>. last accessed 2011-08-08.
- [37] Workflow Management Coalition. Workflow Standard – Terminology & Glossary, February 1999. URL <http://www.wfmc.org/Download-document/WFMC-TC-1011-Ver-3-Terminology-and-Glossary-English.html>. last accessed 2011-08-08.
- [38] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of the International*

- Joint Conference on Artificial Intelligence – Workshop on Information Integration*, pages 73–78, Acapulco, Mexico, August 2003.
- [39] Exagon Consulting. ITIL entwickelt sich zum Renner, June 2006. URL [http://www.exagon.de/Newsarchiv-Details.755.0.html?&tx_ttnews\[tt_news\]=27&cHash=692ece37cc32d93fd2f6de52a1434949](http://www.exagon.de/Newsarchiv-Details.755.0.html?&tx_ttnews[tt_news]=27&cHash=692ece37cc32d93fd2f6de52a1434949). last accessed 2011-08-05.
- [40] Thomas H. Davenport. The New Industrial Engineering: Information Technology and Business Process Redesign. *MIT Sloan Management Review*, 31(4):11–27, 1990.
- [41] Thomas H. Davenport. *Process Innovation: Reengineering Work through Information Technology*. Harvard Business School Press, Boston, MA, USA, 1993.
- [42] Ana Karla Alves de Medeiros, Wil van der Aalst, and A. Weijters. Quantifying Process Equivalence Based on Observed Behavior. *Data Knowledge Engineering*, 64(1):55–74, 2008.
- [43] Patricia Derler and Rainer Weinreich. Models and Tools for SOA Governance. In *Proceedings of the Second International Conference on Trends in Enterprise Application Architecture*, volume 4473 of *Lecture Notes in Computer Science*, pages 112–126. Springer-Verlag, Berlin Heidelberg, 2007.
- [44] Remco Dijkman. A Classification of Differences between Similar Business Processes. In *Proceedings of the 11th International Enterprise Distributed Object Computing Conference (EDOC 2007)*, pages 37–50, Annapolis, MD, USA, October 2007.
- [45] Remco Dijkman. Diagnosing Differences between Business Process Models. In *Proceedings of the Sixth International Conference on Business Process Management (BPM 2008)*, volume 5240 of *Lecture Notes in Computer Science*, pages 261–277. Springer-Verlag, Berlin Heidelberg, 2008.
- [46] Remco Dijkman, Marlon Dumas, and Luciano García-Bañuelos. Graph Matching Algorithms for Business Process Model Similarity Search. In *Proceedings of the Seventh International Conference on Business Process Management (BPM 2009)*, volume 5701 of *Lecture Notes in Computer Science*, pages 48–63. Springer-Verlag, Berlin Heidelberg, 2009.
- [47] Remco Dijkman, Marlon Dumas, Boudewijn van Dongen, Reina Käärrik, and Jan Mendling. Similarity of Business Process Models: Metrics and Evaluation. Technical report, BETA Research School, Eindhoven, The Netherlands, February 2009.
- [48] Remco Dijkman, Marlon Dumas, Boudewijn van Dongen, Reina Käärrik, and Jan Mendling. Similarity of Business Process Models: Metrics and Evaluation. *Information Systems*, 36(2):498–516, 2011.
- [49] Wolfgang Domschke and Andreas Drexl. *Einführung in Operations Research*. Springer-Verlag, Berlin Heidelberg, sixth edition, 2005.
- [50] Marlon Dumas, Wil van der Aalst, and Arthur ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley & Sons, Inc., New York, NY, USA, 2005.

- [51] Julian Eckert. *Cross-organizational Service-based Workflows – Solution Strategies for Quality of Service Optimization*. PhD thesis, Technische Universität Darmstadt, Germany, October 2009.
- [52] Julian Eckert, Stefan Schulte, Michael Niemann, Nicolas Repp, and Ralf Steinmetz. Worst-Case Workflow Performance Optimization. In *Proceedings of the Third International Conference on Internet and Web Applications and Services (ICIW 2008)*, pages 632–637, Athens, Greece, June 2008.
- [53] Julian Eckert, Nicolas Repp, and Wolfgang Martin. *SOA Check 2009: Status Quo und Trends im Vergleich zum SOA Check 2008 und 2007*. IT-Verlag, Sauerlach, Germany, April 2009.
- [54] Marc Ehrig, Agnes Koschmider, and Andreas Oberweis. Measuring Similarity between Semantic Business Process Models. In *Proceedings of the Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM 2007)*, pages 71–80, Ballarat, Victoria, Australia, January 2007.
- [55] Thomas Erl. *Service-Oriented Architecture - Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.
- [56] Thomas Erl. *SOA Principles of Service Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2007.
- [57] Martin Fabini. Governance für komplexe SOA-Unternehmungen: Eine Vision für das Schweizer Gesundheitswesen. In Gernot Starke and Stefan Tilkov, editors, *SOA-Expertenwissen*, chapter 18, pages 309–323. dpunkt.verlag Heidelberg, 2007.
- [58] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. MIT Press, Cambridge, MA, USA, May 1998.
- [59] Stefan Fischer and Paul Müller. Experimentalforschung für das Future Internet – deutsche und europäische Initiativen. *Informatik-Spektrum*, 33(2):122–130, 2010.
- [60] Martin Fröhlich and Kurt Glasner. *IT Governance. Leitfaden für eine praxisgerechte Implementierung*. Gabler Verlag, Wiesbaden, Germany, 2007.
- [61] Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A Survey of Graph Edit Distance. *Pattern Analysis & Applications*, 13(1):113–129, 2010.
- [62] Hartmann J. Genrich. Predicate/Transition Nets. In *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part I, Proceedings of an Advanced Course*, volume 254 of *Lecture Notes in Computer Science*, pages 207–247. Springer-Verlag, Berlin Heidelberg, September 1986.
- [63] Ronald E. Giachetti. *Design of Enterprise Systems – Theory, Architecture, and Methods*. CRC Press, Boca Raton, FL, USA, 2010.
- [64] Matthias Goeken and Stefanie Alter. Towards Conceptual Metamodelling of IT Governance Frameworks. Approach - Use – Benefits. In *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences.*, pages 1–10, Waikoloa, Big Island, Hawaii, January 2009.

- [65] Karl D. Gottschalk, Stephen Graham, Heather Kreger, and James Snell. Introduction to Web Services Architecture. *IBM Systems Journal*, 41(2):170–177, 2002.
- [66] Guido Governatori and Antonino Rotolo. A Conceptually Rich Model of Business Process Compliance. In *Proceedings of the Seventh Asia-Pacific Conference on Conceptual Modelling - Volume 110*, pages 3–12, Brisbane, Australia, 2010.
- [67] Volker Gruhn and Ralf Laue. Einfache EPK-Semantik durch praxistaugliche Stilregeln. In *4. Workshop der Gesellschaft für Informatik e.V. (GI) und Treffen ihres Arbeitskreises Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (WI-EPK)*, pages 176–189, Hamburg, Germany, Dezember 2005.
- [68] Qing Gu and Patricia Lago. A Stakeholder-driven Service Life Cycle Model for SOA. In *Proceedings of the Second International Workshop on Service Oriented Software Engineering (IW-SOSWE 2007)*, pages 1–7, Dubrovnik, Croatia, September 2007.
- [69] Martin Hafner, Joachim Schelp, and Robert Winter. Architekturmanagement als Basis effizienter und effektiver Produktion von IT-Services. *HMD - Praxis der Wirtschaftsinformatik*, 237:54–66, 2004.
- [70] Claus Hagen. Integrationsarchitektur der Credit Suisse. In Stephan Aier and Marten Schönherr, editors, *Enterprise Application Integration – Flexibilisierung komplexer Unternehmensarchitekturen*, pages 61–84. GITO Verlag, Berlin, Germany, second edition, April 2007.
- [71] Michael Hammer and James Champy. *Reengineering the Corporation: A Manifesto for Business Revolution*. HarperBusiness, New York, NY, USA, 2003.
- [72] Brian Hayes. Cloud Computing. *Communications of the ACM*, 51(7):9–11, July 2008.
- [73] Wolfgang Herrmann. Was vom SOA-Hype übrig bleibt. *COMPUTERWOCHE*, 40:14, 2008. URL <http://www.computerwoche.de/index.cfm?pid=411&pk=1224687>. last accessed 2011-08-11.
- [74] Hewlett-Packard. The Eight most important Best Practices in SOA Governance, 2008. URL <http://h20195.www2.hp.com/v2/GetPDF.aspx/4AA1-5985EEW.pdf>. last accessed 2011-08-08.
- [75] Hewlett-Packard and Systinet. SOA Governance – Balancing Flexibility and Control within an SOA, 2007. URL <http://h20195.www2.hp.com/v2/GetPDF.aspx/4AA1-5985EEW.pdf>. last accessed 2011-08-03.
- [76] Jan Hidders, Marlon Dumas, Wil van der Aalst, Arthur ter Hofstede, and Jan Verelst. When are two Workflows the Same? In *Proceedings of the 2005 Australasian Symposium on Theory of Computing - Volume 41*, pages 3–11, Darlinghurst, Australia, 2005.
- [77] Markus Hillenbrand. *Eine serviceorientierte Middleware für die Bereitstellung von Diensten im Internet unter Berücksichtigung von Transparenz, Offenheit und Zuverlässigkeit*. Informatik. Dr. Hut Verlag, München, PhD thesis, Technische Universität Kaiserslautern, Germany, 2008.

- [78] Markus Hillenbrand and Paul Müller. Web Services and Peer-to-Peer. In Ralf Steinmetz and Klaus Wehrle, editors, *Peer-to-Peer Systems and Applications*, volume 3485 of *Lecture Notes in Computer Science*, pages 207–224. Springer-Verlag, Berlin Heidelberg, September 2005.
- [79] Markus Hillenbrand, Joachim Götze, and Paul Müller. Venice – A Lightweight Service Grid. In *Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, Cavtat/Dubrovnik, Croatia, August 2006.
- [80] Markus Hillenbrand, Joachim Götze, and Paul Müller. Web Services Directory based on Peer-to-peer Technology. In *Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, Cavtat/Dubrovnik, Croatia, August 2006.
- [81] Kerrie Holley, Jim Palistrant, and Steve Graham. Effective SOA Governance. IBM Whitepaper, March 2006. URL http://www-07.ibm.com/my/events/soa/download/soagov_mgmt.pdf. last accessed 2011-08-03.
- [82] Bernhard Holtschke, Hauke Heier, and Thomas Hummel. *Quo Vadis CIO*. Xpert.press. Springer-Verlag, Berlin Heidelberg, November 2009.
- [83] Michael N. Huhns and Munindar P. Singh. Service-Oriented Computing: Key Concepts and Principles. *IEEE Internet Computing*, 9(1):75–81, 2005.
- [84] International Organization for Standardization (ISO). ISO 17799. URL <http://www.17799central.com/>. last accessed 2011-08-08.
- [85] International Organization for Standardization (ISO). ISO/IEC 20000-1:2011, 2011. URL http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=51986. last accessed 2011-08-08.
- [86] IT Governance Institute (ITGI). CobiT 4.1: Control Objectives for Information and Related Technology, 2007. URL <http://www.itgi.org/cobit>. last accessed 2011-08-11.
- [87] IT Governance Institute (ITGI). CobiT 5.0: Control Objectives for Information and Related Technology – Exposure Draft, 2011. URL <http://www.isaca.org/Knowledge-Center/Research/ResearchDeliverables/Pages/COBIT-5-Exposure-Draft.aspx>. last accessed 2011-08-02.
- [88] Stefan Jablonski and Christoph Bussler. *Workflow Management: Modelling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, September 1996.
- [89] Ivar Jacobson and Stefan Bylund. *The Road to the Unified Software Development Process*. Cambridge University Press, New York, NY, USA, 2000.
- [90] Christian Janiesch and Axel Korthaus. Validation of a Generic Service Governance Meta Model based on the Comparison of Major Governance Frameworks. In *Proceedings of the 21th Australasian Conference on Information Systems (ACIS)*, pages 1–11, Brisbane, Australia, 2010.

- [91] Christian Janiesch and Michael Niemann. Governance for Platforms in the Internet of Services: Building Blocks for a Governance Framework supporting USDL. In Alistair Barros and Daniel Oberle, editors, *Handbook of Service Description — USDL and its Methods*. Springer-Verlag, New York, USA, 2011.
- [92] Christian Janiesch, Michael Niemann, and Nicolas Repp. Governance in the Internet of Services: Governing Service Delivery of Service Brokers. In *Proceedings of the Pre-ICIS SIGSVC (Special Interest Group Services) Workshop on Services*, Paris, France, December 2008.
- [93] Christian Janiesch, Rainer Ruggaber, and York Sure. Eine Infrastruktur für das Internet der Dienste. *HMD - Praxis der Wirtschaftsinformatik*, 45(261):71–79, June 2008.
- [94] Christian Janiesch, Axel Korthaus, and Michael Rosemann. Conceptualisation and Facilitation of SOA Governance. In *Proceedings of the 20th Australasian Conference on Information Systems (ACIS)*, pages 154–163, Melbourne, Australia, 2009.
- [95] Christian Janiesch, Michael Niemann, and Nicolas Repp. Towards a Service Governance Framework for the Internet of Services. In *Proceedings of the 17th European Conference on Information Systems (ECIS 2009)*, Verona, Italy, June 2009.
- [96] Christian Janiesch, Michael Niemann, and Ralf Steinmetz, editors. The TEXO Governance Framework. SAP Whitepaper, March 2011. URL http://www.internet-of-services.com/fileadmin/IOS/user_upload/pdf/TEXO_GOVERNANCE_FRAMEWORK_WHITEPAPER_1_1.pdf. last accessed 2011-08-05.
- [97] Wolfgang Johannsen and Matthias Goeken. *Referenzmodelle für IT-Governance - Strategische Effektivität und Effizienz mit COBIT, ITIL & Co.* dpunkt.verlag Heidelberg, second edition, 2011.
- [98] Richard Johnson. *Efficient Program Analysis Using Dependence Flow Graphs*. PhD thesis, Cornell University, Ithaca, NY, USA, 1995.
- [99] Richard Johnson, David Pearson, and Keshav Pingali. Finding Regions Fast: Single Entry Single Exit and Control Regions in Linear Time. Technical Report, Cornell University, Ithaca, NY, USA, 1993.
- [100] Richard Johnson, David Pearson, and Keshav Pingali. The Program Structure Tree: Computing Control Regions in Linear Time. In *Proceedings of the ACM SIGPLAN 1994 Conference on Programming Language Design and Implementation*, pages 171–185, Orlando, FL, USA, June 1994.
- [101] Nicolai M. Josuttis. *SOA in Practice: The Art of Distributed System Design*. O'Reilly Media, Inc., Sebastopol, CA, USA, August 2007.
- [102] Aneta Kabzeva, Michael Niemann, Paul Müller, and Ralf Steinmetz. Applying TOGAF to Define and Govern a Service-oriented Architecture in a Large-scale Research Project. In *Proceedings of the 16th Americas Conference on Information Systems (AMCIS 2010)*, Lima, Peru, August 2010.

- [103] Ulrich Kalex. Von der Geschäftsarchitektur zur SOA-Governance. In Gernot Starke and Stefan Tilkov, editors, *SOA-Expertenwissen*, chapter 19, pages 325–340. dpunkt.verlag Heidelberg, 2007.
- [104] Dimitris Karagiannis. A Business Process-Based Modelling Extension for Regulatory Compliance. In *Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI 2008)*, München, Germany, February 2008.
- [105] Holger Karl and Andreas Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, Chichester, West Sussex, UK, illustrated, reprinted edition, 2007.
- [106] Sebastian Kaune. *Performance and Availability in Peer-to-Peer Content Distribution Systems: A Case for a Multilateral Incentive Approach*. PhD thesis, Technische Universität Darmstadt, Germany, February 2011.
- [107] Gerhard Keller and Thomas Teufel. *SAP R/3 Process Oriented Implementation*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998.
- [108] Gerhard Keller, Markus Nüttgens, and August-Wilhelm Scheer. Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK). Technical Report 89, Universität des Saarlandes, Saarbrücken, Germany, January 1992.
- [109] Wolfgang Keller. SOA-Governance: SOA langfristig durchsetzen und managen. In Gernot Starke and Stefan Tilkov, editors, *SOA-Expertenwissen*, chapter 17, pages 289–308. dpunkt.verlag Heidelberg, 2007.
- [110] Marwane El Kharbili and Sebastian Stein. Policy-Based Semantic Compliance Checking for Business Process Management. In *Proceedings of the Workshops collocated with the MobIS2008 Conference: including EPK2008, KobAS2008 and Mod-KollGP2008*, volume 420 of *CEUR Workshop Proceedings*, pages 178–192, Saarbrücken, Germany, November 2008.
- [111] Ekkart Kindler. On the Semantics of EPCs: Resolving the Vicious Circle. *Data and Knowledge Engineering - Special issue: Business Process Management*, 56(1):23–40, 2006.
- [112] Mark Klein and Abraham Bernstein. Toward High-Precision Service Retrieval. *IEEE Internet Computing*, 8(1):30–36, January 2004.
- [113] Gerhard F. Knolmayer and Gabriela Loosli. IT Governance. In Robert J. Zaugg, editor, *Handbuch Kompetenzmanagement. Durch Kompetenz nachhaltig Werte schaffen*, pages 449 – 457. Haupt Verlag, Bern, Switzerland, 2006.
- [114] James Kobiellus. SOA Governance: Preventing Rogue Services, June 2006. URL <http://www.networkworld.com/supp/2006/ndc3/062606-ndc-soa-governance.html>. last accessed 2011-08-05.
- [115] Michael Koch, Markus Hillenbrand, and Paul Müller. A Monitoring Framework for the Venice Service Grid. In *Proceedings of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, Lille, France, September 2010.

- [116] Oliver Kohnke, Torsten Scheffler, and Christian Hock. SOA-Governance - Ein Ansatz zum Management Serviceorientierter Architekturen. *Wirtschaftsinformatik*, 5:408–412, 2008.
- [117] Agnes Koschmider. *Ähnlichkeitsbasierte Modellierungsunterstützung für Geschäftsprozesse*. PhD thesis, Universität Karlsruhe, Germany, November 2007.
- [118] Agnes Koschmider and Andreas Oberweis. Ontology based Business Process Description. In *Proceedings of the Conference on Advanced Information Systems Engineering (CAiSE 2005) - Workshops*, number 2, pages 321–333, Porto, Portugal, June 2005.
- [119] Dirk Krafczig, Karl Banke, and Dirk Slama. *Enterprise SOA: Service-Oriented Architecture Best Practices (The Coad Series)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, November 2004.
- [120] Jochen M. Küster, Christian Gerth, Alexander Förster, and Gregor Engels. Detecting and Resolving Process Model Differences in the Absence of a Change Log. In *Proceedings of the Sixth International Conference on Business Process Management (BPM 2008)*, pages 244–260, Milan, Italy, September 2008.
- [121] Ulrich Lampe. Optimizing the Distribution of Software Services in Infrastructure Clouds. In *Proceedings of the Seventh World Congress on Services (SERVICES 2011)*, Washington DC, USA, July 2011. (accepted for publication).
- [122] Peter Langner, Christoph Schneider, and Joachim Wehler. Petri Net Based Certification of Event-Driven Process Chains. In *Proceedings of the 19th International Conference on Application and Theory of Petri Nets (ICATPN 1998)*, pages 286–305, Lisbon, Portugal, June 1998.
- [123] Vladimir I. Levenshtein. Binary code capable of correcting deletions, insertions and reversals. *Cybernetics and Control Theory (Soviet Physics Doklady)*, 10 (8):707–710, 1966.
- [124] Frank Leymann and Dieter Roller. *Production Workflow: Concepts and Techniques*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.
- [125] Frank Leymann, Dieter Roller, and Marc-Thomas Schmidt. Web services and business process management. *IBM Systems Journal*, 41(2):198–211, 2002.
- [126] Chen Li, Manfred Reichert, and Andreas Wombacher. Discovering Reference Process Models by Mining Process Variants. In *Proceedings of the International Conference on Web Services (ICWS 2008)*, pages 45–53, Beijing, China, September 2008.
- [127] Chen Li, Manfred Reichert, and Andreas Wombacher. Discovering Reference Models by Mining Process Variants Using a Heuristic Approach. In *Proceedings of the Seventh International Conference on Business Process Management (BPM 2009)*, volume 5701 of *Lecture Notes in Computer Science*, pages 344 – 362. Springer-Verlag, Berlin Heidelberg, 2009.
- [128] Dekang Lin. Automatic Retrieval and Clustering of Similar Words. In *Proceedings of the 17th International Conference on Computational Linguistics*, pages 768–774, Montreal, Quebec, Canada, August 1998.

- [129] Bing Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, second edition, July 2011.
- [130] Ruopeng Lu and Shazia Wasim Sadiq. On the Discovery of Preferred Work Practice Through Business Process Variants. In *Proceedings of the 26th International Conference on Conceptual Modeling*, volume 4801 of *Lecture Notes in Computer Science*, pages 165–180. Springer-Verlag, Berlin Heidelberg, November 2007.
- [131] Ruopeng Lu, Shazia Sadiq, and Guido Governatori. Compliance Aware Business Process Design. In *Proceedings of the International Conference on Business Process Management (BPM 2007)*, pages 120–131, Brisbane, Australia, September 2007.
- [132] Jochen Ludewig and Horst Lichter. *Software Engineering*. dpunkt.verlag, Heidelberg, first edition, 2007.
- [133] Matthew C. MacKenzie, Ken Laskey, Francis McCabe, Peter F. Brown, and Rebekah Metz. Reference Model for Service Oriented Architecture 1.0. OASIS Standard, August 2006. URL <http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf>. last accessed 2011-08-12.
- [134] Therani Madhusudan, J. Leon Zhao, and Byron Marshall. A Case-based Reasoning Framework for Workflow Model Management. *Data Knowledge Engineering*, 50:87–115, July 2004.
- [135] Thomas W. Malone, Kevin Crowston, and George A. Herman, editors. *Organizing Business Knowledge – The MIT Process Handbook*. MIT Press, Cambridge, MA, USA, 2003.
- [136] Anne Thomas Manes. The Elephant Has Left The Building, July 2005. URL <http://www.informationweek.com/news/software/bi/164301126>. last accessed 2011-08-11.
- [137] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.
- [138] Eric A. Marks and Michael Bell. *Service-Oriented Architecture: A Planning and Implementation Guide for Business and Technology*. John Wiley & Sons, New York, NY, USA, 2006.
- [139] Miko Matsumura. The Definitive Guide to SOA Governance and Lifecycle Management. WebMethods Whitepaper, March 2007. URL <http://www.scribd.com/doc/7056416/Guide-to-SOA-Governance>. last accessed 2011-08-05.
- [140] Gary McBride. The Role of SOA Quality Management in SOA Service Lifecycle Management. *IBM developerWorks*, March 2007. URL <ftp://public.dhe.ibm.com/common/ssi/ecm/en/raw14091usen/RAW14091USEN.PDF>. last accessed 2011-08-05.
- [141] Michael Meehan. SOA Adoption marked by Broad Failure and Wild Success, 2008. URL http://searchsoa.techtarget.com/news/article/0,289142,sid26_gci1319609,00.html. last accessed 2011-08-05.
- [142] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching.

- In *Proceedings of the 18th International Conference on Data Engineering (ICDE 2002)*, pages 117–128, San Jose, CA, USA, March 2002.
- [143] Ingo Melzer, Sebastian Eberhard, Alexander Hilliger von Thile, Marcus Flehmig, Barbara Rudolph, Wolfgang Dostal, Peter Tröger, Patrick Sauter, Boris Stumm, Matthias Lipp, Jochen Vajda, and Mario Jeckle. *Service-orientierte Architekturen mit Web Services*. Spektrum Akademischer Verlag, Elsevier GmbH, München, fourth edition, 2010.
- [144] Jan Mendling, Boudewijn van Dongen, and Wil van der Aalst. On the Degree of Behavioral Similarity between Business Process Models. In *6. Workshop der Gesellschaft für Informatik e.V. (GI) und Treffen ihres Arbeitskreises Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (WI-EPK)*, volume 303 of *CEUR Workshop Proceedings*, pages 39–58, 2007.
- [145] Jan Mendling, Jan Recker, and Hajo A. Reijers. On the usage of labels and icons in business process modeling. *International Journal of System Modeling and Design*, 1 (2):40–58, 2010.
- [146] André Miede. *Cross-organizational Service Security – Attack Modeling and Evaluation of Selected Countermeasures*. Dr. Hut Verlag, München, PhD thesis, Technische Universität Darmstadt, Germany, December 2010.
- [147] André Miede, Michael Niemann, Stefan Schulte, Julian Eckert, Aneta Kabzeva, Nicolas Repp, and Ralf Steinmetz. Selected Topics in Service Engineering and Management for Enterprise Systems. In *Proceedings of the Pre-ICIS Workshop on Enterprise Systems Research in MIS*, Paris, France, December 2008.
- [148] André Miede, Jean-Baptiste Behuet, Apostolos Papageorgiou, Michael Niemann, Nicolas Repp, and Ralf Steinmetz. Qualitative and Quantitative Aspects of Cooperation Mechanisms for Monitoring in Service-oriented Architectures. In *Proceedings of the Third International Conference on Digital Ecosystems and Technologies (DEST 2009)*, pages 563–568, Istanbul, Turkey, June 2009.
- [149] G. Ayorkor Mills-Tettey, Anthony Stentz, and M. Bernardine Dias. The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs. Technical Report CMU-RI-TR-07-27, Robotics Institute, Pittsburgh, PA, July 2007.
- [150] Mirjam Minor, Alexander Tartakovski, and Ralph Bergmann. Representation and Structure-Based Similarity Assessment for Agile Workflows. In *Proceedings of the Seventh International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development*, volume 4626 of *Lecture Notes in Computer Science*, pages 224–238. Springer-Verlag, Berlin Heidelberg, 2007.
- [151] Tilak Mitra. A case for SOA governance. IBM developerWorks, 2005. URL <http://www.ibm.com/developerworks/webservices/library/ws-soa-govern/>. last accessed 2011-08-03.
- [152] Mariusz Momotko and Kazimierz Subieta. Process Query Language: A Way to Make Workflow Processes More Flexible. In *Proceedings of Advances in Databases and Information Systems, Eighth East European Conference (ADBIS 2004)*, volume 3255

- of *Lecture Notes in Computer Science*, pages 306–321. Springer-Verlag, Berlin Heidelberg, September 2004.
- [153] Alvaro Monge and Charles Elkan. The Field Matching Problem: Algorithms and Applications. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 267–270, 1996.
- [154] Muhammed Yaseen Muriankara. SOA governance framework and solution architecture. *IBM developerWorks*, May 2008. URL <http://www.ibm.com/developerworks/webservices/library/ws-soa-govframe/>. last accessed 2011-08-07.
- [155] Paul Müller and Bernd Reuther. Future Internet Architecture – A Service-oriented Approach. *it – Information Technology*, 50(6):383–389, 2008.
- [156] Shiva Nejati, Mehrdad Sabetzadeh, Marsha Chechik, Steve M. Easterbrook, and Pamela Zave. Matching and merging of statecharts specifications. In *29th International Conference on Software Engineering (ICSE 2007)*, pages 54–64, 2007.
- [157] Michel Neuhaus, Kaspar Riesen, and Horst Bunke. Fast Suboptimal Algorithms for the Computation of Graph Edit Distance. In *Proceedings of the 11th International Workshop on Structural and Syntactic Pattern Recognition*, volume 4109 of *Lecture Notes in Computer Science*, pages 163–172. Springer-Verlag, Berlin Heidelberg, 2006.
- [158] Eric Newcomer and Greg Lomow. *Understanding SOA with Web Services (Independent Technology Guides)*. Addison-Wesley Professional, Boston, MA, USA, December 2004.
- [159] Klaus D. Niemann. *Von der Unternehmensarchitektur zur IT-Governance*. Friedrich Vieweg & Sohn Verlag / GWV Fachverlage GmbH, Wiesbaden, Germany, October 2005.
- [160] Michael Niemann. Governance for Service-oriented Architectures: An Implementation Approach. In *Internet of Services (IoS) PhD Symposium at Interoperability for Enterprise Software and Applications (I-ESA)*, volume 374 of *CEUR Workshop Proceedings*, March 2008.
- [161] Michael Niemann, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. Towards a Generic Governance Model for Service-oriented Architectures. In *Proceedings of the 14th Americas Conference on Information Systems (AMCIS 2008)*, Toronto, ON, Canada, August 2008.
- [162] Michael Niemann, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. SOA Governance: Survey and Model. In *Proceedings of the Pre-ICIS SIGSVC (Special Interest Group Services) Workshop on Services*, Paris, France, December 2008.
- [163] Michael Niemann, Michael Appel, Nicolas Repp, and Ralf Steinmetz. Towards a Consistent Lifecycle Model in Service Governance. In *Proceedings of the 15th Americas Conference on Information Systems (AMCIS 2009)*, San Francisco, CA, USA, August 2009.

- [164] Michael Niemann, Christian Janiesch, Nicolas Repp, and Ralf Steinmetz. Challenges of Governance Approaches for Service-Oriented Architectures. In *Proceedings of the Third International Conference on Digital Ecosystems and Technologies (DEST 2009)*, pages 634–639, Istanbul, Turkey, June 2009.
- [165] Michael Niemann, Julian Eckert, and Ralf Steinmetz. Semantische Analyse zur Unterstützung von SOA-Governance. In *Service Science - Neue Perspektiven für die Informatik - Proceedings der Informatik 2010*, volume 2, pages 299–304, Leipzig, Germany, September 2010.
- [166] Michael Niemann, Kim David Hagedorn, Stefan Schulte, and Ralf Steinmetz. Typische Elemente von SOA-Governance-Ansätzen – Ein Survey. Technical Report KOM-TR-2010-04, Technische Universität Darmstadt, Fachbereich Multimedia Kommunikation (KOM), December 2010.
- [167] Michael Niemann, André Miede, Wolfgang Johannsen, Nicolas Repp, and Ralf Steinmetz. Structuring SOA Governance. *International Journal of IT/Business Alignment and Governance (IJITBAG)*, 1(1):58–75, January 2010.
- [168] Michael Niemann, Sascha Hombach, Stefan Schulte, and Ralf Steinmetz. Das IT-Governance-Framework COBIT als Wissensdatenbank – Entwurf, Umsetzung und Evaluation einer Ontologie. Technical Report KOM-TR-2011-01, Technische Universität Darmstadt, Fachbereich Multimedia Kommunikation (KOM), January 2011.
- [169] Michael Niemann, Melanie Siebenhaar, Julian Eckert, and Ralf Steinmetz. Process Model Analysis using Related Cluster Pairs. In *Business Process Management Workshops: BPM 2010 International Workshops and Education Track*, volume 66 of *Lecture Notes in Business Information Processing*, pages 547–558. Springer-Verlag, Berlin Heidelberg, February 2011.
- [170] Michael Niemann, Melanie Siebenhaar, Stefan Schulte, and Ralf Steinmetz. Comparison and Retrieval of Process Models using Related Cluster Pairs. *Computers in Industry*, 63(2), February 2012. (to be published).
- [171] Daniel Oberle, Nadeem Bhatti, Saartje Brockmans, Michael Niemann, and Christian Janiesch. Countering Service Information Challenges in the Internet of Services. *BISE - Business and Information System Engineering Journal*, 5:370–390, September 2009.
- [172] Daniel Oberle, Nadeem Bhatti, Saartje Brockmans, Michael Niemann, and Christian Janiesch. Effektive Handhabung von Service-Informationen. *Wirtschaftsinformatik*, (5):429–452, October 2009.
- [173] Institute of Electrical and Electronics Engineers (IEEE). IEEE Recommended Practice for Architectural Description of Software Intensive Systems (IEEE Std 1471-2000), 2000.
- [174] Committee of Sponsoring Organizations of the Treadway Commission (COSO). Internal Control – Integrated Framework (COSO report), 1994. URL <http://www.coso.org/IC-IntegratedFramework-summary.htm>. last accessed 2011-07-28.

- [175] Enterprise Architecture Council of the California Franchise Tax Board (FTB). Enterprise Architecture Definition – Business Process Management (BPM), May 2010. URL <http://www.ftb.ca.gov/aboutFTB/Projects/ITSP/BPM.pdf>. last accessed 2011-08-08.
- [176] Office of Governance Commerce (OGC). *ITIL v3: Information Technology Infrastructure Library Version 3*, volume 1-5. London: The Stationary Office, 2007.
- [177] Object Management Group (OMG). Business Process Modeling Notation (BPMN) Version 1.2. Technical Report, Object Management Group (OMG), January 2009. URL <http://www.omg.org/spec/BPMN/1.2/PDF>. last accessed 2011-08-08.
- [178] Object Management Group (OMG). OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.2 with change bars, February 2009. URL <http://www.omg.org/cgi-bin/doc?formal/09-02-03.pdf>. last accessed 2011-08-08.
- [179] Organization for Economic Cooperation and Development (OECD), Directorate for Financial, Fiscal and Enterprise Affairs. *OECD Principles of Corporate Governance*. Organisation for Economic Cooperation and Development, April 1999.
- [180] Hubert Österle. *Business Engineering. Prozeß- und Systementwicklung. Band 1 – Entwurfstechniken*. Springer-Verlag, Berlin Heidelberg, second revised edition, 1995.
- [181] Bo Pang and Lillian Lee. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2:1–135, January 2008.
- [182] Apostolos Papageorgiou, Bastian Leferink, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. Bridging the Gaps towards Structured Mobile SOA. In *Seventh International Conference on Advances in Mobile Computing & Multimedia (MoMM2009)*, pages 288–294, Kuala Lumpur, Malaysia, 2009.
- [183] Apostolos Papageorgiou, Stefan Schulte, Dieter Schuller, Michael Niemann, Nicolas Repp, and Ralf Steinmetz. Governance of a Service-Oriented Architecture for Environmental and Public Security. In *Proceedings of the Information Technologies in Environmental Engineering (ITEE 2009) - Fourth International ICSC Symposium*, pages 39–52, Thessaloniki, Greece, May 2009.
- [184] Apostolos Papageorgiou, André Miede, Dieter Schuller, Stefan Schulte, and Ralf Steinmetz. Always Best Served: On the behaviour of QoS- and QoE-based Algorithms for Web Service Adaptation. In *Proceedings of the Pervasive Computing and Communications (PERCOM) Workshops - Eighth International Workshop on Managing Ubiquitous Communications and Services (MUCS 2011)*, pages 71–76, Seattle, WA, USA, March 2011.
- [185] Michael P. Papazoglou. *Web Services: Principles and Technology*. Pearson Education Limited, Harlow, UK, 2008.
- [186] Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, Frank Leymann, and Bernd J. Krämer. Service-Oriented Computing: A Research Roadmap. In *Service Oriented Computing (SOC), Dagstuhl Seminar Proceedings, Dagstuhl, Germany, May 2006*.

- [187] Mike P. Papazoglou. Service-oriented Computing: Concepts, Characteristics and Directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE 2003)*, pages 3–12, Rome, Italy, December 2003.
- [188] Mike P. Papazoglou and Dimitrios Georgakopoulos. Service-Oriented Computing (Introduction). *Communications of the ACM*, 46(10):24–28, 2003.
- [189] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
- [190] Gregory J. Pottie and William J. Kaiser. Wireless Integrated Network Sensors. *Communications of the ACM*, 43(5):51–58, 2000.
- [191] Nissanka B. Priyantha, Aman Kansal, Michel Goraczko, and Feng Zhao. Tiny Web Services: Design and Implementation of Interoperable and Evolvable Sensor Networks. In *Proceedings of the Sixth ACM Conference on Embedded Network Sensor Systems (SenSys 2008)*, pages 253–266, Raleigh, NC, USA, November 2008.
- [192] Frank Puhlmann. Soundness Verification of Business Processes Specified in the Pi-Calculus. In *Proceedings of the 2007 OTM Confederated International Conference on: On the move to meaningful internet systems: CoopIS, DOA, ODBASE, GADA, and IS - Volume Part I*, volume 4803 of *Lecture Notes in Computer Science*, pages 6–23. Springer-Verlag, Berlin Heidelberg, 2007.
- [193] Konstantin Pussep. *Peer-assisted Video-on-Demand: Cost Reduction and Performance Enhancement for Users, Overlay Providers, and Network Operators*. PhD thesis, Technische Universität Darmstadt, Germany, February 2011.
- [194] Karin Quack. IT-Kosten: Abspecken bis zum Hungertod? *COMPUTERWOCHE*, April 2010. URL <http://www.computerwoche.de/1934854>. last accessed 2011-08-12.
- [195] Joachim Quantz. SOA in der Praxis - Wie Unternehmen SOA erfolgreich einsetzen. Berlecon Research GmbH, 2006. URL http://www.berlecon.de/studien/downloads/Berlecon_SOA.pdf. last accessed 2011-08-05.
- [196] Oliver Raabe, Richard Wacker, Daniel Oberle, Christian Baumann, and Christian Funk. *Recht ex Machina – Formalisierung des Rechts im Internet der Dienste*. Springer-Verlag, Berlin Heidelberg, 2012.
- [197] Corina Radulescu, Hui-Min Tan, Malini Jayaganesh, Wasana Bandara, Michael zur Muehlen, and Sonia Lippe. A Framework of Issues in Large Process Modeling Projects. In *Proceedings of the European Conference on Information Systems (ECIS 2006)*, pages 1594–1605, Goteborg, Sweden, June 2006.
- [198] Dilip Rane and Greg Lomow. SOA Governance: More than just Registries and Repositories. BearingPoint Whitepaper, 2008. URL <http://www.pwc.com/us/en/increasing-it-effectiveness/assets/soa-governance.pdf>. last accessed 2011-08-07.
- [199] Hajo A. Reijers, Ronny S. Mans, and Robert A. van der Toorn. Improved Model Management with Aggregated Business Process Models. *Data Knowledge Engineering*, 68:221–243, February 2009.

- [200] Nicolas Repp. *Überwachung und Steuerung dienstbasierter Architekturen – Verteilungsstrategien und deren Umsetzung*. Books on Demand GmbH, Norderstedt, PhD thesis, Technische Universität Darmstadt, Germany, July 2009.
- [201] Nicolas Repp, Stefan Schulte, Julian Eckert, Rainer Berbner, and Ralf Steinmetz. An Approach to the Analysis and Evaluation of an Enterprise Service Ecosystem. In *Proceedings of the First International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing*, pages 42–51, Barcelona, Spain, July 2007.
- [202] Nicolas Repp, Julian Eckert, Stefan Schulte, Michael Niemann, Rainer Berbner, and Ralf Steinmetz. Towards Automated Monitoring and Alignment of Service-based Workflows. In *Proceedings of the International Conference on Digital Ecosystems and Technologies 2008 (DEST 2008)*, Phitsanulok, Thailand, February 2008.
- [203] Nicolas Repp, Andreas Mauthe, and Ralf Steinmetz. Chancen und Risiken bei Einführung von IT-Governance-Frameworks. *IT-Governance, Zeitschrift des ISACA Germany Chapter e.V.*, 3:9–15, January 2008.
- [204] Nicolas Repp, André Miede, Michael Niemann, and Ralf Steinmetz. WS-Re2Policy: A policy language for distributed SLA monitoring and enforcement. In *Proceedings of the Third International Conference on Systems and Networks Communications*, pages 256–261, Sliema, Malta, October 2008.
- [205] Nicolas Repp, Dieter Schuller, Melanie Siebenhaar, Andre Miedé, Michael Niemann, and Ralf Steinmetz. On distributed SLA Monitoring and Enforcement in Service-oriented Systems. *International Journal On Advances in Systems and Measurements*, 2(1):33–43, June 2009.
- [206] Bernd Reuther. *Ein serviceorientierter Ansatz zur Abstraktion von Kommunikationsprotokollen im Internet*. Dr. Hut Verlag, München, PhD thesis, Technische Universität Kaiserslautern, Germany, 2011.
- [207] Michael M. Richter. Classification and learning of similarity measures. In *Proceedings of the 16th Annual Meeting of the German Society for Classification*, Dortmund, Germany, March 1992.
- [208] Christoph Riedl, Norman May, Jan Finzen, Stephan Stathel, Viktor Kaufman, and Helmut Krmar. An Idea Ontology for Innovation Management. *International Journal on Semantic Web and Information Systems*, 5(4):1–18, 2009.
- [209] Christoph Riedl, Norman May, Jan Finzen, Stephan Stathel, Torsten Leidig, and Roxana Belecheau. Managing Service Innovations with an Idea Ontology. In *Proceedings of the 19th International Conference of RESER (European Association for Research on Services)*, pages 876–892, Budapest, Hungary, September 2009.
- [210] Ingo Rieger and Ralf Bruns. SOA-Governance und -Rollen: Sichern des Mehrwertes einer Service-Orientierten Architektur. *Objektspektrum*, 1:20–24, April 2007.
- [211] Stefanie Rinderle-Ma, Linh Thao Ly, and Peter Dadam. Business Process Compliance. *EMISA Forum*, 28(2):24–29, 2008.

- [212] Jeanne Ross, Peter Weill, and David Robertson. *Enterprise Architecture as Strategy: Creating a Foundation for Business Execution*. Harvard Business School Press, Boston, MA, USA, 2006.
- [213] August-Wilhelm Scheer. *ARIS - Vom Geschäftsprozess zum Anwendungssystem*. Springer-Verlag, Berlin Heidelberg, fourth edition, 2002.
- [214] August-Wilhelm Scheer and Kristof Schneider. ARIS – Architecture of Integrated Information Systems. In Peter Bernus, Jacek Blazewicz, Günter J. Schmidt, Michael J. Shaw, Peter Bernus, Kai Mertins, and Günter Schmidt, editors, *Handbook on Architectures of Information Systems*, International Handbooks on Information Systems, pages 605–623. Springer-Verlag, Berlin Heidelberg, 2006.
- [215] Joachim Schelp and Matthias Stutz. SOA-Governance. *HMD - Praxis der Wirtschaftsinformatik*, 253(253):66–73, February 2007.
- [216] Joachim Schelp, Matthias Stutz, and Robert Winter. SOA-Risiken und SOA-Governance. In Gernot Starke and Stefan Tilkov, editors, *SOA-Expertenwissen*, chapter 41, pages 661–668. dpunkt.verlag Heidelberg, first edition, 2007.
- [217] Tom G. J. Schepers, Maria-Eugenia Iacob, and Pascal A. T. van Eck. A Lifecycle Approach to SOA Governance. In *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC 2008)*, pages 1055–1061, Fortaleza, Ceará, Brazil, March 2008.
- [218] Alexander Schill and Thomas Springer. *Verteilte Systeme – Grundlagen und Basistechnologien*. Springer-Verlag, Berlin Heidelberg, March 2007.
- [219] Helmut Schmid. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing*, Manchester, UK, 1994.
- [220] Philipp Scholl. *Semantic and Structural Analysis of Web-based Learning Resources: Supporting Self-directed Resource-based Learning*. PhD thesis, Technische Universität Darmstadt, Germany, June 2011.
- [221] Dieter Schuller, Apostolos Papageorgiou, Stefan Schulte, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. Process Reliability in Service-Oriented Architectures. In *Proceedings of the Third International Conference on Digital Ecosystems and Technologies (DEST 2009)*, pages 640–645, Istanbul, Turkey, June 2009.
- [222] Dieter Schuller, Julian Eckert, André Miede, Stefan Schulte, and Ralf Steinmetz. QoS-Aware Service Composition for Complex Workflows. In *Proceedings of the Fifth International Conference on Internet and Web Applications and Services (ICIW 2010)*, pages 333–338, Barcelona, Spain, May 2010.
- [223] Dieter Schuller, André Miede, Julian Eckert, Ulrich Lampe, Apostolos Papageorgiou, and Ralf Steinmetz. QoS-based Optimization of Service Compositions for Complex Workflows. In *Proceedings of the Eighth International Conference on Service Oriented Computing (ICSOC 2010)*, pages 641–648, December 2010.
- [224] Roy W. Schulte and Yefim F. Natis. “Service Oriented” Architectures, Part 1. Technical Report SSA Research Note SPA-401-068, Gartner Group, April 1996.

- [225] Stefan Schulte. *Web Service Discovery Based on Semantic Information – Query Formulation and Adaptive Matchmaking*. PhD thesis, Technische Universität Darmstadt, Germany, June 2010.
- [226] Stefan Schulte, Ulrich Lampe, Julian Eckert, and Ralf Steinmetz. LOG4SWS.KOM: Self-Adapting Semantic Web Service Discovery for SAWSDL. In *Proceedings of the Fourth International Workshop of Software Engineering for Adaptive Service-Oriented Systems (SEASS 2010) at the Sixth World Congress on Services (SERVICES 2010)*, pages 511–518, Los Alamitos, CA, USA, July 2010.
- [227] Fabrizio Sebastiani. Machine Learning in Automated Text Categorization. *ACM Computing Survey*, 34:1–47, March 2002.
- [228] Richard W. Selby. *Software Engineering: Barry W. Boehm’s Lifetime Contributions to Software Development, Management, and Research*. Practitioners Series. Wiley-IEEE Computer Society Press, 2007.
- [229] Alkesh Shah. A Value Aligned SOA Maturity Model. Keane, Inc. Online Article, 2007. URL <http://enterpriseinnovator.com/index.php?articleID=12220§ionID=5>. last accessed 2011-08-07.
- [230] Melanie Siebenhaar, Michael Niemann, Julian Eckert, and Ralf Steinmetz. ProMatch.KOM: Tool Support for Process Model Analysis and Improvement. In *Demonstration Track of the Eighth International Conference on Business Process Management (BPM 2010)*, Hoboken, NJ, USA, September 2010.
- [231] Melanie Siebenhaar, Ulrich Lampe, Tim Lehrig, Sebastian Zöllner, Stefan Schulte, and Ralf Steinmetz. Complex Service Provisioning in Collaborative Cloud Markets. In *Proceedings of the Fourth European Conference ServiceWave*, Poznan, Poland, October 2011. (accepted for publication).
- [232] Howard Smith and Peteringar. *Business Process Management – The Third Wave*. Meghan-Kiffer Press, Tampa, FL, USA, 2003.
- [233] Software AG. SOA Governance – Rule your SOA. Software AG Whitepaper, 2007. URL http://www.softwareag.com/Corporate/Images/WP%20SOA%20Governance_tcm16-22130.pdf. last accessed 2011-08-07.
- [234] Software AG. Five Steps for Building the Business Case for SOA Governance. Software AG Whitepaper, 2008. URL http://www.softwareag.com/corporate/images/BusCase_SOA_WP_Nov08-web_tcm16-45795.pdf. last accessed 2011-08-07.
- [235] Software AG. SOA Lifecycle Governance. Software AG Online Article, 2008. URL <http://documentation.softwareag.com/webmethods/centrasite/soal8/overview/soa.htm>. last accessed 2011-08-07.
- [236] Ian Sommerville. *Software Engineering*. Addison-Wesley, Redwood City, CA, USA, ninth edition, 2010.
- [237] Sonic Software Corporation. A New Service-oriented Architecture Maturity Model. Sonic Software Report, 2006. URL http://soa.omg.org/Uploaded%20Docs/SOA/SOA_Maturity.pdf. last accessed 2011-08-07.

- [238] Josef Spillner, Iris Braun, and Alexander Schill. Towards Unified Service Hosting. In *Proceedings of the Fourth International Conference on Software and Data Technologies (ICSOF 2009)*, volume 2, pages 31–36, Sofia, Bulgaria, July 2009.
- [239] Josef Spillner, Matthias Winkler, Sandro Reichert, Jorge Cardoso, and Alexander Schill. Distributed Contracting and Monitoring in the Internet of Services. In *Proceedings of the Ninth IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS 2009)*, volume 5523 of *Lecture Notes in Computer Science*, pages 129–142. Springer-Verlag, Berlin Heidelberg, 2009.
- [240] David Sprott. The SOA Maturity Model. *CBDJ Journal*, 12:4–17, 2005.
- [241] Gernot Starke. SOA Governance: Crucial Necessity or Waste of Time. *InfoQ*, November 2007. URL <http://www.infoq.com/articles/governance-gernot-starke>. last accessed 2011-08-07.
- [242] Ralf Steinmetz. *Multimedia-Technologie: Grundlagen, Komponenten und Systeme*. Springer-Verlag, Berlin Heidelberg, third edition, 2000.
- [243] Ralf Steinmetz and Klara Nahrstedt. *Multimedia Systems*. X.media.publishing Series. Springer-Verlag, Berlin Heidelberg, April 2004.
- [244] Ralf Steinmetz and Klaus Wehrle, editors. *Peer-to-peer Systems and Applications*, volume 3485 of *Lecture Notes of Computer Science*. Springer-Verlag, Berlin Heidelberg, 2005.
- [245] Christoph Strnadl. Bridging Architectural Boundaries Design and Implementation of a Semantic BPM and SOA Governance Tool. In *Proceedings of the Fifth International Conference on Service-Oriented Computing (ICSOC 2007)*, volume 4749 of *Lecture Notes in Computer Science*, pages 518–529. Springer-Verlag, Berlin Heidelberg, 2007.
- [246] Systinet. SOA Governance: Balancing Flexibility and Control Within an SOA. Systinet Whitepaper, 2006. URL http://www.webservices.org/content/download/83830/1225383/file/SOA_Gov906.pdf. last accessed 2011-08-07.
- [247] Product CMMI Team. CMMI for Acquisition, Version 1.2. Technical Report CMU/SEI-2007-TR-017, Carnegie Mellon Software Engineering Institute (SEI), November 2007. URL <http://www.sei.cmu.edu/cmmi/models/ACQ-v12-announce.html>.
- [248] The Open Group. SOA Governance Framework - Technical Standard, August 2009. URL <http://www.opengroup.org/bookstore/catalog/c093.htm>. last accessed 2011-08-07.
- [249] The Open Group. The Open Group Architecture Framework (TOGAF). USA, 2009. URL <http://www.opengroup.org/togaf/>. last accessed 2011-07-27.
- [250] Wil van der Aalst. Formalization and Verification of Event-driven Process Chains. *Information & Software Technology*, 41(10):639–650, 1999.

- [251] Wil van der Aalst and Twan Basten. Inheritance of Workflows: an Approach to Tackling Problems Related to Change. *Theoretical Computer Science*, 270:125–203, January 2002.
- [252] Wil van der Aalst, Ana Karla Alves de Medeiros, and A. Weijters. Process Equivalence: Comparing Two Process Models Based on Observed Behavior. In *Business Process Management*, volume 4102 of *Lecture Notes in Computer Science*, chapter 10, pages 129–144. Springer-Verlag, Berlin Heidelberg, 2006.
- [253] Boudewijn van Dongen, Ana Karla Alves de Medeiros, Eric Verbeek, A. Weijters, and Wil van der Aalst. The ProM Framework: A New Era in Process Mining Tool Support. *Applications and Theory of Petri Nets 2005*, pages 444–454, 2005.
- [254] Boudewijn van Dongen, Remco Dijkman, and Jan Mendling. Measuring Similarity between Business Process Models. In *Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE 2008)*, pages 450–464, Montpellier, France, June 2008.
- [255] Rob J. van Glabbeek and W. Peter Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Journal of the ACM*, 43:555–600, May 1996.
- [256] Axel von Werder. Corporate Governance. In Richard Köhler, Hand-Ulrich Küpper, and Andreas Pfungsten, editors, *Handwörterbuch der Betriebswirtschaft (HWB)*, volume 1 of *Enzyklopädie der Betriebswirtschaftslehre (EdBWL)*, pages 221–229. Schäffer-Poeschel, Stuttgart, sixth edition, 2007.
- [257] World Wide Web Consortium (W3C). Web Services Policy 1.5 - Framework, 2007. URL <http://www.w3.org/TR/ws-policy/>. last accessed 2011-08-07.
- [258] Quinton Wall. Understanding the Lifecycle within a SOA: Design Time. Oracle Whitepaper, April 2006. URL <http://quintonwall.com/wp-content/uploads/2008/08/soa-service-lifecycle-design.pdf>. last accessed 2011-08-07.
- [259] Quinton Wall. Understanding the Lifecycle within a SOA: Run Time. Oracle Whitepaper, August 2006. URL <http://quintonwall.com/wp-content/uploads/2008/08/soa-service-lifecycle-run.pdf>. last accessed 2011-08-07.
- [260] WebLayers SOA Forum. 2007 Survey on SOA Governance, September 2007. URL http://www.bitpipe.com/detail/RES/1209665671_580.html. last accessed 2008-10-05.
- [261] WebMethods. SOA Governance - Enabling Sustainable Success with SOA, October 2006. URL http://www.cioindex.com/it_sourcing/ArticleId/1091/Enabling-Sustainable-Success-with-SOA.aspx. last accessed 2011-08-07.
- [262] Peter Weill. Innovating with Information Systems: What do the most agile firms in the world do? Presentation at the Sixth e-Business Conference, Barcelona Spain, March 2007. URL http://www.iese.edu/en/files/6_29338.pdf. last accessed 2011-08-10.
- [263] Peter Weill and Jeanne W. Ross. *IT Governance - How Top Performers Manage IT Decision Rights for Superiour Results*. Harvard Business School Press, Boston, MA, USA, 2004.

- [264] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag, Berlin Heidelberg, 2007.
- [265] Philipp J. Windley. Governing SOA. InfoWorld Online Article, 2006. URL <http://www.infoworld.com/d/architecture/governing-soa-221>. last accessed 2011-08-07.
- [266] Philipp J. Windley. Teaming up for SOA. InfoWorld Online Article, 2007. URL <http://www.infoworld.com/t/architecture/teaming-soa-620>. last accessed 2011-08-07.
- [267] Phillip J. Windley. SOA Governance: Rules of the Game. *INFOWORLD.COM*, pages 29–35, 2006. URL http://www.cioindex.com/enterprise_architecture/ArticleId/1090/SOA-Governance-Rules-of-The-Game.aspx. last accessed 2011-08-07.
- [268] Matthias Winkler, Jorge Cardoso, and Gregor Scheithauer. Challenges of Business Service Monitoring in the Internet of Services. In *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services (iiWAS 2008)*, pages 613–616, Linz, Austria, November 2008.
- [269] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, San Francisco, CA, second edition, 2005.
- [270] Andreas Wombacher. Evaluation of technical measures for workflow similarity based on a pilot study. In *14th International Conference on Cooperative Information Systems*, volume 4275 of *Lecture Notes in Computer Science*, pages 255–272. Springer, 2006.
- [271] Bobby Woolf. Introduction to SOA Governance. IBM developerWorks, June 2006. URL <http://www.ibm.com/developerworks/library/ar-servgov/>. last accessed 2011-08-07.
- [272] Zhiqiang Yan, Remco Dijkman, and Paul Grefen. Business Process Model Repositories - Framework and Survey. *Information Sciences*, 2009. URL http://cms.ieis.tue.nl/Beta/Files/WorkingPapers/Beta_wp292.pdf. last accessed 2011-08-07.
- [273] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity Search - The Metric Space Approach*, volume 32 of *Advances in Database Systems*. Springer-Verlag, Berlin Heidelberg, 2006.
- [274] Justin Zobel. *Writing for Computer Science*. Springer-Verlag, New York, 2004.
- [275] Sebastian Zöllner, Andreas Reinhardt, Heiko Guckes, Dieter Schuller, and Ralf Steinmetz. On the Integration of Wireless Sensor Networks and Smartphones in the Logistics Domain. In *Proceedings of the 10th GI/ITG KuVS Fachgespräch 'Drahtlose Sensornetze'*, September 2011. (accepted for publication).

LIST OF FIGURES

Figure 1.1	Research agenda for Service-oriented Computing	3
Figure 2.1	SOA roles (e.g, [65, 125, 143, 187])	11
Figure 2.2	Embedding of SOA Governance in the enterprise context [215] . . .	16
Figure 2.3	Elements of Compliance Management [104]	18
Figure 2.4	Levels of business processes (adapted from [264])	22
Figure 3.1	Results of the component analysis	27
Figure 3.2	Example for a RACI chart [86]	34
Figure 3.3	Overview of important aspects of SLCs	36
Figure 3.4	Consolidated phases of the proposed SOA Procedure Models . . .	38
Figure 3.5	Integration of components per author group and quality level . . .	51
Figure 4.1	Survey on Service Life Cycles (cf. Niemann et al. [163], Fig. B.5) . .	57
Figure 4.2	Consolidated Service Life Cycle	60
Figure 4.3	Operational Model for SOA Governance (cf. Niemann et al. [167]) .	65
Figure 5.1	A process model graph (PMG) representing an EPC	83
Figure 5.2	SESE regions	84
Figure 5.3	Computation of Related Cluster Pairs	90
Figure 5.4	Example pair of process models: visualisation of all determined clusters (as grey rectangles)	92
Figure 5.5	Identification of related cluster pairs	96
Figure 5.6	Related cluster pairs	97
Figure 5.7	ProcSim.KOM: Architecture	102
Figure 5.8	ProcSim.KOM: Manual result corrections	102
Figure 5.9	ProcSim.KOM: Result visualisation	103
Figure 5.10	Process model comparison	107
Figure 5.11	Average precision per query model	111
Figure 5.12	Interpolated precision-recall curve (retrieval scenario)	113
Figure A.1	Permitted EPC connections [108]	155
Figure A.2	Well-structured EPC constructs [67]	156
Figure B.1	Approaches per author group	157
Figure B.2	Perceived level of risk of lacking governance [260]	162
Figure B.3	Sufficiency of SOA Governance approach [260]	163
Figure B.4	Employed policy enforcement in organisation [260]	163
Figure B.5	Survey on Service Life Cycles (cf. Niemann et al. [163])	172
Figure B.6	Relative number of mentions per component (normalised)	173
Figure B.7	Integration of components per author group and quality level (founded recommendations only)	174
Figure C.1	Comparison scenario: F1-measure per considered similarity measure	179
Figure C.2	Comparison scenario: Improval of semantic measures	179
Figure C.3	Retrieval scenario: Results overview per measure	182
Figure D.1	ProcSim.KOM: Process model selection	183
Figure D.2	ProcSim.KOM: Configuration dialog	184
Figure D.3	ProcSim.KOM: Result visualisation (comparison scenario)	184

LIST OF TABLES

Table 2.1	Potential business and technical benefits of an SOA [158]	8
Table 3.1	Detailed results of the component analysis	28
Table 5.1	Overview of related work in BPM	80
Table 5.2	Similarity values and node assignments for process models graphs P_1 and P_2 using the measure sim^{SFO} (cf. Figure 5.4)	93
Table 5.3	Cross validation results for process model comparison	108
Table 5.4	Overall results of process retrieval metrics	112
Table B.1	Organisational entities of SOA Governance approaches	158
Table B.2	Core aspects of the definitions	164
Table C.1	Average accuracy by modification class	176
Table C.2	Test case structure: Business domains	177
Table C.3	Number of nodes and relevant models per query model	178
Table C.4	Retrieval scenario: Average precision per query model	180
Table C.5	Retrieval scenario: R-Precision per query model	180
Table C.6	Retrieval scenario: Precision per recall	180
Table C.7	Retrieval scenario: First-5-Precision per query model	181
Table C.8	Retrieval scenario: First-10-Precision per query model	181
Table C.9	Retrieval scenario: First-20-Precision per query model	181

NOMENCLATURE

ARIS	Architecture of Integrated Information Systems
BAM	Business Activity Monitoring
BPEL	Business Process Execution Language
BPM	Business Process Management
BPMN	Business Process Modelling Notation
CMMI	Capability Maturity Model Integration (by SEI)
COBIT	Control Objectives for IT and other Technologies
EPC	Event-Driven Process Chain
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
IaaS	Infrastructure as a Service
IoS	Internet of Services
IR	Information Retrieval
ITIL	IT Infrastructure Library
KPI	Key Performance Indicator
OWL	Web Ontology Language
P2P	Peer-to-Peer
PAIS	Process-Aware Information System
RACI	RACI-Chart: <i>Responsible, Accountable, Consulted, Informed</i>
RAEW	RAEW analysis: <i>Responsibility, Authority, Expertise, and Work</i>
REST	Representational State Transfer
SEI	Software Engineering Institute
SIMM	Service Integration Maturity Model
SLA	Service Level Agreement
SLC	Service Life Cycle
SLCM	Service Life Cycle Management

SMM	SOA Maturity Model
SOA	Service-oriented Architecture
SOA CoE	SOA Centre of Excellence
SOAP	<i>formerly</i> Simple Object Access Protocol
SOC	Service-oriented Computing
SPRM	SAP R/3 Process Reference Model
TOGAF	The Open Group Architecture Framework
UDDI	Universal Description, Discovery and Integration
UML	Unified Modelling Language
WfMS	Workflow Management System
WSDL	Web Services Description Language
WSPL	Web Services Policy Language

FURTHER BASICS

IN this chapter we provide further background on Enterprise Architectures (App. A.1) and EPCs (App. A.2).

A.1 Enterprise Architecture

By the ANSI standard 1471-2000, *architecture* is defined as “the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution.” [173] Important components of an *Enterprise Architecture (EA)* are separate architectures for processes, information, domains, and applications [216]. Combining these, an EA consists of several smaller architectures, and represents the entire enterprise organisation [249]. An EA is a coarse-grained set of architectures, linked by specified interrelations.

Ross et al. [212] distinguish operating model and EA. As operating model they define: “The desired level of business process integration and business process standardization for delivering goods and services to customers.” Based on this, an EA is defined as

“The organizing logic for key business process and IT capabilities reflecting the integration and standardization requirements of the firm’s operating model.” [212]

Further, an EA captures the organising logic in technical choices and policies, while defining data and infrastructure as stable platform to support quickly changing applications [263].

Frameworks for EA specify a common taxonomy, meta models for the EA description, design methods, and reference models serving as blueprints [249]. The Open Group [249], who authored the most popular established EA framework TOGAF, identifies three practice domains: *Business Architecture*, *Information Systems Architecture*, and *Technology Architecture*, where the second is broken down into *Information Architecture* and *Applications Architecture*. As a main objective of the development of an EA, they specify

“providing the fundamental technology and process structure for an IT strategy. This in turn makes IT a responsive asset for a successful modern business strategy.”¹

Further, according to The Open Group [249], an EA targets the achievement of “the right balance between IT efficiency and business innovation, [...] [while EA] assures the needs of the organization for an integrated IT strategy, permitting the closest possible synergy across the extended enterprise.”

As EA, in the course of this thesis, we understand a generic reference model for enterprise organisation, in particular addressing business structure, technology, and information systems, and their complex interrelations. An SOA system is considered a candidate

¹ <http://pubs.opengroup.org/architecture/togaf8-doc/arch/toc.html>, last accessed 2011-08-11

of an IT system that can be designed and developed in a company to become a part of the EA, as *Information System*.

An EA is a description of the overall structure of an entire enterprise, consisting of business entities and their properties, as well as their interrelations. It covers the terminology, the alignment of the business entities, and their interaction with the external environment. The principles used to design this architecture are themselves a central part of the EA [63, 212, 262].

A.2 Event-driven Process Chains (EPC)

As our test cases consist of collections of EPCs, we provide further details here.

EPCs are a method to model business processes, which was introduced by Keller et al. [108] within the scope of the Architecture of Integrated Information Systems (ARIS) [108, 111, 213]. ARIS is a holistic framework for the description and design of IT-based information systems, and thus an all-embracing approach for business process modeling [213, 214]. In general, EPCs are used to represent the control flow perspective of a business process. In addition to the basic variant of EPCs, an extended notation called *enhanced* EPC (eEPC) exists that considers additional elements (e.g., organisational units, supporting systems) of a business process.

Although EPCs are based on Petri nets, which have formal semantics, EPCs only represent a semiformal method for the description of business processes [250]. The method of EPCs is widespread and its underlying concepts can be easily transferred to other modeling approaches, which is the reason for its usage within this thesis. We use basic EPCs, as eEPCs contain additional elements that are specific to the ARIS concept.

In general, a basic EPC can be defined as follows:

Definition 8 (Event-driven process chain). *An event-driven process chain (EPC) represents a directed, connected graph $G = (V, E)$. The set of nodes V consists of three disjoint sets of functions F , events E , and connectors C . The nodes are connected by edges representing the control flow. Functions and events appear in an alternating sequence. [169]*

The basic notation contains the following three types of nodes:

- *Function*: changes the state of an object
- *Event*: represents a state, triggers a function, and results from a function
- *Connector*: describes logical connections between functions and events

Three types of connectors are distinguished [108]:

- **AND** (conjunctive connection): all incoming functions/events must be complete/occur to trigger/produce the subsequent functions/events
- **OR** (disjunctive connection): at least one incoming function/event must be complete/occur to trigger/produce the subsequent functions/events
- **XOR** (adjunctive connection): exactly one incoming function/event must be complete/occur to trigger/produce the subsequent functions/events

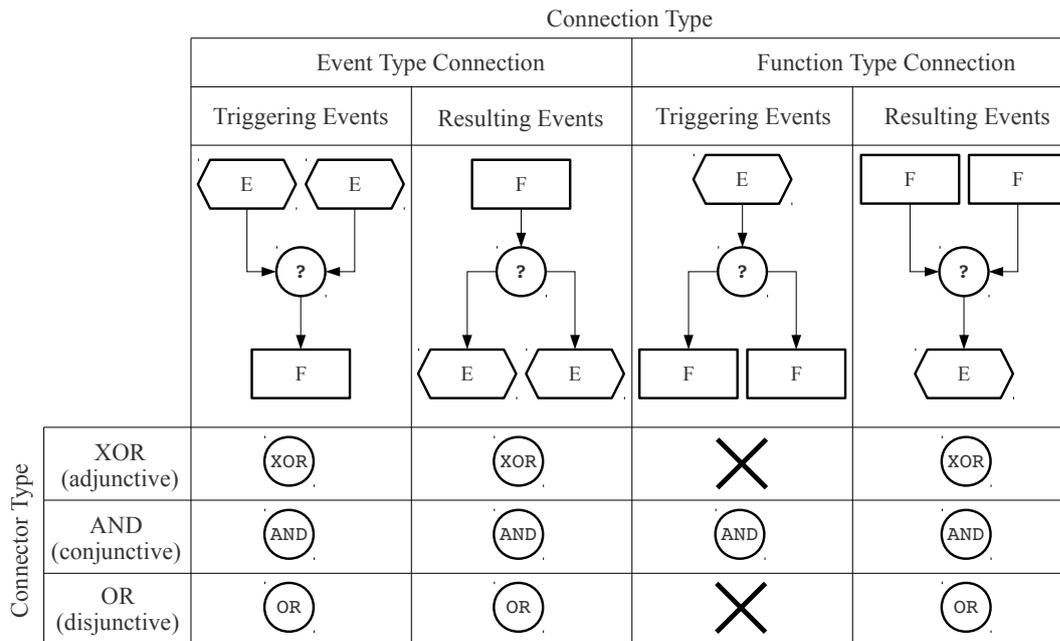


Figure A.1: Permitted EPC connections [108]

In general, functions and events have to appear in an alternating sequence within a process model, i.e., two functions or two events may not be arranged successively, even not, if a logical connector resides between them. This is due to the fact, that an event triggers a function and a function changes the state of an object, which results in a new state represented by an event. Each function and each event have exactly one incoming and one outgoing edge, except for the start and end events, which have either exactly one outgoing or one incoming edge. Connectors can act as split connectors, i.e., they have a single incoming edge and multiple outgoing edges, or as join connectors, i.e., they have multiple incoming edges and a single outgoing edge [45].

Furthermore, functions can be triggered by more than a single event, and multiple events can also be the result of a single function. Unfortunately, not all relations between events and functions are allowed, since events are not able to take a decision. Generally, two types of connections can be distinguished. Within an event type connection, two or more events are connected to a function via a connector node, which can be subclassed into triggering event connections and resulting event connections, depending on the part assigned to the event. A function type connection can be defined analogously [108].

As already stated, EPCs only represent a semiformal method for the description of business processes. But automated processing of EPCs (e.g. simulation, verification) requires a formal semantics to avoid ambiguity and errors (e.g. deadlocks) (cf. [67, 250]). Therefore, several approaches were introduced for the formalization of EPCs (e.g., [122, 250]). As part of these formalisation approaches, so-called well-structured EPCs are introduced, which, informally spoken, comprise only nested, well-structured constructs. Besides basic sequences that consist of functions and events, four well-structured EPC constructs are distinguished (cf. Fig. A.2). Control flow connectors are the critical elements with respect to well-structuredness [67].

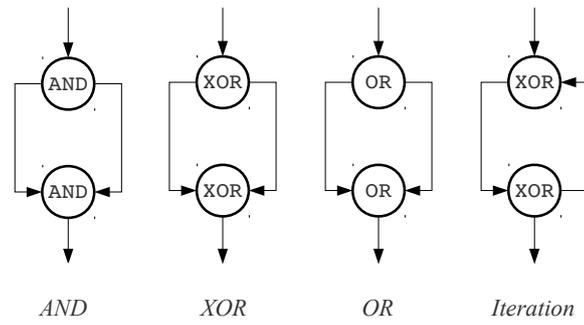


Figure A.2: Well-structured EPC constructs [67]

Although, the application of well-structured constructs is not mandatory to achieve correct EPCs, non-well-structured constructs are not desirable, as they are potential sources of errors [250].

DETAILED RESULTS OF THE COMPONENT ANALYSIS OF SOA GOVERNANCE APPROACHES

THIS supplementary chapter provides further details concerning the results of the structural analysis of SOA Governance performed in Chapter 3 and the survey of Service Life Cycles performed in Chapter 4.

B.1 Components – Selected Results in Detail

This section provides information on the analysis of selected components. We provide further details for the components *Organisational Structure* (cf. App. B.1.1), Roles and Responsibilities (cf. App. B.1.2), and *SOA Procedure Model* (cf. App. B.1.3).

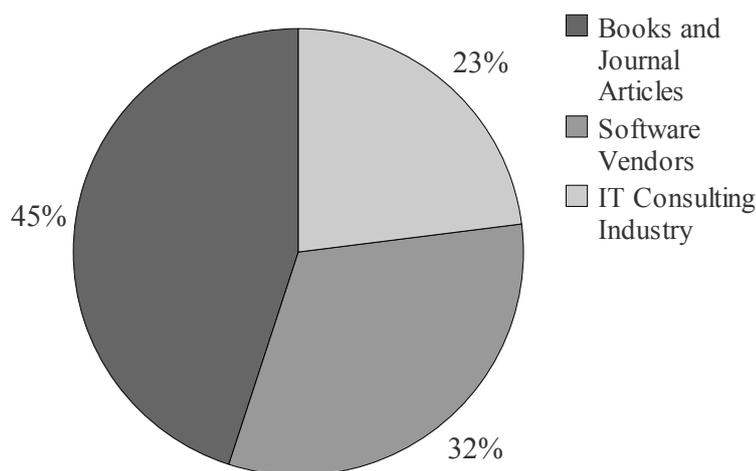


Figure B.1: Approaches per author group

In the analysis, approaches authored in three different author groups have been investigated: books and journal articles, publications sponsored by software vendors, and authored in the IT consulting industry (cf. Fig. B.1).

B.1.1 Organisational Structure

Table B.1 provides an overview of mentioned organisational entities and their consideration by selected approaches.

Table B.1: Organisational entities (group roles) of SOA Governance approaches

<i>Approach</i>	SOA Governance Board	SOA Board	SOA Centre of Excellence (CoE)	Service Development Team	Application Development Team	IT Operations
Schepers et al. [217]			•			
Bernhardt and Seese [15]		•	•			
Derler and Weinreich [43]				•	•	
Kohnke et al. [116]			•			
Bieberstein et al. [17, 18]		•	•	•		•
Marks and Bell [138]	•		•			
Brown et al. [27]	•	•	•			
Schelp and Stutz [215]		•	•			
Rieger and Bruns [210]		•	•	•		•
Hewlett-Packard [74, 75]	•		•			
Software AG [233, 234]	•		•			
Oracle [1]			•			
The Open Group [248]	•	•	•	•	•	•
Everware-CBDI Allen [3]	•		•			
Bearing Point [198]	•					
ZapThink [20]			•			
Windley [265, 266, 267]	•		•			
Berlecon [195]		•				

B.1.2 Roles and Responsibilities

Two further roles are outlined here: the *tester* and the *administrator*.

TESTER. Testers assess the general functionality and meets functional and non-functional requirements of a service before their putting into operation. In particular, services are to be tested concerning the interaction with other services, whether they meet their functionality requirements in the service infrastructure [17, 20, 27, 210, 248].

ADMINISTRATOR. Administrators manage the technical layer of an SOA system. They operate the server and infrastructure programmes such as the Enterprise Service Buses, the application servers, or databases. The supervision of the operation of SOA Governance-related systems such as registries, repositories, or tools for policy checks are also part of their duties. In large organisations, these roles are split up

onto more specific competencies such as system administrator or database administrator [17, 43, 210, 233, 234, 248].

B.1.3 SOA Procedure Models

In this section we provide descriptions of all SOA Procedure Models (cf. Sect. 3.1.6) that were suggested as a *founded recommendation*.

Schepers

Schepers et al. [217] define an SOA Governance Model that is based on a life cycle, consisting of six stages. Its main task is to ensure a consistent realisation path from strategic considerations (SOA strategy development) to the adoption and implementation of the architecture (including, i.e. Service Level Management). In the first step, “Define a SOA Strategy”, business goals are translated into abstract SOA goals. Using the integrated maturity model, the outline of the SOA program is determined, and a role model is defined. In the next step, “Align organization to SOA”, special organisational entities are established, service owners and service categories are defined. The third step, “Manage service portfolio” addresses the identification and prioritisation of services and the definition of the corresponding portfolio. Once services are defined and implemented, in the fourth phase “Control service life cycle”, their development throughout their life cycle is monitored. As setup for the service life cycle, fundamental principles are to be defined such as service granularity, and service integration into the SOA, the definition of change procedures, change impact analysis procedures, as well as the deployment of service registries and repositories for service and metadata administration. Step five, “Incorporate policy enforcement” deals with the enforcement of guidelines and policies, both in service development and service operation. Policies are stored in a repository and can in some cases be enforced automatically. In the last step, “Service level management”, SLAs are defined for every service, that must be approved by the service user, covering quality of service (i.e. performance, reachability) as well as cost. Based on these agreements, service benefit and cost are evaluated. This evaluation, as well as further feedback information from any of the steps, can result in the development of new services, or in service change or abolishment and hence is input for the cycle’s first phase.

Marks and Bell

The SOA life cycle defined by Marks and Bell [138] is composed of three steps. In step 1, the high-level organisation, governance processes, ownership of services, budgeting, and funding models are determined. In the second step, policies for the service life cycle (design, building, and the operation of services) are defined (also called “rules of engagement”). The third step is the implementation and integration of SOA. This comprises policy enforcement techniques, integration of services management, messaging platforms, service registries and metadata repositories, development tools, and security solutions.

IBM

The SOA Governance life cycle proposed by IBM is outlined in several contributions [26, 27, 81, 140, 154] (and detailed in the last one). It consists of four iterative phases: “Plan”, “Define”, “Enable”, and “Measure”. In the “Plan” phase, the construction of the SOA Governance model starts, covering the analysis of the current situation, the current maturity, existing organisational structure and governance methods, and the utilised tools. In this phase, an SOA vision and strategy is determined, and principles for business and technical parts of SOA are defined. Implementation processes are formulated and prioritised, as well as the effort for these intentions is estimated. In the second phase “Define”, the governance model is refined and built, based on information concerning principles, policies, procedures, goals, and vision currently existing, researched and investigated in the Plan phase. The CoE is being founded and governance methods and guidelines are defined and linked to metrics. In a “transition roadmap”, these methods are planned and chronologically organised. Third, in the “Enable” phase, the defined solution is deployed. Roles and decision rights are assigned, and the metrics collection and reporting mechanisms are installed. In the last phase, “Measure”, the defined metrics are captured and reported, and it is determined how well the processes, policies, and mechanisms installed meet SOA requirements. This phase of continuous improvement is being performed continually – the resulting data serves as input for the next cycle iteration.

Software AG

At Software AG [233], authors define an SOA roadmap consisting of three “typical SOA strategy phases”, using the same basis as “for common SOA maturity models”. Phase 1 “Modernizing Production Systems” targets the reuse of existing systems, such as mainframe applications, the definition of service interfaces and encapsulation of productive applications, as well as the combination of these with further services to create initial SOA applications. In phase 2, “Loosely Coupled Discoverable Services”, new roles (e.g., service architect or service designer) and tools (e.g. service registries and repositories) are introduced to manage services, service interfaces, and further SOA artefacts. In Phase 3, “Composite Applications and Business Processes”, service usage is improved concerning composition time, and service integration into applications. New applications are formed by service assembly, not development. The goal is to realise an enterprise-wide SOA.

The Open Group

The procedure model outlined by The Open Group [248], called *SOA Governance Vitality Method (SGVM)*, is a cycle consisting of four phases: “Plan”, “Define”, “Implement” and “Monitor”. In the first phase, “Plan”, the current state of the SOA and SOA Governance is analysed and compared with reference architectures. Governance principles are defined and the vision and extend of the governance approach is planned. In the next phase, “Define”, a governance roadmap is created, that identifies involved processes and roles and determines concrete methods required. These are implemented in the “Implement” phase. In the “Monitor” phase, the SOA system and the governance is controlled by defined metrics and governance methods are reviewed and eventually changed during a new iteration of the life cycle.

Oracle

Afshar [1] defines “Six steps to successful SOA Governance”. In step 1, goals, strategy and constraints are defined. In step 2, policies and procedures are defined. Step 3 defines metrics, while in step 4, governance mechanisms are installed. Step 5 addresses the analysis and improvement of processes. Step 6 represents the evaluation of the system, a refinement of policies and procedures in order to proceed with step 1 on the next level of maturity (cf. Sect. 3.1.10).

Bearing Point

The roadmap by BearingPoint [198] for implementing SOA Governance consists of four steps. An initial assessment of business and IT environment is performed, interviewing stakeholders, and formulating a consensus for a preliminary governance strategy. Using this strategy, in the second step, tools are selected, governance policies and processes are developed, and new organisational structures are set up. Stakeholders are continuously consulted, quality assurance and regular audits are performed. As third step, the accountable board performs the implementation of the SOA Governance by monitoring and controlling the life cycles of services and policies and performing policy enforcement. Finally, the overall effectiveness of the SOA system is measured, using predefined set of metrics and measurements, reporting the results, and generating recommendations for improvement.

Bieberstein

Bieberstein et al. [17] provide a three step procedure for the introduction of SOA Governance. Step one comprises the introduction of fundamental governance functions (e.g., so-called guiding principles), a roadmap, and an initial management process, a first estimation of the SOA, and experimenting and collecting of experiences. In step two, these experiences are integrated in a refined governance model, governance methods are introduced, and competencies are set up. In step three, the achieved competencies and experiences are transferred to the persons involved in the SOA operation.

Windley

The procedure model by Windley [267] consists of consolidated recommendations, describing the ordering and type of processes of forming SOA governance. First, the vision concerning what is expected to be achieved by governance mechanisms is to be formulated by all involved persons. The implementation of policies starts with the introduction of a “interoperability framework”, defining technical standards for the implementation. Further policies are defined, concerning service operation. It is important to monitor effectiveness and acceptance of these policies continuously and revise them if required. The supporting infrastructure (e.g., registries and repositories) are to be introduced timely in order to have the developers get used to them and to avoid later additional service changes.

Bernhardt and Seese

Bernhardt and Seese [15] proceed from an SOA life cycle that is composed of the phases *service proposition, design, implementation, provisioning, consumption, and management*. Based

on the OASIS SOA Reference Model [133] they define 41 different governance policies that provide the central aspect of their approach. Policies are to ensure that all SOA life cycle activities are executed in a manner that sustains the company's objectives.

Further approaches (Proposals)

At webMethods [261], authors define an SOA Governance life cycle, that is similar to the service life cycle. It consists of design time, runtime, and change time governance. At BEA Systems [10], SOA Governance is defined as service life cycle governance. They define 6 service life cycle steps that are enhanced by governance duties.

According to Schelp and Stutz, the SOA roadmap is considered to be the implementation plan for the SOA strategy. It serves as transparent proof for the benefit of the implementation. An iterative SOA life cycle serves the introduction of new SOA processes and the adjustment of existing architecture processes. [215]

B.2 The SOA Forum Survey on SOA Governance

WebLayers published a survey on the perception of SOA Governance among "The SOA Forum" industry consortium. The consortium comprises 1300 companies being "large enterprises and Government Agencies that reflect both early adopters and mature implementations of SOA" [260]. Selected results are outlined in Figures B.2, B.3, and B.4 below.

What level of risk do you feel your organization has by putting services in production that are not effectively governed?

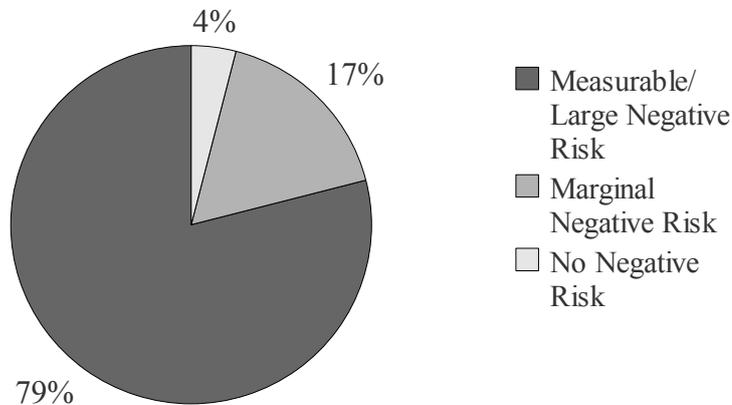


Figure B.2: Perceived level of risk of lacking governance [260]

B.3 Definitions for SOA Governance – Details

In the following, the definitions are sorted by the main aspects *processes, policies, organisational structures, and business goals*. Table B.2 provides a structured overview.

Is your current SOA Governance approach sufficient?

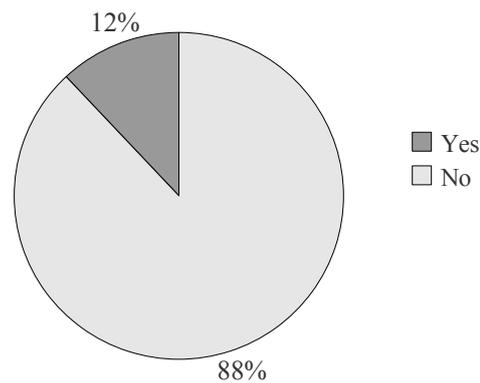


Figure B.3: Sufficiency of SOA Governance approach [260]

What policy enforcement methods are employed in your organization?

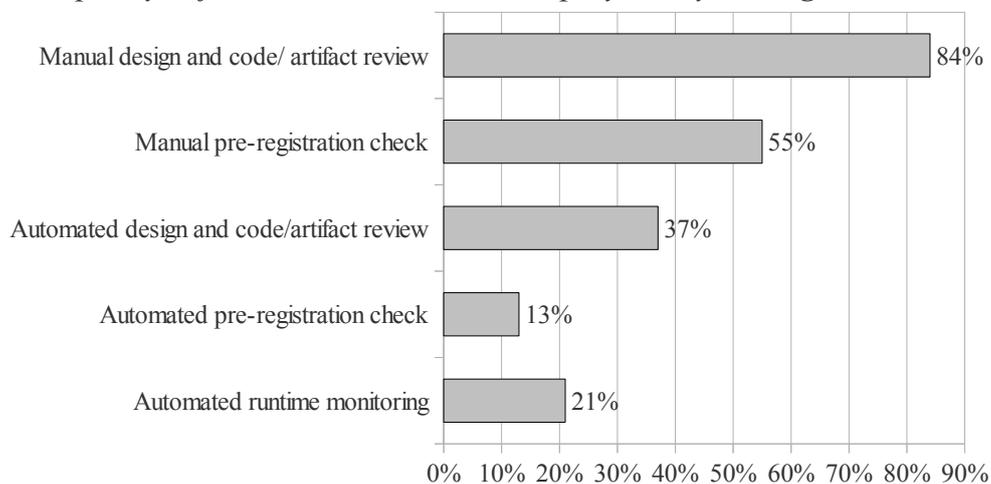


Figure B.4: Employed policy enforcement in organisation [260]

Processes

Some definitions emphasize the integration of processes. Authors at Software AG [233] provide the following definition:

“[SOA Governance] defines the decision-making authority for developing and/or modifying SOA artefacts; and it has both a strategy and a life cycle. In addition, it encompasses people (i.e., roles), technologies (i.e. tools) and processes (i.e., production) – further emphasizing the far-reaching effects SOA Governance has on the organization.”

– Software AG [233]

“SOA Governance refers to the organization, processes, policies and metrics required to manage an SOA successfully.”

– Marks and Bell [138]

Table B.2: Core aspects of the definitions

<i>Definition</i>	Business Goals and Objectives	Organisational Structures, SOA Management	Policies, Guidelines	Processes, Practices	Metrics, Controls	Roles, Decision Rights	Reference to IT Governance	Technical Aspects	Compliance, Regulations
Bernhardt and Seese [15]	•	•	•	•					•
Bieberstein et al. [17, 18]	•	•	•	•	•	•			
Marks and Bell [138]	•	•	•	•	•				
Schelp and Stutz [215]	•			•			•		
Rieger and Bruns [210]		•	•					•	
Hewlett-Packard [74, 75]								•	
Brauer and Kline [25]			•	•				•	
Software AG [233, 234]				•		•			
Oracle [1]	•		•	•		•			
The Open Group [248]							•		
Everware-CBDI Allen [3]	•	•	•	•			•		
BearingPoint [198]	•		•	•	•			•	
ZapThink [20]							•		
Anil John (via [116, 267])			•	•		•	•		
Fabini [57]		•							
Keller [109]	•								
Our definition (cf. Sect. 3.2)	•	•	•	•	•	•	•	•	•

“[...] SOA Governance should be considered an extension of existing IT Governance that deals with the decision rights, processes and policies that are put into place to encourage the adoption and operation of an SOA that may cross ownership boundaries.”

– Anil John (via Windley [266, 267] and Kohnke et al. [116])

Policies

For the majority of authors, *policies* are an important part of an governance approach for SOA.

“A SOA Governance in general represents a organisational and technical framework with guidelines and policies, that manadatorily determine, how services are to be developed and operated companywide.”¹

– Rieger and Bruns [210]

“SOA Governance is a set of solutions, policies, and practices which enable companies to implement and manage an enterprise SOA.”

– Brauer and Kline [25]

“[...] SOA Governance can be defined as the interaction between policies (what), decision-makers (who), and processes (how) in order to ensure SOA success.”

– Oracle (Afshar) [1]

Organisational Aspects

According to Everware-CBDI Allen [3],

“[SOA Governance is] the part of IT Governance that refers to the organizational structures, policies and processes that ensure that an organization’s SOA efforts sustain and extend the organization’s business and IT strategies, and achieve the desired outcomes.”

Fabini [57] emphasises that

“SOA Governance is a management structure including creational and administrative elements.”²

Business Goals, SOA Success

Many definitions emphasise the alignment with the *company’s goals and objectives*. According to Keller [109],

“SOA Governance is about creating conditions that allow an SOA to grow in a company.”³

Authors at BearingPoint [198] define:

“SOA governance is the discipline of making all SOA programs within an enterprise consistent and aligned with holistic business goals and objectives through a well structured set of top-down policies, procedures and controls.”

– Rane and Lomow [198]

Schelp and Stutz [215] describe the term as follows:

1 translated from German

2 translated from German

3 translated from German

“Thus SOA Governance is defined as evolutionary enhancements or specialisation of IT Governance [...], by determining adequate mechanisms for decision support [...], that account for SOA holistically supporting the achievement of specified company objectives.”⁴

Bieberstein et al. [17] provide the most comprehensive, quite generic definition:

“Governance provides an overarching structure to prioritize and then support the enterprise business objectives on a strategic, functional, and operational level. The governance model defines ‘what to do’, ‘how to do it’, ‘who should do it’, and ‘how it should be measured’. It defines the rules, processes, metrics, and organizational constructs needed for effective planning, decision making, steering, and control of the SOA engagement to meet the enterprise business needs and challenging targets.”

Bernhardt and Seese [15] provide the following description:

“SOA Governance consists of the organizational structures, processes, and policies an organization puts in place to ensure that the adoption, implementation, and operation of SOA in an organization is done in accordance with best practices, architectural principles, government regulations, and laws, that is, in a manner that sustains and extends the organization’s strategies and objectives.”

Further Definitions

“[SOA Governance is] the application of SOA to IT Governance”

– Bloomberg [20]

“SOA Governance is about managing the quality, consistency, predictability, change and interdependencies of services.”

– Hewlett-Packard and Systinet [75]

“SOA Governance should be viewed as the application of Corporate Governance, IT Governance and EA Governance to Service Oriented Architecture.”

– The Open Group [248]

Concluding, it is remarkable that only few approaches mention the relationship to IT Governance, and only one approach explicitly integrates the aspect of compliance.

B.4 Selected Approaches in Detail

B.4.1 Brauer and Kline

Brauer and Kline [25] (at HP Labs and Systinet) see SOA Governance in the context of the life cycle of business services. They define two key infrastructure solutions supporting SOA Governance: the business service registry and business service management.

⁴ translated from German

As main instruments of their SOA Governance approach, the authors mention the business service registry and business service management. They define a five-stages-service life cycle and a detailed SOA roadmap that shows elements of a maturity model. In general, their approach addresses service security, service auditing, service level compliance (SLA Monitoring), and service life cycle management. The policy catalogue defined addresses standards compliance, SLA specification, service configuration and security-related issues. The authors state that a governance model should focus on “people, processes, and technology”. However, they do not clearly specify what is meant by “people” and “processes”. Summarising, this approach addresses SOA Governance almost exclusively on the technical level, while, however, lacking a detailed description of governance methods.

B.4.2 *Bieberstein et al.*

Bieberstein et al. [17] propose an SOA Governance Model. They identify six governance processes and three steps for launching the SOA Governance Model, combined with an SOA roadmap. The SOA strategy and SOA objectives should be defined in a way that both business and IT units have a clear understanding of them. According to them, policies, defined by governance positions, form the basis for any decisions. Their model is completed by a set of best practices.

In a further publication, Bieberstein et al. [18] describe an approach to guide an SOA successfully, emphasising transformation of organizational structures and behavioural practices. They propose the Human Services Bus (HSB) as a new organizational institution, streamlining cross-department processes, thus optimally exploiting the SOA approach. Compared to others, their approach lacks a maturity model, metrics, an SOA life cycle, a service life cycle, and policy enforcement techniques.

B.4.3 *WebMethods*

The SOA Governance approach at WebMethods [261] consists of two parts: *Architecture Governance* and *Service Life Cycle Governance*. Architecture Governance comprises issues such as corporate technology standards, the definition of an SOA topology and determination of an SOA platform strategy. Service Life Cycle Governance is divided into design time, runtime, and change time governance, and focuses on the regulation of service design through corresponding policies, and three different types of enforcement mechanisms. Additionally, they mention organisational changes and define an SOA life cycle. Further techniques such as maturity models, metrics, or governance processes are not part of the approach.

B.4.4 *Software AG*

The approach by Software AG [233, 234] identifies maturity and governance levels. Besides this six-level-maturity model they define an SOA service life cycle, incorporating services, related artefacts and roles. They provide a five-step SOA adaptation plan as well as a set of best practices. However, the governance processes are not explicitly defined. A policy framework, based on best practices, is used in order to ensure the successful

long-time operation of an SOA. They consider new roles as well as a new governance team necessary. An SOA life cycle, a metrics model, impact on employees' behaviour, and policy enforcement techniques are not explicitly included.

B.4.5 *BEA Systems*

The approach of BEA Systems [10] emphasises the importance of a service life cycle for SOA Governance as most critical requirement of a successful SOA Governance approach. Central policy definition and enforcement regulating the design, building, provisioning, and operation of services affect the whole SOA system referring to quality insurance, monitoring, and SLA management. The primary goals are reduction of development costs and faster "time-to-service". They define a service life cycle with six phases. It is the task of a central policy definition and enforcement authority to regulate the design, building, provisioning, and operation of services. Main goals are quality insurance, monitoring, and SLA management inside the SOA system.

B.4.6 *Oracle*

The SOA Governance approach proposed at Oracle [1] consists of nine "key areas of interest", combined with a structured set of best practices. It is completed by an SOA adaptation model defining a cycle of six steps that supports continuous improvement of the SOA.

The approach at Oracle is characterised by a policy framework. The author, Afshar [1], considers governance policies to be the central tool of every governance approach. Eight policy domains define the decision fields and topics that have to be managed and controlled by policy enactment. These cover architecture, technology, information, financial, portfolios, people, project execution, and operational, each of them complemented by a concrete list of best practices. In particular, they define new roles and responsibilities in the domain people and demand a new organisational entity as well as the concrete definition of incentives in order to have impact on employees' behaviour. As one policy category, under project execution, they define service life cycle governance formulating the main stages of such a cycle. Additionally, Afshar [1] describes an SOA maturity model consisting of six steps and supporting continuous improvement of the SOA. Concluding, the author presents a comprehensive governance policy framework covering a large number of aspects or problem fields of an SOA system.

B.4.7 *IBM*

Authors at IBM define SOA Governance as extension of IT Governance that focuses on the service life cycle and composite applications. The IBM SOA Governance model comprises a service life cycle and an SOA Governance life cycle, both congruently consisting of four phases [26, 81, 271].

The approach focusses on service life cycle management, decision rights, policies and measures. The four phases of the life cycles are: plan, define, enable, measure, and model, assemble, deploy, manage, respectively. These mutually congruent life cycle phases form

the core of IBM's approach and are based on best practices. Among others, organisational changes, employee training, and, implicitly, new SOA roles are included.

B.4.8 *Marks and Bell*

Marks and Bell [138] introduce an SOA Governance Framework. SOA Governance, according to them, consists of three basic steps. The setup of an overarching governance model determining fundamental principles as high-level organisation, services ownership and funding issues is the initial step. Second, service-related basic policies concerning, e.g. designing, building and operation are created. The third step consists of the implementation and integration of the actual SOA Governance.

Marks and Bell [138] define an SOA Governance framework identify organisation, SOA processes, policies, metrics, and behaviour as crucial to success. They define policies in six different domains, based on best practices: enterprise, business, process, compliance, technology standards, and security policies. They propose policy enforcement models and define new roles for several new tasks being introduced along with the SOA. The proposed SOA life cycle consists of design-time, publishing and discovery, and run-time governance aspects. A SOA roadmap and a service life cycle are mentioned, but not specified in detail. The only technique not considered by Marks and Bell is an SOA maturity model. This concept is one of the most comprehensive approaches considered by the analysis.

B.4.9 *Schelp and Stutz*

According to Schelp and Stutz [215], an SOA Governance model is composed of a set of management activities combined with organisational structures based on governance principles. The activities comprise three groups: implementation, management and control of an SOA. The components of organisational structure are SOA strategy, SOA organisational structure and SOA operational structure. These two aspects determine the way to best govern an SOA.

B.4.10 *Paul Allen*

A recent proposal for an SOA Governance Framework was made by Allen [3] (Everware-CBDI). He defines an SOA Governance Framework that consists of five views: an organisational view, a process view, a policy view, an infrastructure view, and a maturity view. The first view defines organisational structures, roles and responsibilities that are needed by SOA Governance. The process view describes management processes at the one hand and operational processes at the other. In the policy view, several types of governance policies are described. The infrastructure view provides the technical means to support governance, e.g. by policy enforcement, or change management. In the maturity view, maturity assessment for the first four views is provided. Best practices are only partially considered, as part of the service life cycle. Allen defines a task "communication" as part of the infrastructure view that covers impact on behaviour.

B.4.11 *Weill and Ross*

Weill and Ross [263] identify six interacting components for effective design of IT Governance. Their main focus lies on the intention to influence behaviour by IT metrics and accountabilities. The goal is to create target-oriented incentives in order to evoke specific desirable behaviour. The framework, however, does not allow for SOA characteristics.

B.5 *Service Life Cycle Survey – Details*

Design and Development

SERVICE CATEGORISATION In order to organise the service portfolio properly, services are sorted according to criteria such as the level of the technical department or business area, that a service has been defined for. Many approaches use a *service taxonomy* [15, 138, 198, 233]. It consists of determined categories and subcategories, that a service is classified into. As part of governance, it is important to define a taxonomy, that fits the number of services in the SOA system concerning their granularity. Further approaches define service domains as categories [20, 217].

DESIGN TIME POLICIES Design time policies define the manner services are designed, the technical standards to use, and the architectural requirements to meet. According to Brown et al. [27], they affect the design patterns used during the realisation of a service. The limitation of choice of technical standards and interface formats ensures the interoperability, i.e., the seamless service interaction [1]. It is recommended to restrict interface design to secure and encrypted communication [195], as well as to make use of accepted standards as WSDL, SOAP, and REST, rather than use proprietary solutions of software vendors, avoiding a “buy-in” effect [20]. This ensures flexibility when choosing a platform, as well as when deciding to provide services externally outside the company’s network. Additionally to enacting adequate architectural and design guidelines, processes are to be defined that regulate frequently occurring procedures, e.g., the exception request. [1, 20, 27, 74, 138, 195, 198, 210, 215, 217, 233, 248, 267]

DOCUMENTATION GUIDELINES By defining and enforcing documentation guidelines, SOA Governance approaches address the realisation of service reuse in different contexts. They provide the aspects and the extend/scope of service documentation. Service documentation usually comprises documentation from technical, functional, and business view. The latter integrates the service into corresponding business processes. The standard UDDI (Universal Description, Discovery and Integration) provides a well structured support, simplifying service retrieval using a database-based approach. Documentation scope is recommended to be adjusted to the maturity level of the SOA initiative and the degree of service reuse aimed at [3] [1, 15, 265].

Runtime

OPERATIONAL GUIDELINES Runtime policies comprise authorisations and commitments for persons or roles for the announcement, ownership, instantiation, and usage of services on the one hand. On the other, guidelines control and regulate service operation. Guidelines cover security requirements (e.g., the utilised encryption), access control (use

of specific authentication methods), service availability, and service performance, correct logging of events, and correct billing of services.

Additionally to the definition and enactment of runtime policies, their enforcement is also addressed by the governance approach. For the (at least partial) automated enforcement of operational guidelines, the governance approach is also in charge of establishing the needed software tools such as policy management and Web service management systems. [1, 3, 15, 43, 74, 75, 138, 198, 210, 215, 217, 233, 248]

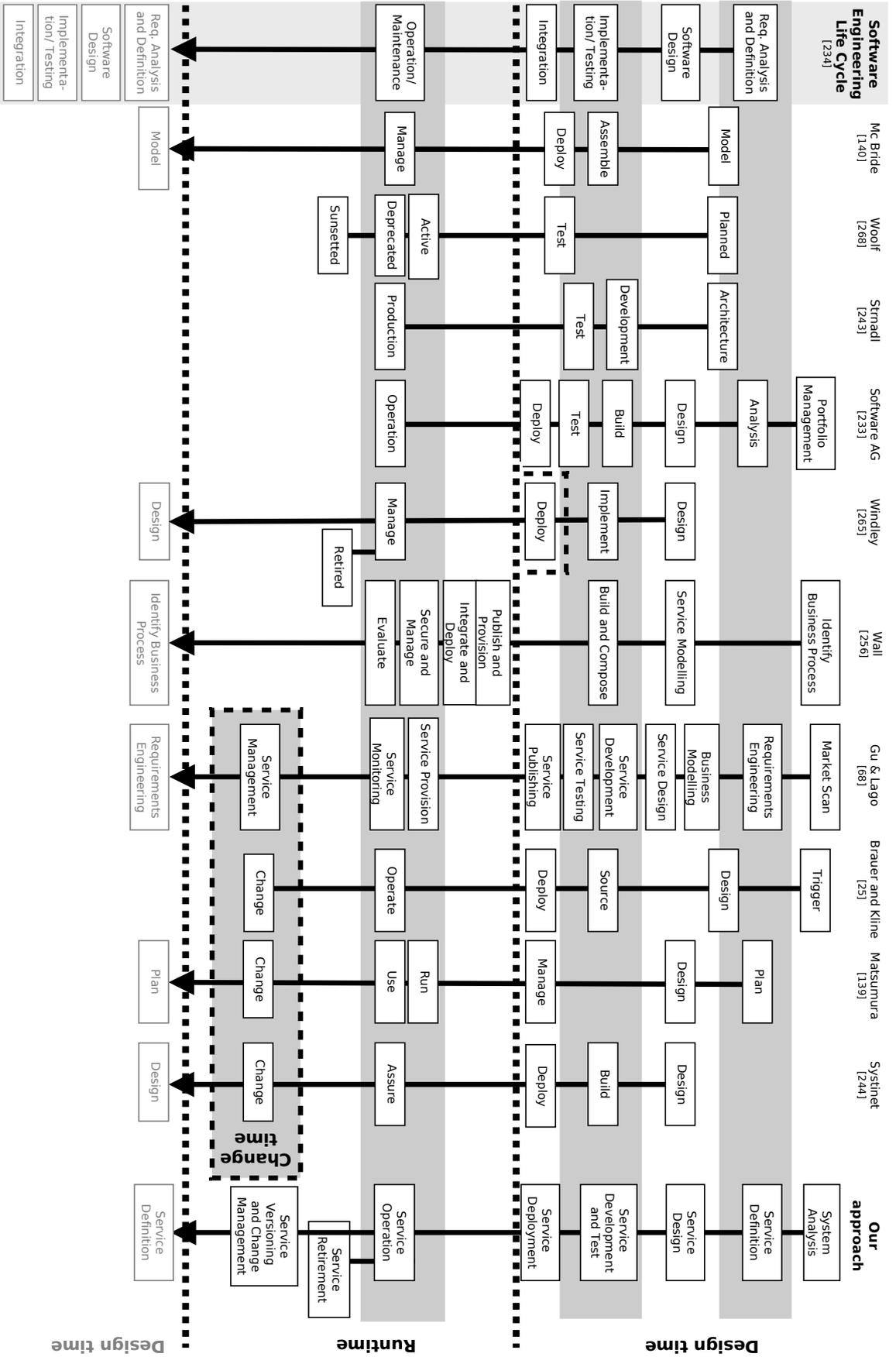


Figure B.5: Survey on Service Life Cycles (cf. Niemann et al. [163])

B.6 Components Analysis – Details

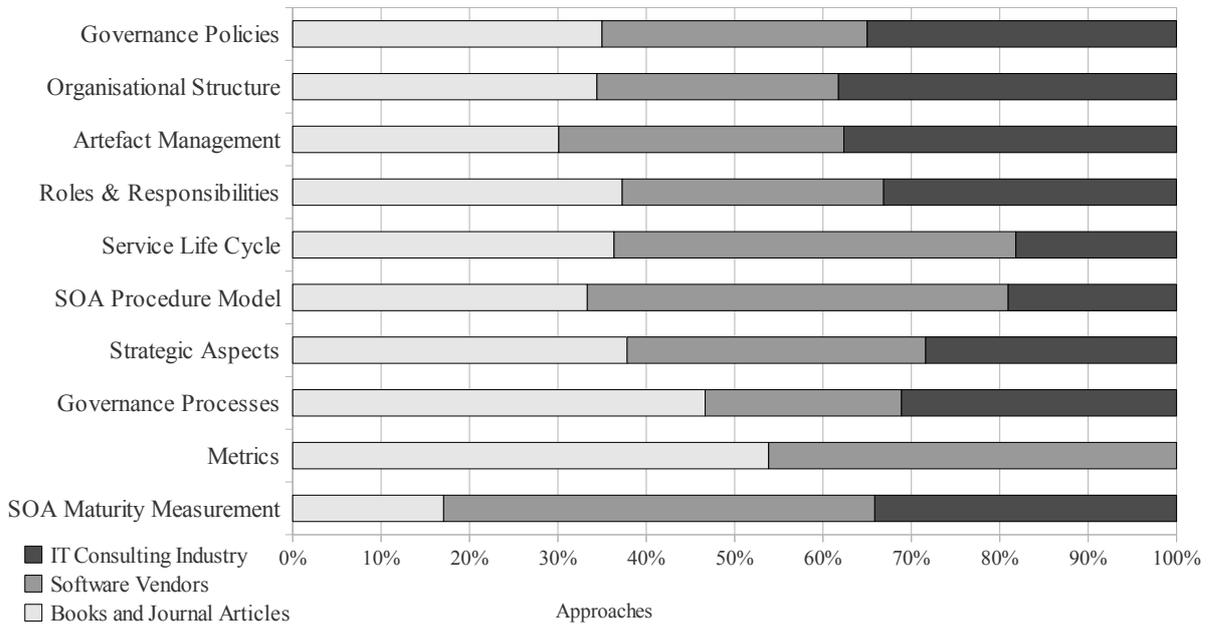


Figure B.6: Relative number of mentions per component (normalised)

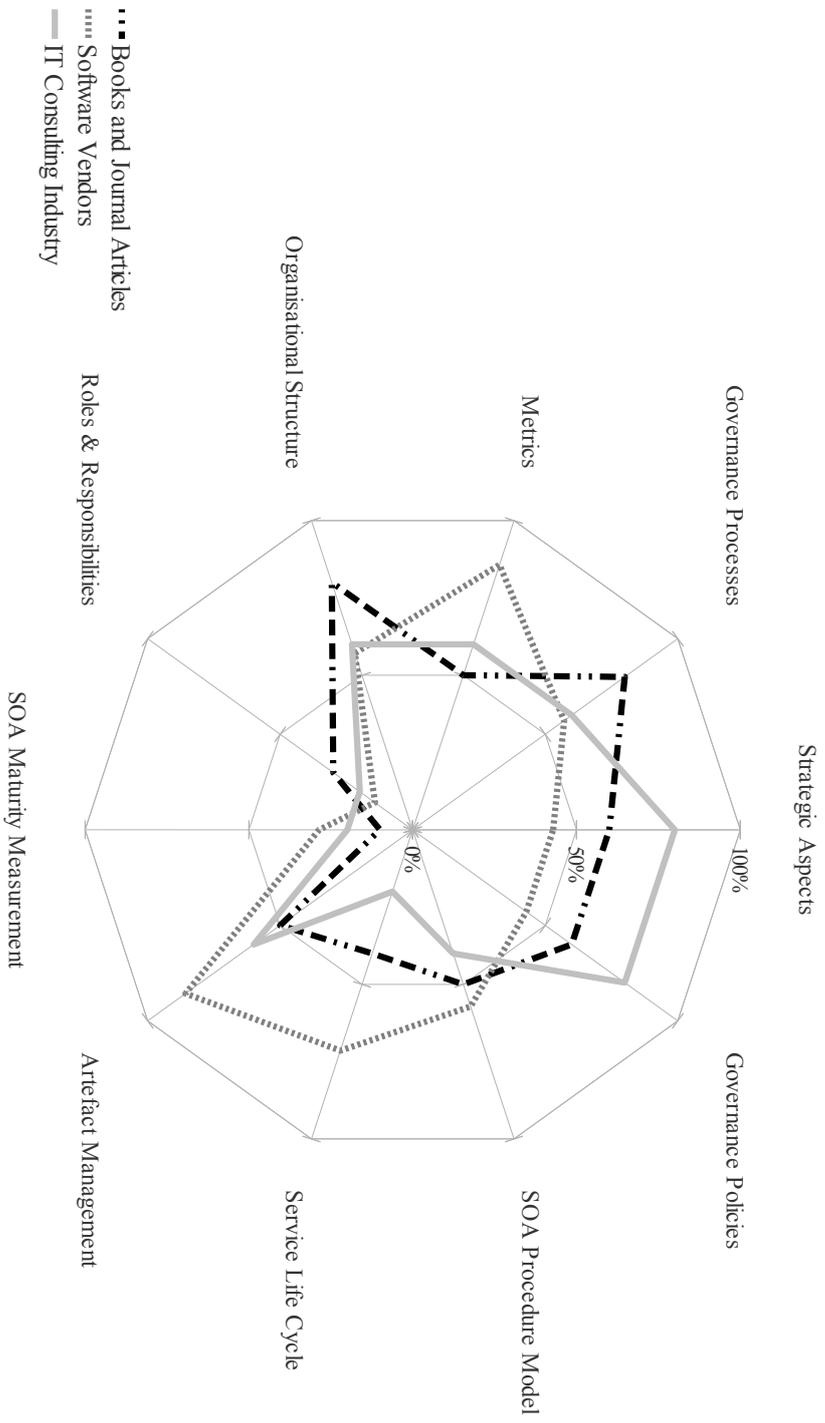


Figure B.7: Integration of components per author group and quality level (founded recommendations only)

EVALUATION DETAILS

PRIOR TO OUR evaluations outlined above, we performed a preliminary evaluation of the comparison approach using a separate test case (cf. Sect. C.1). Further, we provide detailed information on the SPRM test case in Section C.2. Section C.3 and C.4 provide result details for the evaluations of the process comparison and the process evaluation scenario, respectively.

C.1 Preliminary Evaluation of Comparison Scenario

The approach has been evaluated using a subset of the “Handels-H”¹ reference catalogue [11]. The “Handels-H” reference model describes the structure of information systems of merchandising concerns, covering the description of processes concerning all activities of the supply chain ranging from procurement to distribution as well as supporting processes (e.g. accounting). As process description language, the basic form of EPC is used.

In order to assess the quality of the matching approach, a metric for automated matching tasks suggested by Melnik et al. [142] is used, the *matching accuracy*. The metric evaluates the quality of matching algorithms which require human quality assessment. Given an approximately good matching result containing plausible match candidates, the metric will return a bad result, if the intended match candidates are not part of it (see also Sect. 5.6.1.1).

C.1.1 Evaluation Setup

Our evaluation data set consists of 50 process models, ranging from 8 to 50 elements each. It is divided into two sets. The first set contains 25 models that have been randomly selected from the process reference catalogue. For the second set, we modified these 25 process models analogously to the evaluation performed in [47] by changes in text and structure.

- The *text-based modifications* aim at renaming descriptions of functions and events in a meaningful way to account for varying terms used by different model designers. They include using synonyms, changing verbs to nouns and vice versa, and reformulation of descriptions in other words.
- *Structural modifications* consider the possibility that tasks can be performed in a different order, can be parallelised, or are divided into different units of work. They include removing a function and/or event or groups of functions (or events), adding a function and/or event, moving a function and/or event or groups of functions (or events), changing structure by parallelising sequences using AND/OR/XOR or

¹ online browsable at <http://www.semture.de/downloads/cubetto/handels-h/index.html>, last accessed 2011-08-16

vice versa, and changing the order of several elements within a sequence or structure.

We further subdivided the evaluation data into five classes. In each of these, we performed a number of structural and text-based modifications to each model, respectively. For example, for every model pair in class B, one model has been changed by applying 4-5 structural *and* 4-5 text-based modifications (cf. Tab. C.1).

Two persons annotated all process model pairs with the respective differences. The suggested matching result of two models is compared to the annotations and the number of correctly suggested correspondences is determined. Finally, the accuracy is computed for each model pair (cf. Tab. C.1).

As string-based approach, we applied the Jaccard coefficient (cf. Eq. 5.8). As semantic similarity measure we employed the distributional semantic first order measure that uses Wikipedia (cf. Eq. 5.16). We used equal weighting: $0.5 \cdot \text{sim}^{\text{Jac}} + 0.5 \cdot \text{sim}^{\text{SFO}}$. By further preliminary experiments, we found out that a threshold of 0.1 yields the best result for this metric.

C.1.2 Results

The combined approach using both string-based and semantic similarity measures performs best on average.

Table C.1: Average accuracy by modification class

	Class					avg.
	A	B	C	D	E	
Number of modifications	4-6	8-10	12-14	16-18	≥ 20	
String-based similarity	0.91	0.96	0.81	0.82	0.66	0.83
String-based and semantic sim.	0.93	0.94	0.86	0.88	0.65	0.85

In general, with increasing diversity of the models, a declining tendency of the accuracy is observed. The string-based approach is, especially for the classes C and D, improved by the consideration of the semantic similarity metric. Overall, the average accuracy is increased by 2 percentage points. The overall average savings rate is 85%, i.e., effort savings of 85% when performing internal process pre-assessments can be realised. Even savings of 65%, the smallest value measured, represents a significant savings amount – here achieved when comparing highly different models.

C.2 Test Case Details

This Section provides details on the test case SPRM that has been used for the evaluation in the process comparison (cf. Sect. 5.6.1) and process retrieval (cf. Sect. 5.6.2) scenarios.

Table C.3 provides detailed the node counts for each query model as well as the number of relevant models the test case contains for each query model.

Table C.2 outlines the domains of the test case, the number of models per domain, and the origin domain per query model.

Table C.2: Test case structure: Business domains

Business Domain (Text ID)	ID	German Title	No. of Models	Query ID
Sales and Distribution (1Ve)	27	Vertrieb	76	1, 5, 6, 7, 9
Financial Accounting (1Ex)	6	Externes Rechnungswesen	54	
Treasury (1Tr)	23	Fiskus	48	
Asset Accounting (1An)	0	Anlagenmanagement	45	8
Customer Service (1Ku)	9	Kundenservice	41	2, 3, 4, 10
Procurement (1Be)	2	Beschaffung	37	
Project Management (1Pr)	19	Projektmanagement	36	
Plant Maintenance (1In)	8	Instandhaltung	35	
Product Data Management (1Pr)	16	Produktdatenmanagement	26	
Enterprise Controlling (1Un)	24	Unternehmenscontrolling	22	
Quality Management (1Qu)	20	Qualitätsmanagement	20	
Environment, Health and Safety (1En)	4	Umwelt, Gesundheit und Sicherheit	19	
Revenue and Cost Controlling (1Er)	5	Erlös- und Kostencontrolling	19	
Compensation Management (1Ve)	26	Vergütungsmanagement	18	
Production Planning and Procurement Planning (1Pr)	18	Produktions- und Beschaffungsplanung	17	
Production (1Pr)	17	Produktion	17	
Personnel Time Management (1Pe)	15	Personalzeitwirtschaft	12	
Training and Event Management (1Ve)	25	Veranstaltungsmanagement	12	
Personnel Development (1Pe)	14	Personalentwicklung	10	
Recruitment (1Pe)	13	Personalbeschaffung	9	
Payroll (1Pe)	11	Personalabrechnung	7	
Benefits Administration (1Ar)	1	Arbeitgeberleistungsadministration	6	
Real Estate Management (1Im)	7	Immobilienmanagement	6	
Organizational Management (1Or)	10	Organisationsmanagement	5	
Personnel Administration (1Pe)	12	Personaladministration	4	
Inventory Management, Warehouse Management and Transportation (1Be)	3	Bestandsführung, Lagerverwaltung und Transport	3	
Retail (1Wa)	28	Warenwirtschaft	1	
Travel Management (1Re)	21	Reisemanagement	1	

Table C.3: Number of nodes and relevant models per query model

	Query ID										avg.
	1	2	3	4	5	6	7	8	9	10	
No. of events	3	5	5	11	5	6	41	15	16	11	11,8
No. of functions	1	3	3	5	3	3	9	4	4	5	4,0
No. of connectors	1	2	2	6	1	2	20	6	4	5	4,9
Sum (No. of items)	5	10	10	22	9	11	70	25	24	21	20,7
No. of relevant models	12	13	7	5	13	11	16	18	9	6	11,0

C.3 Evaluation of the Comparison Scenario

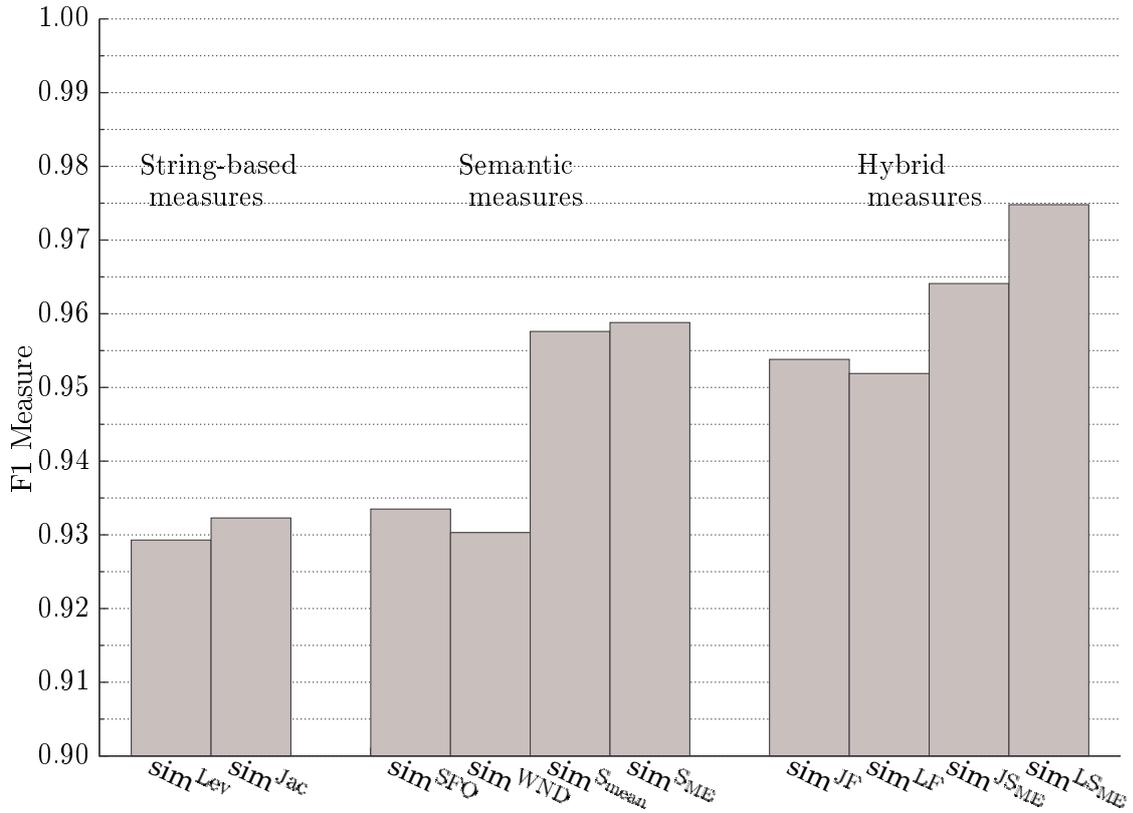


Figure C.1: Comparison scenario: F1-measure per considered similarity measure

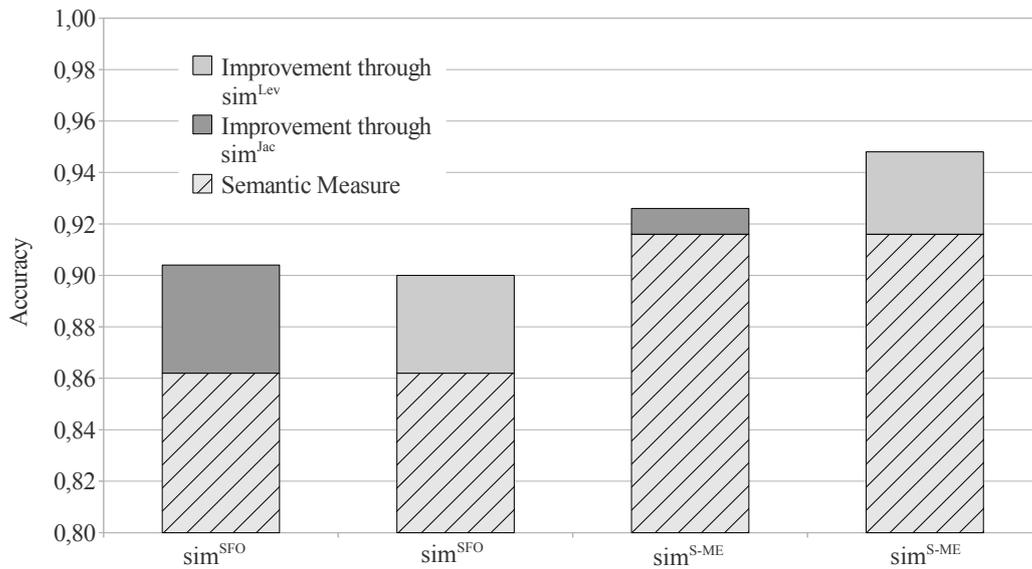


Figure C.2: Comparison scenario: Improval of semantic measures

C.4 Evaluation of the Retrieval Scenario

Table C.4: Retrieval scenario: Average precision per query model

Query ID	Lucene	Indri [48]	sim^{LSME} (node-based)	sim^{JSME}	sim^{LSME} (cluster-based)	sim^{JSME}	Label Sim. [48]
1	0.7979	0.6988	0.7998	0.7927	0.7378	0.6880	0.7568
2	0.8616	0.8687	0.9366	0.9431	0.7460	0.5559	0.7722
3	0.8757	0.8494	0.7053	0.6352	0.6257	0.5871	0.6950
4	1.0000	0.9073	0.9667	0.9667	0.9667	0.8761	0.9846
5	0.8797	0.4633	0.8116	0.8083	0.5659	0.4903	0.7625
6	0.7843	0.7722	0.9361	0.9335	0.7578	0.6913	0.8687
7	0.7781	0.5985	0.8382	0.8544	0.7578	0.7643	0.8108
8	0.4804	0.6178	0.7411	0.7626	0.6467	0.5567	0.6757
9	0.8429	0.8591	0.8732	0.8944	0.7626	0.7293	0.8687
10	1.0000	0.9749	1.0000	1.0000	1.0000	1.0000	0.8301
avg.	0.8211	0.7610	0.8609	0.8591	0.7567	0.6939	0.8025

Table C.5: Retrieval scenario: R-Precision per query model

Query ID	Lucene	sim^{LSME} (node-based)	sim^{JSME}	sim^{LSME} (cluster)
1	0.7500	0.7500	0.7500	0.5833
2	0.6923	0.7692	0.8462	0.6923
3	0.8571	0.7143	0.5714	0.7143
4	1.0000	0.8000	0.8000	0.8000
5	0.9167	0.6923	0.6923	0.6154
6	0.6364	0.9091	0.9091	0.6364
7	0.7500	0.7500	0.7500	0.7500
8	0.4444	0.6111	0.6111	0.5556
9	0.7778	0.7778	0.7778	0.6667
10	0.6667	1.0000	1.0000	1.0000
avg.	0.7491	0.7774	0.7708	0.7014

Table C.6: Retrieval scenario: precision per recall (interpolated precision-recall curve)

Recall intervall	Lucene	Indri	sim^{LSME} (node-based)	sim^{JSME}	Label Sim. [48]
[0.00 – 0.05)	1.0000	1.0000	1.0000	1.0000	1.0000
[0.05 – 0.15)	1.0000	0.9028	1.0000	1.0000	1.0000
[0.15 – 0.25)	0.9261	0.9022	1.0000	1.0000	1.0000
[0.25 – 0.35)	0.9261	0.9022	1.0000	1.0000	0.9744
[0.35 – 0.45)	0.9261	0.8519	1.0000	1.0000	0.9699
[0.45 – 0.55)	0.9261	0.8158	1.0000	1.0000	0.9557
[0.55 – 0.65)	0.8401	0.6782	0.9318	0.8919	0.7991
[0.65 – 0.75)	0.6869	0.5718	0.7908	0.7921	0.7069
[0.75 – 0.85)	0.6270	0.3147	0.6738	0.7198	0.5311
[0.85 – 0.95)	0.4274	0.2582	0.5288	0.5713	0.3179
[0.95 – 1.00]	0.4024	0.1987	0.4193	0.4220	0.1669

Table C.7: Retrieval scenario: First-5-Precision per query model

Query ID	Lucene	$\text{sim}^{\text{LS}_{\text{ME}}}$	$\text{sim}^{\text{JS}_{\text{ME}}}$
(node-based)			
1	1.0000	1.0000	1.0000
2	1.0000	1.0000	1.0000
3	1.0000	0.8000	0.6000
4	1.0000	0.8000	0.8000
5	1.0000	1.0000	1.0000
6	1.0000	1.0000	1.0000
7	1.0000	1.0000	1.0000
8	0.6000	1.0000	1.0000
9	1.0000	1.0000	1.0000
10	0.8000	1.0000	1.0000
avg.	0.9400	0.9600	0.9400

Table C.8: Retrieval scenario: First-10-Precision per query model

Query ID	Lucene	$\text{sim}^{\text{LS}_{\text{ME}}}$	$\text{sim}^{\text{JS}_{\text{ME}}}$	$\text{sim}^{\text{LS}_{\text{ME}}}$
		(node-based)		(cluster)
1	0.7000	0.7000	0.7000	0.7000
2	0.9000	1.0000	0.9000	0.7000
3	0.6000	0.6000	0.6000	0.6000
4	0.5000	0.5000	0.5000	0.5000
5	0.8000	0.8000	0.8000	0.5000
6	0.7000	1.0000	1.0000	0.6000
7	0.9000	1.0000	1.0000	0.9000
8	0.4000	0.9000	1.0000	0.9000
9	0.7000	0.8000	0.8000	0.6000
10	0.6000	0.6000	0.6000	0.6000
avg.	0.6800	0.7900	0.7900	0.6600

Table C.9: Retrieval scenario: First-20-Precision per query model

Query ID	Lucene	$\text{sim}^{\text{LS}_{\text{ME}}}$	$\text{sim}^{\text{JS}_{\text{ME}}}$	$\text{sim}^{\text{LS}_{\text{ME}}}$
		(node-based)		(cluster)
1	0.5000	0.5000	0.5000	0.4500
2	0.6000	0.6500	0.6500	0.5500
3	0.3000	0.3000	0.3000	0.3000
4	0.2500	0.2500	0.2500	0.2500
5	0.6000	0.5000	0.5500	0.4500
6	0.4500	0.5000	0.5000	0.4500
7	0.6000	0.6000	0.6000	0.6000
8	0.5000	0.5500	0.5500	0.5000
9	0.3500	0.4000	0.4000	0.3500
10	0.3000	0.3000	0.3000	0.3000
avg.	0.4450	0.4550	0.4600	0.4200

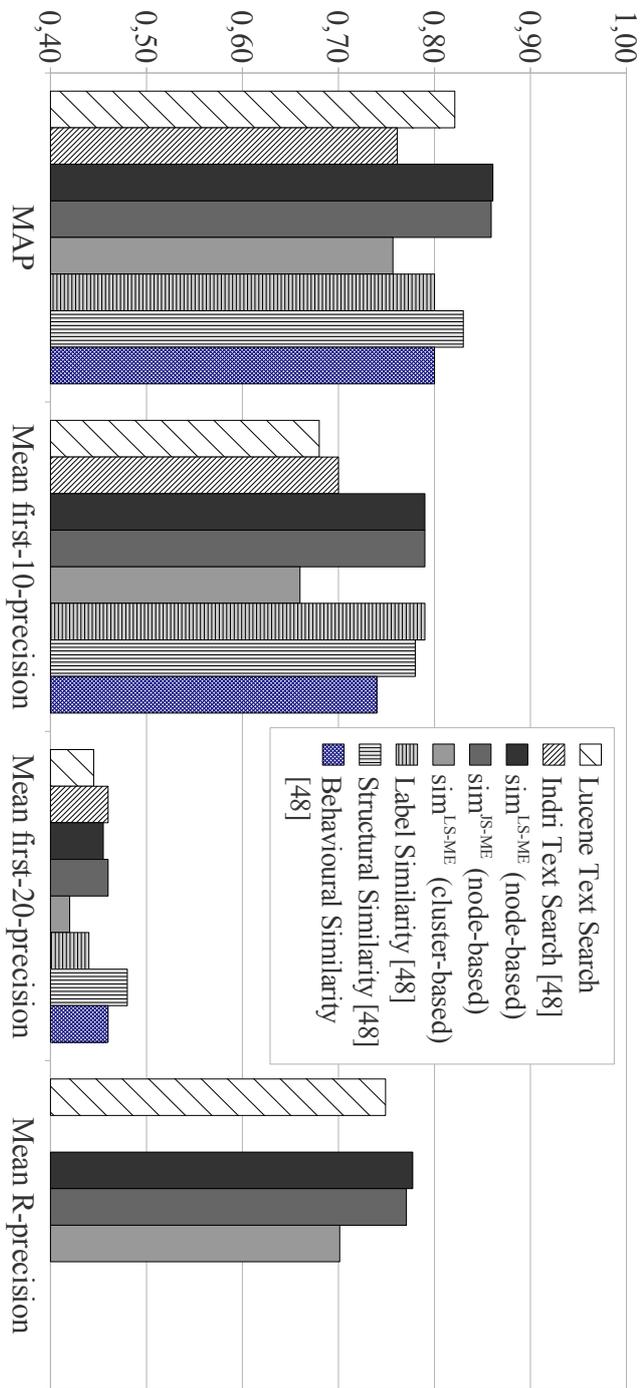


Figure C.3: Retrieval scenario: Results overview per measure

IMPLEMENTATION DETAILS

SOME DETAILS of our implementation could not be outlined in the main part. We introduce an additional feature of ProcSim.KOM called neighbourhood consideration. Further, we provide supplementary examples of the realised graphical user interface.

Neighbourhood Consideration

As optional feature of the step (A1) *determine potential match candidates per node* (cf. p. 91), a neighbourhood comparison between two matching candidate nodes can be performed. The idea is to consider the relative positions of the nodes within the graphs.

To determine the set of neighbours for a given node, a breadth first search with limited depth is performed. The neighbourhood of a node is defined as follows.

Definition 9 (d-Neighbourhood). *The neighbourhood with depth d of a node v is the set of its neighbours. A node n is a neighbour of a node v , iff there exists a path from v to n with at most d edges.*

Given two graphs, the procedure determines the neighbourhood N_v of node v in one graph and the r neighbourhoods N_{v_1}, \dots, N_{v_r} of all matching candidates $M(v)$ in the other graph. For each matching candidate $c \in M(v)$, the corresponding neighbourhoods N_{v_1} and N_{v_i} are compared which results in an aggregated value indicating the similarity of the neighbourhoods. The matching candidate c , whose neighbourhood is the most similar to the neighbourhood of the given node v , is the best matching candidate. Now, the similarity of this node pair $\text{sim}(v, c)$ obtained by string-based, semantic, or hybrid similarity metrics can be modified, for example, the value can be multiplied by a constant factor.

However, different settings in preliminary experiments did not yield better results.

Graphical User Interface

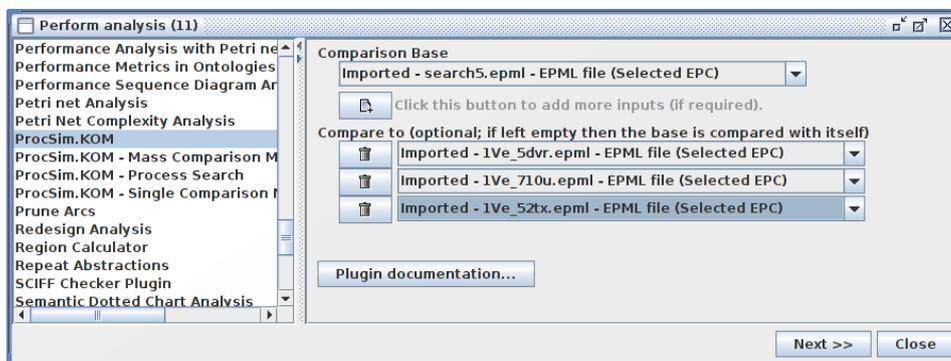


Figure D.1: ProcSim.KOM: Process model selection

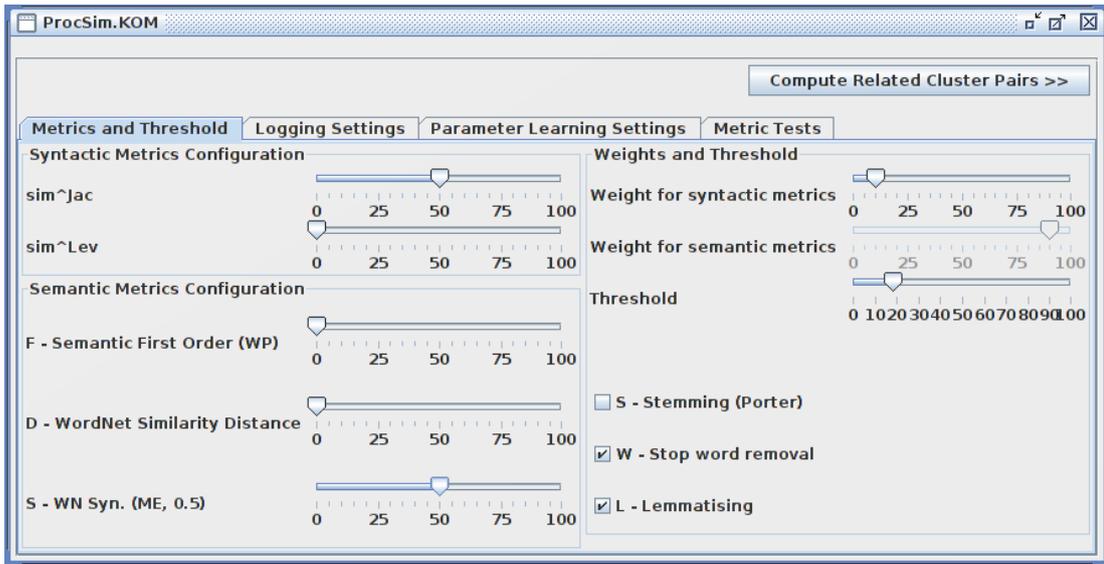


Figure D.2: ProcSim.KOM: Configuration dialog

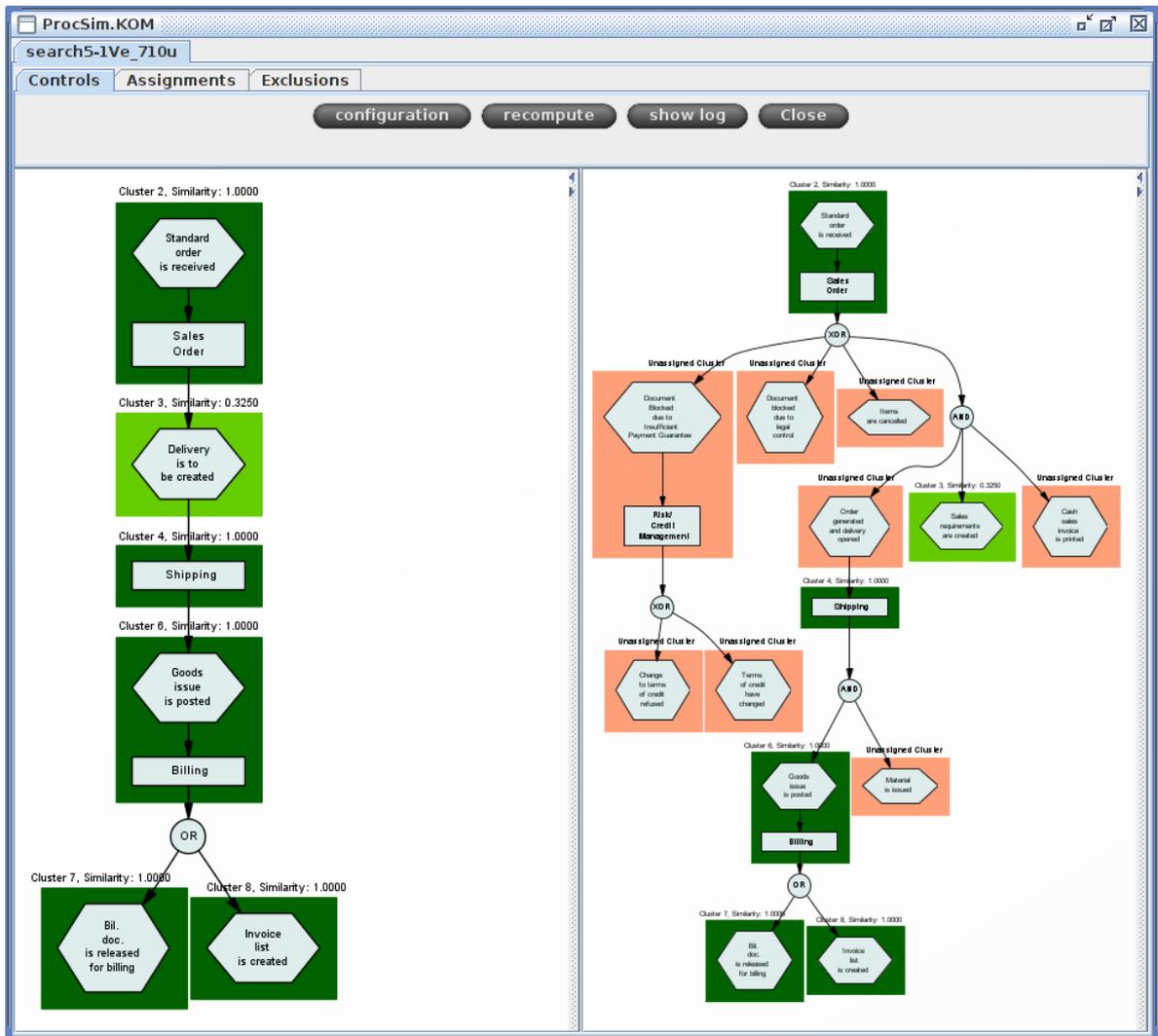


Figure D.3: ProcSim.KOM: Result visualisation (comparison scenario)

AUTHOR'S PUBLICATIONS

E.1 Main Publications

- [1] Michael Niemann, Melanie Siebenhaar, Stefan Schulte, and Ralf Steinmetz. Comparison and Retrieval of Process Models using Related Cluster Pairs. *Computers in Industry*, 63(2):168–180, February 2012.
- [2] Michael Niemann, Melanie Siebenhaar, Julian Eckert, and Ralf Steinmetz. Process Model Analysis using Related Cluster Pairs. In *Business Process Management Workshops: BPM 2010 International Workshops and Education Track*, volume 66 of *Lecture Notes in Business Information Processing*, pages 547–558. Springer-Verlag, Berlin Heidelberg, February 2011.
- [3] Michael Niemann, Julian Eckert, and Ralf Steinmetz. Semantische Analyse zur Unterstützung von SOA-Governance. In *Service Science - Neue Perspektiven für die Informatik - Proceedings der Informatik 2010*, volume 2, pages 299–304, Leipzig, Germany, September 2010.
- [4] Michael Niemann, André Miede, Wolfgang Johannsen, Nicolas Repp, and Ralf Steinmetz. Structuring SOA Governance. *International Journal of IT/Business Alignment and Governance (IJITBAG)*, 1(1):58–75, January 2010.
- [5] Marc Billeb, Achim Schäfer, Mourad Abbou, Michael Niemann, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. Einfluss regulatorischer Anforderungen im Internet of Services. *IT-Governance, Zeitschrift des ISACA Germany Chapter e.V.*, 6:8–13, October 2009.
- [6] Daniel Oberle, Nadeem Bhatti, Saartje Brockmans, Michael Niemann, and Christian Janiesch. Effektive Handhabung von Service-Informationen. *Wirtschaftsinformatik*, (5):429–452, October 2009.
- [7] Daniel Oberle, Nadeem Bhatti, Saartje Brockmans, Michael Niemann, and Christian Janiesch. Countering Service Information Challenges in the Internet of Services. *BISE - Business and Information System Engineering Journal*, 5:370–390, September 2009.
- [8] Michael Niemann, Michael Appel, Nicolas Repp, and Ralf Steinmetz. Towards a Consistent Lifecycle Model in Service Governance. In *Proceedings of the 15th Americas Conference on Information Systems (AMCIS 2009)*, San Francisco, CA, USA, August 2009.
- [9] Michael Niemann, Christian Janiesch, Nicolas Repp, and Ralf Steinmetz. Challenges of Governance Approaches for Service-Oriented Architectures. In *Proceedings of the Third International Conference on Digital Ecosystems and Technologies (DEST 2009)*, pages 634–639, Istanbul, Turkey, June 2009.

- [10] Michael Niemann, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. SOA Governance: Survey and Model. In *Proceedings of the Pre-ICIS SIGSVC (Special Interest Group Services) Workshop on Services*, Paris, France, December 2008.
- [11] Michael Niemann, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. Towards a Generic Governance Model for Service-oriented Architectures. In *Proceedings of the 14th Americas Conference on Information Systems (AMCIS 2008)*, Toronto, ON, Canada, August 2008.
- [12] Michael Niemann. Governance for Service-oriented Architectures: An Implementation Approach. In *Internet of Services (IoS) PhD Symposium at Interoperability for Enterprise Software and Applications (I-ESA)*, volume 374 of *CEUR Workshop Proceedings*, March 2008.

E.2 Further Publications

- [1] Christian Janiesch and Michael Niemann. Supporting USDL by a Governance Framework. In Alistair Barros and Daniel Oberle, editors, *Handbook of Service Description — USDL and its Methods*, pages 415–444. Springer, New York, NY, USA, March 2012.
- [2] Melanie Siebenhaar, Georg Wilhelm, Sebastian Zöller, Michael Niemann, and Dieter Schuller. Governance in emergenten Softwaresystemen – Kennzahlen und Change Management-Prozess. In *Multikonferenz Wirtschaftsinformatik 2012*, March 2012.
- [3] Christian Janiesch, Michael Niemann, and Ralf Steinmetz, editors. The TEXO Governance Framework. SAP Whitepaper, March 2011. URL http://www.internet-of-services.com/fileadmin/IOS/user_upload/pdf/TEXO_GOVERNANCE_FRAMEWORK_WHITEPAPER_1_1.pdf. last accessed 2011-08-05.
- [4] Michael Niemann, Sascha Hombach, Stefan Schulte, and Ralf Steinmetz. Das IT-Governance-Framework COBIT als Wissensdatenbank – Entwurf, Umsetzung und Evaluation einer Ontologie. Technical Report KOM-TR-2011-01, Technische Universität Darmstadt, Fachbereich Multimedia Kommunikation (KOM), January 2011.
- [5] Michael Niemann, Kim David Hagedorn, Stefan Schulte, and Ralf Steinmetz. Typische Elemente von SOA-Governance-Ansätzen – Ein Survey. Technical Report KOM-TR-2010-04, Technische Universität Darmstadt, Fachbereich Multimedia Kommunikation (KOM), December 2010.
- [6] Melanie Siebenhaar, Michael Niemann, Julian Eckert, and Ralf Steinmetz. Pro-Match.KOM: Tool Support for Process Model Analysis and Improvement. In *Demonstration Track of the Eighth International Conference on Business Process Management (BPM 2010)*, Hoboken, NJ, USA, September 2010.
- [7] Aneta Kabzeva, Michael Niemann, Paul Müller, and Ralf Steinmetz. Applying TOGAF to Define and Govern a Service-oriented Architecture in a Large-scale Research Project. In *Proceedings of the 16th Americas Conference on Information Systems (AMCIS 2010)*, Lima, Peru, August 2010.

- [8] Christian Janiesch, Michael Niemann, and Nicolas Repp. Towards a Service Governance Framework for the Internet of Services. In *Proceedings of the 17th European Conference on Information Systems (ECIS 2009)*, Verona, Italy, June 2009.
- [9] André Miede, Jean-Baptiste Behuet, Apostolos Papageorgiou, Michael Niemann, Nicolas Repp, and Ralf Steinmetz. Qualitative and Quantitative Aspects of Cooperation Mechanisms for Monitoring in Service-oriented Architectures. In *Proceedings of the Third International Conference on Digital Ecosystems and Technologies (DEST 2009)*, pages 563–568, Istanbul, Turkey, June 2009.
- [10] Nicolas Repp, Dieter Schuller, Melanie Siebenhaar, Andre Miedé, Michael Niemann, and Ralf Steinmetz. On distributed SLA Monitoring and Enforcement in Service-oriented Systems. *International Journal On Advances in Systems and Measurements*, 2(1):33–43, June 2009.
- [11] Apostolos Papageorgiou, Stefan Schulte, Dieter Schuller, Michael Niemann, Nicolas Repp, and Ralf Steinmetz. Governance of a Service-Oriented Architecture for Environmental and Public Security. In *Proceedings of the Information Technologies in Environmental Engineering (ITEE 2009) - Fourth International ICSC Symposium*, pages 39–52, Thessaloniki, Greece, May 2009.
- [12] Christian Janiesch, Michael Niemann, and Nicolas Repp. Governance in the Internet of Services: Governing Service Delivery of Service Brokers. In *Proceedings of the Pre-ICIS SIGSVC (Special Interest Group Services) Workshop on Services*, Paris, France, December 2008.
- [13] Yohei Kunichika, Harald Holz, Oleg Rostanin, and Michael Niemann. Workflow Management System and Workflow Management Method, December 2008. US Patent Application. Publication No. US 2008/0313024 A1. URL <http://ip.com/patapp/US20080313024>, last accessed 2011-08-10.
- [14] André Miede, Michael Niemann, Stefan Schulte, Julian Eckert, Aneta Kabzeva, Nicolas Repp, and Ralf Steinmetz. Selected Topics in Service Engineering and Management for Enterprise Systems. In *Proceedings of the Pre-ICIS Workshop on Enterprise Systems Research in MIS*, Paris, France, December 2008.
- [15] Nicolas Repp, André Miede, Michael Niemann, and Ralf Steinmetz. WS-Re2Policy: A policy language for distributed SLA monitoring and enforcement. In *Proceedings of the Third International Conference on Systems and Networks Communications*, pages 256–261, Sliema, Malta, October 2008.
- [16] Julian Eckert, Stefan Schulte, Michael Niemann, Nicolas Repp, and Ralf Steinmetz. Worst-Case Workflow Performance Optimization. In *Proceedings of the Third International Conference on Internet and Web Applications and Services (ICIW 2008)*, pages 632–637, Athens, Greece, June 2008.
- [17] Michael Niemann, Melanie Siebenhaar, Julian Eckert, Stefan Schulte, Nicolas Repp, and Ralf Steinmetz. SOA in Practice. Technical Report KOM-TR-2008-04, Technische Universität Darmstadt, Fachbereich Multimedia Kommunikation (KOM), June 2008.

- [18] Julian Eckert, Nicolas Repp, Michael Niemann, Marc Billeb, Achim Schäfer, and Ralf Steinmetz. IT Organization as a Limiting Factor for the Success of Service-Oriented Architectures. *EFL Quarterly*, (2):4–5, April 2008.
- [19] Yoshiro Matsui, Michael Niemann, Harald Holz, and Oleg Rostanin. Workflow Management System, Workflow Management Method, Workflow Management Program, and Recording Medium, March 2008. US Patent Application. Publication No. US 2008/0059967 A1. URL <http://ip.com/patapp/US20080059967>, last accessed 2011-08-10.
- [20] Nicolas Repp, Julian Eckert, Stefan Schulte, Michael Niemann, Rainer Berbner, and Ralf Steinmetz. Towards Automated Monitoring and Alignment of Service-based Workflows. In *Proceedings of the International Conference on Digital Ecosystems and Technologies 2008 (DEST 2008)*, Phitsanulok, Thailand, February 2008.

CURRICULUM VITÆ

Personal Information

Michael Niemann

★ 01. August 1978 in Tübingen, Baden-Württemberg

German/ Deutsch

Education

- since 06/2007 Doctoral candidate at the Department of
Electrical Engineering and Information Technology
Technische Universität Darmstadt, Germany
- 04/2006–06/2006 Diploma thesis at Ricoh R&D Group, Tokyo, Japan
in the research project “Virtual Office of the Future”
in conjunction with the German Research Centre for
Artificial Intelligence (DFKI), Kaiserslautern, Germany
- 10/1999–08/2006 Studies of Joint Degree of
Business Administration and Computer Science
Technische Universität Kaiserslautern, Germany
- 02/2003–07/2003 Studies of Business Administration, Computer Science,
and Russian language
Moscow State University (MSU), Moscow, Russia
- 09/1989–06/1998 Nordpfalzgymnasium Kirchheimbolanden, Germany,
majored in Mathematics, French, and English

Professional Experience

- Since 06/2007 Research assistant at the research group Service-oriented
Computing (SOC) at the Multimedia Communications Lab (KOM)
Technische Universität Darmstadt, Germany
- 01/2007–03/2007 Student assistant at the Knowledge Management Group
of the German Research Centre for Artificial Intelligence (DFKI),
Kaiserslautern, Germany
- 06/2004–03/2006 Student assistant at the chair of Economic Law
Technische Universität Kaiserslautern, Germany
- 06/2004–09/2005 Student assistant at the chair of National Economy
Technische Universität Kaiserslautern, Germany

- 10/2005–12/2005 Student assistant in the project “Virtual Office of the Future”
at Ricoh R&D Group, Tokyo, Japan
- 07/2003 Trainee at the Embassy of the German Federal Republic (BRD)
in Moscow, Russia

Teaching Activities

- 10/2008–12/2010 Tutor at the “Erstsemesterprojektwoche etit”
Technische Universität Darmstadt, Germany
- 2009 Lab exercise “Multimedia Communications Lab II – SOA Integration
Project”, in cooperation with Software AG Darmstadt
Technische Universität Darmstadt, Germany
- 2008, 2009 Seminar “Communication Systems and Multimedia:
Advanced Topics of Future Internet Research”
Technische Universität Darmstadt, Germany
- Since 2007 Tutor for various Bachelor- and Master theses
(and Studien- and Diplomarbeiten)
Technische Universität Darmstadt, Germany

Awards and Honors

- 03/2010 Journal Best Article Award (“Fourth Annual Excellence in Research
Journal Awards”) of the International Journal of IT/Business Align-
ment and Governance (IJITBAG) 01/2010 for:
*Michael Niemann, André Miede, Wolfgang Johannsen, Nicolas Repp, and
Ralf Steinmetz: Structuring SOA Governance.*
- 10/2008 Best Paper Award at the International Conference on Systems and
Networks Communications (ICSNC) 2008 for:
*Nicolas Repp, André Miede, Michael Niemann, Ralf Steinmetz: WS-Re2Policy:
A Policy Language for Distributed SLA Monitoring and Enforcement.*
- 08/2008 Best Paper Award at the 14th America’s Conference on Information
Systems (AMCIS) 2008 for:
*Michael Niemann, Julian Eckert, Nicolas Repp, and Ralf Steinmetz:
Towards a Generic Governance Model for Service-oriented Architectures.*

Scientific Activities

- Reviewer Informatik 2010, 2011
IEEE International Conference on Digital Ecosystems and
Technologies (IEEE DEST) 2009
Kommunikation in Verteilten Systemen (KiVS) 2009
Innovations 2008
America’s Conference on Information Systems (AMCIS) 2008, 2009,
2010

- Comm. Member Informatik 2011 – Workshop on “Governance in Verteilten Systemen (GVS)”
Informatik 2010 – Workshop on “Governance in Verteilten Systemen (GVS)”
Informatik 2009 – Workshop on “Governance in Verteilten Systemen (GVS)”
America’s Conference on Information Systems (AMCIS) 2009 – Mini-track “Enterprise Systems: Governance”
Kommunikation in Verteilten Systemen (KiVS) 2009, Workshop “Service-oriented Computing”

Memberships

- Studenteninitiative bonding e.V.
Association for Information Systems (AIS)

ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG

ICH versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe.

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, im Dezember 2011