



UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft



International Graduate School
Dynamic Intelligent Systems

Model-Based Evaluation of Service-Oriented Enterprise Architectures

**A thesis submitted to the
University of Paderborn
in partial fulfilment of the requirements for the degree of
„DOKTOR DER NATURWISSENSCHAFTEN“
(Dr. rer. nat.)**

submitted by:
Martin Assmann
Sighardstraße 46
33098 Paderborn

First supervisor: Prof. Dr. Gregor Engels
Second supervisor: Prof. Dr. Wilhelm Schäfer

Paderborn, November 2009

Abstract. Enterprise Architecture (EA) has undergone many changes since the IT has found its way into the world of enterprises. The introduction of Service-Oriented Architecture (SOA) is such a change with major consequences. The introduction of an SOA increases the flexibility and thus the productivity of an enterprise architecture, but unfortunately also its complexity. This makes the transformation of an enterprise architecture to an SOA-like enterprise architecture to a challenging and risky task. To overcome the change- and complexity-related problems when introducing SOA, Enterprise Architecture Management (EAM) systems are required. The approach of this thesis suggests a method on how to establish Enterprise Architecture Management that is especially suited for an SOA introduction. This thesis suggests a variant of an EAM system that is especially suited for the introduction of an SOA. The presented method on creating such an EAM system includes guidance on how to define a meta model for Service-Oriented Enterprise Architecture (SOEA), which is harmonized with the respective enterprise architecture. The SOA introduction is especially supported by defining SOA quality criteria and corresponding metrics. Some metrics have to be ascertained by experts. Other metrics have their measuring points within the SOEA models (instances of the SOEA meta model) and their calculation is automatable. Creating and maintaining SOEA models as well as applying the automatable metrics are supported by an eclipse-based tool. As metrics only produce measures that are hard to interpret, indicators are introduced. They allow interpreting the measures concerning the quality criteria. With the help of this EAM system, the transformation of an enterprise to a service-oriented enterprise can be planned and the level of goal-achievement (SOA-conformance of the EA) can be monitored steadily. By this, the contribution of this work aims at the reduction of the risk when introducing an SOA.

Zusammenfassung. Seit die Informationstechnologie Einzug in die Unternehmenswelt gehalten hat, werden Unternehmensarchitekturen ständig neuen Veränderungen unterworfen. Die Einführung einer Service-Orientierten Architektur (SOA) ist eine solche Veränderung mit weitreichenden Folgen. Eine SOA erhöht zwar die Flexibilität und damit auch Produktivität einer Unternehmensarchitektur, leider aber auch deren Komplexität. Dadurch wird die Transformation einer Unternehmensarchitektur zu einer Service-Orientierten Unternehmensarchitektur zu einer herausfordernden und risikobehafteten Aufgabe. Um den weitreichenden Veränderungen und der neuen Komplexität Herr zu werden, wird ein Enterprise Architecture Management (EAM) System benötigt. Diese Arbeit unterbreitet eine Variante eines EAM-Systems, das besonders für die Einführung einer SOA geeignet ist. Die hier aufgezeigte Methode zur Erschaffung eines solchen EAM Systems beinhaltet die Erzeugung eines Metamodells für eine Service-Orientierte Unternehmensarchitektur, das auf die jeweilige Unternehmensarchitektur abgestimmt wird. Die Einführung der SOA wird zudem durch SOA-Qualitätskriterien und dazu passenden Metriken unterstützt. Einige dieser Metriken müssen durch Experten ausgewertet werden. Andere Metriken haben ihre Messpunkte innerhalb der SOEA-Modelle (Instanzen des SOEA-Metamodells) und können deshalb prinzipiell automatisch ausgewertet werden. Sowohl das Anlegen und Pflegen solcher Modelle als auch die Auswertung der automatisch auswertbaren Metriken wird durch ein eclipse-basiertes Werkzeug unterstützt. Da die Resultate von Metriken nur schwer interpretierbare Messzahlen sind, werden Indikatoren eingeführt. Sie erlauben die Interpretation der Messzahlen bezüglich der Qualitätskriterien. Mit Hilfe dieses EAM-Systems kann die Transformation einer Unternehmensarchitektur zu einer service-orientierten Unternehmensarchitektur geplant und der Zielerreichungsgrad (SOA-Konformität der Unternehmensarchitektur) ständig überwacht werden. Damit zielt der Beitrag dieser Arbeit darauf ab, das Risiko eines Fehlschlags bei der Einführung einer Service-Orientierten Architektur zu verringern.

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Task Definition	5
2	Basic Concepts, Requirements and Related Work	11
2.1	Enterprise Architecture	12
2.1.1	Enterprise Architecture Definitions	12
2.1.2	Comparison of EA definitions	21
2.1.3	Choices for EA-specific Dimensions	22
2.2	Enterprise Architecture Management	25
2.2.1	Enterprise Architecture Management Definitions	25
2.2.2	Comparison of EAM definitions	33
2.2.3	Choices for EAM-specific Dimensions	36
2.3	Service-Oriented Architecture	38
2.3.1	About the Service Paradigm and SOA	38
2.3.2	Comparison of Existing Definitions	48
2.3.3	Choices for SOA-Specific Dimensions	51
2.4	Requirement Derivation	54
2.5	Evaluation of Related Work	58
3	Solution Concept	69
3.1	Designing a Solution Concept	69
3.2	Thesis Structure	76
4	Service-Oriented Enterprise Architectures	79
4.1	SOA Definition	80
4.1.1	SOA as an Evolutionary Product of Enterprise Architecture	80
4.1.2	SOA Service Definition	96
4.2	EA Definition Frame	100
5	Formalization of an SOEA Modelling Language	103
5.1	Deriving an SOA Meta Model	104
5.1.1	How to derive an SOA Meta Model?	105
5.1.2	Deriving Concepts from the Given SOA Definition	108
5.1.3	Deriving a Meta Model from Identified Concepts	112
5.2	Deriving an Enterprise Architecture Meta Model	117
5.2.1	Deriving a minimal EA meta model	117
5.2.2	Defining an individual EA meta model	120
5.3	Union of the SOA and the EA Meta Model	125
5.3.1	Transformation to Table Representation	128

5.3.2	Matching Artefacts	131
5.3.3	Joining Artefacts.....	135
5.3.4	Algorithmic Concerns.....	138
5.3.5	Application of the Merging Algorithm.....	140
6	Defining Quality Criteria and their Metrics	143
6.1	The Quality Criteria Catalogue.....	146
6.1.1	Defining Structural Quality Criteria	146
6.1.2	Defining SOA Service Quality Criteria.....	152
6.2	Metrics for Quality Criteria	158
6.2.1	Metrics for Structural Quality Criteria	160
6.2.2	Metrics for Service Criteria	184
7	Interpretation of Measures	197
7.1	Indicators for Service Orientation Metrics	197
7.1.1	Indicators for Structural Criteria.....	200
7.1.2	Indicators for Service Criteria	206
7.2	Report on Service Orientation of an EA.....	210
7.3	Recommendation of Improvements.....	212
7.3.1	Defining Remedial Actions	212
7.3.2	Strategy for Prioritizing Actions.....	213
8	Tool support with an Eclipse-Based Prototype	215
8.1	SOEA Meta Model Implementation with EMF.....	217
8.2	Creation of a Graphical Editor.....	220
8.3	Implementing Service Orientation Metrics	225
9	Summary, Conclusion and Outlook	229
9.1	Summary.....	229
9.2	Conclusion.....	230
9.3	Outlook	234
	References.....	237
	Appendix A.....	247

Figures

Fig. 1-1: Main tasks of the SOA introduction.....	6
Fig. 1-2: Main task of the thesis.....	7
Fig. 1-3: General solution concept for quality evaluation problems.....	9
Fig. 2-1: Overview of the ISA [Krcmar05].....	13
Fig. 2-2: Layers of Enterprise Architecture based on [Nieman05].....	14
Fig. 2-3: Artefacts of Enterprise Architecture based on [Engels08].....	15
Fig. 2-4: EA approach of the IAF as in [Capgem01].....	16
Fig. 2-5: Overview of the Zachman Framework [Zachma87].....	16
Fig. 2-6: Overview of ARIS [Scheer98].....	18
Fig. 2-7: Application layer of the EA meta model from [Braun05].....	20
Fig. 2-8: Terms defined in the enterprise ontology [Uschol95].....	20
Fig. 2-9: Adequacy of alternatives for characterizing enterprise architecture.....	23
Fig. 2-10: Preferred alternatives for characterizing dimensions of EA.....	24
Fig. 2-11: Enterprise architecture pyramid based on [DernGe03].....	26
Fig. 2-12: Elements of information architecture based on [DernGe03].....	27
Fig. 2-13: Elements of software architecture based on [DernGe03].....	27
Fig. 2-14: Enterprise Architecture Management as in [Engels08].....	29
Fig. 2-15: Overview of the TOGAF Framework [HarenV09].....	31
Fig. 2-16: The TOGAF Architecture Development Method [HarenV09].....	32
Fig. 2-17: The management cycle.....	34
Fig. 2-18: Adequacy of alternatives for characterizing dimensions of EAM.....	36
Fig. 2-19: Preferred characterization alternatives for EA and EAM.....	38
Fig. 2-20: Web Service Triangle from [Dostal05].....	40
Fig. 2-21: Primitive view of how SOA modularizes automation logic as in [ErlTho06]	43
Fig. 2-22: How components of the SOA relate [ErlTho06].....	43
Fig. 2-23: SOA Pattern example from [ErlTh09].....	44
Fig. 2-24: Elements of a service as in [Krafzi06].....	44
Fig. 2-25: Main elements of an SOA as in [Krafzi06].....	45
Fig. 2-26: Adequacy of alternatives for characterizing dimensions of SOA.....	52
Fig. 2-27: Preferred alternatives for dimensions of EA, EAM and SOA.....	53
Fig. 2-28: Requirements derived from preferred characterization alternatives.....	55
Fig. 2-29: The development corridor based on [Engels08].....	56
Fig. 2-30: Categorization of requirements.....	57
Fig. 2-31: Appropriateness of existing approaches.....	66

Fig. 3-1: Listing of categorized requirements.....	69
Fig. 3-2: Solution concept for R1 and R2.....	70
Fig. 3-3: Solution concept extended by R5 and R6.....	70
Fig. 3-4: Solution concept extended by R4.....	71
Fig. 3-5: Solution concept extended by R3.....	72
Fig. 3-6: Solution concept extended by R7, R10 and R11.....	73
Fig. 3-7: Solution concept extended by R9, R12 and R13.....	74
Fig. 3-8: Realization of the basic solution concept.....	75
Fig. 3-9: Sequence of realizing the steps of the solution concept.....	78
Fig. 4-1: Contribution of chapter 4.....	79
Fig. 4-2: IT-landscape with two monolithic applications.....	80
Fig. 4-3: IT-landscape with two component based applications.....	82
Fig. 4-4: IT-landscape with middleware connecting applications.....	82
Fig. 4-5: IT-landscape with middleware (EAI) with separation of GUI.....	84
Fig. 4-6: IT-landscape with basic services.....	85
Fig. 4-7: IT-landscape with basic and orchestrated services.....	87
Fig. 4-8: IT-landscape with orchestration engine.....	89
Fig. 4-9: IT-landscape with orchestration engine and human interaction services.....	90
Fig. 4-10: IT-landscape with orchestration engine and complex event processor.....	92
Fig. 4-11: IT-Landscape with business process monitoring.....	94
Fig. 4-12: Major concepts combined by Service orientation.....	96
Fig. 4-13: Example of an SOA service contract.....	98
Fig. 4-14: Structure of an SOA service.....	99
Fig. 4-15: Essential parts of an enterprise architecture (compare [Assman08]).....	101
Fig. 5-1: Contribution of chapter 5.....	104
Fig. 5-2: Section of the SOA meta model from [CDBISA08].....	106
Fig. 5-3: Part of the SOA meta model from [OASISR08].....	107
Fig. 5-4: Part of the SOA meta model from [Baresi03].....	107
Fig. 5-5: Overview on adequacy of existing SOA meta models.....	108
Fig. 5-6: Layers in Service-Oriented Enterprise Architecture.....	109
Fig. 5-7: Diagram with SOA meta model.....	116
Fig. 5-8: Essential parts of an enterprise architecture (compare [Assman08]).....	118
Fig. 5-9: Diagram with minimal EA meta model.....	119
Fig. 5-10: Diagram with EA meta model.....	125
Fig. 5-11: Simplified meta model merging process.....	127
Fig. 5-12: Generic example for table transformation.....	128
Fig. 5-13: Transformation of ‘Interface’ to table representation.....	130
Fig. 5-14: Bayesian network for artefact matching.....	132

Figures

Fig. 5-15: Matching example.....	134
Fig. 5-16: Joining example before joining.....	137
Fig. 5-17: Joining example after joining.....	138
Fig. 5-18: Algorithm for meta model merging.....	139
Fig. 5-19: Integrated meta-model of EA and SOA concepts.....	141
Fig. 6-1: Realization of basic solution concept.....	143
Fig. 6-2: Contribution of chapter 6.....	145
Fig. 6-3: Relation between quality criteria, metrics, and indicators.....	146
Fig. 6-4: SOA reference architecture.....	147
Fig. 6-5: Structure of an SOA service.....	157
Fig. 6-6: ISA meta model as in [Vascon07].....	159
Fig. 6-7: Adaptation of LCOIS metric.....	160
Fig. 6-8: Adaptation of NOIS metric.....	160
Fig. 6-9: Template for metric description.....	161
Fig. 6-10: Measuring points for multi-channel services.....	164
Fig. 6-11: Measuring points for SOA service reuse.....	165
Fig. 6-12: Measuring points for legacy adaptation.....	166
Fig. 6-13: Measuring points for middleware saturation.....	167
Fig. 6-14: Measuring points for middleware usage.....	168
Fig. 6-15: Measuring points for standard communication.....	169
Fig. 6-16: Measuring points for event enablement.....	172
Fig. 6-17: Measuring points for business object events.....	173
Fig. 6-18: Measuring points for process events.....	174
Fig. 6-19: Measuring points for orchestration invocation.....	175
Fig. 6-20: Measuring points for application invocation.....	176
Fig. 6-21: Measuring points for SOA service matching.....	177
Fig. 6-22: Measuring points for application invocation.....	177
Fig. 6-23: Measuring points for human support.....	178
Fig. 6-24: Measuring points for service realization.....	179
Fig. 6-25: Measuring points for application realization.....	179
Fig. 6-26: Measuring points for Orchestration engine existence.....	180
Fig. 6-27: Measuring points for orchestrated SOA services.....	181
Fig. 6-28: Measuring points for orchestrated processes.....	181
Fig. 6-29: Measuring points for BPM application.....	183
Fig. 6-30: Measuring points for KPI usage.....	183
Fig. 6-31: Measuring points for KPI messages.....	184
Fig. 6-32: Measuring points for stored business objects.....	186
Fig. 6-33: Measuring points for accessed business objects.....	187

Fig. 6-34: Measuring points for orchestration	189
Fig. 6-35: Measuring points for loose coupling.....	190
Fig. 6-36: Measuring points for functional compactness.....	191
Fig. 6-37: Measuring points for service registration.....	195
Fig. 7-1: Contribution and structure of chapter 7.....	198
Fig. 7-2: Template for indicator definition	199
Fig. 7-3: Table view of report on service orientation	211
Fig. 7-4: Bar diagram of report on service orientation	211
Fig. 7-5: Diagram of report on service orientation	212
Fig. 7-6: Precedence graph for quality properties.....	214
Fig. 8-1: Contribution of chapter 8	216
Fig. 8-2: Versions of used technologies.....	217
Fig. 8-3: GMF dashboard after first step	217
Fig. 8-4: Ecore model with basic elements.....	218
Fig. 8-5: Ecore model diagram with basic elements.....	219
Fig. 8-6: GMF dashboard after second step.....	219
Fig. 8-7: Tree syntax editor generated from “Domain Gen Model”	220
Fig. 8-8: Eclipse workspace with “Graphical Def Model”	221
Fig. 8-9: Eclipse workspace with “Tooling Def Model”	222
Fig. 8-10: Creation of the mapping model.....	223
Fig. 8-11: Defining node labels	224
Fig. 8-12: Correcting interchanged mappings	224
Fig. 8-13: Screenshot of the graphical editor.....	225
Fig. 8-14: Menu bar entry of the OCL plugin.....	226
Fig. 8-15: Metric management window.....	226
Fig. 8-16: Metric creation window	227
Fig. 8-17: Measure calculation window	228
Fig. 9-1: Basic solution concept	229
Fig. 9-2: Final table showing the fulfilment of requirements	233

1 Introduction

“Change is the only constant” is a citation often used by business analysts. As described in [WoodsD06], over the years the factor change has steadily increased. It is pointed out that several average life cycle times, namely those for products, applications, and business processes, have been decreased by orders of magnitude during the last decades. During this time, enterprise architectures have significantly changed several times to keep up with these decreasing life cycle times. Service-Oriented Architecture is the latest concept targeting the challenge ever shortening lifecycles, which enterprises are confronted with.

In the last years, Service-Oriented Architecture (SOA) has grown from a hype to a seriously relevant enterprise topic. On the one hand, this is indicated by the growing number of enterprises selling SOA solutions like IBM, Oracle, and SAP. On the other hand, it is indicated by the number of publications on the topic SOA. Often-cited publications are “SOA – concepts technology and design” by Thomas Erl (compare [ErlTho06]) and “Enterprise SOA: Service-Oriented Architecture best practices” by Dirk Krafzig et al. (compare [Krafzi06]).

SOA is a style for enterprise architecture. For this reason, it cannot be compared with software architecture styles like Enterprise Application Integration (compare [KaibMi04]). The transformation to SOA concerns the whole enterprise and should be done by persons that understand themselves as enterprise architects. The variety of affected parts of the enterprise makes the transformation to SOA a challenging task.

The challenge of introducing SOA brings up governance and technical challenges, which have to be mastered at the same time. The technical challenge is about mastering new technologies to develop and implement SOA services. The governance challenge includes convincing managers and employees of the concept, managing finances, changing established development processes and transforming the enterprise architecture. The architectural challenge targets the question on how to transform the structural elements of an enterprise to a service-oriented style. This question shall be focused here.

Firstly, the architecture challenge is tough because an enterprise architect will have to manage the transformation of the current enterprise architecture to a service-oriented one. Greenfield approaches are rather rare, as enterprises cannot afford

discarding their existing assets. Therefore, the introduction of a Service-Oriented Architecture heavily influences the existing enterprise architecture.

Secondly, the architecture challenge is tough, because since the birth of SOA, in the beginning of the current decade, no standard definition for SOA has been formulated. Hence, there are several opinions of what SOA is. The only thing they have in common is that there should be services with which SOA should bridge the gap between the business world and the IT-world in enterprises. It is shown later on that SOA definitions reach from very technical to very abstract ones.

This thesis shows how the architectural challenge is faced with a model-based approach that allows modelling an enterprise architecture and evaluating its service orientation with the help of service orientation quality criteria. The modelling and evaluation are realized in a proof-of-concept implementation using the eclipse modelling framework (EMF).

This thesis was created in cooperation with the Wincor-Nixdorf International GmbH. Therefore, the focus does not lie on SOA as an artificial concept but on an SOA suitable in the enterprise context. The author has collected experiences at Wincor Nixdorf in parallel to elaborating the results of this thesis. He has influenced several projects pushing service orientation. Many interesting problems and their solutions have indirectly contributed to this thesis.

The next section shall motivate why SOA is important for an enterprise and why this thesis was created.

1.1 Motivation

A common problem that enterprises of medium to big size are facing nowadays is the lack of flexibility concerning their IT. Business processes can change within days, but the IT-architects cannot keep pace. Their huge IT-landscapes with hundreds of applications and dozens of technologies cannot be rearranged in the same time as processes can, at least not with a justifiable effort. Regarding the trend of ever-faster change that can be observed since several decades (compare [WoodsD06]), this issue will become more and more delicate in the future.

As markets change more quickly these days, business processes have to change in the same manner. A business process change usually implies a change in IT. If a

competitor is able to change his IT faster or more efficient than his rivals can do, then he will experience a serious advantage. He could serve his customers with more customized solutions or just be faster in delivering solutions while offering the same or even a lower price.

Furthermore, the IT-related costs of many enterprises have become the lion's share regarding their investments. That means that savings in this sector are especially desirable because of the great saving potential.

In order to be able to keep pace with the changing market situations, not only the portfolio of an enterprise has to change, but also its internal architecture has to be changed. This can be compared with the natural evolution process. There are suited animals and less suited animals for any given environment. They have a portfolio, which means they have capabilities like running, hunting, hiding, sneaking, etc. Be there a big cat animal with a physique perfectly suited for the savannah environment. As a change, be there trees rapidly spawning and growing in that environment. Now climbing becomes a helpful capability for the big cat. Just like an enterprise that can expand its portfolio, the big cat can learn climbing. Therefore, it might survive the change, but there will be another animal with a different genome and thus a different physique that fits better to the current environment. Changing the physique means changing the architecture for an enterprise. The previously perfectly suited animal will lack efficiency in the changed environment. The better-suited animal might have a different muscle profile as climbing stresses muscles in a different way than running. Furthermore, it might have a different blood circulation, as it is generally colder in the tree-crowded environment. Both might have the same capabilities, but one of them is more efficient due to the differences in its physique. The same goes for enterprises, changing the portfolio due to a different market situation helps surviving, but decreases efficiency if the architecture is not changed accordingly.

Service-Oriented Architecture can be regarded as a big step in the evolution of enterprises. It promises making the enterprise architecture more flexible, which means that not only the current change of the market can be overcome but also further changes in the future can be adapted in an easier way. Analogously, the big cat would not only change its blood circulation but also its evolutionary speed by having offspring with the age of two and not with the age of three.

For further pointing out a situation that SOA could be a solution for, a realistic scenario is described in the following. The scenario is given for a fictive company

named Crimson & Wiley (CW). CW attends two main business segments. Firstly, it sells ATMs with the related software. Secondly, it sells point of sale systems (POS systems) with related software for retailers. The company consists of a banking division and a retail division.

Both divisions started with completely different hardware products with a low software share. Over time, the software portfolio was growing. The monitoring for the hardware systems was developed early. Both divisions developed their own solutions, because of several reasons. At first, not enough information exchange has taken place between the divisions. Secondly, the systems seemed to be so different that a common solution would not be profitable. Another reason might be that there were persons in both divisions wanting to take responsibility for the monitoring development and therefore did not seek the cooperation between divisions. Once having two different software development departments in different organization structures, there were several development projects that were not checked for synergy or reuse effects.

As markets change over time, so did the banking and retail market in this scenario. Both business segments were growing together. That means that POS systems got more and more similar to ATMs. POS had to adapt card readers early and nowadays it is even possible to withdraw money from a POS system. The software-supported assignment of cash transports in both segments also has been wandering into the portfolio of CW.

At a certain point of time, the management of CW recognized the potential savings that lay in the IT. The question how to introduce consolidations in the two divisions came up. A big bang approach with consultants merging organization structures and consolidating software systems was not desired. The consequences on staffing would have been too hard, facing the fact that both organizations should be able to work at full capacity in mid term. The technical consequences of this approach were also estimated as risky, because the quality of service could suffer too much from the abrupt change.

This is where SOA comes into play. With the introduction of SOA, the reuse and consolidation between the two divisions could be realized in form of a managed evolution, not a revolution. The definition of SOA services that are registered in a central registry fosters the reuse of software in the future. Then, the development process may start with looking up suitable SOA services for the new development project. Furthermore, SOA services adapt the existing software systems and because

of the implementation transparency of SOA services the stepwise exchange of legacy systems is eased.

However, not only software systems can be reused and consolidated but also businesses processes have to be merged. For example, the assignment of cash transports can be planned for ATM and POS system networks in one process, so that the overall transport ways may be shortened. The orchestration of SOA services also delivers supports in this scenario.

The evolutionary merging of two divisions is a scenario where SOA is beneficial. The given scenario is similar to a possible mergers and acquisition (M&A) scenario where SOA could have similar benefits. SOA is a considerable strategic option for enterprises that foresee manifold changes in their business processes like the one from the cash transport assignment. Otherwise, the problems of integrating software systems, merging and altering business processes with the underlying IT, and increasing the enterprise-wide reuse of software systems will be harder to tackle.

This thesis will clarify what SOA is and show how it attempts to meet the expectations. If SOA holds all of its promises, then it will greatly reduce the IT costs of an enterprise. The other side of the coin is the complexity of the SOA introduction and the risk that it might fail if not enough knowledge about SOA is at hand. Methods for introducing an SOA in an enterprise would be very beneficial because they can reduce the risk of failure. Therefore, a first task definition for this thesis is given in the next section.

1.2 Task Definition

As pointed out in the previous section, reducing the risk of an SOA introduction is the goal of this thesis. In this section, the scope of this goal is further elucidated.

The SOA introduction contains six main tasks being depicted in Fig. 1-1. This thesis cannot cover all the tasks, but will focus on tasks that are hardly covered by existing solutions and are regarded as risky.

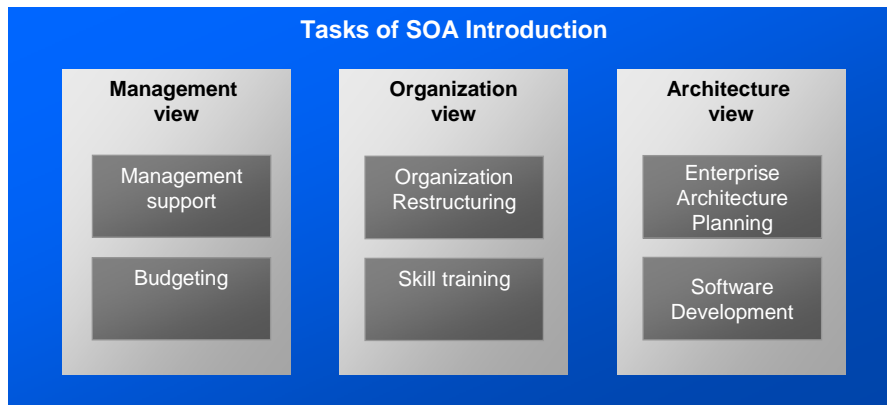


Fig. 1-1: Main tasks of the SOA introduction

This thesis takes the management support for granted. This is usually the case when an adequate business case has been presented to the management to show the profitability of the SOA. Information on the creation of an adequate business case for SOA is given in [Assmanb09].

After the acceptance of the business case, the management support should be given and a budget should be granted. The budgeting for an SOA introduction is similar to budgeting in any other strategic project. This is the reason why budgeting is not covered in this thesis.

In the mid term, new roles should be created and employees should be found for these new roles. A role defines a set of tasks an employee is responsible for. This restructuring of the organization is not trivial, but as long as it is clear which new tasks have to be handled, these tasks and also similar existing have to be combined into new roles. A new role, for example is the enterprise architect. The modelling of dependencies between business processes and IT-systems as well as the design of new SOA services supporting changing business processes belong to his tasks.

The skill training requires trainers who are familiar with the concepts of service orientation and related technologies. These trainers have to be trained or bought from consulting companies.

One of the more delicate tasks is the software development process. The adaptation of the existing software development process premises the adequate skill training of developers and software architects. Especially defining the right SOA services is a completely new task to be concerned. The problem is not tackled here, but a diploma

thesis describing a method for SOA service tailoring was created (compare [UecanE08]) parallel to this thesis. This thesis will use the diploma thesis' results on the quality of SOA services for the last and remaining task from Fig. 1-1.

Probably, the most problematic task of the SOA introduction is the planning of the enterprise architecture (EA), including a plethora of elements like business processes, SOA services and applications. What is the service-oriented style for an enterprise architecture? How would an individual EA applying this style look like? What changes have to be initiated in order to transform the EA to an SOA-like EA? As the answers on these questions are not easy to retrieve, the task of EA planning is tackled here. This main task of this thesis is illustrated in Fig. 1-2. The person responsible for the planning of the enterprise architecture is called the enterprise architect. He should be supported with tools and methods tailored for the EA planning part of the SOA introduction.

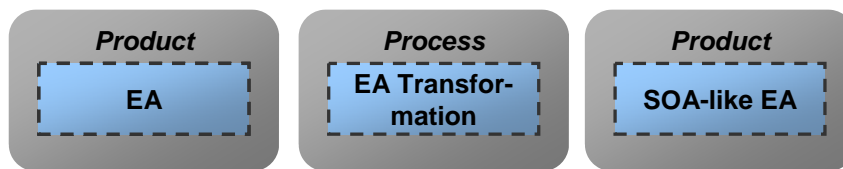


Fig. 1-2: Main task of the thesis

The three topics related to the three boxes are examined in the next chapter. The first topic concerns enterprise architecture. That means what are the relevant elements in an enterprise architecture and how can these elements be categorized or modelled.

The discipline related to EA transformation is Enterprise Architecture Management (EAM). EAM and SOA fit together in a harmonic way because EAM usually defines ways how to change an enterprise architecture, but not what to change. However, SOA defines style of EA and defines what has to be changed, but not how to do this. Therefore, SOA and EAM complement one another. This is why the second examined topic is EAM.

The third topic is Service-Oriented Architecture. However, not the term SOA itself but SOA in the context of an enterprise architecture has to be clarified.

The existing work is examined for each topic and the characterizing dimensions are elaborated. At the same time the alternatives for each characterizing dimension are discussed and the best choice for a solution to the here stated problem is given. From

these choices, the final requirements for a solution are derived at the end of chapter 2. In chapter 3, the task definition is refined and an overview on the structure of the remaining thesis is given.

It is very probable that most enterprise architectures have structures and elements not fitting into a Service-Oriented Architecture. These have to be identified and changed afterwards. This can hardly be done without the help of models, due to the immense size of process- and IT-landscapes in enterprises. Without the right abstraction level, thus leaving out the right details, an architect can probably not identify the right structures to be changed. Models are often sneered at because they are seen as an expensive way of documentation, but with increasing complexity of a system, they get indispensable for planning and restructuring.

In the following, a very general solution concept on how to prove quality properties of real world systems (like service-orientation of an EA) with the help of models is introduced. The approach of this thesis will follow this general concept.

The general solution concept proving quality properties of a real world system is depicted in Fig. 1-3. Quality properties are specified for a given real world system. The real world system can be anything from a material object like a car to an immaterial software system like a route planning software. Its quality properties are often given in texts of natural language. Usually, it should be possible to understand the quality properties without technical background knowledge. However, the quality properties are often hard to prove. For example, how to decide how high the maintainability of a software system is? Inspecting the real world system for this purpose is tedious because the source code contains much more information than needed. As long as the real world system is too complicated for understanding it by just inspecting it, a model of the system that abstracts from irrelevant details is created. Modelling languages with a well-defined syntax, like the UML, are often used for this purpose. However, also drawings without neither a defined syntax nor semantics can be used.

For example, the modelling language could be UML class diagram. Using class diagrams, the system can be better understood and planned, but still, it is hard to tell whether the maintainability is given or not. For this reason, the quality properties have to be refined and transformed to quality criteria. The number of god classes (compare [RielAr05]) could be a refined quality criterion indicating the maintainability of a software system. Using a class diagram, the evaluation is easy. In addition, this could

be automated with little effort. However, the quality criteria must fit to the modelling method chosen. Otherwise, the evaluation is not possible (e.g. when using state charts together with criteria for class diagrams).

As models are mostly too complicated to be processed on paper, a decision on tool support has to be taken. This is not easy because a plethora of tools designed for modelling is available. At first, a choice for the type of modelling language the tool supports has to be taken. An example for a generic language tool is a drawing program. Other tools support languages having a formally specified syntax, like “Enterprise Architect” for the UML. In addition, there are tools for languages with formal semantics like the Dynamic Meta Modelling editor (compare [Hausma05], [Banden09]). However, the tool decision is not done with the choice of a modelling language, because the tool can and should support the evaluation of quality criteria. An example for such a modelling tool is an architect’s modelling tool that is able to calculate the statics of the building modelled.

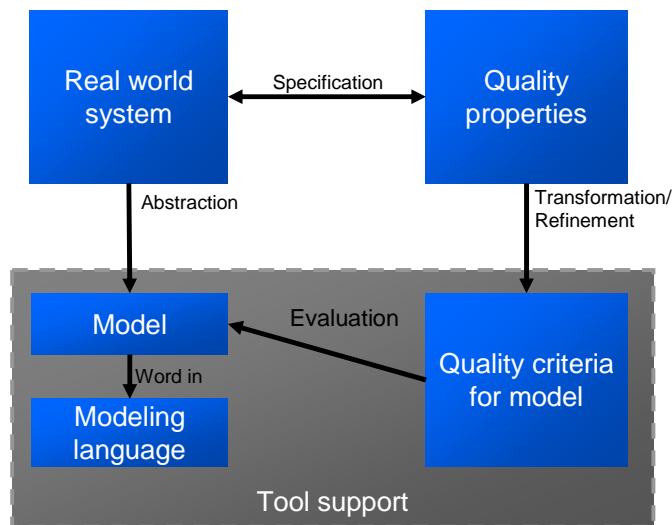


Fig. 1-3: General solution concept for quality evaluation problems

The approach of this thesis will follow this general approach. The elaboration of the corresponding solution approach for this thesis is given in chapter 3. In the following chapter 2, the basic concepts and requirements for this thesis are examined.

2 Basic Concepts, Requirements and Related Work

This chapter covers the basic concepts building the foundation of the thesis. In addition, the requirements for a solution to the task given in section 1.2 will be refined here.

The basic concepts are enterprise architecture (EA), Enterprise Architecture Management (EAM) and Service-Oriented Architecture (SOA). The relations of the basic concepts are depicted in Fig. 1-2. It shows that SOA is the architectural style an enterprise architecture should be transformed to and EAM is an approach to manage enterprise architecture transformations in general.

On the one hand, the related work for SOA focuses on how a Service-Oriented Architecture should look like but merely on how to transform an existing architecture to it. On the other hand, the EAM-related work places the focus on how to transform an existing architecture, but not on how it should look like in detail.

For this reason, the concepts shall be integrated in one approach. In order to do this, the characterizing dimensions for each topic are elaborated and the alternatives are discussed. Characterizing dimensions are found during the comparison of existing definitions. From each dimension, the best alternative concerning a solution for the given task is chosen. From the collection of chosen alternatives the requirements for the approach of this thesis, which combines the three basic concepts, are derived.

The following section 2.1 gives an overview on enterprise architecture definitions, identifies the characterizing dimensions, and picks the adequate alternatives for the approach of this thesis. The sections 2.2 and 2.3 do the same for EAM and SOA.

In section 2.4 the complete set of requirements is derived from the chosen characterizing dimension alternatives. Finally, in section 2.5 the existing approaches are evaluated concerning the requirements elaborated.

2.1 Enterprise Architecture

This section covers enterprise architecture definitions and the choices for the alternatives of the characterizing dimensions. Thus, the first subsection covers EA definitions, the second subsection covers their comparison, and the third subsection covers the choice of alternatives concerning the characterizing dimensions.

In [Engels08], enterprise architecture is regarded as an ambiguous term. On the one hand, it is seen as architecture concerning the structure of a system – i.e. enterprise architecture is concerning the structure of an enterprise. On the other hand, it is seen as a discipline to manage these architectures. In this thesis, architecture is used in the sense of system structure and not of a discipline. The discipline to manage enterprise architecture is referred as Enterprise Architecture Management (EAM). Approaches claiming focusing on EA but meaning EAM (concerning the given categorization), will be examined in the EAM section.

2.1.1 Enterprise Architecture Definitions

In the previous sections, the term enterprise architecture was often mentioned. The term enterprise architecture is described differently by different authors. The definitions from [Krcmar05], [Nieman05], [Engels08], [Zachma87], [Braun05], and [Uschol95] are reflected and afterwards compared concerning their characterizing dimensions.

The Information System Architecture (ISA) as described in [Krcmar05] is a layered approach to describe an enterprise architecture. The business architecture is the top layer in the hierarchy. According to the author, it is very closely related to the process and organization architecture that build the layers below the business architecture. The third layer consists of the application, data, and communication architecture. The application architecture describes functions with which business processes are realized. The data architecture describes the distribution and the relation of the data entities in the enterprise. The communication architecture holds the information about the data transfers between the applications. The infrastructure architecture describes the platform for the layer above. In detail, these are the information and communication technologies, operating systems, and hardware.

Fig. 2-1 depicts the ISA in a gyroscopic way, which means that these architectures and their description must be balanced. Furthermore, the business architecture is

depicted by an arrow, such that it should be integrated in all the lower layers of the enterprise architecture. By this gyroscopic figure, a set of views is defined for this approach. There is no language suggested, in which the architecture shall be described. This means the user is free to define his own language and may decide freely on the meta content.

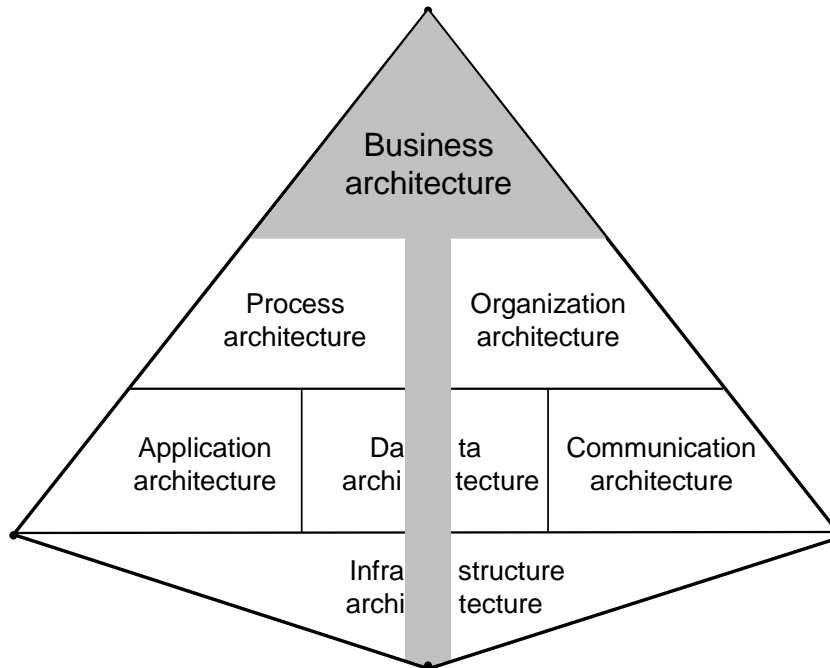


Fig. 2-1: Overview of the ISA [Krcmar05]

The second EA definition source is depicted in Fig. 2-2, as found in [Nieman05]. Niemann describes enterprise architecture as a layered architecture containing a business architecture, an application architecture, and an infrastructure architecture. These layers are not strictly separated from each other. This means they have intersections at certain points but the meta content is not entirely cohesive. A cohesive meta content means that all meta concepts can be set into relation with each other. For example, a formal meta model offers such meta content cohesion. Concrete integration points as well as a concrete language are not named in this approach. Furthermore, the layers can be quite different from enterprise to enterprise, as the concrete meta content in these layers may vary.

Even though enterprise architectures may vary in their appearance according to [Nieman05], the business architecture generally holds information about the goals and the strategy of an enterprise, about the business processes, and about the organization. Furthermore, the application architecture holds information about services or

interfaces, about the applications, and the data held by them. The infrastructure architecture consists of the development environment, the test environment, and the IT-technologies used.

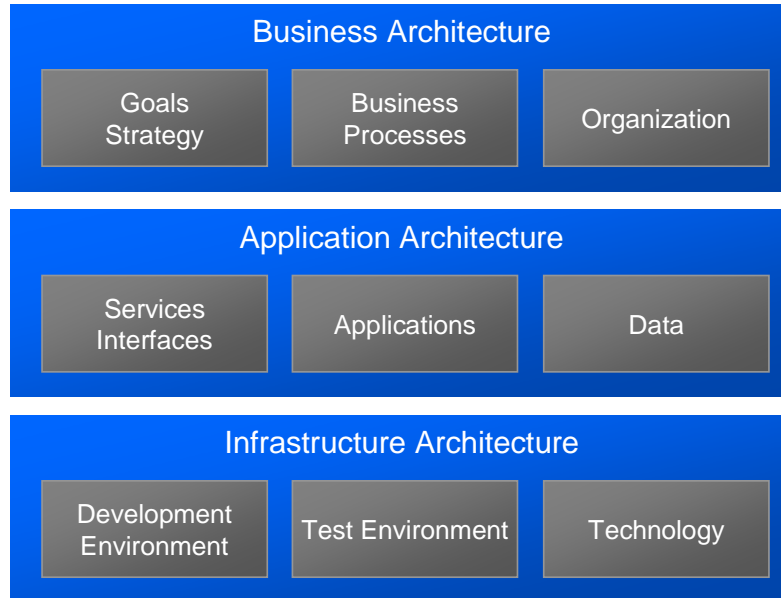


Fig. 2-2: Layers of Enterprise Architecture based on [Nieman05]

In Quasar Enterprise (compare [Engels08]), enterprise architecture is depicted as in Fig. 2-3. The authors of [Engels08] do not follow a layered approach to describe enterprise architecture. Still, a clear distinction between different kinds of architectures is given. Furthermore, Quasar Enterprise is based on the Integrated Architecture Framework (IAF, compare [Capgem01]) which is also a layered EA approach, depicted in Fig. 2-4.

The architectures named in Quasar Enterprise are the business architecture and the application landscape architecture. The latter consists of the information system architecture and the technology infrastructure architecture. Quasar Enterprise does also concern EAM. For this reason, it is also mentioned in subsection 2.2.1.

The main artefacts of the business architecture are business processes, business objects, and the organization of the enterprise. The business architecture influences the application landscape and by that it influences the information system architecture (IS architecture) and the technology infrastructure architecture (TI architecture).

The artefacts of the IS architecture are application services, logical components and technical components. Application services gather small-grained business functions. The realization of an application service makes use of several logical components. Logical components are abstract, e.g. a monitoring application, a travel booking system etc. A logical component is realized with a technical component, e.g. IBM Tivoli as a monitoring application.

Again, a distinction is made between logical and technical elements within the technology infrastructure. In this case, platforms, namely the system software components and hardware components are distinguished. Part of a logical platform could be an application server and the corresponding realization in the technical platform would be an Oracle Application Server.

The Quasar Enterprise approach provides several small examples or even meta models to describe most of the EA elements. There is no underlying holistic and cohesive meta model, but some concepts can be found in different meta model descriptions. These concepts can be used as integration points.

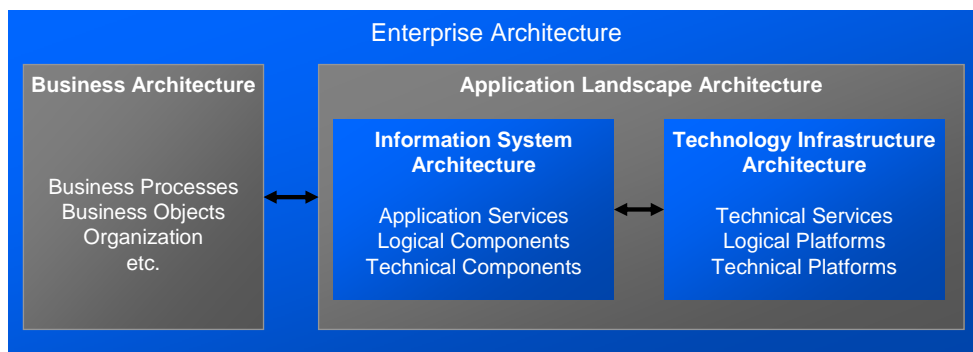


Fig. 2-3: Artefacts of Enterprise Architecture based on [Engels08]

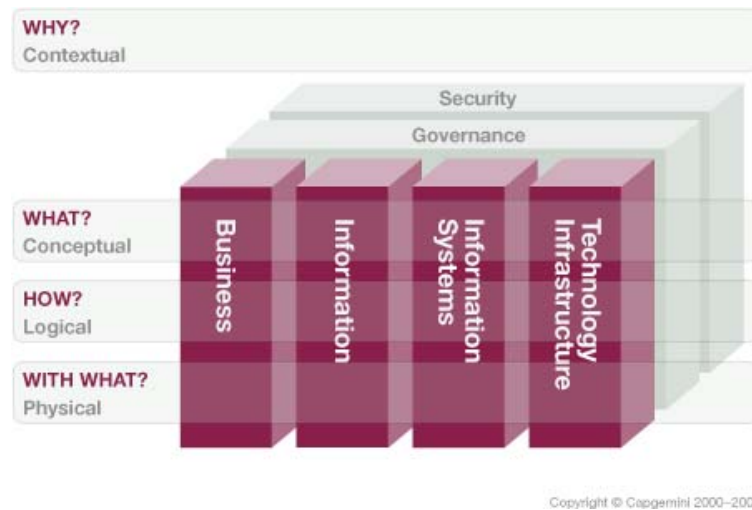


Fig. 2-4: EA approach of the IAF as in [Capgem01]

The next source for an EA definition is the Zachman Framework. In 1987, John Zachman had the idea in mind to “keep the business from disintegrating”. According to Zachman, the information system architecture has to be managed if the disintegration shall be stopped. Motivated by his idea he developed the Zachman Enterprise Architecture Framework, of which an overview is depicted in Fig. 2-5:

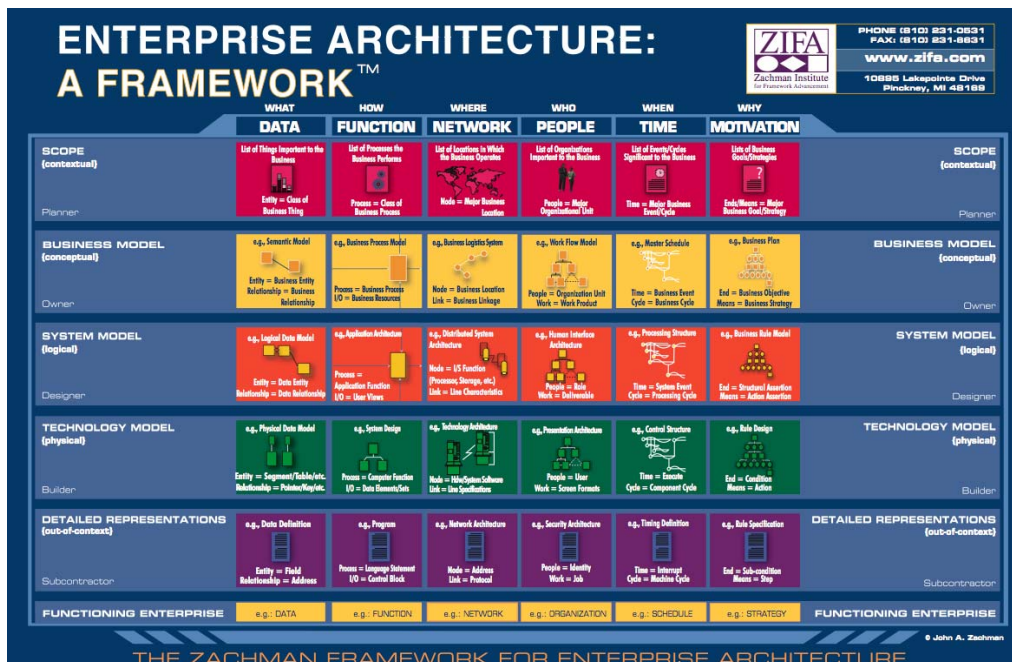


Fig. 2-5: Overview of the Zachman Framework [Zachma87]

The Zachman Framework defines layers and perspectives. The five layers are the contextual, the conceptual, the logical, the physical layer, and the out-of-context layer. For each of these layers different kinds of perspectives are introduced. First of these perspectives is the data perspective, answering the question what is examined. Further perspectives are the function perspective describing the how, the network perspective describing the where, the people (or organization) perspective describing the who, the time (or schedule) perspective describing the when and the motivation (or strategy) perspective describing the why. An enterprise architecture can be described quite completely with all these perspectives on all of the layers. Not all of the perspectives are important for all enterprises, so the user has to choose a suited set of perspectives. The Zachman Framework gives hints on how to realize these perspectives on the specific abstraction layer, for example a business process model for the functional perspective on the conceptual layer. The Framework also defines concepts that have to be followed when designing an enterprise architecture:

- Every cell describes an aspect of the architecture and is unique
- The six perspectives cover all models required for the development of the system
- The layers are hierarchical and have to be designed top-down
- The order of the perspectives has no meaning
- The perspectives are treated as abstractions without intersections that shall improve the handling of the systems complexity
- According to [Minoli08] the framework is recursive, so that an instance of the framework can describe the whole enterprise and the next instances describe the divisions of the enterprise.

The Zachman Framework does not present a concrete language to describe the suggested set of views on the EA. The meta content is described textually only. Thus, cohesion of the underlying meta concepts is not given.

Another framework is the Architecture of Integrated Information Systems (ARIS) as described in [Scheer98] and [Scheerb02]. Its EA approach is a little bit different from the typical layered view on enterprise architecture. Its focus lies more on the coherence of the elements describing the architecture.

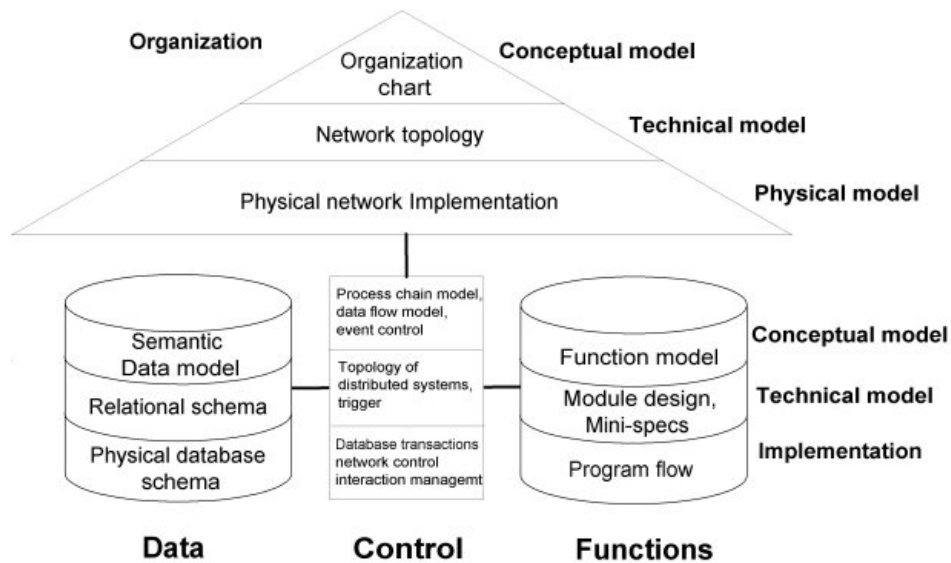


Fig. 2-6: Overview of ARIS [Scheer98]

The four main elements as depicted in Fig. 2-6 are the views on Organization, Data, Control, and Functions. Each of these blocks is layered, whereas the layers are the conceptual the technical and the physical layer. Such a block is considered as a view or perspective on the architecture.

The function view describes the activities that are needed to perform a process and the relations of these activities. Usually a function is related to an information object and describes an action on this object. Examples are creating an invoice or to take an incident. Furthermore, a function has a relation to an element of the organization view that means there is a responsible and or an executing organization unit for this function. Functions can also be related to events. On the one hand, a function like “processing order” can require the occurrence of an event like “order created”. On the other hand, the function itself can create an event like order processed.

The data view describes the information objects and their relations as well as their environment like in-house, client or provider. Possible upcoming events that trigger functions are also described in this view.

The organization view contains the elements of the company organization structure. This reaches from organizational units over roles to jobs and jobholders. Those elements of this view contain the responsible and accountable persons for functions

defined in the function view. Furthermore, the organization view describes resources that are required to perform functions.

The control view brings all the elements of the other views together. It contains the processes acting as the glue for functions, data, and organization. A process defines on which events it is invoked and the sequence in which the functions are executed. The execution of processes then leads to the fulfilment of business goals that are not depicted explicitly in the model.

To sum up, ARIS describes a fixed set of views and suggests languages like the Event-driven Process Chain (EPC). These have integration points that are indicated by the lines in Fig. 2-6. Not for every EA part a language like EPC is given, which leads to partly defined but not entirely fixed meta content predetermination.

Other approaches to describe enterprise architecture are ontologies and meta models. Meta models are a specification of terms and a syntax for a formal language. The purpose of the meta model is to describe the set of possible models. A model is named an instance of this meta model and describes a part of a real world situation (compare [Zelevs99]). However, an ontology has many similarities with a meta model. It shall describe the relations between real world concepts including their semantics. This can be done with a formal language, but also with natural language. Especially the description of semantics, which is more stressed in ontologies, is often given in natural language.

In [Braun05] a layered meta model is suggested to describe enterprise architectures. The layers are the strategy, the organization, and the system/application layer. The application layer is depicted in Fig. 2-7. The three layers have defined integration points, so that they are not strictly separated from each other and a meta content cohesion is established. The meta model fixes the predetermined meta content as the usable elements are exactly described. This approach suggests one and only one holistic language to describe an enterprise architecture.

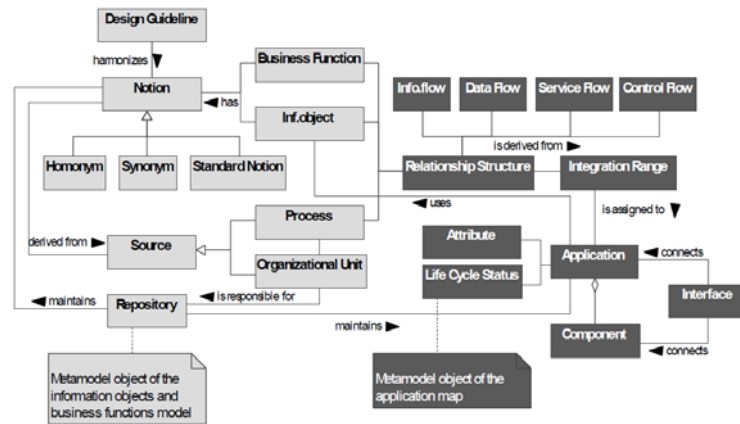


Fig. 2-7: Application layer of the EA meta model from [Braun05]

In [Uschol95] an extensive enterprise ontology is given. A variety of the described terms is given in Fig. 2-8. The approach contains the definition of these terms and their relations to each other. Just like in the approach of [Braun05], the language and the meta content is exactly defined here, because the variety of elements is fixed to the about 80 described terms. This ontology is not layered and therefore the meta concepts are completely cohesive. Maybe as a compensation for not layering the ontology concepts, categories were defined for the different terms. There is nothing said about views in this approach, which is interpreted as there is only one fixed view on the complete model.

ACTIVITY etc.	ORGANISATION	STRATEGY	MARKETING	TIME
Activity	Person	Purpose	Sale	Time Line
Activity Specification	Machine	Hold Purpose	Potential Sale	Time Interval
Execute	Corporation	Intended Purpose	For Sale	Time Point
Executed Activity Specification	Partnership	Purpose-Holder	Sale Offer	
T-Begin	Partner	Strategic Purpose	Vendor	
T-End	Legal Entity	Objective	Actual Customer	
Pre-Condition	Organisational Unit	Vision	Potential Customer	
Effect	Manage	Mission	Customer	
Doer	Delegate	Goal	Reseller	
Sub-Activity	Management Link	Help Achieve	Product	
Authority	Legal Ownership	Strategy	Asking Price	
Activity Owner	Non-Legal Ownership	Strategic Planning	Sale Price	
Event	Ownership	Strategic Action	Market	
Plan	Owner	Decision	Segmentation	

Fig. 2-8: Terms defined in the enterprise ontology [Uschol95]

2.1.2 Comparison of EA definitions

In the previous section, it is shown that there are different definitions of enterprise architecture. In this section, the characterizing dimensions of EA approaches are defined and afterwards a comparison of the approaches is given.

The *meta content predetermination* is the first characterizing dimension. As enterprise architectures are very individual, predetermining the meta content is done in different forms. Approaches defining a formal meta model will fix the meta content and do not leave any degree of freedom. Therefore, the alternatives fixed, partly variable, and variable are identified.

The second characterizing dimension is identified as *meta content cohesion*. Often there are several layers or diagrams describing an EA. There are pieces of information located in different layers or diagrams that shall be related to each other during the EA planning. For example, which business process is affected by the removal of a certain application interface? If the different meta concepts of an EA are not described cohesively, this will hardly be doable. Some approaches with layers have integration points, allowing a transitive information recovering. Others are completely cohesive, e.g. by providing a holistic meta model.

The third characterizing dimension in which the definitions vary is the *suggested language*. That means in which extent is the abstraction level fixed and how formal are the modelling languages to be used. With the suggested language, also the level granularity is influenced. A meta model fixes the granularity; a self-decided language leaves everything open. The spectrum of suggested languages reaches from self-decided modelling languages, to partial suggestions (with small meta models or diagram types), to meta modelling, or an ontology with a fixed formal syntax. Meta models and ontologies are very similar in this context as they specify the modelling elements and their relations. These languages are formal in their syntax but not in their semantics. Theoretically, it is possible to define formal syntax with formal semantics, but this is only mentioned for the sake of completeness.

The fourth characterizing dimension is the *suggested view*. Most observed references suggest a set of different views on the enterprise architecture in form of different layers or diagram types. Possible alternatives are a single fixed view, a defined set of views, or individual views on the enterprise architecture model.

	Meta content predetermination	Meta content cohesion	Suggested language	Suggested view
[Krcmar05]	Variable	Integration points	Self-decided	Set of views
[Nieman05]	Partly variable	Integration points	Self-decided	Set of views
[Engels08]	Partly variable	Integration points	Partly suggested	Set of views
[Zachma87]	Partly variable	None	Partly suggested	Set of views
[Scheer98]	Partly variable	Integration points	Partly suggested	Set of views
[Braun05]	Fixed	Holistic	Meta model	Fixed
[Uschol95]	Fixed	Holistic	Ontology	Fixed

The view on enterprise architecture in this thesis will be given in section 4.2. In the next section, the best choices (concerning the task from section 1.2) for the alternatives are discussed.

2.1.3 Choices for EA-specific Dimensions

Having taken a look on what is EA, the choices concerning the EA dimensions for the approach of thesis can be identified. Fig. 2-9 depicts the identified options and evaluates them concerning their adequacy. Afterwards the choices will be discussed.

The *meta content predetermination* should not be fixed as enterprise architectures are too individual and the stakeholders usually have their own nomenclature in mind. Preventing the relearning by allowing individual contents is regarded as the better alternative. Partly individual means that there are fixed contents and that nomenclatures as well as meta elements can be changed. Completely individual model contents are also acceptable as long as the enterprise architect is able to define the required content.

Category	Alternative	Adequacy
Meta content predetermination	Variable	+
	Partly Variable	+
	Fixed	-
Meta content cohesion	None	-
	Integration points	O
	Holistic	+
Suggested language	Self-decided	-
	Partly suggested	O
	Meta model / Ontology	+
	Meta model with formal semantics	-
Suggested view	Individual views	+
	Set of views	O
	Fixed	-

Fig. 2-9: Adequacy of alternatives for characterizing enterprise architecture

The *meta content cohesion* heavily influences the quality of the predictable consequences of changes in the enterprise architecture. The alternative supporting this feature best is a holistic model. Models with integration points between diagrams or layers are also possible, but the transitive way to find related pieces of information from different layers is more complicated.

The often very complex enterprise architecture has to be described somehow if the enterprise architect wants to deal with it. To do so, a *language* has to be *suggested*. A self-decided language as suggested in [Krcmar05] is probably not very helpful, as the machine readability that is needed in highly complex EAs is not guaranteed. The same goes for partly suggested languages. At least they are more helpful, because they build a frame for the syntax. A formal syntax, e.g. in form of a meta model, greatly improves the usefulness of a model because of its machine-readability. Together with an informal description, the different stakeholders have less freedom in interpreting the semantics of the EA description. It is still possible to interpret the description in different ways with a formal syntax (like in a meta model) and informal semantics. Formal semantics could improve this fact but they are rather complicated. Firstly, no language for enterprise architectures with formally defined semantics exists. Secondly, the stakeholders using the language would not be used to work formal specifications. Thirdly, the effort of defining an EA with formal semantics (already given an appropriate language) is very high and will probably outweigh the advantages.

The *suggested view* is fixed or a set of views in the examined approaches. However, the best answer on the question is completely dependent on the user. It is hard to foresee, which view he will find the most interesting. Therefore, the best solution for an approach that shall be suitable for any kind of enterprise architect should provide individual views rather than a predefined set of views on the enterprise architecture.

An overview of the preferred alternatives for the dimensions characterizing enterprise architecture is illustrated in Fig. 2-10. Having completed the examination of the first basic concept, the second basic concept – EAM – is elicited in the following two sections.

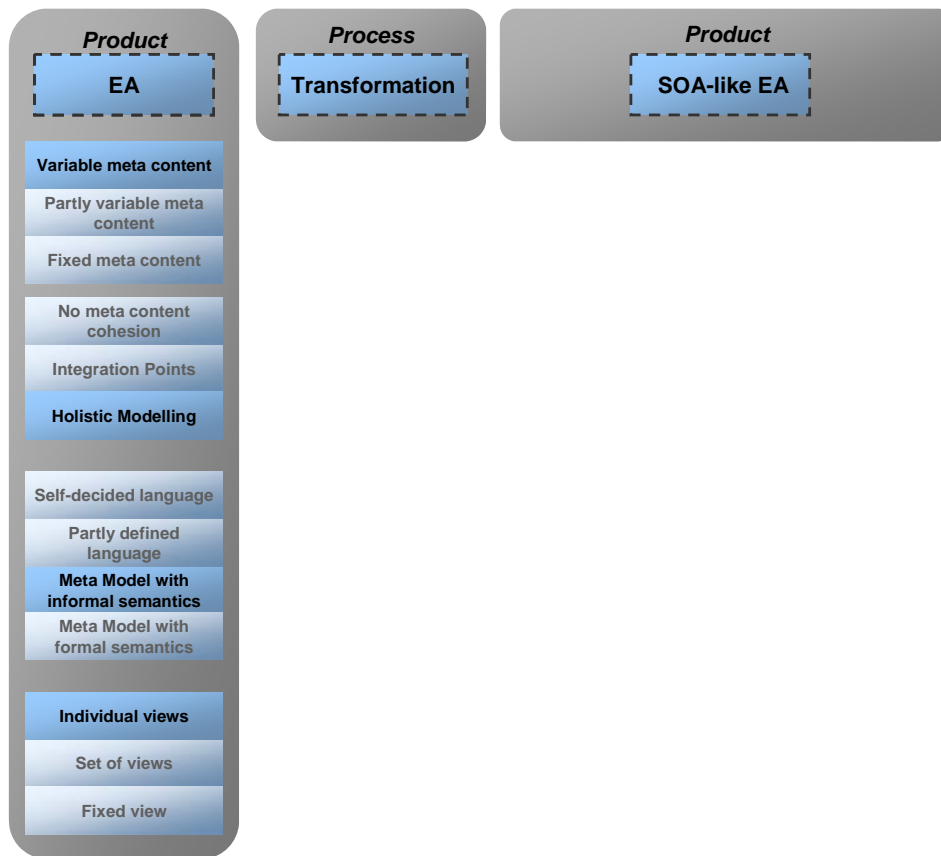


Fig. 2-10: Preferred alternatives for characterizing dimensions of EA

2.2 Enterprise Architecture Management

The existing EAM definitions are presented and compared concerning their characterizing dimensions in this section. EAM is the discipline or process that controls the transformations of an EA. In the manner of the previous section, the EAM definitions are examined and compared concerning their characterizing dimension. Afterwards, the choice of the best alternatives is discussed.

2.2.1 Enterprise Architecture Management Definitions

In this section, the term Enterprise Architecture Management is addressed. The approaches from [DernGe03], [Engels08], [Nieman05], [Matthe08], [Haren07], and [Bieman94] are compared here.

In [DernGe03] an architectural pyramid is described that covers Enterprise Architecture. Central figure is the pyramid as in Fig. 2-11. The EA pyramid and the process concerning the EA transformation are woven together in this approach. For this reason, the parts concerning EAM of this approach will be elicited predominantly.

The top layer is the strategy of the enterprise and its influence shall be given on all levels beneath. The strategy layer itself comprises of strategic goals like “Technological leadership for hybrid drives until 2015” or “Introduction of service orientation until 2010”. The business drivers, being part of the business architecture, are derived from these strategic goals. Business drivers are the main factors influencing revenue. Common examples for these influencing factors are customer needs and competition. In addition, the business layer contains business processes designed for realizing the business drivers. The organizational architecture, containing structural descriptions of business units and departments, is also regarded as part of the business architecture. So far, this reads as a normal EA definition.

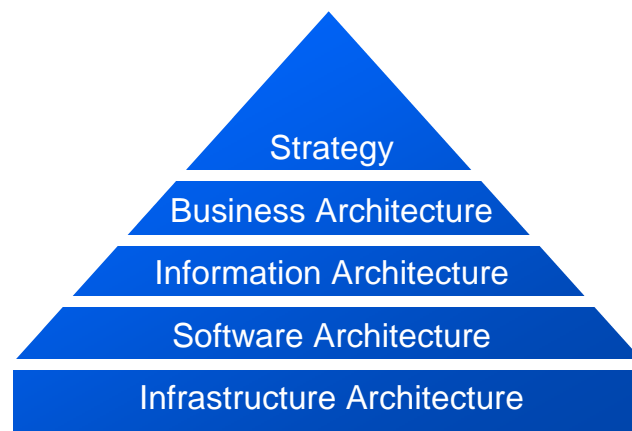


Fig. 2-11: Enterprise architecture pyramid based on [DernGe03]

However, the information architecture also has elements concerning the management of the EA. In general, the information architecture is the mediating layer between the business and the IT architecture. It contains the information concepts including business objects. It also comprises the information systems being abstracted from technological details like specific software products. Furthermore, the interactions between different IS can be abstracted in this layer in form of information object flow. A domain model is an often used to depict the static part of the information architecture.

Information systems are often planned only concerning the information systems itself, but not concerning their context. To prevent that different elements of the IT-architecture are not fitting to each other, the information architecture acts as a guiding frame. It contains the information system portfolio with the as-is and target state in its focus (see Fig. 2-12). The as-is IS portfolio lists the existing information systems and evaluates them concerning certain criteria, e.g. process support and target platform. Furthermore, it depicts the integration level of the IS landscape and information flow between IS. The information objects are often also called business objects are closely related to information systems. Their knowledge is decisive for the further development of information and IT architecture. The target IS portfolio describes the desired state of the application landscape in the future. Planning as-is and target architecture is based on the IS portfolio in this approach. The elements of the lower architecture layers are derived from the target IS portfolio. The as-is and target modelling is the first major extension of EA approaches in the direction of EAM.

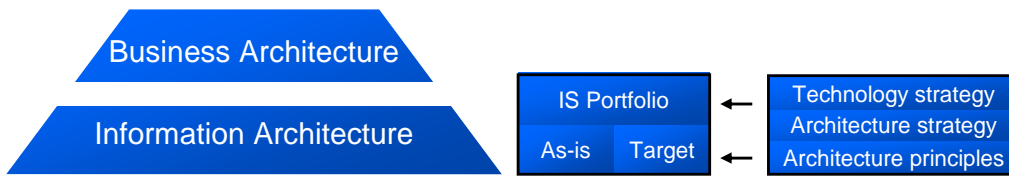


Fig. 2-12: Elements of information architecture based on [DernGe03]

Furthermore, the IS-portfolio is heavily influenced by three concepts: Firstly, the technology strategy, which has major influence on the IT basic infrastructure. For example, the strategy primarily to develop internally used software in Java. Secondly, the architecture strategy that describes the how the target architecture is enforced. Thirdly, the architecture principles, e.g. that Event-Driven Architecture is generally to be preferred over a synchronous message architecture. These strategies are also part of EAM and not of EA.

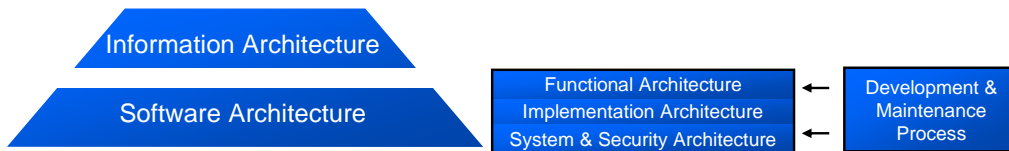


Fig. 2-13: Elements of software architecture based on [DernGe03]

The software architecture layer is divided into a static and a dynamic part. The dynamic part concerns the software development and maintenance process for the applications used within the enterprise. The static part concerns the applications themselves.

The dynamic part, the software development process, is seen as a crucial part of the IT-architecture as it has great influence on the flexibility of applications. It describes how the realization of new requirements is planned, designed, implemented, and rolled out. If there is a sophisticated and flexible software development process the application landscape can be changed in a more efficient and flexible way. Often there are many applications in an enterprise and each of them can have a different development and maintenance process. Considering the software development process is also a part of EAM.

The static part is covering the applications. Every application has a security architecture and a system architecture that represent the mapping from the software architecture to the infrastructure architecture. Besides an application has an implementation architecture describing which components the software consists of

and how these are interacting with each other. In addition, an application has a functional architecture that describes which functionality an application provides and the information needed and provided for business processes supported by the application. The rest of the EA pyramid is not covered, as it does not contain any EAM relevant information.

In the following, the approach is summarized. The planning strategy is realized as a reengineering approach with a target and as-is planning of the IS-portfolio. The change strategy of this approach is evolutionary, as the transformation from as-is to target architecture is realized over time and not in a big bang approach. Strategies for the architecture development must be formulated in an informal way. A measurement of the realization of the strategies is not proposed.

Another approach that regards architecture as a discipline is described [Engels08]. The tasks of Enterprise Architecture Management are depicted in Fig. 2-14. The Quasar Enterprise approach follows the Integrated Architecture Framework (IAF, compare [Capgem01]). According to IAF, the following four layers are determined. The contextual layer tries to clarify why something is done. The conceptual layer gives the answer on what should be done. The technical layer answers the question how something is done, and the physical layer gives insight into what things are finally realized.

In [Engels08] the EA change strategy is represented by the arrows in the figure. In order, the steps are analyzing and defining a business architecture, defining the ideal application landscape, defining an integration strategy, and defining an integration platform. All these steps have to underlie a constant evolution. This evolutionary planning strategy is realized using three planning instances, the as-is, the target and the ideal enterprise architecture. A measurement mechanism is not suggested in this approach.

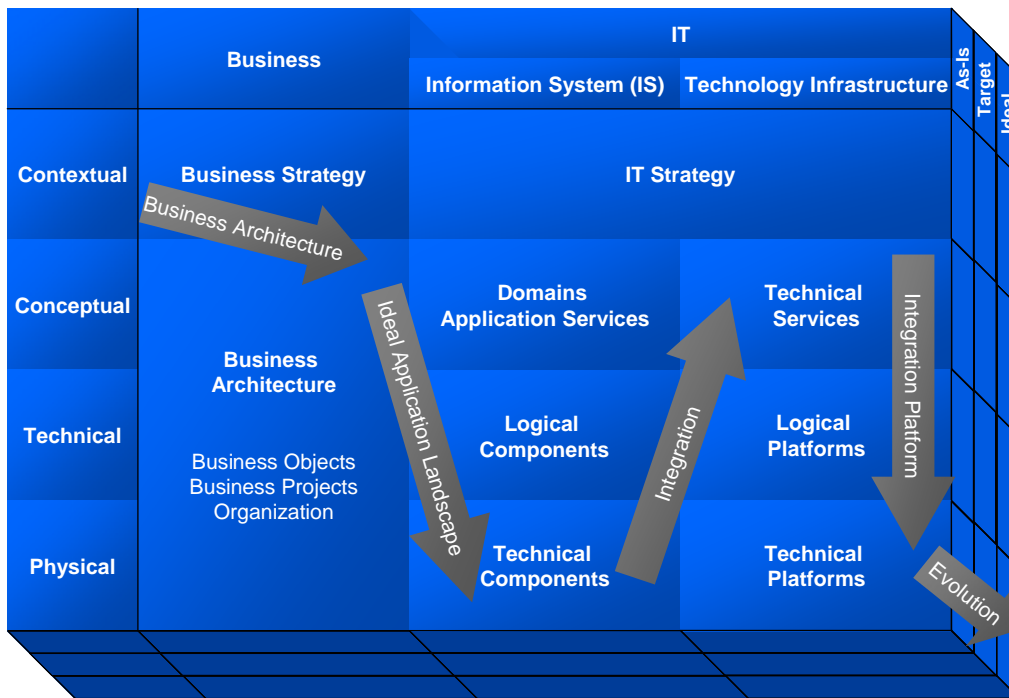


Fig. 2-14: Enterprise Architecture Management as in [Engels08]

In addition, [Nieman05] states something about EAM. The EAM tasks are defined there as the processes, methods, tools, responsibilities and standards that are required, so that IT systems do what they are intended to do in an efficient way. According to [Nieman05], the following tasks have to belong to EAM:

- The strategic process of documenting, analyzing and planning the enterprise architecture
- The operative process implementing the planned enterprise architecture
- The definition of documentation techniques
- The analysis and planning techniques
- The evaluation techniques
- The tools and their integration in processes
- The performance indicators and controlling

Something completely different in the area of EAM is the enterprise architecture pattern catalogue [Matthe08]. It presents interesting insights on the tasks concerning Enterprise Architecture Management. Tasks are formulated as questions and are called concerns. Patterns can have relations to concerns, meaning that they are useful for

working on the concern. The relevance of concerns has been validated in an industrial case study.

In addition to the concerns, the EA pattern catalogue presents three different kinds of patterns: methodology, view, and information patterns. Concerns address the question: “Which goal is to be achieved for which stakeholder?”. An example for this is: “Which business objects are exchanged over which interfaces?”. For a concern at least one methodology pattern is given, which provides the reader with steps to be taken in order to address the given concern. Methodologies range from group discussions to visualizations to formal methods like metric calculations. For each methodology pattern there exists at least one view pattern that providing languages for the methodology pattern. Furthermore, such a view pattern is a way to visualize the data contained in information patterns. The underlying meta-models for the views given in one or more V-patterns are provided with the information pattern.

The catalogue does not give recommendations on methods combining several of these patterns or how to embed them into a process. Concerns are distributed among the categories “Application Landscape Planning”, “Infrastructure Management”, “Interface, Business Object, and Service Management” and “Support of Business Processes”.

In [Matthe08] no planning or change strategy is specified. However, concrete architecture strategies are given as concerns in an informal way. For this reason, the catalogue can be used to identify relevant strategies (or concerns) for EAM. Unfortunately, no measurement method for the concrete concerns is suggested.

Enterprise Architecture Management is also covered by a well-known framework. The Open Group Architecture Framework (TOGAF) as described in [Haren07] and [HarenV09] consists of seven parts from which the first one is an introduction. The others are depicted in Fig. 2-15. Part two and three cover an architecture development method, whereas part three contributes guidelines and techniques for the development method. The fourth part describes the Architecture Content Framework, which provides a structural model for architectural content created by the architect. It allows the major work products of the architect to be consistently defined, structured, and presented. Part five presents the enterprise continuum. Its purpose is to aid in communication between enterprise architects and to aid in organizing re-usable architecture and solution assets. It presents a classification mechanism for architecture assets, such that architectures from different contexts can be made comparable. Part

six contains two reference models, a technical and an integrated information reference model. They provide taxonomies, each with a visual representation for their domain. Part seven contains the architecture capability framework. In order to provide an architecture capability an enterprise has to have organization structures, processes, roles, responsibilities, and skills of employees. The framework shows these artefacts, their dependencies, and ways how to manage them.

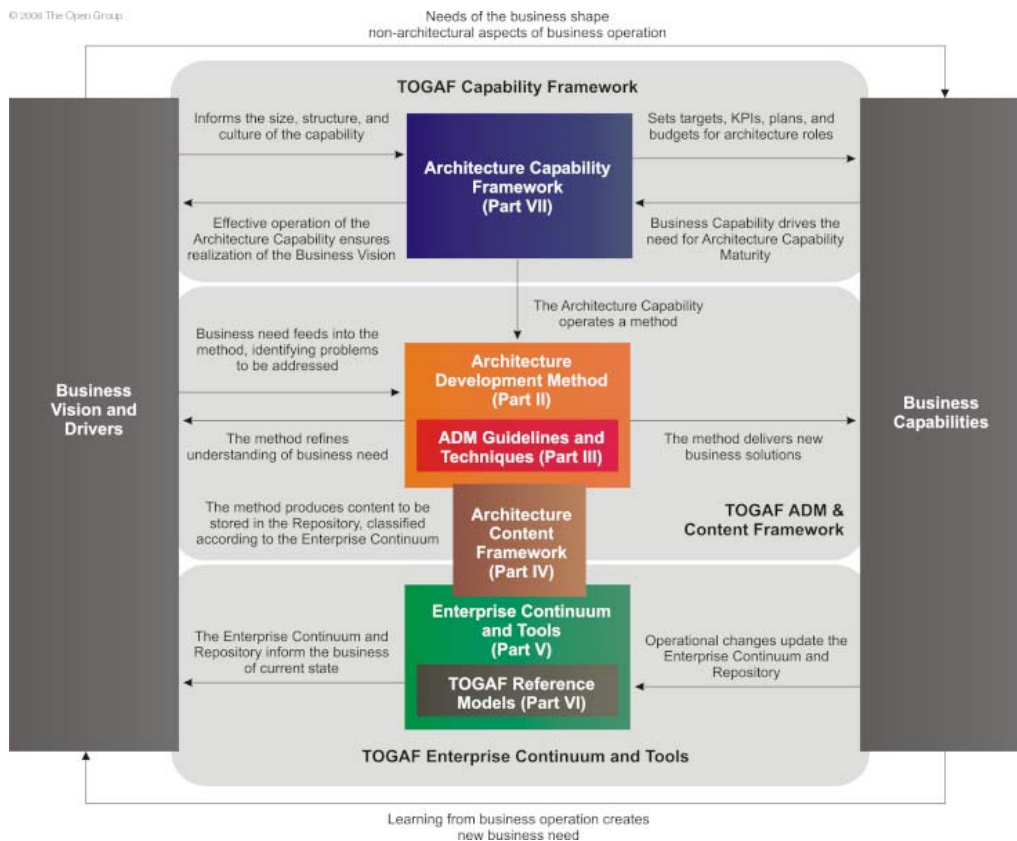


Fig. 2-15: Overview of the TOGAF Framework [HarenV09]

The Architecture Development Method (ADM), as depicted in Fig. 2-16, constitutes the core of the TOGAF Framework. The ADM is iterative, over the whole process, between phases, and within phases. For each iteration of the ADM, a decision has to be taken concerning the breadth of coverage, the level of detail and the time horizon. These decisions should be assessed concerning the resources and competences needed and the value generated. As the ADM is generic, it is expected to be extended as needed for the specific context. However, the ADM of TOGAF suggests a reengineering-like method for architecture planning but no measurement mechanism on architecture strategies.

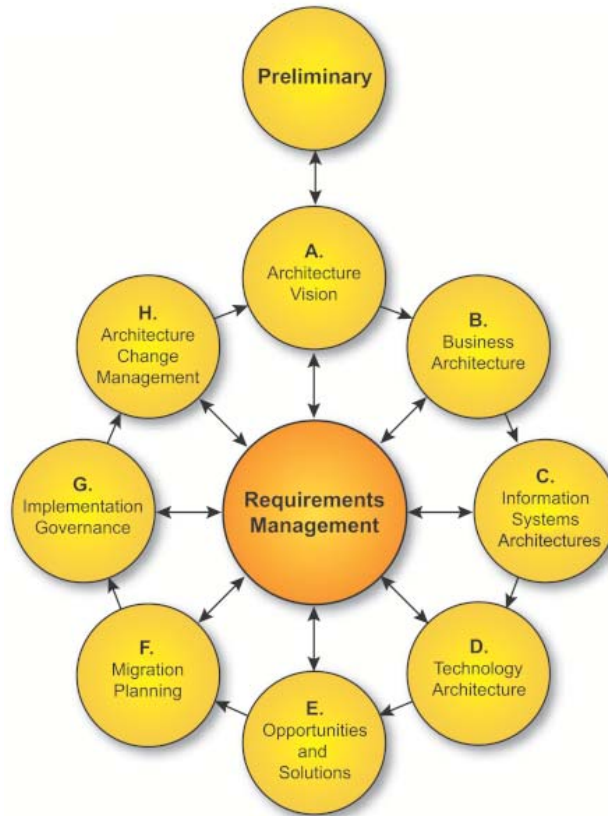


Fig. 2-16: The TOGAF Architecture Development Method [HarenV09]

So far, it has been pointed out that EAM concerns the planning of the EA to transform it into a form of higher quality. From Enterprise Architecture Management a parallel to the concept of Component Based Architecture (CBA) exists. In [MyersG73] this concept is described. As there are several degrees of freedom in tailoring components, there are also different quality attributes for Component Based Architectures.

Two major quality attributes are coupling and cohesion. These attributes and metrics for them are presented in [Steven74] and [Bieman94]. According to [Steven74] coupling between components is regarded as low if there are only simple and obvious interfaces between them. In addition, the number of interfaces should be small and the exchanged content shall mainly be data. If the interfaces are complex, refer to internal elements of other components and have control over other components, then the coupling is regarded as strong.

Cohesion is closely related to coupling as it refers to the “relatedness” of the internal parts of a component. A high cohesion means that the component can hardly be split into separate components, because the internals are working together very close. A low cohesion means that it is possible to identify subcomponents within a component that have low coupling. That is why coupling and cohesion are closely related to each other.

Most interesting for an approach taking care of the quality of architectures are the formal metrics that are defined for coupling and cohesion. In [Bieman94] the quite weakly defined term of cohesion is now defined by the number of data tokens that are used in more than one slice of a program respectively component. A program slice is the portion of program text that affects a specified program variable (compare [WeiseR91]). There are deterministic rules how to compute these figures. The result is a comparable degree of cohesion for components. The idea of measuring quality with formal methods makes sense if the automation helps to examine a great number of components. This approach could be used analogously for enterprise architectures and their quality attributes, as they are often very large.

2.2.2 Comparison of EAM definitions

Having presented several approaches concerning EAM, the characterizing dimensions of EAM are formulated in order to be able to compare the existing definitions.

At first, the term management is examined a little further. Typical for management in many areas (e.g. motor management or total quality management) is the establishment of a management system. Such a system implements the management cycle as depicted in Fig. 2-17.

The management cycle is closely related to a concept widely spread and used in electrical and mechanical engineering – the feedback control system. Without the last step of measuring the results, it is comparable to an open loop control system. An example in mechanical engineering is a car going at constant speed. Without measuring the speed constantly, it is impossible to hold the same speed all the time. The only thing the driver can influence is the angle of the gas pedal. However, the same angle of the gas pedal does not guarantee a constant speed, as there are too many speed-influencing factors. The speed of a car is influenced by the road incline, wind,

air pressure influencing the efficiency of the engine, and so on. The driver – or the cruise control system – needs to measure the current speed and to adjust the gas pedal to the right angle. Only then, the desired speed will be kept over time. For the enterprise architecture, this means that all four steps of the management cycle have to be implemented.

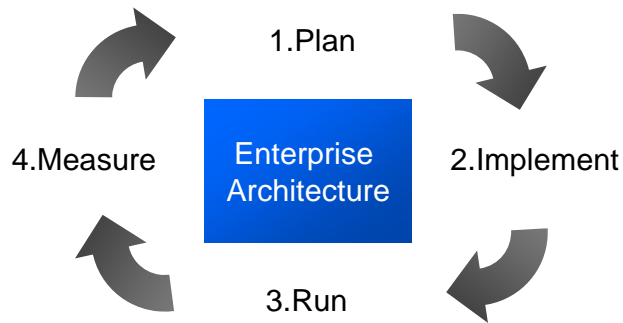


Fig. 2-17: The management cycle

The first step is to plan the object to be managed. Goals and strategies have to be formulated that steer the transformation direction of the managed object. It will be hard to find adequate ways to change the current situation so that it will fulfil given requirements without making a plan of complex structures. Planning is usually influenced by architecture strategies. Following the SOA style is one of them.

The second step is to implement what was planned before. Again modelling is helpful for this task, as concrete changes should be derived from a model and packed into portions of adequate size.

The third step is running the implemented solution. In the context of enterprise architecture, it means to operate the enterprise. This lies out of the responsibility of the management system.

The fourth step is crucial for management systems. In order to control whether the previously formulated strategies have been fulfilled, a measurement has to take place, so that the planned system can be compared to the actual state..

Planning, implementing and operating are what is mostly done anyways, but measuring and controlling the result is not a matter of course. On the one hand, the implementation of concrete changes has to be measured and on the other hand, it has

to be checked whether all the architecture strategies have been followed. Controlling the strategy gets more and more important the more complex the managed structure is. Usually, enterprise architectures are highly complex.

The first characterizing dimension of EAM is the planning coordination that is primarily related to step one. On the one hand, there can be several instances in an enterprise planning the enterprise architecture. On the other hand, this can be done centrally. A centralization of planning is suggested in all approaches. Only in enterprises not following an EAM approach, the planning is often decentralized.

Most of the characterizing dimensions are leaned to this definition of a management cycle. The first characterizing dimension is also related to step one and identified as the *planning strategy*. It concerns the way to find change projects and has two main alternatives. The defensive alternative is to wait on opportunities for a change. When an element of the enterprise architecture is created or changed due to pure functional reasons, then it is implemented in a way that it fits in the target architecture. The more aggressive alternative is to reengineer the enterprise architecture. Reengineering requires to model the as-is state and to have a target state for the architecture.

The second characterizing dimension is the *change strategy* related to step two. What size are the steps to be implemented? A complete rebuild is possible if the existing system is not very complex or very different from the target system. Furthermore, partial rebuilds exchanging whole components of a system are possible. The last option is the smoothest way, as it suggests a steady evolution of the system.

The third characterizing dimension is the *architecture strategy formulation*. A strategy is usually formulated informally or not at all. However, there is also the possibility to formulate an architecture strategy as a metric like in the case of the CBA concepts coupling and cohesion.

This leads to the fourth characterizing dimension - the *measurement of architecture strategy*. The measurement can be omitted, expert-based or be supported in an automated way. Even for experts it is a challenging task to decide whether a strategy is realized or not if there are no concrete metrics for the strategy. A strategy formulation that is machine-readable is a premise for the automated measurement of architecture strategy realization. If this is given, the measurement of architecture strategy could be automated.

	Planning strategy	Change strategy	Architecture strategy formulation	Architecture strategy measurement
[DernGe03]	Reengineering	Evolution	Informal	None
[Engels08]	Reengineering	Evolution	Informal	None
[Matthe08]	Reengineering	Evolution	None	None
[Nieman05]	Reengineering	Evolution	Informal	None
[Haren07]	Reengineering	Evolution	Informal	None
[Bieman94]	-	Partial rebuild	Metric-based	Expert-based

2.2.3 Choices for EAM-specific Dimensions

Category	Alternative	Adequacy
Planning strategy	Opportunity	-
	Reengineering	+
Change strategy	Rebuild	-
	Partial rebuild	O
	Evolution	+
Architecture strategy formulation	None	-
	Informal	O
	Metric-based	+
Architecture strategy measurement	None	-
	Expert-based	O
	Automated	+

Fig. 2-18: Adequacy of alternatives for characterizing dimensions of EAM

This subsection covers the identification of adequate alternatives of the characterizing dimensions of EAM. The results are depicted in Fig. 2-18.

The first characterizing dimension is the *planning strategy*. A minimalist's choice would be to make use of opportunities. If an artefact of the enterprise architecture shall be changed or build anyway, then the new artefact can be designed in a way that is conform to the new target architecture. However, this approach is not very effective, because old components that are hindering others might not be changed. Therefore a reengineering approach with the modelling of an as-is and target architecture is suggested.

Transforming something into something different, can be done following different *change strategies*. If the existing system is not very complex or very different from the target system, a complete rebuild can be considered. For an enterprise architecture, this is not an option because the existing system is much too valuable to be wasted in favour of a completely new architecture. For software that is build out of components sometimes a partial rebuild is done. If a component is written in an undesired language or the fixing of the known bugs would take more effort than rebuilding, then single components might be built from the scratch again. This strategy is also hardly favourable for enterprises as the existing system and the target system will usually have too much commonality. All of the analyzed references suggest a rather smooch advancement in transforming the enterprise architecture. This smooth advancement is here referred to as evolution. It is regarded as the best option for the introduction of an SOA.

The *formulation of the architecture strategy* is often done informally. The disadvantage of strategies or goals that are defined informally is that it is hard to tell whether they are fulfilled or not. This also depends on the understanding of the individual employee. For this reason, the formulation of metrics for strategies is helpful as there is less freedom for interpretation and the result is measurable.

The *architecture strategy measurement* is not considered in EAM approaches. However, its omission may lead to planned actions that do not follow the wanted strategy. For this reason, the measurement is regarded as essential. There is a disadvantage if experts try to follow these strategies with their actions. For complex structures and complex strategies, it gets harder to have everything in mind and to evaluate the situation in the right way. This is why an automated support for strategy measurement is suggested.

Having found the desired alternatives for Enterprise Architecture Management, they are illustrated in Fig. 2-19. Finally yet importantly, SOA has to be taken care of, which is done in the following two sections.

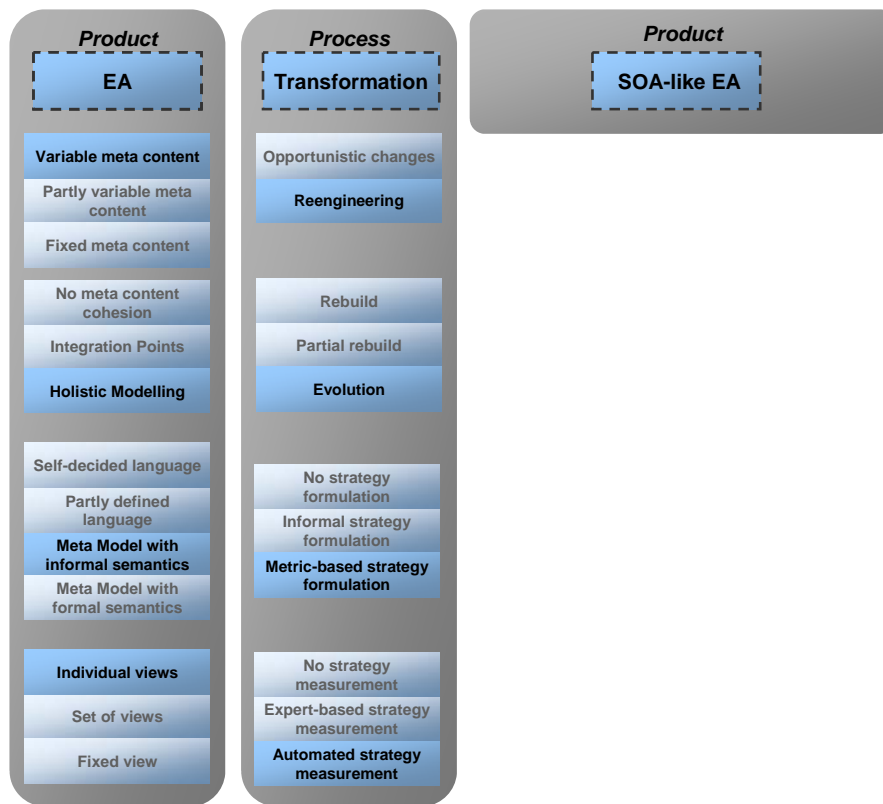


Fig. 2-19: Preferred characterization alternatives for EA and EAM

2.3 Service-Oriented Architecture

The existing SOA definitions are presented and compared concerning their characterizing dimensions in this section. In the manner of the previous section, the SOA definitions are examined and compared concerning their characterizing dimension. Afterwards, the choice of the best alternatives is discussed.

2.3.1 About the Service Paradigm and SOA

Defining Service-Oriented Architectures, the term service has to be defined. At least since the SOA-hype started, the term service has several meanings in an enterprise. On the one hand, there are services in the business sense that are brought by an enterprise for a client. These are mostly complex processes with mandatory accounting and specific Service Level Agreements (SLA). On the other hand, there are SOA services, whose character will also be enlightened in this section. In the following, service and

SOA definitions from the sources [OASISR06], [Dostal05], [WoodsDo6], [Sieder07], [ErlTho06], [ErlTho09], [Krafzi06], [Winter08], and [Bianco07] are presented.

From the Organization for the Advancement of Structured Information Standards (OASIS) point of view, which has established an official standard with its SOA Reference Model (compare [OASISR06]), a service is strongly related to needs and capabilities, because it should bring them together. A capability is the ability to perform work. Capabilities already exist in the environment of an enterprise, they are owned by IT-systems, people and organisations.

Though capabilities already exist, services as meant in the context of SOA (SOA services) not necessarily exist in the same enterprise. A service is more than a capability, as it makes one or more capabilities accessible by an interface and disposes of a specification of the work that it can perform, included in a so-called service contract. If SOA is new to the reader, it is recommended to read the OASIS Reference Model (compare [OASISR06]). It provides a vocabulary for Service-Oriented Architectures and allows people to achieve a common understanding when talking about services.

“A service is a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description. A service is provided by a service provider. Moreover, the service is to be consumed by a service consumer. However, the potential consumers of the service might not be known to the service provider in advance. Furthermore, the consumers could make use of the service beyond the scope that was originally conceived by the provider. A service is accessed by means of a service interface, where the interface comprises the specifics of how to access the underlying capabilities. There are no constraints on what constitutes the underlying capability or how the service provider implements the access to the service.

Thus, the service could carry out its described functionality through one or more automated and/or manual processes. These could invoke other available services. The consequence of invoking a service is a realization of one or more real world effects. These effects may include:

1. Information returned in response to a request for that information,
2. A change to the shared state of defined entities, or

3. Some combination of (1) and (2).

Note, the service consumer in (1) does typically not know how the information is generated, e.g. whether it is extracted from a database or generated dynamically. In (2), he does typically not know how the state change is effected.

The service concept above emphasizes a distinction between a capability that represents some functionality created to address a need and the point of access where that capability is brought to bear in the context of SOA. It is assumed that capabilities exist outside of SOA. In actual use, maintaining this distinction may not be critical (i.e. the service may be talked about in terms of being the capability) but the separation is pertinent in terms of a clear expression of the nature of SOA and the value it provides.”

(OASIS Reference Model [OASISR06])

The OASIS is a global consortium that drives the development, convergence, and adoption of e-business and web service standards. It does not clearly describe how a Service-Oriented Architecture itself looks like as it provides a reference model only. Realizations of it can have many different faces. The OASIS regards SOA as an architecture that follows the concepts of visibility, interaction, and effect. Furthermore, there are entities (people and organizations) offering capabilities and acting as service providers. The ones who make use of services are called service consumers. The service description allows prospective consumers to decide if the service is suitable for their current needs and establishes whether a consumer satisfies any requirements of the service provider.

Dostal ([Dostal05]) firstly specifies three actors in a Service-Oriented Architecture; the service provider, the service requester and the service registry as shown in Fig. 2-20. This well-known service triangle is also quite abstract, but depicts a basic idea of the service paradigm.

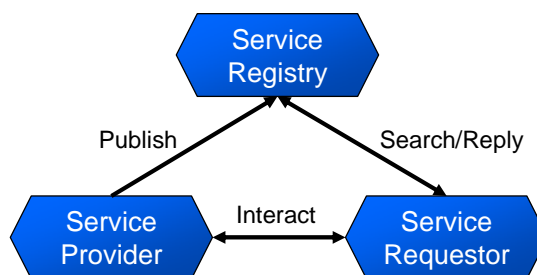


Fig. 2-20: Web Service Triangle from [Dostal05]

At first the providing service is published in the registry, afterwards the requestor searches the registry and hopefully finds a suitable service for its demands and can then interact with the provider. A service, as defined by Dostal, is a program or a software component underlying the concept of information hiding. This means access may only happen via a publicly described interface. This form of encapsulation is compared with the plug-in concept that is similar but less useful due to inflexible interfaces.

A service should be represented by a web service, which needs a properly described interface from the point of view in [Dostal05]. The Web Service Description Language (WSDL, compare [Christ01]) is the method of choice so far, but semantic issues are not properly covered by that and therefore additional documentation is useful.

In [WoodsDo6] SOA is motivated with the need of new processes that have arisen in enterprises due to changing markets and globalization. These new processes span several segments in an enterprise. Moreover, existing systems like SCM and CRM do not properly support the processes as the systems focus on their segments only.

According to [WoodsD06] the flexibility demand increases with the shortening of life cycles. During the last hundred years process execution times, product life cycles and process life cycles (change management) have drastically decreased. Especially the shortening of process life cycles is hard to realize with traditional enterprise architectures, however these maybe efficient but just as long as flexibility is not required.

Because of ongoing changes in the enterprise environment, more flexibility of IT is demanded that can only be brought by a new architecture. That architecture should solve integration needs so that application-spanning workflows should be easily realizable. According to [WoodsD06] an architecture not holding process control in a separate and maintainable entity but having it scattered over applications (where it is often even hard-coded) and humans will hardly prove out to be efficient in the next decade.

According to Siedersleben, SOA is a concept providing an architecture for (software) system landscapes (compare [Sieder07]). System landscapes are characterized by their evolving structure just like networks of motorways. They are never finished and never perfect. In such a network, road works permanently exist and

sometimes previously made design decisions turn out to be false afterwards. Coping with these facts alone demands a flexible architecture, but in addition, system landscapes contain redundancies that are not wanted but are inevitable up to certain extent and system landscapes are massively heterogeneous. It is desirable that an architecture minimizes the negative impact of these characteristics. Therefore, Siedersleben suggests three attributes for an appropriate architecture. Firstly, systems are loosely coupled which means they communicate asynchronously and react robust on failures of other systems. Secondly, systems exclusively communicate over well-defined interfaces and thirdly the workflow control is a separate component, so that systems do not need to “know” much of each other. The service is an ideal element to support these attributes.

Thomas Erl ([ErlTho06]) defines services by their attributes that clearly belong to his principles of service orientation. Services (should) adhere to the principles of reusability, formal contract, loose coupling, abstraction, composability, autonomy, statelessness, and discoverability. Erl thinks of services as a way to offer work for other entities following the previously named principles.

Thomas Erl states that a Service-Oriented Architecture cuts into four logical components:

- Messages
- Operations
- Services
- Processes

The first item, messages, represents units of communication containing the data required to complete some or all parts of a unit of work. An operation stands for a simple unit of work. Several units of work comprise to a unit of processing logic called a service. This service represents the logic required to process messages in order to complete a unit of work. Finally yet importantly, a process is the coordinated aggregation of units of work and contains business rules that determine which service operations are used to complete a unit of automation. Fig. 2-21 depicts the relationship between these components:

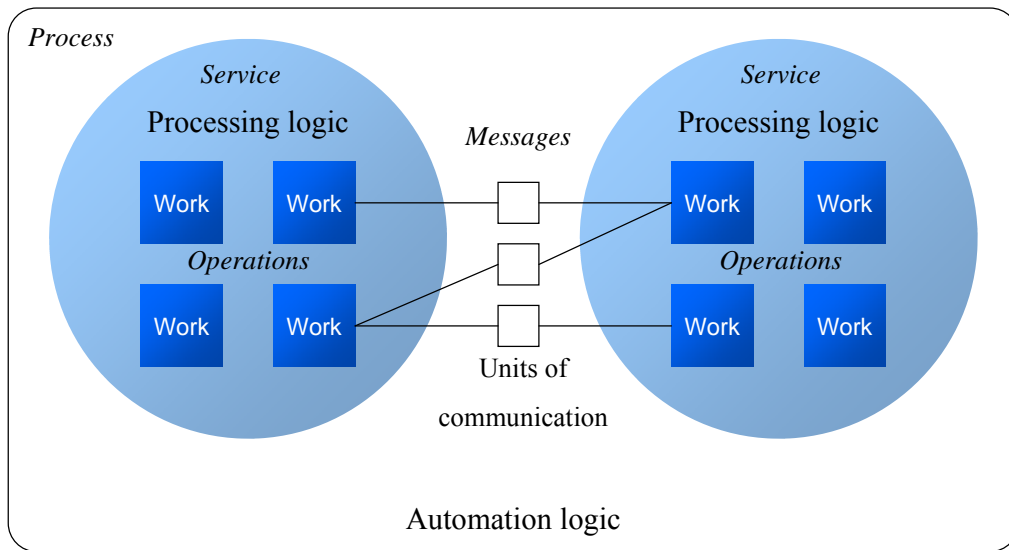


Fig. 2-21: Primitive view of how SOA modularizes automation logic as in [ErlTho06]

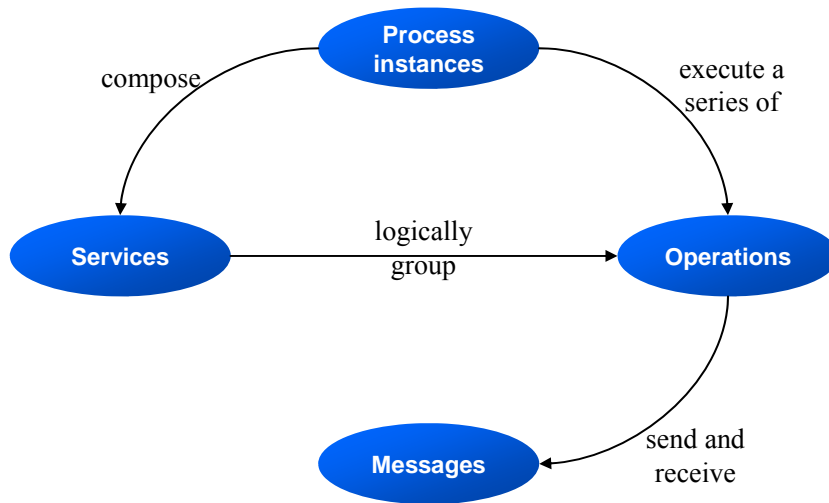


Fig. 2-22: How components of the SOA relate [ErlTho06]

As the figure above shows, operations send and receive messages to perform work. A service logically groups a collection of related operations. A process instance can compose services and at the same time may only require a subset of the functionality offered by the services.

The SOA Design Pattern Catalogue [ErlTh09] contains over 80 patterns for the design of a Service-Oriented Architecture. There are basic and compound patterns that

consist of several basic patterns. The patterns are described following an informal schema that can be seen in Fig. 2-23:

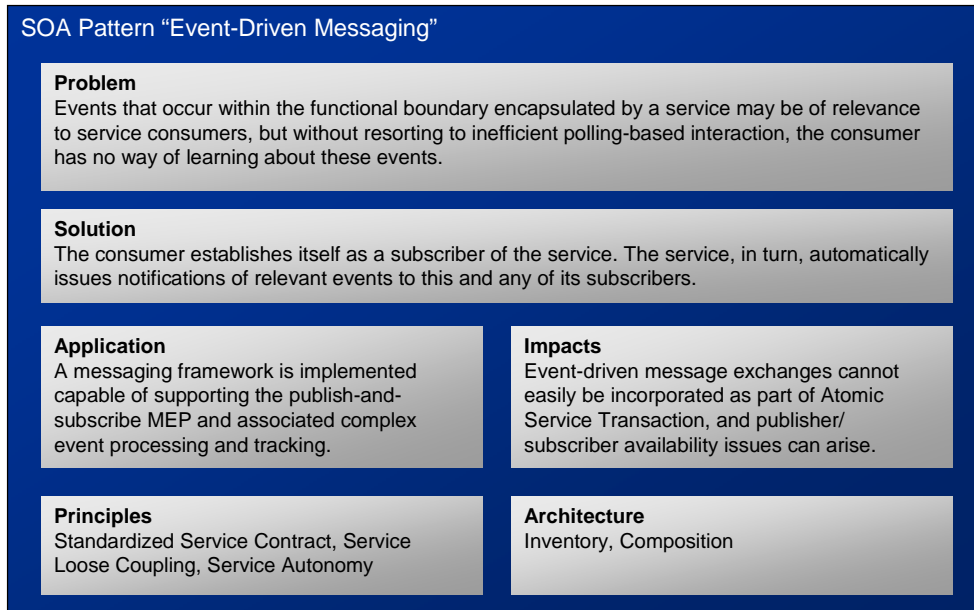


Fig. 2-23: SOA Pattern example from [ErlTh09]

These patterns can be useful to get an idea in where the development of single aspects of an SOA landscape should be directed. This is very useful if a single problem occurs that fits to a pattern. However, within the development of a system landscape, there are always several topics to be covered at the same time and these will have intersections. Therefore, the problem of system landscape is not solved completely with these patterns but they help choosing the right direction.

Another author being observed here is Krafzig ([Krafzi06]) who states that a service is built up as shown in the figure below. Krafzig’s service definition is more detailed and therefore less abstract.

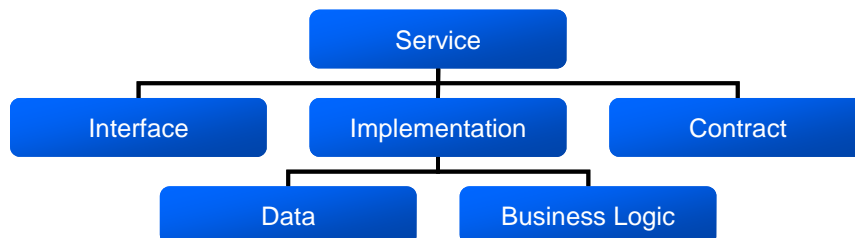


Fig. 2-24: Elements of a service as in [Krafzi06]

The contract includes all descriptions of the service. The form of these descriptions is not fixed at all and does not have to be formal, though formal descriptions like IDL or WSDL for interfaces may have great benefits. The contract describes:

- Interface
- Purpose
- Constraints
- Functionality
- Availability
- Accessibility
- Visibility

The interface, as described in the contract, enables interaction between service provider and service consumer. Functionality is made accessible by it. The implementation provides business logic and data and thereby fulfils the contract.

A service is a black box from the client perspective. It typically encapsulates a high-level business concept and consists of several parts shown in the figure above. They are coarse grained and impose a strong vertical slicing of the underlying applications.

From Krafzig's point of view, the whole concept of SOA focuses on the definition of business infrastructure. When using the term service a business service, like get reservation, is meant. Services provide the structure of the SOA because they often remain unaltered while frontends, service implementations, as well as business processes often are subject to change. Krafzig's main elements of an SOA are shown in the figure below:

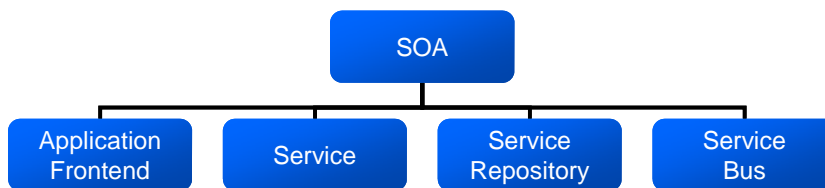


Fig. 2-25: Main elements of an SOA as in [Krafzi06]

Application frontends are the active players that initiate activities of enterprise systems. Usually they enable users to invoke services and receive the results. Services are governed in the services repository, i.e. they are registered there and their meta

information is given to requesting service users. The service bus connects all participants with each other.

The introduction of SOA in enterprises is not only a technical problem but also a governance problem. As it takes investments to establish an SOA, like hiring SOA specialists buying tools and training people, managers are often sceptical. As long as they do not see the monetary benefits, they are usually hard to convince. For this reason, “The economic justification of Service-Oriented Architecture” is covered in [Winter08] and in [Assmanb09].

The authors of the study in [Winter08] have identified two general approaches for a business case. The first one is the SOA infrastructure business case and the second the SOA business platform business case. The former concentrates on the benefits of the technical infrastructure and the IT organization. This means that the reuse of services and the reduced software development costs stand in the foreground. The business case for the business process platform includes the infrastructure approach and additionally covers a broader more comprehensive way of evaluating SOA. The concept for the business process platform includes service-enabled solutions, business intelligence, and a unified technology foundation.

The two business case approaches come with a blueprint covering quantitative and qualitative benefits as well as upcoming costs. Within the blueprint for the infrastructure, business case the quantitative benefit section covers categories like development efficiency, maintenance efficiency, application lifecycle extension, and consolidation. A category contains an example for benefits and metrics, like less component development effort as measurement and percentage of overall development costs as a metric. An example for a qualitative benefit is business IT-alignment. Costs are categorized like quantitative benefits with categories like hardware/software costs and IT change management.

The blueprint for the business process platform additionally contains the quantitative benefit categories business process quality and productivity, innovation and insight. Qualitative benefits cover the categories Time-to-market, mergers and acquisitions, and business network transformation.

The business case blueprints presented in [Winter08] can be used as basis to create an SOA business case in an enterprise. It covers an important governance topic, namely convincing the management of the SOA concept. In the context of this work, it

can be seen as a tool to create the right circumstances for an SOA introduction so that the use of sophisticated tools and methods will be financed.

In the technical report [Bianco07], the ATAM Framework (Architecture Tradeoff Analysis Method) [Kazman00] is used to evaluate Service-Oriented Architectures. The authors present several quality criteria that can be used to evaluate an SOA. As the report was developed using the ATA Method, it shall provide means for a quick analysis in the beginning of a project but is not intended to cover a whole lifecycle as EAM intends to.

The SEI definition of SOA refers only to a landscape that is implemented with services, whereas services can be implemented by technologies like Web Service or CORBA. Advanced concepts like Event-Driven Architecture (EDA) and Complex Event Processing (CEP) are only mentioned as an outlook on emerging technologies. With this quite narrow definition of SOA, the authors state that SOA descriptions are best given as a run time view – the component & connector view as proposed in [Kazman00].

For the focused concepts, which are mainly services interfaces, Enterprise Service Bus (ESB – as middleware concept) and orchestration, interesting insights are given. Firstly, a high-level technology discussion is lead. The first question discussed is whether service communication should be implemented with SOAP, REST or other messaging solutions. Furthermore, the discussion covers the topics service middleware (ESB or point-to-point) and service binding (static or dynamic). The topic service orchestration is supposed to be realized with BPEL, which is a common way to implement orchestrations but not the only one (compare “hard-wired orchestration” in section 4.1). These discussions give an overview for architects which technologies are mainly used today and what could be the right solution for its ones own requirements.

In addition to the technology discussion, questions concerning quality attributes are given. The quality attribute is discussed and sample evaluation questions are given for each question. The examined quality attribute topics are:

- Target platform
- Synchronous versus asynchronous services
- Granularity of services
- Exception handling and fault recovery
- Security

- XML optimization
- Use of a registry of services
- Legacy systems integration
- BPEL and service orchestration
- Service versioning

Unfortunately, [Bianco07] lacks the full relation to Enterprise Architecture Management, but still gives interesting and useful insights on quality questions of Service-Oriented Architectures.

2.3.2 Comparison of Existing Definitions

As to be read on the previous pages, SOA and service definitions are nearly as manifold as the general statements given in the section before. The authors are using different ways to define Service-Oriented Architectures. The characterizing dimensions are presented in the following.

The most general and abstract picture for SOA is given by the [OASIS06] followed by [ErlTho06], [Dostal05] and then [Krafzi06]. Many authors do not establish a relationship to enterprise architecture but focus on abstract descriptions like [OASISR05] or technologies suitable for SOA like [Bianco07]. At the same time [Bianco07] presents an evaluation method for SOAs, which is implicitly a relation to Enterprise Architecture Management. Others like [Krafzi06] give describe SOA in an enterprise context. The *definition context* is the first dimension for SOA definitions with the alternatives abstract, technical, enterprise-oriented, EA-oriented or mixed.

Furthermore, there is a parallel of the SOA definitions to EA definitions - the formalization. For EA the formalization of the model later used had to be defined. Here, only the *definition formalization* is of interest. These alternatives are a meta model as in [OASISR05] or informal descriptions as in [Krafzi06]. Thus, the alternatives are formal syntax descriptions or informal description.

Another characterizing dimension of SOA definitions are their *SOA service definitions*. The SOA service is the building block or in other words the structuring element of the Service-Oriented Architecture. The service of course does not exist for self-purpose. There are always entities that consume services and those that provide services. In [OASISR05], [Dostal05], and [Krafzi06] these are mentioned namely. Erl does not mention these roles, but also regards service as work that is offered by some

entity and is consumed by some other entity. Next to this purpose, descriptions always include that an SOA service has a description or contract, an implementation, and one or more interfaces. The differences are shown in the formulation of quality attributes for services. For now, it is distinguished between none, few and comprehensive.

The term service registry is always used for describing an entity that allows searching for registered services. Sometimes it is also called service repository. In this thesis, it is distinguished between the terms registry and repository. A service registry is found in nearly every description of an SOA, though the implementation varies and reaches from documents to sophisticated search engines. The OASIS Reference Model does not directly include something similar but this is because it is more general and abstract. It demands the visibility of services, which means that there have to be mechanisms that allow entities to find services. The implementation of such mechanisms is most probably represented by a service registry.

The term service repository is used in [Krafzi06] as one of the elements forming an SOA. He uses this term equivalent to service registry. However, according to several major software provider like HP, Software AG and IBM a repository contains a service registry and is far more than that. The functionality is similar to the functionality of a Configuration Management Database plus the functionality of a service registry. A repository manages user defined configuration items including SOA services, of course. These items can be attached with document links, they can be versioned, they can have statuses, and on top, a workflow management system can be used that implements internal processes working on these configuration items. Thus, a service repository is far more than a service registry.

Due to their close relation, the service registry and the service repository build a characterizing dimension – the *discoverability support*. The alternatives are none, registry only, and repository (the repository includes the registry).

The next dimension is the *middleware*. The term Enterprise Service Bus (ESB) is mentioned in some definitions. In [OASISR05] and [ErlTho06], it is not mentioned at all, but many other authors regard it as an essential element and all important software vendors sell according products. Definitions of the ESB are sometimes vague and always concern the communication of SOA services. The alternatives in this dimension are not mentioned, middleware, and ESB.

Moreover, there is a dimension that is named *workflow management* here. This term is rarely mentioned in definitions, but instead the term orchestration is used the more. The orchestration can be restricted to services that are calling each other in their program code (hard-wired) and through an engine that is able to interpret workflow or orchestration languages (soft-wired). The option not to mention workflow management is also possible.

The seventh characterizing dimension is *Event-Driven Architecture (EDA)*. The idea behind is that asynchronous messages are sent when services are invoked and have done some work. This concept can be supported by a complex event processor that correlates different events according to rules specified or by a broker forwarding events to subscribers. The alternatives are not existent, event broker and complex event processor.

The last dimension is *business process monitoring* that only rarely appears in SOA definitions. From all other authors' statements emerges the same impression (at the latest when looking at the detailed architecture descriptions) that services abstract from applications and are composed to processes. That means processes are covered in most definitions but not their management. Business process monitoring is an important concept of a Service-Oriented Architecture that may be explicitly supported by the Event-Driven Architecture by using the events as information source for business process monitoring. The alternatives for it are none, and simple and EDA supported.

In the following tables, the definitions are compared and evaluated concerning the characterizing dimensions defined. The best choices of alternatives for the approach of this thesis are discussed in the next subsection.

	[OASISR06]	[Dostal05]	[WoodsD06]	[Krafzi06]
Definition context	Abstract	Technical	Enterprise-oriented	Enterprise-oriented
Definition formalization	Meta model	Informal	Informal	Informal
SOA Service definition	Few	Few	Few	Few
Discoverability support	Registry	Registry	Registry	Registry
Middleware	Middleware	ESB	Middleware	ESB
Workflow	Soft-wired	Soft-wired	Soft-wired	Soft-wired

Management				
Event-driven Architecture	Event broker	Event broker	Complex Event processor	None
Business process monitoring	None	None	Simple	None

	[ErlTho06]	[Sieder07]	[Winter08]	[Bianco07]
Definition context	Abstract Technical	Enterprise-oriented	Enterprise-oriented	Technical, EA-oriented
Definition formalization	Informal	Informal	Informal	Informal
SOA Service definition	Comprehensive set	Few	None	Few
Discoverability	Registry	Registry	Repository	Registry
Middleware	Middleware	ESB	ESB	Middleware
Workflow Management	Soft-wired	Soft-wired	Soft-wired	Soft-wired
Event-driven Architecture	Event broker	Event broker	None	None
Business process monitoring	None	EDA supported	simple	None

2.3.3 Choices for SOA-Specific Dimensions

Category	Alternative	Adequacy
Definition context	Abstract	-
	Technical	-
	Enterprise-oriented	O
	EA-oriented	+
	Mixed	O
Definition formalization	Informal	-
	Meta model	O
SOA Service definition	Without quality attributes	-
	With few quality attributes	O
	With a comprehensive set of attributes	+
Discoverability	None	-
	Registry	O
	Repository	+

Middleware	None	-
	Middleware	+
	Enterprise Service Bus	+
Workflow Management	None	-
	Hard-wired	O
	Soft-wired	+
Event-driven Architecture	None	-
	Event broker	O
	Complex event processor	+
Business process monitoring	None	-
	Simple monitoring	O
	EDA-based monitoring	+

Fig. 2-26: Adequacy of alternatives for characterizing dimensions of SOA

Fig. 2-26 shows the different alternatives for the characterizing dimensions of Service-Oriented Architectures.

For the task given in this thesis it should be clear that EA-oriented *definition context* for the SOA is desirable. This means that SOA is clearly regarded as a style of EA and its definition is based on concepts concerning the EA. Definitions that at least use an enterprise context and an adequate vocabulary are suitably to a limited extent only.

With respect to the formalization of the EA modelling the SOA *definition formalization* should also be given, so that the benefits of the formalization can be used for an integrated approach. Informal definitions without a meta model will be hard to integrate into an SOA-like EA meta model.

The *SOA service definition* should be based on a comprehensive set of quality attributes, because this is a premise for a measurement of their quality (needed in EAM context).

The *discoverability* of SOA services should at least be supported by a registry. The better choice is a repository supporting further tasks in the service lifecycle.

The *middleware* should be provided either by a middleware technology like web service or CORBA or by an ESB. According to some definitions, the ESB is a little more than just a middleware, as it may enforce simple rules and monitor service

usage. These functions could also be implemented with the help of an Event-Driven Architecture. Therefore, it is as useful as a simple middleware if an EDA is present.

The *workflow management* should be soft-wired in every case. It offers the most flexibility and probably because of this, none of the examined references suggests something different.

The approach of the *Event-Driven Architecture* should be realized with a complex event processor that is able to analyze complex patterns in event occurrences. This is very beneficial for business process monitoring.

The *business process monitoring* should be based on the complex event processing, because there is probably no other source of information that delivers the pieces of information so fast and so easy from all parts of the IT-landscape. Without EDA-support, it will be hard to retrieve all the needed information in the right quality.

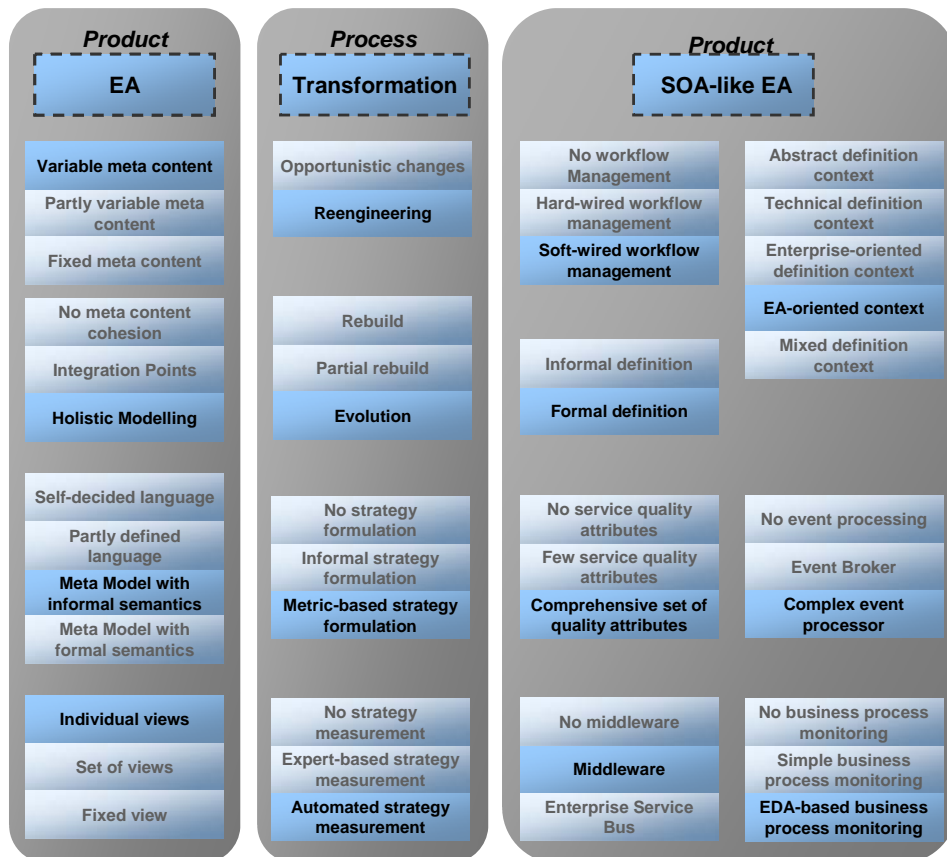


Fig. 2-27: Preferred alternatives for dimensions of EA, EAM and SOA

Fig. 2-27 depicts the preferred alternatives for all three topics examined. From these chosen alternatives, the requirements for the realization of the approach of this thesis are derived in the next section.

Having examined all the relevant topics for the thesis task, the requirements for the approach can be derived from the chosen alternatives.

2.4 Requirement Derivation

In the previous sections, the best alternatives of the characterizing dimensions were chosen. In order to realize the approach, the final requirements are derived in this section. Fig. 2-28 illustrates the overview on the requirements derived from the chosen options.

Firstly, the requirements arising from the chosen EA dimension alternatives are identified. Supporting an *individual views* is a requirement as-is. The description of the enterprise architecture with a structured syntax means that there has to be an *enterprise architecture formalization* in form of a meta model. As enterprises are very individual in their structure, there should not be a fixed meta model for all enterprises. Instead, the formulation of an *individual EA meta model* should be possible. The *holistic modelling of the enterprise architecture* is also directly taken as a requirement. Nearly all the requirements so far should be concerned in the *tool support*. The individual meta model does not because it will be overridden by another requirement (integrated language for EA and SOA).

Secondly, the EAM alternatives are focused. The evolutionary character of the approach together with the quite low SOA experience level of many architects leads to the requirement of a *methodical approach*. The methodical approach formulates a sequence of actions to get to the desired result. As the evolutionary approach requires many small improvement steps, *improvement suggestions* on how to redesign the enterprise architecture are noted as a requirement. The reengineering approach also benefits from these suggestions. Moreover, reengineering requires the *as-is and target modelling*. The metric-based strategy formulation requires the formulation of *SOA conformance criteria*, as SOA is to be seen as the strategy (or style) for the enterprise architecture. Improvement suggestions and as-is & target modelling should be tool supported. The methodical approach cannot completely be tool supported, as the creation of an individual tool might be part of the method. The SOA conformance

criteria should also be tool supported but this is a premise for the requirement automation of criteria checks that will be identified in the next paragraph.

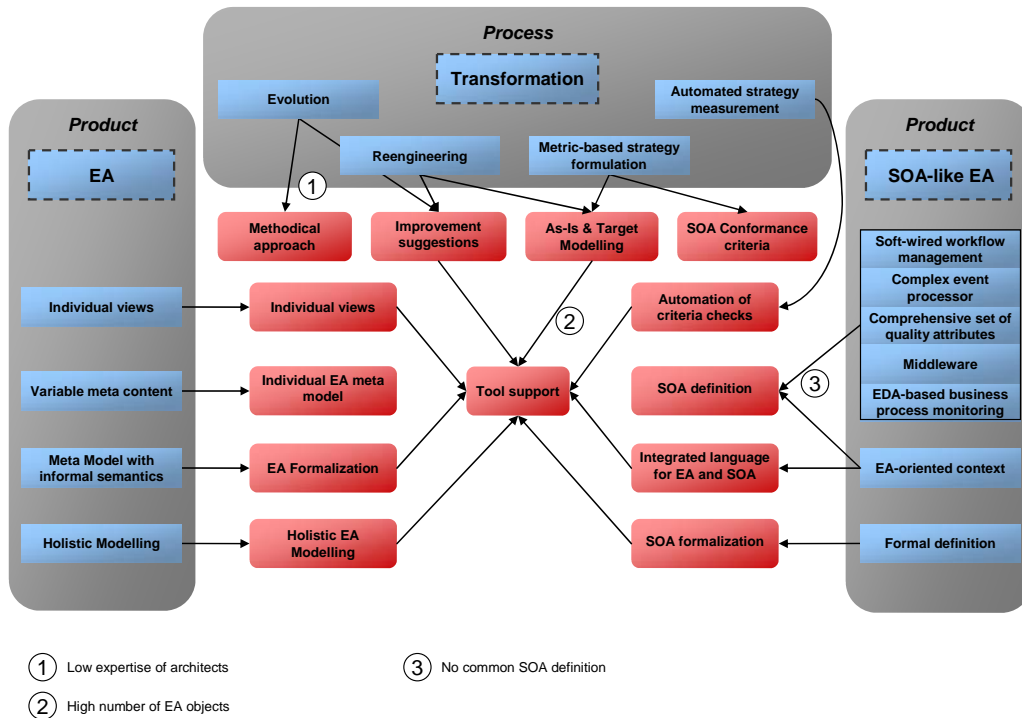


Fig. 2-28: Requirements derived from preferred characterization alternatives

A reengineering approach is based on the idea of *as-is & target modelling*. Due to the high number of assets in an EA (2), a tool should supported the modelling. As suggested in [Engels08] there should the picture of an ideal landscape acting as a lighthouse for the target landscape (compare Fig. 2-29). Here, the explicit modelling of the target architecture is not followed, but the target architecture is defined by a set of conformance criteria. An enterprise architecture fulfilling all criteria would be equivalent to the ideal landscape. The decision for the criteria approach lies in the easier and continuous measuring of the goal achievement. With an ideal landscape model and an existing landscape, a comparison of both models had to be done each time something is changed. In this thesis, the approach of formulating *SOA conformance criteria* is favoured, because these can be continuously checked and separately formulated. Being continuously checkable means the *automation of conformance criteria checks* is implemented based on the formally described EA model.

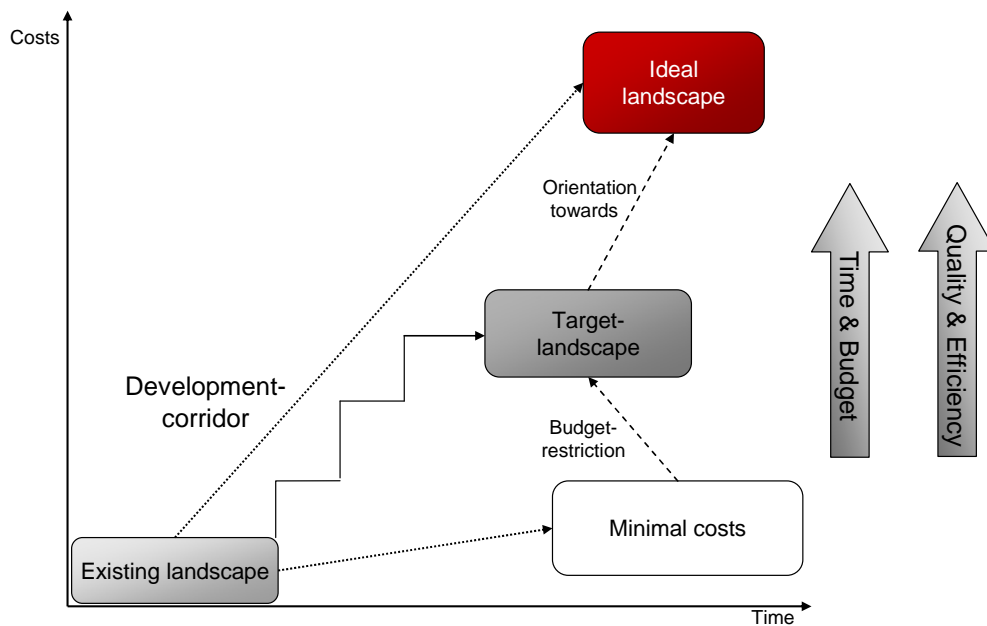


Fig. 2-29: The development corridor based on [Engels08]

Thirdly, the requirements derivable from the SOA alternatives are concerned. If the conception of an SOA is decoupled from the enterprise architecture and the EA management system, then the restructuring of the EA to a service-oriented style is unnecessarily complicated. As SOA is a style of enterprise architecture, it has to be defined in the same terms as enterprise architecture itself. That means a common language has to be found, which is able to express SOA conformance criteria in terms of enterprise architecture. Therefore, an *integrated language for SOA and EA* is required.

The alternatives from soft-wired workflow management to EDA-based business process monitoring are all concerning the content of the SOA definition. As none of the given definitions covers all of the SOA contents, a definition is given in this thesis. The result shall be an EA-oriented SOA definition.

The *formal definition of SOA* is also a requirement, because it is a premise for the automated checks of the SOA conformance criteria and for the integration with the formal EA meta model.

From now on, the requirements will be referenced with R followed by a number. The sequence was not chosen randomly but with respect to categories that have been found for them. The categories of requirements are depicted in Fig. 2-30.

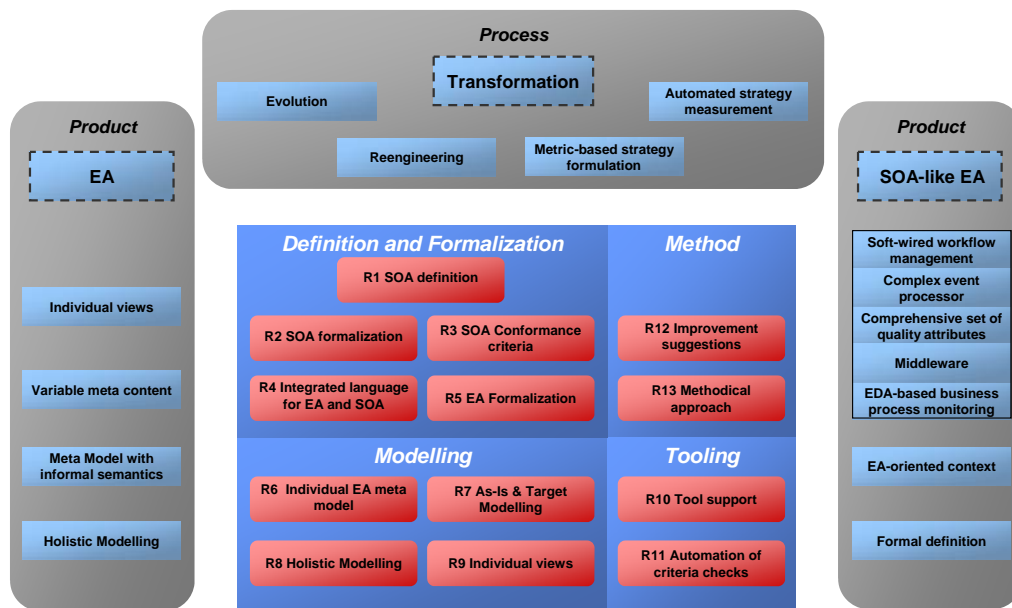


Fig. 2-30: Categorization of requirements

For the sake of readability, the requirements are also given in form of a list:

- R1 SOA definition
- R2 SOA formalization
- R3 SOA conformance criteria
- R4 Integrated language for EA and SOA
- R5 EA Formalization
- R6 Individual EA meta model
- R7 As-Is & Target Modelling
- R8 Holistic EA Modelling
- R9 Individual views
- R10 Tool support
- R11 Automation of criteria checks
- R12 Improvement suggestions
- R13 Methodical approach

In this section the requirements derived from the three different fields EA, EAM and SOA were elaborated, which are the foundation for the rest of the thesis. In the next section the approaches in the field of EAM and SOA is evaluated concerning the fulfilment of these requirements.

2.5 Evaluation of Related Work

This section examines a part of the related approaches that have been previously presented. This time they are not examined concerning a single basic concept but concerning the whole set of requirements presented in the previous section. In the following the approaches of TOGAF, Quasar Enterprise, Zachman Framework, ISA, and the approach from the SEI are concerned.

The fulfilment of a criterion is reflected with the help of three categories. The options not fulfilled, partly fulfilled, and mostly fulfilled are possible. Every option covers a third of the range of values. That means if less than a third of a requirement is fulfilled it is rated as not fulfilled, between a third and two thirds it is rated partly fulfilled and above two thirds it is rated as mostly fulfilled.

Zachman Framework

- **R1 SOA definition**
The Zachman framework was created in the late 80's and since it was never modified concerning SOA-principles. This is the reason why the Zachman framework does not have SOA elements defined. Thus, R1 is not fulfilled.
- **R2 SOA formalization**
If R1 is not fulfilled then R2 cannot be fulfilled either.
- **R3 SOA conformance criteria**
If R1 is not fulfilled then R3 cannot be fulfilled either. R3 is not fulfilled.
- **R4 Integrated language for EA and SOA**
As R1 is not fulfilled, there is no integrated language. R4 is not fulfilled.
- **R5 EA Formalization**
As R1 is not fulfilled, R5 is not fulfilled.
- **R6 Individual EA meta model**
There is no meta model predefined, thus an individual model can be defined. As this is not part of the framework, R6 is only partly fulfilled.

- **R7 As-Is & Target**

For the Zachman approach, the whole EA model is split up in layers and perspectives. Each perspective of a layer has a recommendation of an adequate model, e.g. the function view of the conceptual layer can be expressed with a business process model. The framework does not specify a certain languages, such as BPMN or activity diagrams. Furthermore, it is up to the user, which set of models he chooses to work with. As freedom in modelling languages is wanted but target modelling is not supported in the framework the criterion R1 is regarded as partly fulfilled.
- **R8 Holistic EA**

The Zachman Framework does not make efforts to integrate the views between the layers. The set of models is not integrated and it will cause high efforts to be kept consistent. R8 is not fulfilled.
- **R9 Individual views**

The Zachman Framework offers a variety of views from which an adequate set should be chosen. R9 is fulfilled.
- **R10 Tool support**

The Zachman Framework is not supported by a specific tool. This would also be hard to realize as modelling languages can be chosen freely. However, it is possible to manage diagrams with tools chosen independently. For example, there is an add-in for the Sparx Enterprise Architect. R10 is partly fulfilled.
- **R11 Automation of criteria checks**

As R10 is not given R11 cannot be fulfilled either.
- **R12 Improvement suggestions**

As the Zachman is a framework, it cannot give any suggestions for improving a specific enterprise architecture. R12 is not fulfilled.
- **R13 Methodical approach**

The Zachman Framework specifies a generic top-down sequence in that the model should be created. Additionally, there is a set of rules. More methodical guidelines are not given. R13 is partly fulfilled.

TOGAF Assessment

- **R1 SOA definition**
TOGAF has no explicit SOA definition. However, it provides concepts for business services and for logical applications. With this TOGAF is closer to an SOA definition than e.g. the Zachman Framework is. R1 is partly fulfilled.
- **R2 SOA formalization**
As TOGAF provides extensive meta models the parts of the SOA definition is also covered. R2 is partly fulfilled.
- **R3 SOA conformance criteria**
TOGAF does not provide any details on how to evaluate the quality of a Service-Oriented Architecture. R3 is not fulfilled.
- **R4 Integrated language for EA and SOA**
There is a full meta-model for the EA with the partly fulfilled SOA definition. Therefore, R4 is also partly fulfilled.
- **R5 EA Formalization**
As R3 is not fulfilled and R4 is only partly fulfilled R5 is not fulfilled.
- **R6 Individual EA meta model**
There is an EA meta model predefined, thus an individual model cannot be defined. R6 is not fulfilled.
- **R7 As-Is & Target**
TOGAF provides a full meta model and smaller views on the meta model. There are no notations for the languages specified, but this is not important. However, target modelling is not concerned in TOGAF. R7 is partly fulfilled.
- **R8 Holistic EA**
TOGAF integrates all the layers it defines and provides a holistic modelling. The requirement is completely fulfilled.

- **R9 Individual views**
TOGAF offers a considerable set of views. However, individual views are not supported. Therefore, R9 is partly fulfilled.
- **R10 Tool support**
TOGAF is supported in Enterprise Architect with the help of an MDG add-on. Therefore, R10 is partly fulfilled.
- **R11 Automation of criteria checks**
As R9 is not given, R11 cannot be fulfilled, either.
- **R12 Improvement suggestions**
As R3 and R10 are not fulfilled, R12 cannot be fulfilled, either.
- **R13 Methodical approach**
TOGAF defines steps and their order on how to manage enterprise architecture. R13 is fulfilled

Quasar Enterprise

- **R1 SOA definition**
Quasar Enterprise defines SOA services and puts them in the context of the enterprise architecture via a meta-model. Not all SOA-related concepts are described, but even attributes of services are described. Therefore, R1 is fulfilled.
- **R2 SOA formalization**
There is a meta model given by Quasar Enterprise. However, service attributes are not formalized. R2 is partly fulfilled.
- **R3 SOA conformance criteria**
There are no specific criteria given by Quasar Enterprise. The quality concerning service orientation of the Service-Oriented Enterprise Architecture cannot be evaluated. R3 is not fulfilled.
- **R4 Integrated language for EA and SOA**

The EA meta-model contains the SOA elements defined by Quasar Enterprise. R4 is fulfilled.

- **R5 EA Formalization**
As there is a meta model for the integrated language but R3 is not fulfilled R5 is partly fulfilled.
- **R6 Individual EA meta model**
There is an EA meta model predefined, thus an individual model cannot be defined. R6 is not fulfilled.
- **R7 As-Is & Target**
Quasar Enterprise provides a meta model for Service-Oriented Enterprise Architecture. There are no notations for the languages specified, but this is not important for this requirement. Furthermore, the method concerns target modelling. R7 is fulfilled.
- **R8 Holistic EA**
The Enterprise architecture definition of Quasar Enterprise does not strictly separate the EA Layers, so that relations between elements from different layers can be recognized.
- **R9 Individual views**
Quasar Enterprise offers a set of views and leaves it open to define others. However, individual views are not further concerned in the method. Therefore, R9 is partly fulfilled.
- **R10 Tool support**
The tool “Visualize IT” could not be evaluated directly. Probably not all the requirements are fulfilled by the tool. Especially the support for conformance criteria checks cannot be implemented, as those are not defined in Quasar Enterprise. R10 is partly fulfilled.
- **R11 Automation of criteria checks**
As R3 is not fulfilled, R11 is also not fulfilled.
- **R12 Improvement suggestions**

The Quasar Enterprise Method suggests keeping a model of the ideal enterprise architecture. An improvement would be any step of the current situation towards the ideal model. Unfortunately, there is little advice on how to fill the ideal model with the suitable content as no conformance criteria are defined. Furthermore, there is no support in defining small projects improving the current situation. Therefore, R12 is not fulfilled.

- **R13 Methodical approach**
Quasar enterprise delivers a comprehensive methodical approach. Therefore, R13 is fulfilled.

Information System Architecture (ISA)

- **R1 SOA definition**
The ISA approach does not define elements of a Service-Oriented Architecture, but resides on a substantial definition of enterprise architecture. R1 is not fulfilled
- **R2 SOA formalization**
As no SOA definition is given, formalization cannot be given either. R2 is not fulfilled.
- **R3 SOA conformance criteria**
Due to the lack of an SOA definition, conformance criteria do not exist either. R3 is not fulfilled.
- **R4 Integrated language for EA and SOA**
As R1 is not fulfilled R4 is not fulfilled either.
- **R5 EA Formalization**
As R3 and R4 are not fulfilled this requirement is not fulfilled either.
- **R6 Individual EA meta model**
There is no meta model predefined, thus an individual model can be defined. As this is not part of the framework, R6 is only partly fulfilled.

- **R7 As-Is & Target**
The ISA approach does not suggest any specific forms of modelling but demands that the views should always be aligned to the enterprise strategy. As target modelling is not concerned, R7 is partly fulfilled.
- **R8 Holistic EA**
The integration between the EA layers is a weakly enforced focus of the ISA approach. Therefore, R8 is partly fulfilled.
- **R9 Individual views**
The ISA suggests a set of views but leaves open their granularity. Therefore, R9 is partly fulfilled.
- **R10 Tool support**
There is no dedicated tool support for ISA.
- **R11 Automation of criteria checks**
As R3 is not fulfilled R11 is not fulfilled either.
- **R12 Improvement suggestions**
The ISA cannot give any detailed improvement suggestions. R12 is not fulfilled.
- **R13 Methodical approach**
There is a concept in which order the EA model shall be made but not more. R13 is partly fulfilled.

SEI – Evaluating a Service-Oriented Architecture

- **R1 SOA definition**
An SOA definition is given by the SEI approach. R1 is fulfilled.
- **R2 SOA formalization**
Even though there is an SOA definition, no efforts have been made to describe SOA in a formalized way (e.g. with a meta model). R2 is not fulfilled.

- **R3 SOA conformance criteria**
There is a catalogue of questions on service-oriented enterprise architecture. However, this catalogue does by far not cover the whole bandwidth of SOA criteria. Still R3 is fulfilled.
- **R4 Integrated language for EA and SOA**
The SEI approach focuses on the SOA evaluation and does not combine it with Enterprise Architecture Management. Therefore, R4 is not fulfilled.
- **R5 EA Formalization**
There is no formalization suggested by the approach, neither for the language nor for the criteria. R5 is not fulfilled.
- **R6 Individual EA meta model**
There is no meta model predefined, thus an individual model can be defined. As this is not part of the framework, R6 is only partly fulfilled.
- **R7 As-Is & Target**
The SEI Approach does not suggest modelling the EA in order to be able to evaluate it. Therefore, R7 is not fulfilled.
- **R8 Holistic EA**
The SEI approach does not focus on holistic Enterprise Architecture Modelling. R8 is not fulfilled.
- **R9 Individual views**
The SEI approach does not concern EA modelling and therefore does not state anything about individual views. R9 is not fulfilled.
- **R10 Tool support**
There is no tool support for the SEI approach. R10 is not fulfilled.
- **R11 Automation of criteria checks**
As R10 is not fulfilled R11 is not fulfilled either.
- **R12 Improvement suggestions**
There are considerations of conceptual solutions for some key concepts of SOA, e.g. synchronous or asynchronous messaging. These can be used to find

improvements of the current architecture. Unfortunately, these are only available for very fundamental key concepts. R12 is partly fulfilled.

▪ **R13 Methodical approach**

The SEI approach indirectly defines a method by presenting an evaluation example. R13 is partly fulfilled.

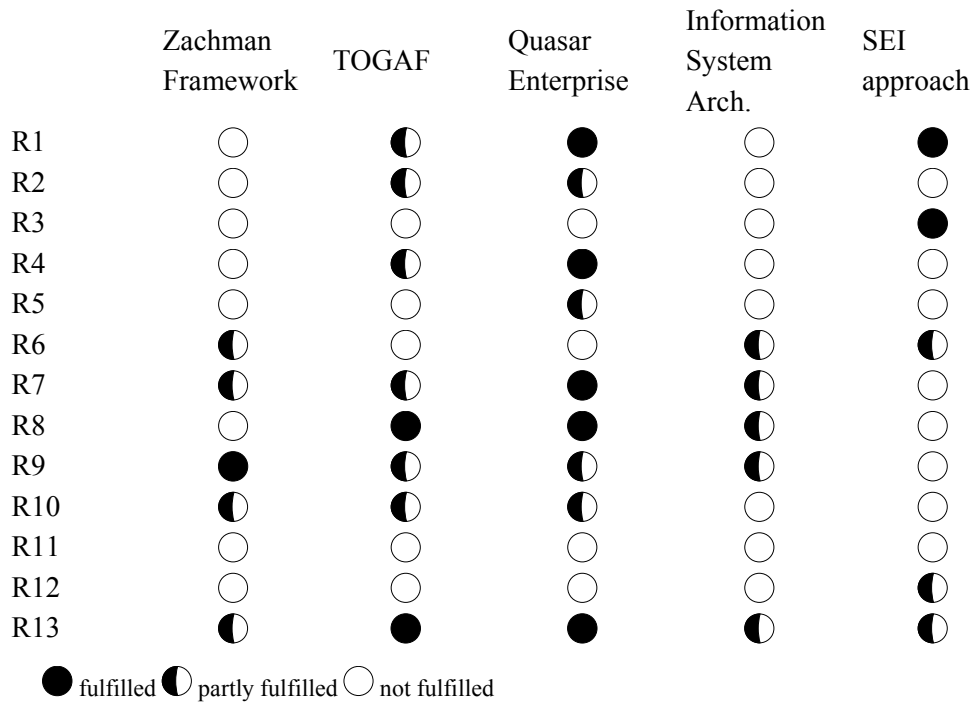


Fig. 2-31: Appropriateness of existing approaches

First of all, Fig. 2-31 shows that none of the existing approaches can fulfil the requirements placed. There are two EAM approaches, the Zachman Framework and the Information System Architecture, lacking an SOA definition. This makes them also inappropriate for fulfilling other requirements. However, the Zachman Framework describes an interesting variety of modelling approaches. The SEI approach is the only approach that offers detailed conformance criteria for a Service-Oriented Architecture. Overall, Quasar enterprise fulfils the most requirements of the existing approaches. However, it still lacks conformance criteria, formalization, and automation.

Generally spoken, the existing approaches either concentrate either on SOA or on Enterprise Architecture Management. For this reason, the thesis concept combining both approaches will be presented in the next chapter.

3 Solution Concept

This chapter describes the solution concept that will be followed in this thesis. The proposed solution shall fulfil the derived requirements as far as possible. For this purpose, the concept is stepwise defined and illustrated in the next section. In the second section of this chapter, an overview on the chapters covering the single realization steps is given.

3.1 Designing a Solution Concept

In this section, the solution concept is elicited stepwise. It is aligned with the general solution concept depicted in Fig. 1-3 and aims at realizing the requirements from section 2.4. At the end of the section, it will be shown that the elaborated solution is aligned with the general solution approach.

In the following, the realization of each requirement is discussed with respect to the previously mentioned general solution concept. Each time when a portion of the solution has been defined, the central solution figure (starting with Fig. 3-2) will be updated. To have the requirements present, they are repeated in Fig. 3-1.

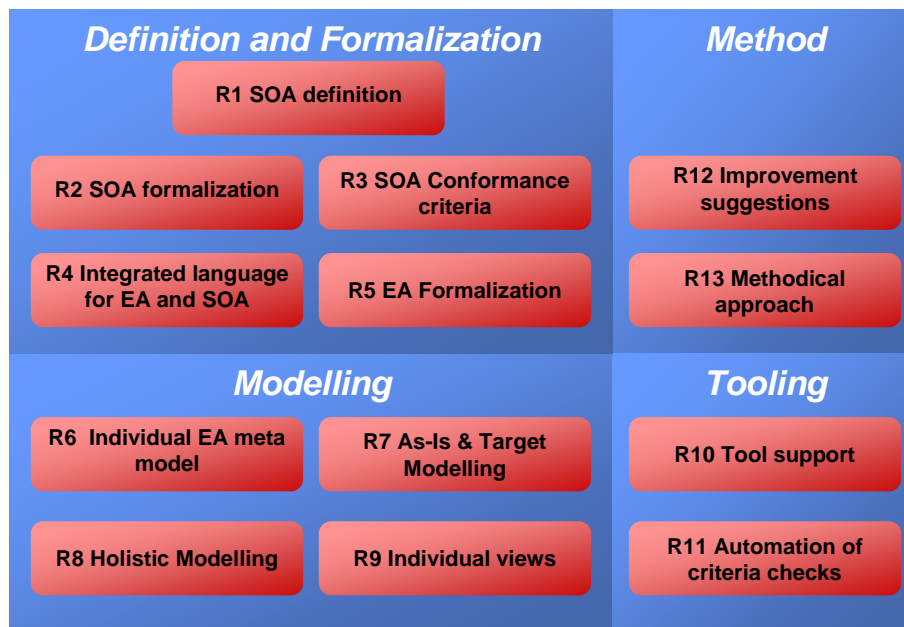


Fig. 3-1: Listing of categorized requirements

R1 demands an SOA definition. The definition of SOA shall preferably include all concepts identified in the related work section. R2 demands that this definition shall be formalized, which will be done in form of a meta model. A UML conform meta model is chosen, because it will ease the realization of the tool support. The definition of SOA and the derivation of the SOA meta model are described in chapter 4. Fig. 3-2 illustrates this part of the solution concept. Requirements are illustrated with ellipses. They are preferably situated on the upper left corner of the items they are related to.

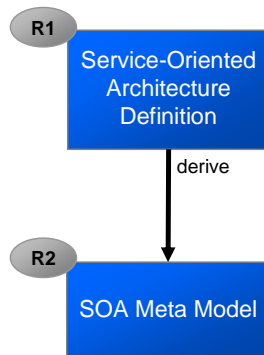


Fig. 3-2: Solution concept for R1 and R2

With R5, the formalization of EA is demanded. This requirement is closely related to R6 demanding the freedom to bring in an individual EA meta model into the solution. Individual EA meta model does not mean an entirely individual meta model. To ensure that it is an EA meta model, a frame for an EA definition is given and a small set of fixed meta model entities will be defined. By this, the holistic modelling as demanded by R8 will also be realized.

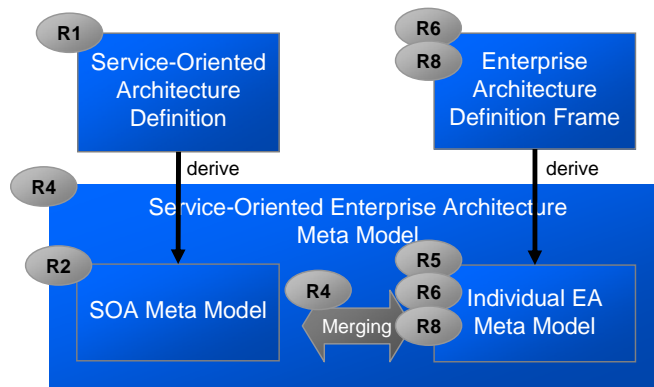


Fig. 3-3: Solution concept extended by R5 and R6

An integrated modelling language for SOA and EA is demanded by R4. The integration of the language is also a reason for having chosen meta models as formal language definitions for SOA and EA. If having commonalities, two meta models can be merged to one. To ensure some commonalities an EA definition frame is given and a minimal EA meta model is derived from it. The two resulting meta models shall be merged, which is not a simple task. For this reason, a method to realize the merging will be described here. The result of the merging will be the Service-oriented Enterprise Architecture (SOEA) meta model. It realizes R4, which demands an integrated modelling language for SOA and EA. The meta model merging and the resulting SOEA meta model also support the fulfilment of R8.

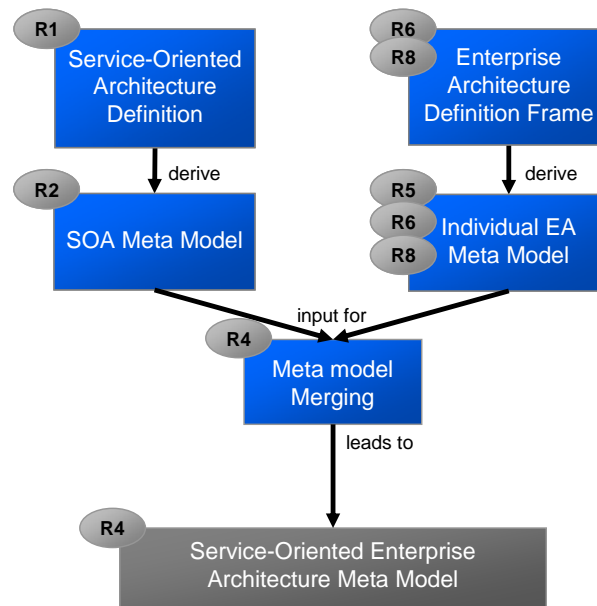


Fig. 3-4: Solution concept extended by R4

Furthermore, the method has to provide conformance criteria for service orientation as demanded by R3. These criteria are formulated in form of an SOA quality criteria catalogue. This catalogue will be divided in architecture quality criteria and Service quality criteria. The former mainly concerns the structure of elements in the SOEA. The latter concerns SOA services only and formulates requirements for well-designed SOA services. In Fig. 3-5 the SOA quality criteria catalogue is added to the solution concept. Furthermore, the SOEA model as instance of the SOEA meta model is depicted. Taking up the information for this model is not covered here, as the sources for information will be different from enterprise to enterprise. The creation of the

model has to be realized by the enterprise architect and his team as an individual piece of work.

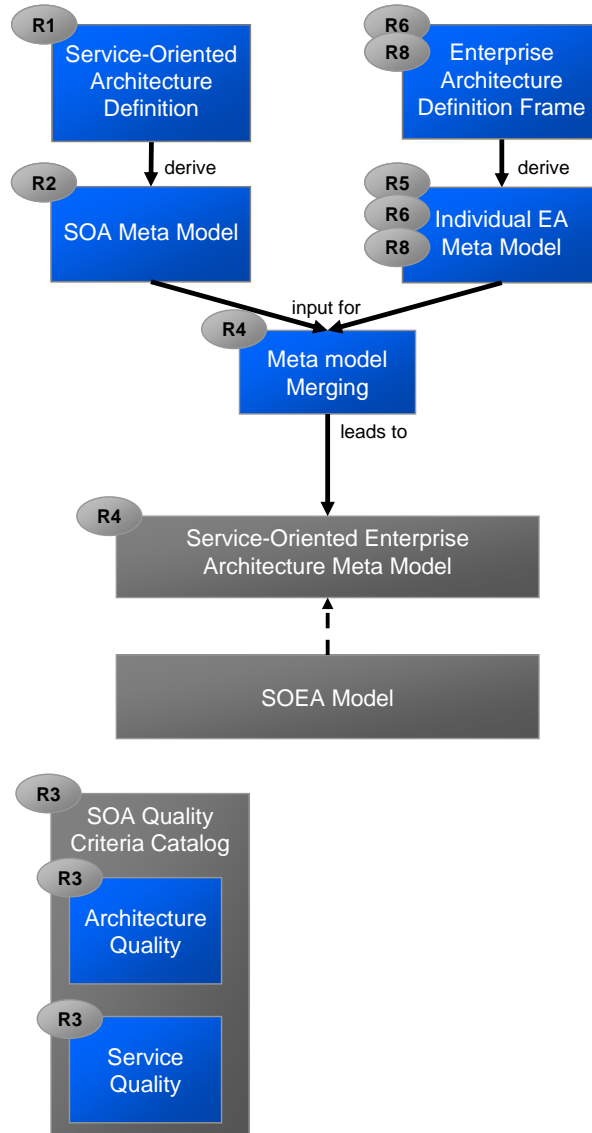


Fig. 3-5: Solution concept extended by R3

In order to make the SOA conformance criteria applicable to the SOEA meta model in an automated way (required because of R11), the criteria have to be made measurable and then automated. The former is done by defining metrics and indicators for the SOA conformance criteria. A metric defines a measure, as well as a measuring point. An indicator defines how to interpret the measures. Automation is achieved by tool support that is demanded in R10. The as-is and target modelling as demanded by

R7 are realized by the automated checking of the conformance criteria. That means that there is only one model at a time. An extra target model with the ‘perfect’ SOA-like EA (perfect in the sense of service-orientation) is not needed because all changes that lead to the fulfilment of the conformance criteria will lead the EA model towards the ‘perfect’ SOA-like EA. To have an overview how far the EA is away from the target, a check of the conformance criteria should result in a report on the fulfilled and unfulfilled criteria. The update of the solution concept is depicted in Fig. 3-6.

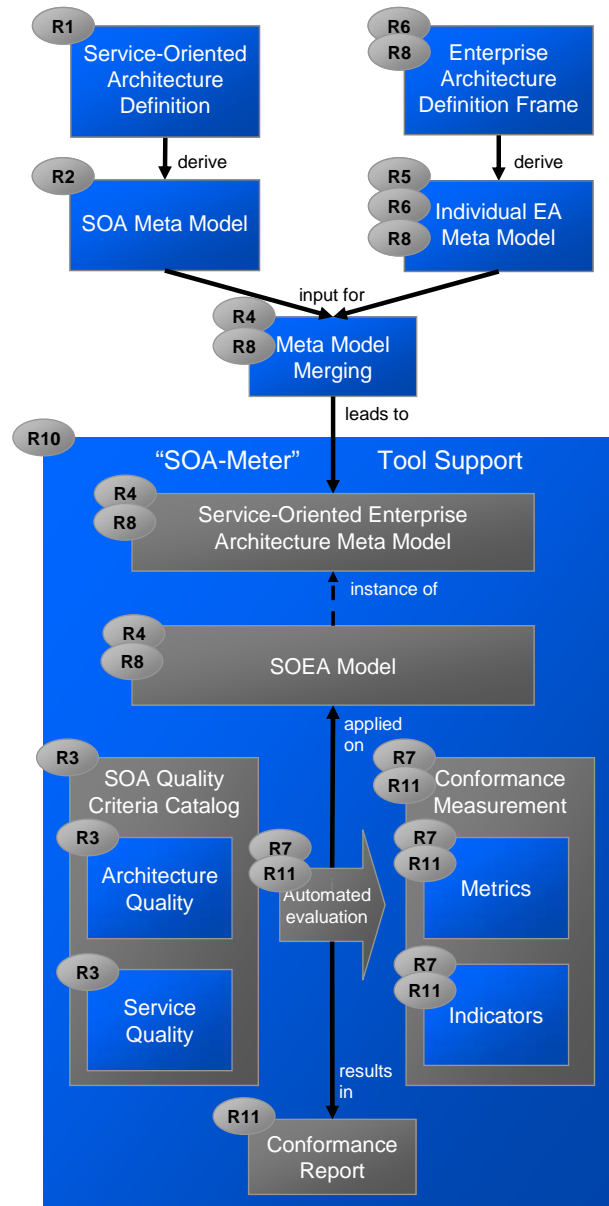


Fig. 3-6: Solution concept extended by R7, R10 and R11

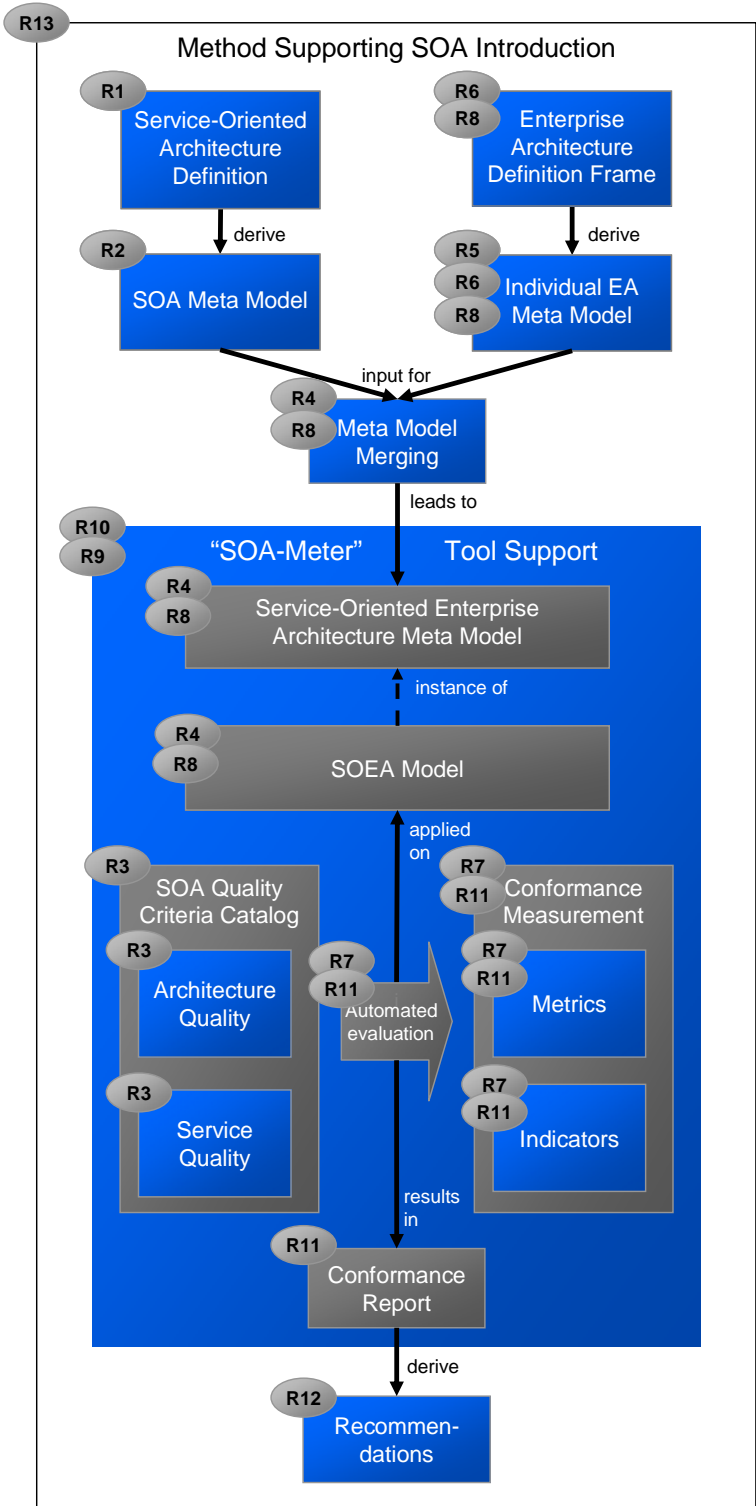


Fig. 3-7: Solution concept extended by R9, R12 and R13

The individual views as demanded in R9 have to be realized via the tool support. From the report of SOA conformance some suggestions and recommendations on changes of the architecture should be given. The derivation of these will be covered in this thesis. For this reason, the fulfilment of R12 is given. Finally yet importantly, the methodical approach as in R13 is given by the concept that has been described in this section. All the steps described can be followed by an enterprise architect, so that he receives a system allowing SOA conformance checks. That means the whole figure of the solution concept describes the demanded method at the same time. The complete solution concept can be seen in Fig. 3-7.

Now the general solution concept is picked up again. It will be shown how the basic concept has been realized with the solution concept elaborated in this chapter. It will not be explained here, why some of the realizations have been chosen. This will be done in the according chapters. The chapter content is explained in the next section.

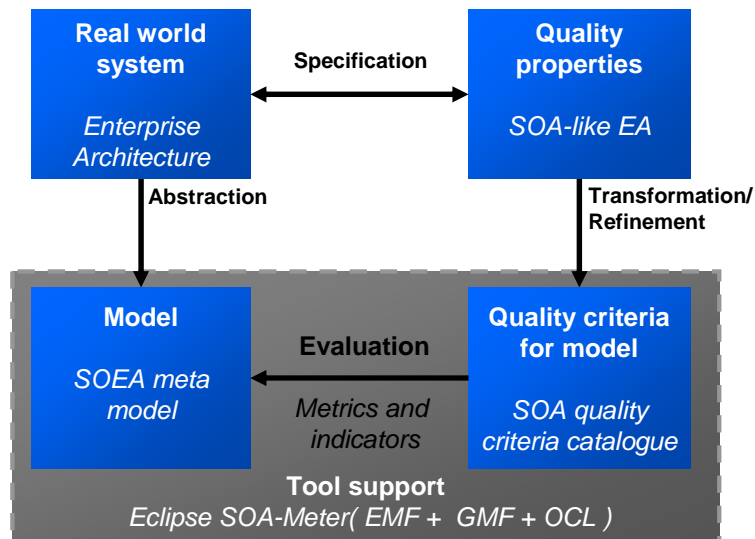


Fig. 3-8: Realization of the basic solution concept

Figure Fig. 3-8 depicts the realization of the basic solution concept. The real world system is the enterprise architecture that has been recognized as the object of change. The high-level quality property that the EA should fulfil is to be an SOA-like EA. The refinement of this criterion will lead to the SOA quality criteria catalogue. The criteria will be evaluated with the help of metrics and indicators working on the SOEA meta model. The SOEA meta model is the realization of the modelling for our approach. SOEA meta model, as well as the metrics and indicators for the SOA quality criteria

catalogue will be implemented in an eclipse-based prototype. The prototype mainly uses the eclipse modelling framework (EMF), the graphical modelling framework (GMF) and a plugin using the Object Constraint Language (OCL, compare [OCLspe06]) for the evaluation of the metrics defined on the meta model.

Having defined a solution concept, it has to be clarified in which sequence the realization is described. This is done in the next section.

3.2 Thesis Structure

On the basis of the defined solution concept, the sequence of the realization steps will be set. For this reason, the structure of the remaining thesis is described in this section. In Fig. 3-9 the distribution of the steps over the chapters is illustrated. The pentagon on the upper right corner indicates that the solution concept item is covered in the according chapter.

The first step is to define Service-Oriented Architecture in the enterprise context including the motivation and benefits of this architectural style. Part of this section is an illustration of the development of enterprise architectures since the upcoming of IT systems. Afterwards a frame for the definition of EA is given. It leaves open the necessary degrees of freedom to define an individual EA. The EA and SOA definitions are given in chapter 4.

Afterwards, the formal part for modelling is described. Chapter 5 contains the description on how the Service-Oriented Architecture meta model is derived from the SOA definition. In addition, a minimal enterprise architecture meta model is given. For architects that do not have made up their mind about an EA meta model, an example on how to create such a meta model is described. The resulting EA meta model is then used as example for the following merging process. The resulting meta model is named the Service-Oriented Enterprise Architecture (SOEA) meta model. It has the advantage that the resulting meta model can be used for enterprise architecture planning and for evaluating service orientation of the enterprise at the same time. The description of the merging method includes an example of an SOEA meta model that completes chapter 5.

The resulting SOEA meta model has the ability to describe enterprise architectures that are not service-oriented as well as service-oriented ones. To identify the parts of the architecture that are not SOA conform an SOA quality criteria catalogue is put up

in chapter 6. The catalogue is divided into two major parts. Firstly, the service quality criteria, which only concern the set of services within the enterprise. Criteria like granularity, coupling and cohesion are covered there. Secondly, the architecture quality criteria, which evaluate the enterprise architecture concerning its conformance to the Service-Oriented Architecture style are described. Architecture quality criteria focus on relations between EA elements. Service quality criteria focus on services themselves. In addition to the quality criteria, the metrics for measuring these criteria are elaborated in chapter 6.

Chapter 7 concerns the results of the measuring defined in chapter 6. Therefore, the indicators for the metrics are defined, so that every measure can be interpreted in the right way. Moreover, chapter 7 presents the way a report on service orientation could look like and which improvement recommendations can be derived from the possible report results.

The description of the tool support that has not been treated so far will be given in chapter 8. It enlightens the eclipse based prototype that allows defining meta models, allows formulating conformance criteria in the Object Constraint Language (OCL compare [OCLspe06]), and allows executing these conformance checks on a given model. The thesis is completed in chapter 9 giving a conclusion and an outlook on possible further developments of the work presented.

With sequence in mind, the realization of the solution concept is elaborated stepwise. It begins with chapter four delivering the required definitions.

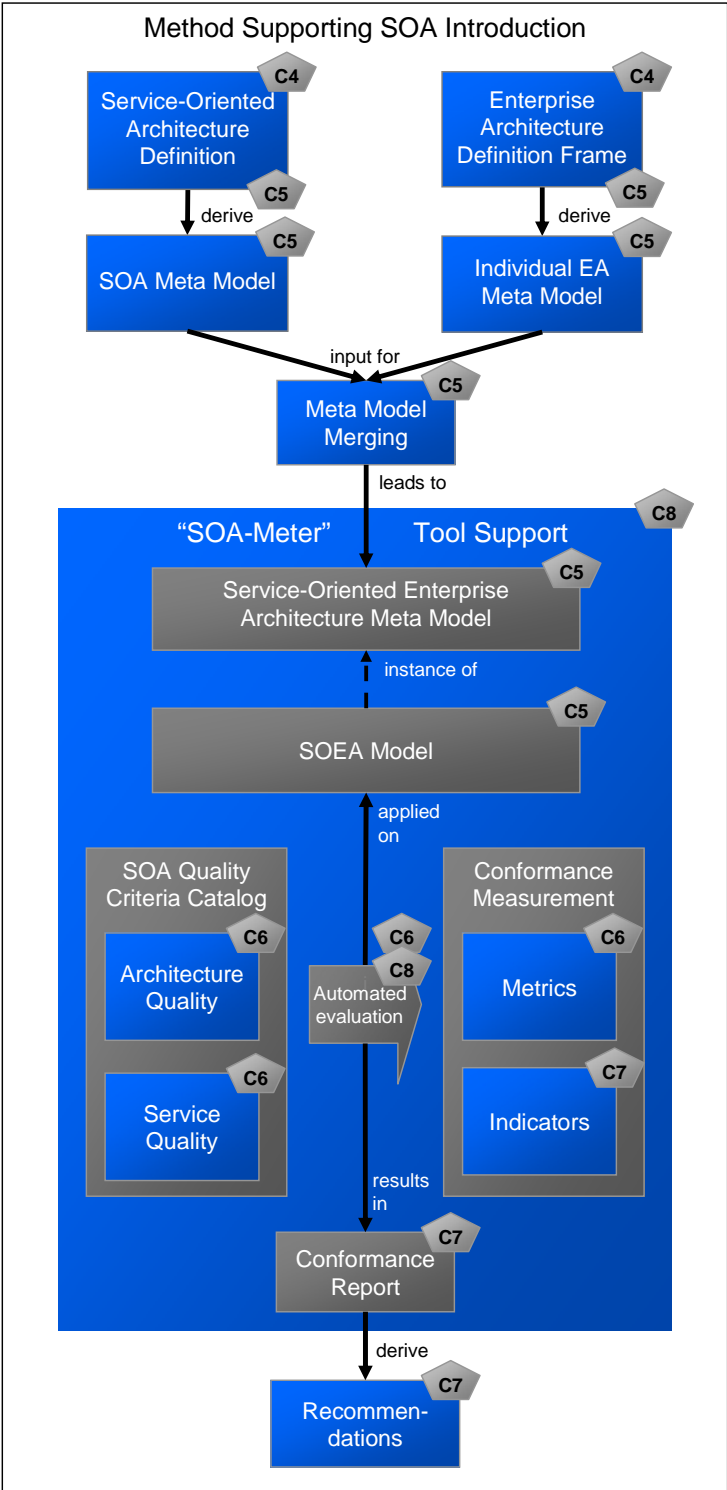


Fig. 3-9: Sequence of realizing the steps of the solution concept

4 Service-Oriented Enterprise Architectures

Within this chapter an own, comprehensive definition of Service-Oriented Architecture is given. In addition, a frame for Enterprise Architecture is described. The SOA definition is given to fulfil R1. R6 and R8 are at least partly fulfilled by providing the EA definition frame. Fig. 4-1 shows that these definitions are needed to be able to derive the formal meta models of each kind of model.

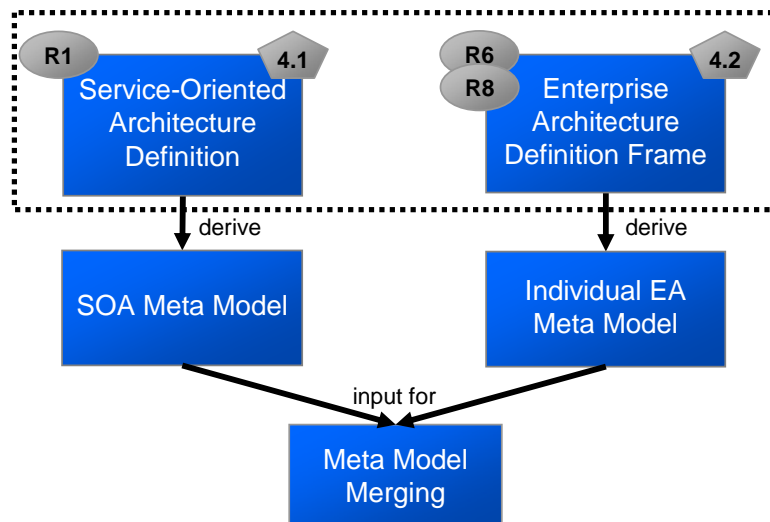


Fig. 4-1: Contribution of chapter 4

Section 4.1 contains the two-part SOA definition for this thesis. The first of the two subsections (4.1.1) provides the SOA reference architecture. To provide this definition in an EA context, the historical development of enterprise architecture is examined in this subsection. The historical outline of enterprise architecture leads to the conclusion that SOA is an evolutionary product of enterprise architecture. The second subsection (4.1.2) contains the SOA service definition. Purpose of the SOA definition is to provide the basis for the derivation of a formal SOA meta model and for the quality criteria catalogue.

In section 4.2 an enterprise architecture definition frame is described. It allows the use of an individual EA model and will be used to derive a minimal EA meta model.

4.1 SOA Definition

This section embraces the evolutionary description of the SOA reference architecture and the SOA service definition.

4.1.1 SOA as an Evolutionary Product of Enterprise Architecture

The historical outline of the EA evolution shall lead to a better understanding of SOA and an EA-oriented definition. For this reason, this section depicts the evolution of enterprise architecture from the early beginning up to the stage of SOA as a style for EA. SOA can be regarded as the latest step in the evolution of EA. It combines many of the previously gained capabilities and discards some of the early ones.

To illustrate the evolution process of enterprise architectures a running example is introduced. It is always focused on the layers reaching from the applications up to process definitions. For every evolution step, reasons are given why the systems have been built the way they are and why they did not suffice anymore, so that the next evolution step was developed.

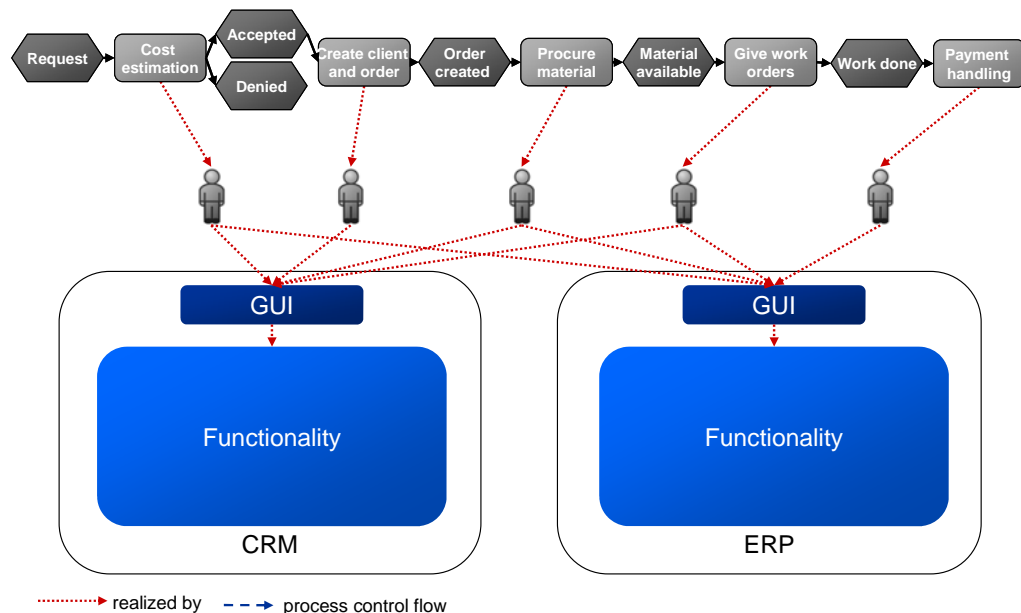


Fig. 4-2: IT-landscape with two monolithic applications

The running example of enterprise architecture starts with a very simple setting, being depicted in Fig. 4-2. There is a process reaching from a customer request to production up to payment handling. All steps involve human work and IT systems so that IT systems always have to be operated via a graphical user interface (GUI). The monolithic IT systems, namely a Customer Relationship Management system (CRM) and an Enterprise Resource Planning system (ERP), are characterized by integrated GUIs, huge functionality and no communication with other systems. We are aware of the fact that the notions ERP and CRM are not as old as that kind of monolithic systems, but the tasks they fulfil exist for a longer time.

In the beginning, monolithic systems were often developed directly on purpose, because efficiency was a major design driver. Furthermore, the process life cycle was much longer in the past, and processes could be implemented directly in the applications. Therefore, the coupling of functionality in this system is very strong. Often users and developers either were the same or had close contact.

The reasons why these systems did not suffice anymore are relatively simple: Over time, the monolithic systems had to be adapted to (slowly) changing processes. Maintenance effort is relatively high because every change concerning the system requires testing the whole system. In addition, previous updates of a system make its architecture more complex and hinder the implementation of new updates. IT-responsible persons recognized that maintenance was a growing cost factor that should be reduced by improving the architecture of the usually growing enterprise systems.

This leads to the novelty of components within the applications, depicted in Fig. 4-3. The components are connected by interfaces. Often proprietary interface technologies were used so that components could be exchanged but it was hard to integrate them with other components. This means they were still closely coupled. The functional expansion or replacement of a component is easier than the one of a monolithic system. However, there is also a disadvantage because somebody has to decide how the components are tailored. If they are not tailored well, it will happen that simple updates often concern several components and communication effort between these components is very high.

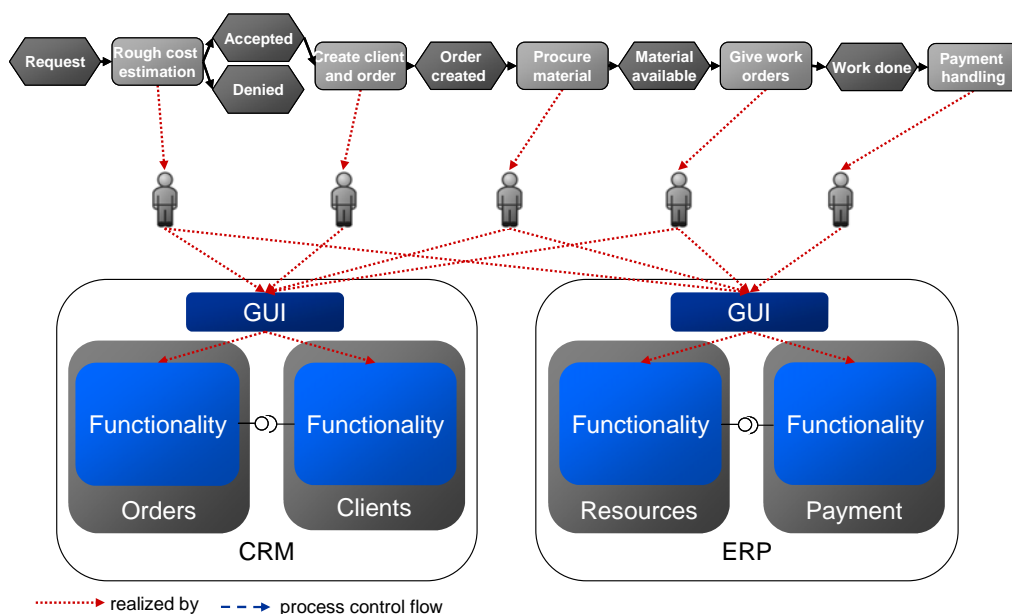


Fig. 4-3: IT-landscape with two component based applications

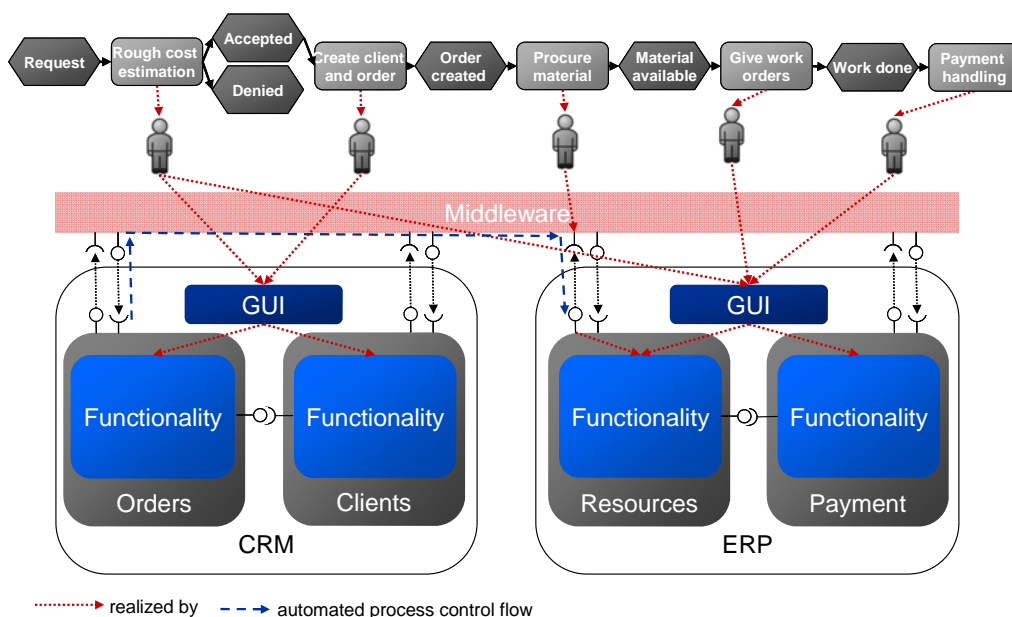


Fig. 4-4: IT-landscape with middleware connecting applications

However, component based systems were a big step that reduced the maintenance effort of single software applications. In the following time, requirements arose from the process side concerning the integration of different components and applications.

Building ever-new interfaces between components implemented with different technologies became a serious cost problem.

Enterprise Application Integration has been a hype topic several years ago. Its goal is to integrate several applications with each other. The main concept to achieve this is middleware technology like CORBA (compare [OMGCOR92]) or ActiveMQ (compare [Apache03]) which strongly decreased the integration effort in enterprise systems. A simple reason for this is the 1:n communication pattern instead of the n:n communication pattern. Without a middleware, the communication pattern is n:n because if each of the n components wants to interact with the other components then n-1 different interfaces have to be implemented for every component in the worst case. The worst case occurs if all components have different technologies. On the other hand, if a component offers an interface in the middleware technology then it can communicate with all other applications, i.e. a 1:n communication is possible. For these reasons, middleware is an indispensable concept for an enterprise architecture.

Middleware

A middleware is a communication technology that is used by the as much software interfaces as possible. It greatly reduces the integration and communication costs of within an enterprise.

Obviously, something has radically changed in Fig. 4-4. The human actors two and three from left hand side do not have to access the CRM system anymore. Before they had to look up the order being stored in the CRM system, now the order data they need for procurement and giving work orders is transferred to the ERP system in a nightly batch run. This saves a lot of effort and reduces the number of mistakes in the process. Drawback of the middleware is that a new technology is required within the enterprise systems. Its introduction always comes with additional costs.

Still, the integrations problems were not solved completely. There are systems, especially older ones, which offer their functionality only via the graphical user interface so that the middleware cannot adapt to it and automation is hindered once again. To reach this functionality the GUI design had to be changed.

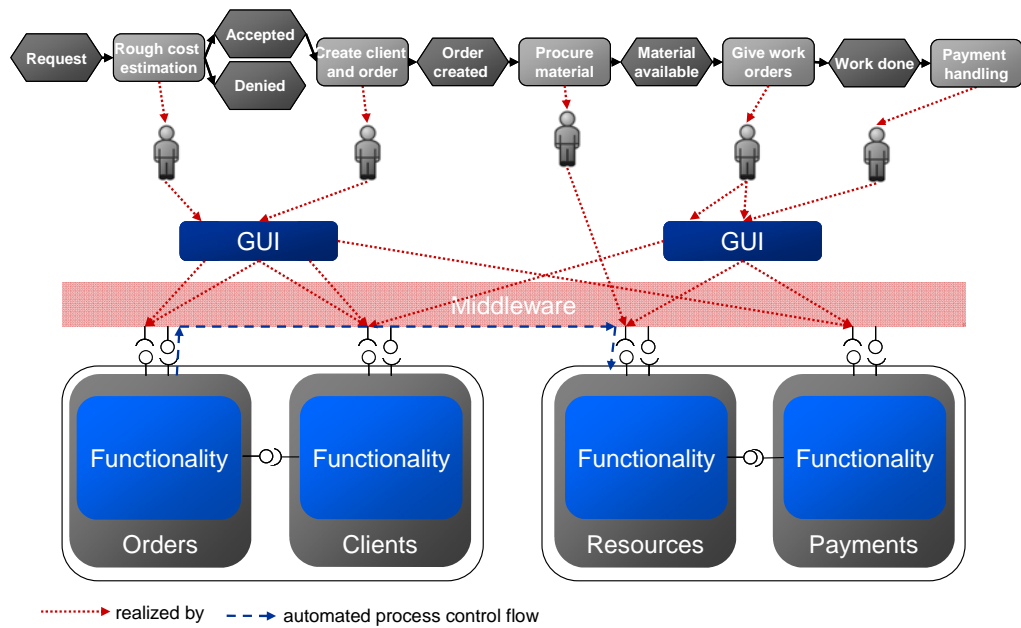


Fig. 4-5: IT-landscape with middleware (EAI) with separation of GUI

The separation of GUIs implies the disclosure of functionality of software components. It increases the flexibility of systems because all functionality is now reachable for automation purposes via the middleware. Of course, this disclosure of functionality increases the design and implementation effort slightly. Fig. 4-5 shows that the graphical user interfaces now have to use middleware interfaces that can be used by other automated components.

Disclosure of Functionality

Each functionality of a software has to be offered with an interface technology suitable for automation purposes (e.g. the middleware technology). It is not acceptable that functionalities are only reachable via a graphical user interfaces.

Now one might ask what is left to optimize, what is the new requirement from the business side that makes the present architecture more sufficient. The main driver is flexibility, as process lifecycles are steadily decreasing. Especially since the beginning of the new decade this topic has been discussed by IT and business experts. Maybe the fact that business experts found their way into the discussion about IT flexibility has been given birth to service orientation. Because from the view of IT experts an IT landscape with separated GUIs and Enterprise Application Integration is already

relatively flexible. For business experts it is not that perfect because the tailoring of functionalities was made with respect to technical and maybe organizational circumstances. This means that the building blocks of functionality that are reusable from the process view are not the ones that were developed by IT-experts. For an enterprise that wants to react on steady process changes the adequate tailoring of business functions implemented by IT is helpful. This brings the service concept into play. In its simplest form a service is a business function that is implemented by IT. For a service, it is only interesting what it does and not how it does something. This especially means that a service implementation is not bound to a certain application.

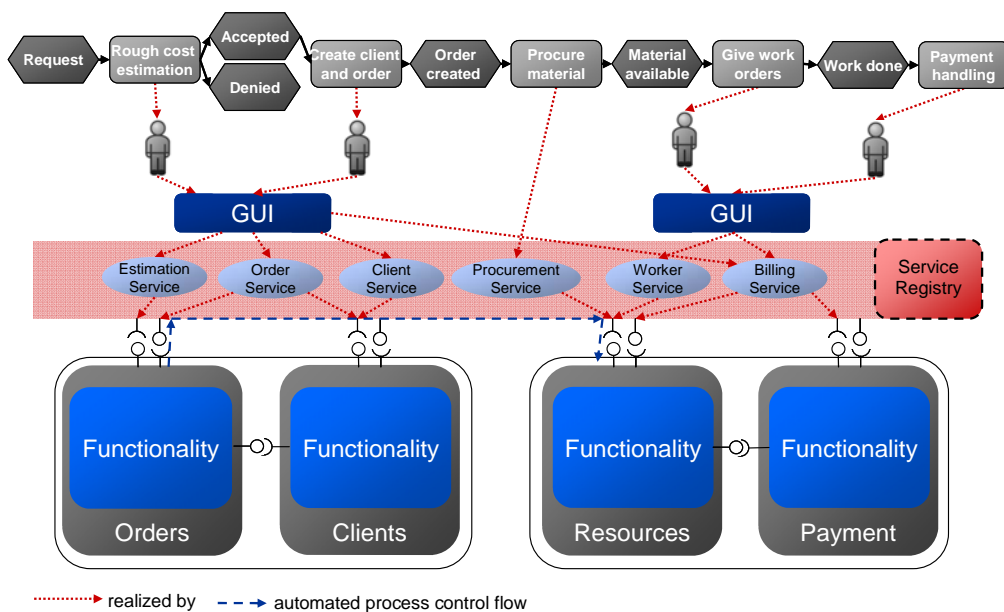


Fig. 4-6: IT-landscape with basic services

Fig. 4-6 depicts an IT-landscape that offers services that are consumed by the GUIs or directly by applications. For example, the procurement service is executed in a batch run every 24 hours. The picture shows an ideal landscape where all functions are exposed by services. However, this is not necessary. Especially during the transformation phase to a service-oriented enterprise there will exist both forms.

There is a new concept showing up at this stage, named service registry. It keeps all the information about services that are required to use them. This includes behavioural descriptions, usage costs, availability etc. Potential users of SOA services can search for services and use them afterwards.

The SOA services represent a new form of abstraction bringing up advantages and drawbacks. The drawbacks occur in form of increased effort for identifying, implementing, and maintaining services as well as the related service registry. The first advantage is the increased reusability of services if they are tailored well and documented in the service registry. Thusly, the SOA services themselves are used properly by developers. Moreover, functions spanning several applications can be offered with SOA services. This important concept will be referred to as IT-business alignment:

IT-Business Alignment

A SOA service bundles functionality with respect to business needs and not with respect to technical needs. As the SOA service is unaware of its implementation, it may use several technical applications for its realization. This business-oriented bundling of functionality is major contribution to the IT-business alignment.

The role of the GUIs is again important in this scenario, because with changing processes the GUIs have to be transformed, too. For this purpose, the use of web-based interfaces e.g. implemented with portal and servlet technology, can save effort when changing a process and therefore increases flexibility.

Now there is a very premature kind of Service-Oriented Architecture, as it uses the service concept for the first time. However, flexibility regarding the implementation of new processes has not yet reached its climax. This is because until now the control of the process flow is distributed over humans and the whole application landscape. Humans can learn new things quite easily, but applications have to be reprogrammed in their specific language. This is tedious because the programmer has to find the code fragments that change the desired process control flow among other uninteresting code fragments. Consequences of code changes are at least, recompiling, testing, and down time for restarting. If there was a way to extract process flow information and make it available in an explicit way, the effort of changing processes could be reduced drastically.

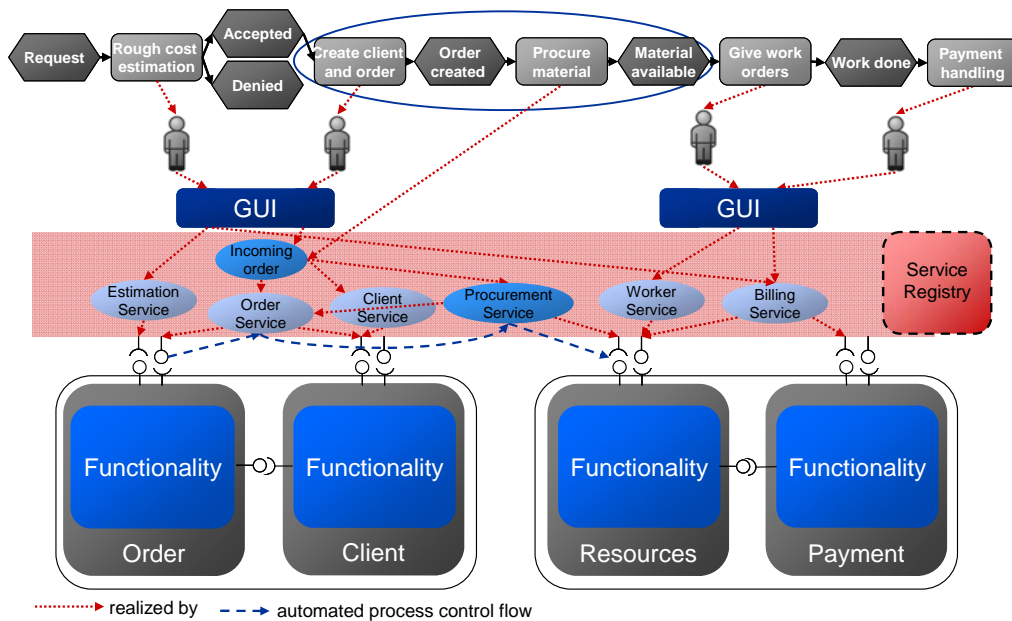


Fig. 4-7: IT-landscape with basic and orchestrated services

Fig. 4-7 shows the orchestrated incoming order service. It represents the process part that begins after the “Accepted” event and ends with the “Material available” event. The incoming order orchestration contains the process control flow information of the named process part and the calls of the subordinate basic services. This service orchestration can be realized with any technology that is able to call the other services and can be used like a service operation, for example a web service that is deployed on an application server. This means the process control flow is now made more explicit as it is separated from other implementation details. However, it still exists in form of a programming language, which is the reason why we call this a “hard-wired” orchestration.

Within the “Procurement” service, something has been changed, too. As mentioned before, required data from the “Order” component was transferred in a daily batch run to the “Resources” component. By that, the data was held redundantly in both applications and was not available in real time. When the batch run was implemented, it was the only practicable solution to have a batch run each night, due to performance restrictions. The situation has changed over time and now the data load could also be transferred in real time, which would speed up the process in a noticeable way. The triggering (process control flow) of the batch run was implemented in the “Order” component. Now a redesign has been realized, so that the old batch run was removed from the system by finding the appropriate code fragments in the “Order” component.

Instead, the process control flow was moved to the “Procurement” service. The according operation now retrieves the data directly from the “Order” Component. This change is regarded as a stable solution (change not predicted for longer time, reuse is likely, small granularity), which is the reason why this hard-wired orchestration is now a usual service operation of the “Procurement” service.

Another advantage of service orientation is observable within this transformation. The exchange of the two solutions did not bother any users of the “Procurement” service, because its interface remained unchanged. Only the hidden part, the implementation was exchanged. This probably saves some effort in the transformation process.

In this case, we have shown two examples for hard-wired orchestrations. One of them is regarded as reusable and became a service operation of a basic service. The other embraces a bigger process part and is less likely to be reused. Just like the process part it represents, it is more likely to be changed. In both variants, the process control flow was made explicit in the service layer, thus the effort for process changes was decreased. The price to pay for this flexibility is the increasing number of services that have to be maintained.

Until now, service orchestrations could be used to easily implement the process control flow in a relatively well-isolated form. Still one has to read the code of several services to gain knowledge about the process control flow. Furthermore, changing the flow requires reprogramming several services. There is a restricted amount of logic that has to be programmed within service orchestrations. Mainly, the parallel and alternative paths from process models are implemented within it. To meet decisions for the alternative paths parameters of service operations are evaluated and forwarded. A further desirable enhancement regarding the quick implementation of new processes is to have a (visual) language that combines concepts of process models and that is interpretable at the same time. Interpretable mainly means that there are service-operations instead of non-executable activity-rectangles.

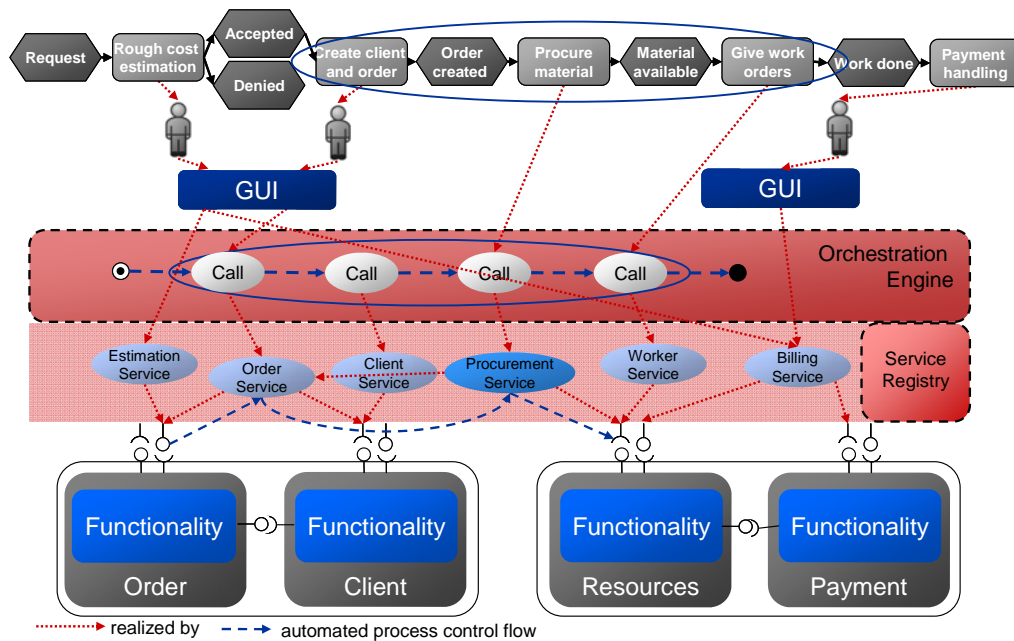


Fig. 4-8: IT-landscape with orchestration engine

Fig. 4-8 introduces a new component of a Service-Oriented Architecture, the Orchestration Engine. It is able to interpret special process models. The most common language for these process models is the Business Process Execution Language (BPEL, compare [OASIS07]). The concept behind the orchestration is also known for a longer time. It is called Workflow Management. It first brought up the idea of making the process control flow more explicit and modelling it in an interpretable language. In combination with the middleware, it is possible to access most of the business functions available. At the same time, the process-oriented tailoring of services eases the modelling of executable process models.

In the previous figure, the hard-wired incoming order service has been introduced. It is now replaced by the soft-wired orchestration. It is called soft-wired because it is modelled in language such as BPEL, which makes it easier to change. This invited us to add the “give work orders” process step to the orchestration. Before this step was triggered by a human person that also sent the work orders via mail. The sending has been automated and thereby the whole step could be automated and integrated in the orchestration. Furthermore, the orchestration is now much nearer to the process description (similar modelling concepts like visual modelling of activities, forks, and joins) but still executable because it is interpretable by the orchestration engine.

What has been reached until now is quiet good for flexibility, but the reader might ask what all this technical stuff is good for if there are many humans involved in a process. With the means provided by now, human work is a serious problem because it cannot be integrated in any orchestration. The second person from the left in Fig. 4-8 just initiates the orchestration. Our orchestrations are only interpretable if no humans are involved. Every time a human interacts in the process, the process control flow hold by the orchestration engine is lost. However, human interaction can also be seen as part of a service implementation. “Send order confirmation” is a business function and therefore a possible service operation. On the one hand, its implementation could be an employee who writes letters and sends them via mail to the clients address, on the other hand, this could be fully automated with an application that retrieves the required data and sends the confirmation via email. Theoretically, nothing speaks against human interaction within service implementation; the problem lies in the technical realization.

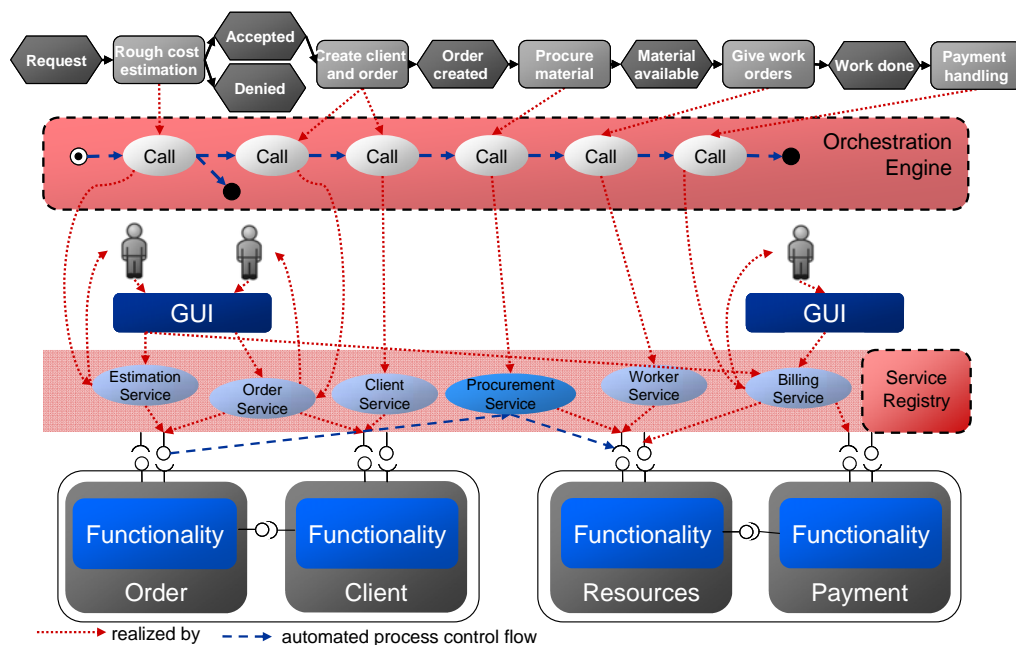


Fig. 4-9: IT-landscape with orchestration engine and human interaction services

Human actions are triggered like any other service operation in Fig. 4-9. Of course, some extra effort is required to provide this possibility. Generally, if a service with human interaction is triggered, the potential actors have to be informed. The task has to be delegated to an employee who will actually execute the action. After completing the task, he has to transfer the results back to the orchestration engine, which then

regains control of the process flow. A role concept for human actors is very helpful to be able to inform those and only those employees being able to fulfil the task.

BPEL4People (compare [OASISP05]) is an extension of the Business Process Execution Language that provides support for human interaction. It comprises a role concept, task delegation and support for scenarios like escalation and the four eyes scenario.

The integration of human work in service operations is laborious but allows to model whole processes in an interpretable language like BPEL. For a Service-Oriented Architecture, orchestration with the integration of human work is indispensable because it greatly increases the flexibility if processes that have to be reorganized concerning their control flow:

Orchestration

Orchestration is the isolation of the process control flow by using an executable process modelling language for processes and a respective execution engine. Human work tasks are treated just as electronically initiated steps within this concept. The executable process models are referred to as orchestrations and their execution engines as orchestration engines.

Now, there seems not much left that can be done to improve the flexibility. This is true as far as no interaction with enterprises in the process is required. This scenario is named choreography because enterprises act on their own and are not steered by a central unit like the services in an orchestration. However, we will not follow this concern now, as there is a more pressing one.

Meanwhile our enterprise can implement new processes quite fast and efficient. That might be faster than most of the processes are even executed. Short process execution times have become a critical aspect nowadays. For this reason, the processes should be supported by the enterprise architecture, too.

Now what could make our architecture generally slow? Time is often wasted between actions and not during their execution. There is only one process in the figure, but the real world is a little more complex, because the figure depicts only a model of the real world and abstracts in this way from details. No one should be able nor should he have interest in modelling all activities of an enterprise in a single

executable process model. There will always be several processes and they have to be invoked when certain conditions are fulfilled. These conditions are mostly distributed over the enterprise or base on external events. Events can also be regarded as a condition that is fulfilled and maybe a reaction is necessary in a certain time. Generally, there are two possibilities to check these conditions.

First one is checking them on a regularly basis or even wait until one can be sure that they will be fulfilled. Checking conditions on a regular basis means that one could lose nearly a whole cycle length if the conditions become fulfilled just after they were checked. This is the way it is often done today.

Second one is to create small notifications when something changes and immediately when all conditions have become fulfilled. This is not as easy as it might sound, so we will see what the consequences are.

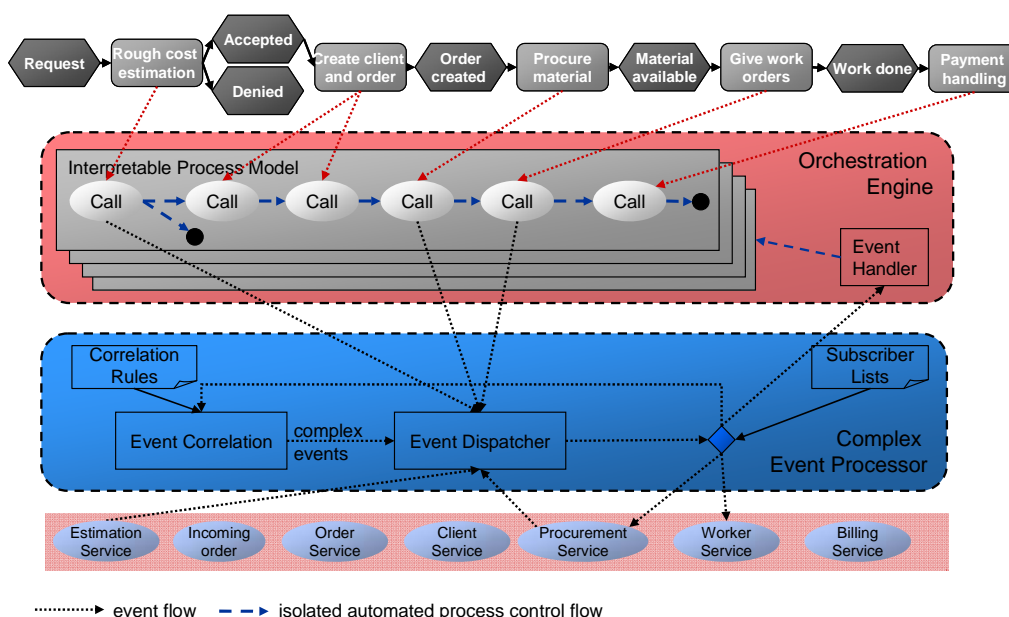


Fig. 4-10: IT-landscape with orchestration engine and complex event processor

In Fig. 4-10 several new concepts are introduced. On top of the service layer, a complex event processor (CEP) is added that receives all events (notifications) generated by event producers (any services or applications may create events). The CEP also holds subscriber lists. Any application can subscribe for a type of event. If so, the dispatcher forwards the event to the subscriber immediately after its occurrence.

The event correlator is a component of the CEP that receives a copy of all events. It checks the flow of events on occurrence patterns where several events are involved in a certain time. These patterns are specified in correlation rules. For example, if an error occurs in a process usually nothing is done, because a single error is not regarded as a problem that requires intervention. However, if the error occurs ten times within an hour, a process for quality improvement should be invoked. The event correlator can recognize this easily, but on the basis of single events this would not be possible. If such a rule is fulfilled the correlator fires a new (complex) event, which then is treated as any other event by the dispatcher.

Another very important event consumer is the orchestration engine. It can invoke new processes if a certain event occurs. This means a further decoupling of the IT-landscape. The process control flow within processes is made explicit with the help of soft-wired orchestration. With an Event-Driven Architecture and an event-listening orchestration engine, the control flow between different processes can also be made explicit. Without events, any kind of application or human has to check some conditions in the system until a reactive process is launched.

For example, a batch script might check the log files for errors every night. As it is executed every 24 hours, only a delayed reaction is possible. If it finds more than ten errors, it invokes a quality insurance process. In order to do so, it has to know its information source and any kind of application or address where the process can be invoked. This means that the control flow logic is hidden in an application or in human minds. With the event handler of the orchestration engine this control flow can be made explicit and a lot of time can be saved as reactions are executed immediately.

For the development process of enterprise architectures this means that complex and simple events have to be identified and implemented, but this will be outweighed by the shorter execution times, the information gain, and the automated initiation of reactive processes. For these reasons, Complex event processing is an indispensable part of a Service-Oriented Architecture:

Complex Event Processing

Complex event processing concerns the automated generation of business events and their correlation within a complex event processor. The generated events can be used to trigger other processes or for covering information needs.

This kind of architecture is very flexible and allows a quick process execution. However, with the frequent business process changes a new drawback occurs in enterprises. A business process model, just like a piece of code or any other result of human work, is rarely perfect. It undergoes several (minor) changes to improve its quality during its lifetime. One of the problems with the quality assurance of processes is the recognition of errors. The reason for this is that every business process can have different characteristics that mirror its quality. The present architecture does not support the monitoring of processes, so that their weaknesses cannot be recognized as soon as possible. However, due to ever more frequent changing business processes, this becomes a driving demand in enterprises.

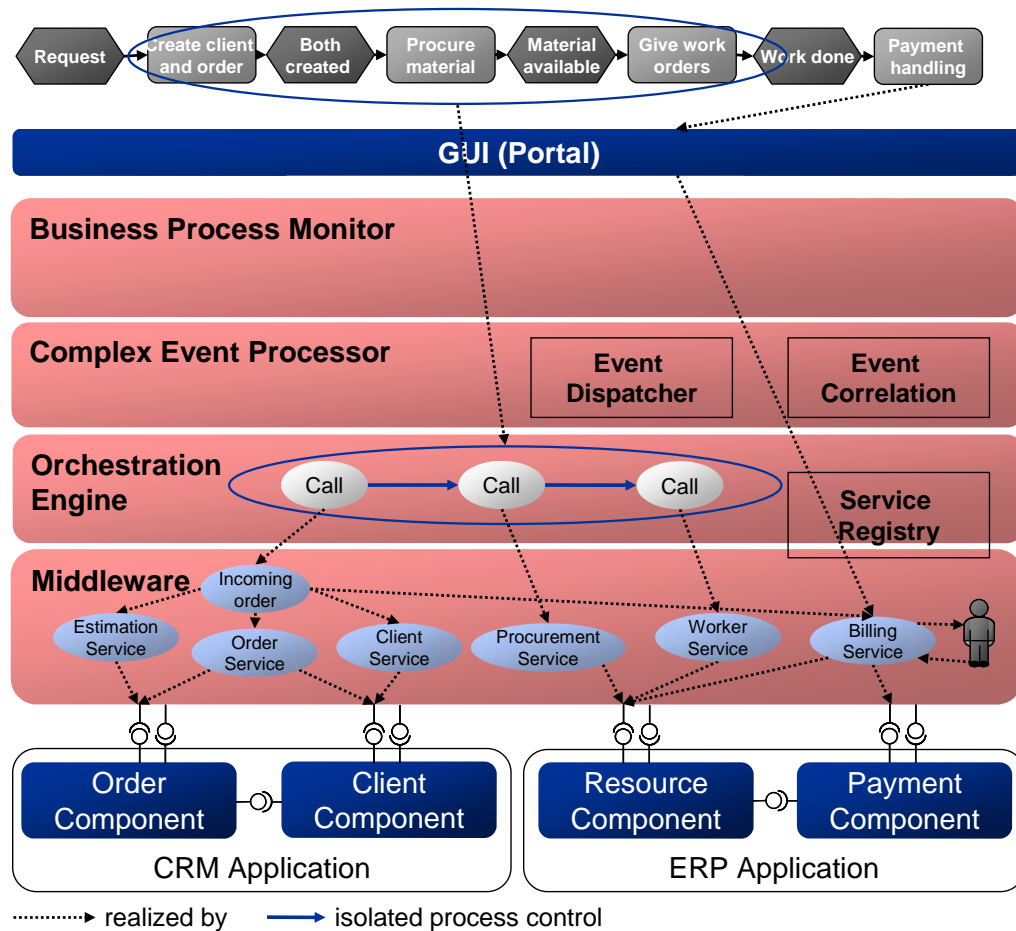


Fig. 4-11: IT-Landscape with business process monitoring

The novelty in Fig. 4-11 is the business process monitor. For every business process, it holds several performance indicators, like the execution time between specific steps or the number of used resources during process execution. In order to gain knowledge about these it has to retrieve the information from the IT-landscape. If it would hold an interface to every component that provides information required for a performance indicator, it would relatively soon deliver wrong results, as the sources for specific pieces of information can change their locations over time. Furthermore, it would hardly be possible to get informed about the exact point in time when an action was executed.

A more sophisticated method to retrieve the required information is to get it delivered. The events we introduced in the previous section are the method of choice for this task. Any performance indicator has to be calculated with any kind of event and some rules what do to with several events. The example with the number of used resources requires a “resource used” event that is added up with every occurrence in the context of a certain process. For the execution time, start and end events of the according steps are read and the duration time is calculated. Furthermore, a performance indicator can have thresholds and upon the crossing of a threshold, an alert is created. This can be done in form of an event or visualization on a dashboard.

By this, the quality characteristics of a business process are monitored in real-time, enabling business analysts to react immediately on quality problems. With low thresholds and indicator trend analyses, even pro-active reaction can be invoked.

Just like events, performance indicators have to be identified at first. These indicators are heavily influenced by the business goals the enterprise aims at. The realization of those performance indicators has to be planned. This means to identify which events are required to measure it and how the events have to be processed.

Business process monitoring offers the chance to

Business Process Monitoring

Business process monitoring concerns the definition of performance metrics for processes, the information retrieval for measures as well as the graphical processing of the data. Monitoring is, unlike reporting, executed nearly in real-time. This has the advantage, that reactions can be immediately triggered if a performance indicator is out of its range.

Fig. 4-11 implicitly contains a set of concepts. The 6 most important were pointed out in this section and are now summarized in Fig. 4-12. Some of them are more business related and some of them are more IT related. The more business-related concepts are Business Process Monitoring (BPM), Orchestration, and IT-business alignment. The more IT-related concepts are disclosure of functionality, middleware, and complex event processing. Orchestration, BPM, middleware, and complex event processing have been explicitly described in the previous paragraphs. IT-business alignment concerns the SOA services that bundle functionality in a more business suitable way. Disclosure of functionality demands that every functionality is offered by machine processable interfaces, or in other words, GUIs must not be the only way to access a certain functionality

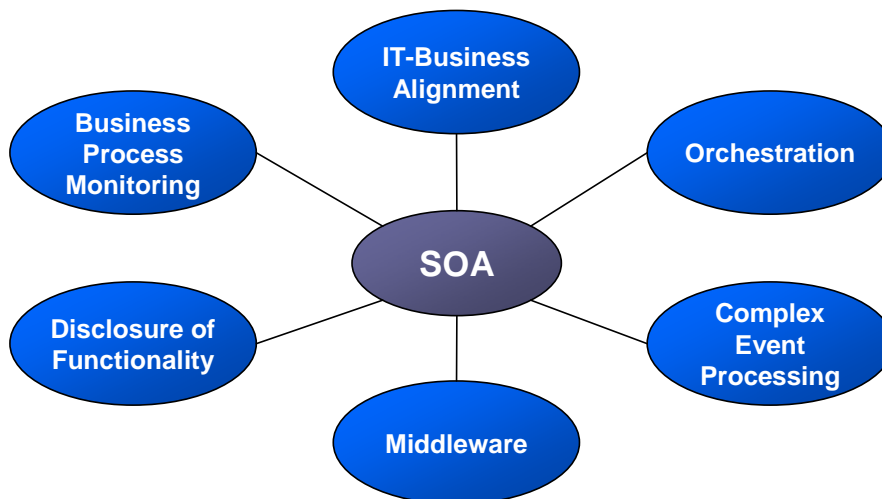


Fig. 4-12: Major concepts combined by Service orientation

The evolution of EA was outlined in this section. Moreover, an SOA reference architecture that depicts the latest style in this evolutionary process was described. SOA is regarded as an architectural style for enterprise architecture that aims at an optimal IT realization (automation) of business processes. The reference architecture is a major part of the SOA definition. The missing part – the SOA service definition will be given in the next section.

4.1.2 SOA Service Definition

This subsection provides the definition of SOA services. The SOA definition starts with the differentiation of the SOA service from the business service and the Web

Service (compare [Christ01]). All three terms can be referred to as service but they have different meanings. This is why they are often confused.

A business service is a business product of an enterprise that requires payment and is directed towards clients only. Usually, there exists a service level agreement (SLA) for each service sold to a client. The SLA specifies the performance of the service and defines penalties if the promised performance could not be delivered. SOA Services represent collections of business functions belonging together because they work on the same topic or business object. They are not contracted with a client and usually of a finer granularity. That means several SOA services will be required to deliver a business service.

An SOA service is regarded as a business function containing several operations. SOA Services are loosely coupled, but they can use each other with preferably low integration effort. Operations of an SOA service are grouped according to business demands. A possible and widespread concept is to identify business objects and their manipulations/actions and then derive a service according to the business object and its operations according to its manipulations/actions. This part adheres from technology and is called the business interface.

Web services are a possible implementation form for SOA services. However and as just mentioned, the SOA service adheres from technology. SOA services can also be implemented with other technologies like CORBA (compare [OMGCOR92]) or JMS (compare [SunMic00]). The Web Service is just a widespread variant of the SOA service implementation.

Having cleared the differences between business services, SOA services, and web services, the definition of the SOA service is rendered more precisely in the following. As an SOA service with interfaces is not self-explaining it needs to be described by a service contract. The contract comprises three blocks of information. In Fig. 4-13 the structure of a service contract is depicted with the example of a cash order service.

The first and biggest part contains the general description items including operation semantics and details like a responsible person, related documents, and interfaces. The description of service operations is mostly given in textual form. Possible other forms are use case diagrams or visual contracts (compare [Lohmann06])

The second block contains information about the operational level agreements (OLA) the service can deliver. For example, only if the SOA services used in a process promise a degree of performance then the degree of performance of the whole process can be predicted.

The third block describes the monitoring and reporting information a service can deliver. Monitoring information constitutes the events that a service sends upon execution. The information carried by these events is noted here. The reporting information describes the performance indicators for which reports can be generated.

SOA-Service Contract for Cash Order Service (COS)

General Description
 The service allows the management of cash orders.
Operations: Plan, Create, Add, Change Date, Change Value, Cancel, Delete, Finish
Description of Operations:
 Plan: Creates cash order with long time in advance. Checks calendar collisions.
 Input: Date, denominations and lot sizes, location
 Output: Cash Order
Responsibility: Managed Services (John Doe)
Technical Interfaces: Web Service, JMS
WSDL Location: http://depb3334:9080/WSRouter_EAI/SRVO.wsdl
 ...

Operational Level Agreement
 Reachable 24/7 at 99.7% availability, Cash Order creation incl. cheks in less than 90 sec.
 Plan cash order needs to be 48h in advance
 ...

Monitoring: Cash order created (EventID, ProcessID, client lots, location)
 Cash order planned on critical date(EventID, ProcessID, client, collison reaseon)
Reporting:
 Monthly report on number cash orders and average execution time
 ...

Fig. 4-13: Example of an SOA service contract

Services are not stand-alone components without any relations to each other. It should be the usual case that services use each other. It works the same way as with components in the context of component-based architectures. In the context of service orientation, this concept got a new name – orchestration. A reason might be that orchestration can be realized in two ways. First, by adding service interface calls in the implementation (hard-wired) of a service or second by using an orchestration engine (soft-wired). Detailed descriptions on this were given in section 4.1.1 .

Fig. 4-14 depicts the structure of a service consisting of one business interface and technical interfaces implementing the business interface. Furthermore, an extensive

description of the service, called service contract describes what the service does. A service contract should not hold information on how it is implemented (except from the available technical interfaces). Although the realization of the service has a determined appearance, e.g. a CRM system that is used by a call centre agent, this appearance is made transparent to the users. Next to this transparency, there are other characteristics for well-built services like adequate granularity and statelessness. These are covered in detail in chapter 5.

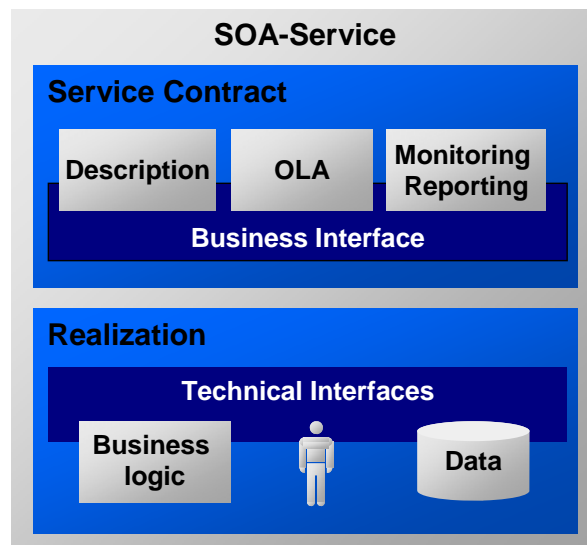


Fig. 4-14: Structure of an SOA service

Next to the description of the service expressed in natural language, in diagrams, or whatever preferred, there should be an Operation Level Agreement (OLA). The OLA is similar to an SLA but it contains no financial details like penalties for unperformed work. OLAs are to be seen as helpful information not as part of a contract. Furthermore, there should be a part in the service contract that states the information that can be monitored (event messages sent) and reported.

This section has, together with the previous section, provided the EA-oriented SOA definition as being used as a foundation for this thesis. A short form of this definition is given as conclusion of this section:

Service-Oriented Architecture

A Service-Oriented Architecture is a productive system of processes, SOA services and applications that work together using the following concepts:

- Middleware
- Disclosure of functionality
- IT-business alignment
- Orchestration
- Complex Event Processing
- Business Process Monitoring

The mediating element is the SOA service. It has a business interface described in a service contract. The business interface offers a set of functionalities and completely abstracts from the realization of these functionalities. An SOA service can have different technical realisations.

After giving a definition frame for an individual EA in the next section, the required formalisation of both definitions can be elaborated in the next chapter.

4.2 EA Definition Frame

According to the requirement R6 an individual EA meta model is required. This section realizes a first step for this requirement by describing an EA definition frame. This frame will be used to define a minimal EA meta model in the next chapter. With the minimal EA meta model the individuality of an own enterprise architecture is restricted as little as possible.

The representation of enterprise architecture is chosen as in fig. 4-15 from [Assman08]. The enterprise architecture is depicted there as a layered structure and as a hierarchical structure. The layers depicted in fig. 4-15 serve only as a categorization and do not indicate a separation of the underlying EA meta model. A distinction between the business, the service, the application, and the infrastructure layer is made in the layered view.

In the hierarchical view, the enterprise architecture can be roughly divided in business and IT architecture. The main elements from the other definitions within the spectrum between business architecture and infrastructure architecture are strategy and goals, organization architecture, process architecture, information architecture and software/application architecture. In addition, the service layer has been added. However, this layer is optional because this representation form shall still allow expressing the structure of non-SOA architectures.

More important are the elements that are in direct relation with the service layer. These are the elements with dashed frames, namely the process architecture, the information architecture, the software architecture, and especially its applications. As the service layer is embedded between these elements, they have to be present in any EA definition.

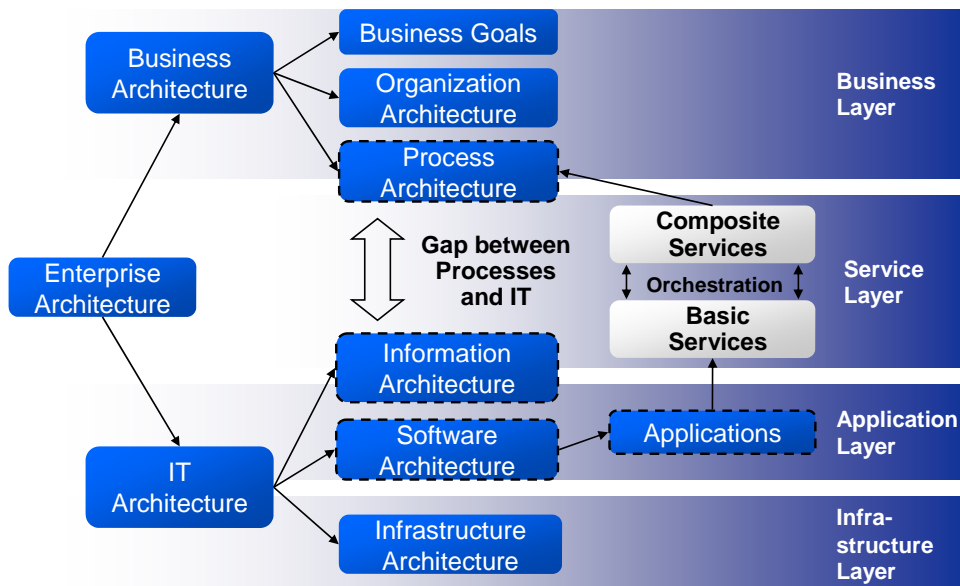


Fig. 4-15: Essential parts of an enterprise architecture (compare [Assman08])

In the rest of this section, the elements shown are described shortly to get an impression of their content. A more concrete meta model for the definition frame will be given in the next chapter.

The goals as part of the business architecture display the strategy an enterprise wants to follow in the mid or long term. Goals like “Market leadership in ATM selling” or “Best-in-Class for Cash Management” are embedded there.

The organization architecture may reflect divisions and departments of an enterprise. At least there should be a role model so that actors in processes and responsible persons can be named anonymously.

There are usually different types of processes in an enterprise. Some are representing the main business; others are internal like development and administration processes. All these kinds of processes are part of the process architecture. When modelling this architecture, mostly the processes with a long lifetime are concerned. Ad-hoc processes being executed only a few times and never occurring again are usually not in scope.

The service architecture containing basic and composite services is part of a Service-Oriented Enterprise Architecture. However, it is possible that these elements do not exist in an enterprise architecture.

The information architecture is an abstraction of the application architecture. It contains the information or business objects that are treated within the processes. Such an information object could be a contract, an order or even an event message generated by a broken ATM. A model of this architecture could also depict logical applications, which are types of applications that are then realized by one or more real technical applications.

The application architecture contains the applications installed and running in an enterprise. Furthermore, the interfaces between the applications are objects of interest here. On this level of architecture, specific technologies such as programming languages and transport protocols play a decisive role.

The infrastructure architecture contains elements that are necessary to operate applications such as operating systems and hardware. It is regarded as optional whether the instances and locations of hardware systems are modelled in this context. Models containing this information can become very large.

This section has described a frame in that an enterprise architecture definition should be. It is a prerequisite for the formalization of a minimal EA meta model. The section also concludes the chapter. Having defined SOA and EA, the formalization of both may be elicited in the next chapter.

5 Formalization of an SOEA Modelling Language

The content of this chapter is depicted in Fig. 5-1. It covers the creation of meta-models for Service-Oriented Architecture, enterprise architecture and the merging process of the two meta models. By that, a modelling language for enterprise architectures is derived that can also express Service-Oriented Architectures. The purpose of this modelling language is to make service orientation of the enterprise measurable and at the same time to use the model for enterprise architecture planning. The measurement of service orientation will be based on a set of criteria that are evaluated with the help of metrics, which is covered in chapter 6.

Before eliciting the single steps depicted in Fig. 5-1, the decision for the meta model approach shall be discussed shortly. This has been done in more detail during the elaboration of the requirements in chapter 2.

In order to change an enterprise architecture in such a comprehensive way SOA requires, planning activities are needed. The planning should be based on a model. Otherwise, the complexity of the whole enterprise architecture would not be controllable. Such models could be informal, e.g. in natural language or drawings. A high ambiguity concerning syntax and semantics are major drawbacks of this approach. For this reason, as a tradeoff between formalism and informalism, a formal meta model with informally described semantics has been chosen. Informal means that the semantics of the meta model elements are given in natural language.

The requirements touching these facts are R2 “SOA formalization”, R4 “Integrated language for EA and SOA”, and R5 “EA Formalization”. R2 and R5 will clearly be fulfilled by the formal meta models. R4 will be partly fulfilled as the two halves of the integrated language are provided by the meta models.

The following sections in this chapter will cover the topics in the dashed frame of Fig. 5-1. The way to find a suitable SOA meta model is described in section 5.1. Afterwards, a minimal EA meta model and an exemplary EA meta model are elicited in section 5.2. Finally, an algorithm to merge the two meta models is elaborated in section 5.3.

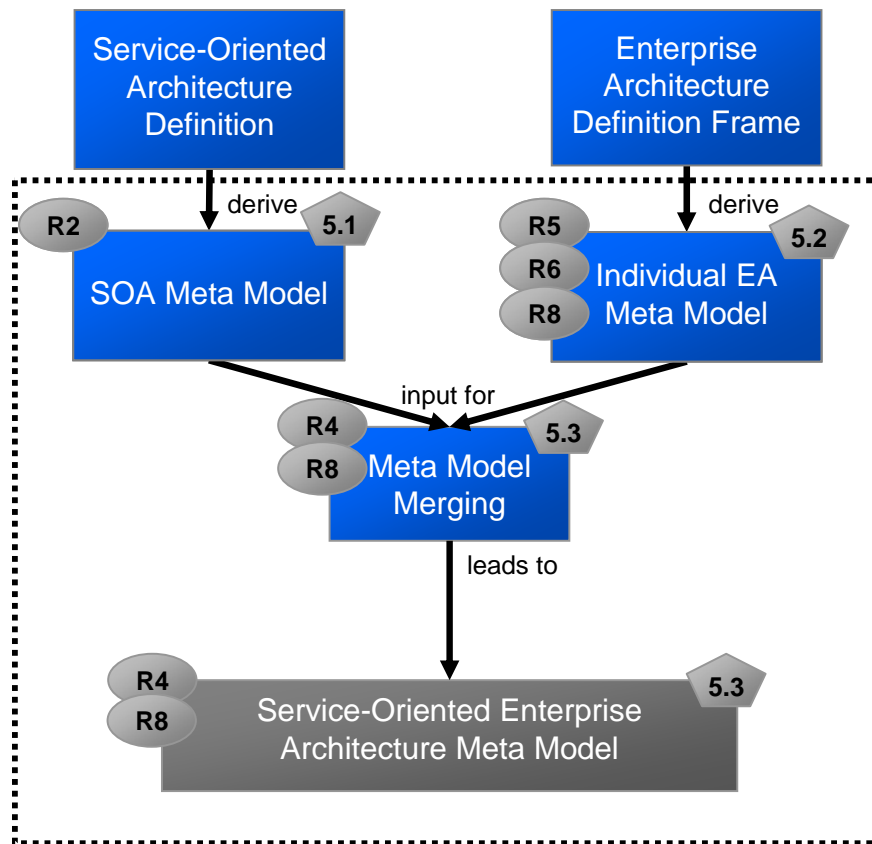


Fig. 5-1: Contribution of chapter 5

5.1 Deriving an SOA Meta Model

The meta model for the Service-Oriented Architecture is derived in this chapter. It is not intended to be changed by an enterprise architect. If the meta model was variable, the metrics that will be defined for the conformance criteria would have a variable basis. Most of the metrics will rely on the SOA meta model. Changing the meta model would result in a much more complicated method. Therefore, this option is not considered here. If an enterprise architect wants to define his own SOA meta model, he can do so but has to be aware that neither the conformance criteria nor their metrics might be applicable anymore for his own approach.

The question whether an existing SOA meta model should be chosen is answered in subsection 5.1.1. It is pointed out that the manual derivation – as described in subsection 5.1.2 from the SOA definition of this thesis is most adequate for the

approach. The result of the manual derivation is an SOA meta model that is to be used in the meta model merging process (compare section 5.3).

5.1.1 How to derive an SOA Meta Model?

In order to derive a suitable SOA meta model, some criteria for adequateness have to be fulfilled. The criteria for a suitable SOA meta model are:

- The SOA definition from this thesis must completely be reflected in the SOA meta model. Otherwise, it cannot be used for planning and evaluating a fully-fledged SOA (R1).
- The detail level should be similar to the detail level of an enterprise architecture meta model, so that the detail level after the merging is consistent (R4). The SOA meta model should only be as detailed as needed. Otherwise, user acceptance is lowered and the effort/benefit ratio is decreased.
- The SOA meta model must be connected, so that a holistic modelling is possible (R8).

Existing approaches are examined concerning their suitability in the remainder of this subsection. It will be shown that the SOA meta model should be derived manually from the SOA definition in chapter 4.

The existing approaches concerned here are the SOA meta model from CBDI Service Architecture & Engineering (compare [CBDISA08]), the OASIS Reference Architecture for SOA (draft status, compare [OASISR08]), and the approach from [Baresi03].

According to [CBDISA08] the “objective in making the model available is to provide a detailed concept model [...] that can form the basis for coherent cross lifecycle asset recording and management.” An asset management or more precisely an IT asset management covers financial, contractual and inventory functions over assets. There are about 90 asset types in the sense of [CBDISA08] reaching from a business process to a single network address. For this reason, the abstraction level of the CBDI SOA meta model is very low. A part of the meta model is showing this is depicted in Fig. 5-2.

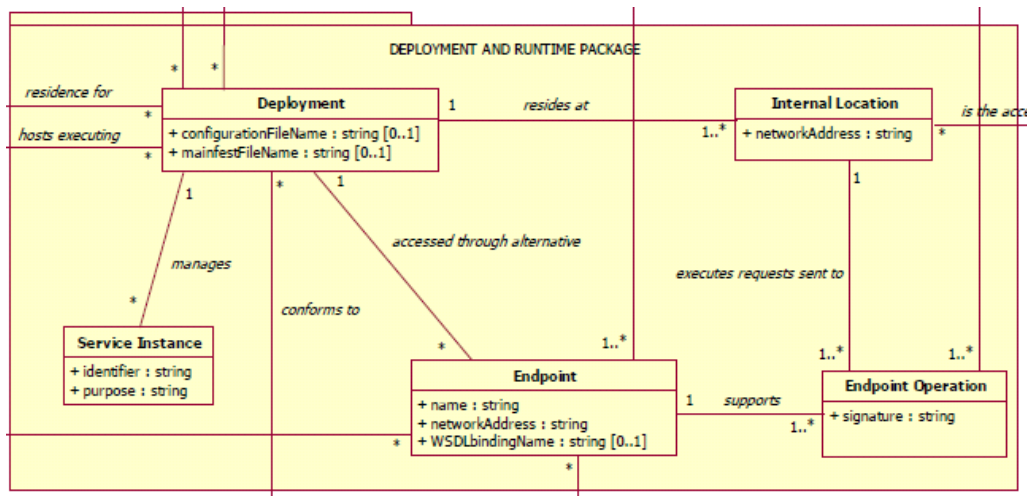


Fig. 5-2: Section of the SOA meta model from [CDBISA08]

The purpose of the CBDI SOA meta model is therefore different from the purpose of the meta model required in this thesis. The CBDI SOA meta model is e.g. to be used to design and implement a software that manages the complete inventory of hard and software assets of an enterprise. On the one hand, asset management works on a low abstraction level and covers probably thousands or even millions of assets. On the other hand, enterprise architecture planning works on a high abstraction level. Hence, the magnitude of the targeted SOA meta model is definitely smaller. Furthermore, the CBDI SOA meta model does not address the concept of business process monitoring.

The OASIS SOA reference architecture [OASISR08] also defines a meta model for a Service-Oriented Architecture. It is given in form of several parts. One of them is depicted in Fig. 5-3. Due to the OASIS, the reference architecture is intended to cover “the issues involved in constructing, using, and owning an SOA-based system.” This SOA meta model is even more complicated than the one from CBDI. It comprises social, governance and other related issues. The detail level reaches down to the log file specific actions are recorded in. The purpose of the SOA reference model is also directed towards SOA asset management. In addition, it is intended to clarify SOA governance issues, like decision processes for SOA service development. For these reasons, the OASIS SOA meta model was not built for the purpose required here and thus does not deliver the needed granularity. In addition, the OASIS SOA reference model also does not cover the concepts complex event processing and business process monitoring adequately. For these reasons, the SOA meta model provided by [OASISR08] does not fulfil the criteria for an adequate meta model, neither.

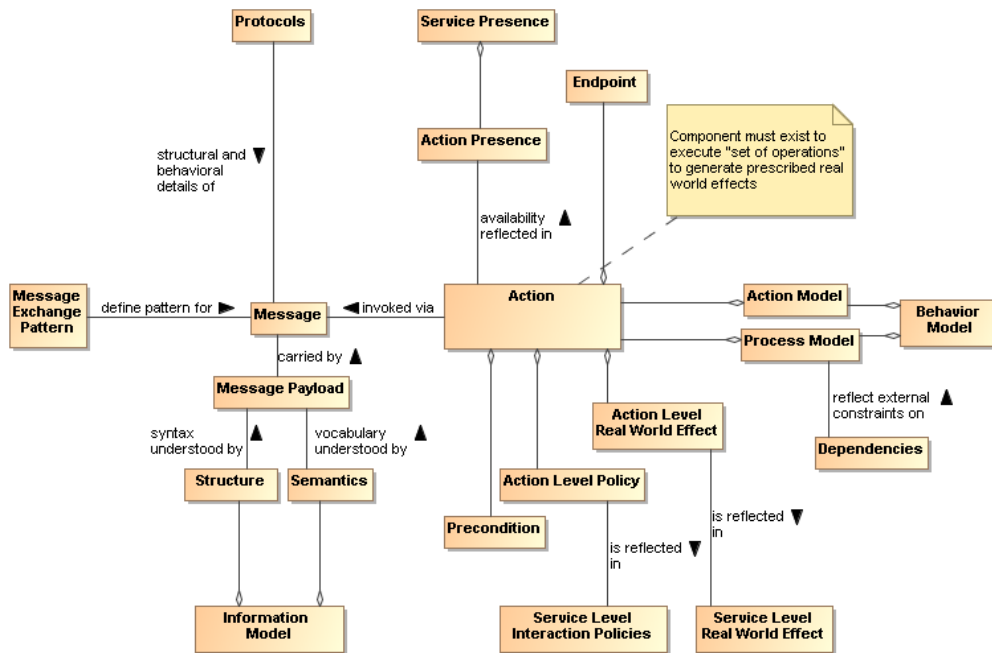


Fig. 5-3: Part of the SOA meta model from [OASISR08]

Already in 2003, a meta model for SOA was presented in [Baresi03]. The intention of this meta model was to be able to predict the possible configurations of this architecture. A configuration means which services can communicate with each other and which sequence of operation can be realized. This kind of meta model is not adequate because it was not designed to plan enterprise architecture. None of the high-level concepts like Event-Driven Architecture or workflow management has been considered.

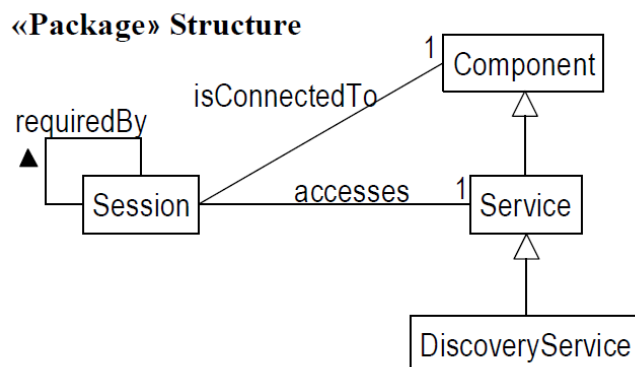


Fig. 5-4: Part of the SOA meta model from [Baresi03]

Criteria \ Meta Model	CBDI	OASIS	[Baresi03]	Targeted solution
Main Purpose	SOA Asset management	SOA Asset Management, SOA Governance	Service Configuration determination	EA Planning
SOA definition conformance	5/6	4/6	2/6	6/6
Granularity (# concepts)	~90	~120	~15	~20

Fig. 5-5: Overview on adequacy of existing SOA meta models

The adequacy of existing meta models is summed up in Fig. 5-5. Neither of the approaches has been chosen, because of the different purpose and granularity level of the approaches. There would have been the possibility to adapt the CBDI or OASIS meta model. At first, they had to be reduced to the appropriate detail level, i.e. leaving out 4/5 of the concepts. Eventually, some further adjustments of the core had to be taken. In addition, the missing main concepts had to be added to the model. After all these steps, the remaining content of the existing models would be diminished to a very low level. Furthermore, the OASIS and the CBDI SOA meta model have been developed in parallel to this work and were not published until 2008.

Several existing meta model approaches have been examined, but none of them fulfils the criteria for a meta model that is adequate for this approach. The too high detail level and the missing concepts like business process monitoring or Event-Driven Architecture are most problematic. For this reason, the derivation of the SOA meta model is elaborated in the next section.

5.1.2 Deriving Concepts from the Given SOA Definition

In this section, the definition of a fully developed SOA from chapter 4 is reconsidered and a meta model is derived. The definition is analyzed concerning concept candidates for an SOA meta model. In order to identify all the concepts that are needed to describe Service-Oriented Architectures, the reference architecture (compare subsection 4.1.1) and the SOA service definition (compare subsection 4.1.2) will be discussed concerning their conceptual elements. The candidate elements of the SOA meta model will be marked in *italic* style.

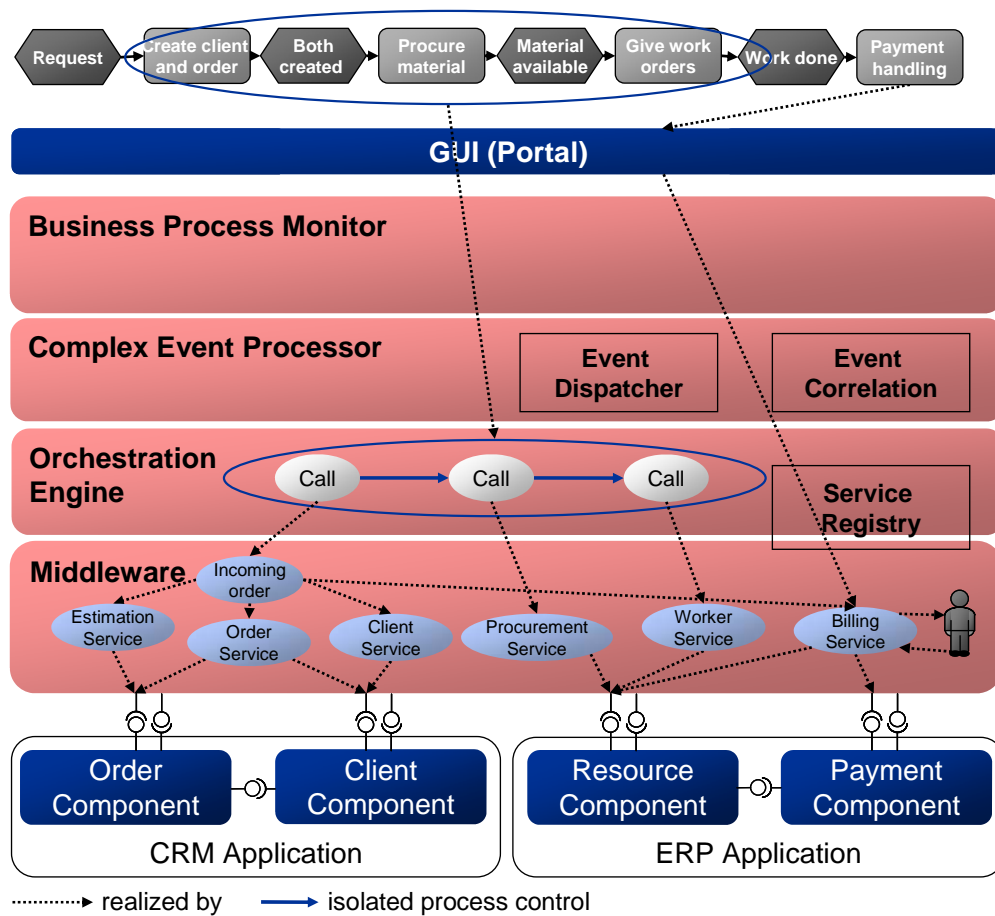


Fig. 5-6: Layers in Service-Oriented Enterprise Architecture

The reference architecture from section 4.1 is shown again in Fig. 5-6. The major concepts of service-orientation from Fig. 4-12 are one by one revisited in order to derive candidate elements of an SOA meta model.

The most obvious candidate element related to middleware is the *interface*. An interface offers the functionality of an *application* or *application component*. In addition, the interface is implemented in a *technology* of which one is the *middleware* technology. Furthermore, interfaces can have write or read access on *business objects*.

The IT-business alignment primarily concerns the *SOA service*. A SOA service consists of a *service contract* and a *service realization*. The service contract describes the *business interface* of the SOA service. The service realization specifies a *service*

interface with which the business interface is instantiated. All the SOA services have to be registered in a *service registry*. A *service repository* is an application containing a service registry. In addition, the service repository offers functionalities that support architects and developers. It contains a model of the elements used in the software development process for Service-Oriented Architectures. Preferable features of a repository are a free definable object meta model and taxonomy of software development elements, a role model support, a versioning of objects and attached documents and a lifecycle support for arbitrary object types.

The disclosure of functionality mainly concerns the applications and their interfaces. Due to this concept, *graphical user interfaces* may not contain the only possibility to reach a piece of functionality. Each piece of functionality has to be reachable by interfaces suitable for automation, preferably in the middleware technology. If a proprietary interface cannot be used by an SOA service, then a *service integration adapter* (SIA) is created for this interface. The adapter itself is not an SOA service but only a copy in a different technology.

Orchestration concerns the automated execution of *business process steps*. An *orchestration* is regarded as executable pattern of linked business process steps. Such an orchestration is executed by an *orchestration engine*. The automated execution of business process steps does not distinguish between completely electronically work or and steps in that employee *roles* are involved.

The main element for complex event processing is the *complex event processor*. Its *event correlator* correlates the *events* the *event dispatcher* has received. The correlation is based on predefined *correlation rules*. An event correlated from several other events is called *complex event*.

The business process monitoring focuses *business processes* and their performance. Therefore, *performance indicators* are defined for a business process. The *business process monitor* is an application that supports the determination of performance indicators, the required information retrieval, and the presentation of the monitored indicators. The examination of the business process monitoring concludes the candidate concept examination.

As the concepts were gathered together indiscriminately, some of them have to be reconsidered for combining or skipping, which is done after listing the candidate

concepts. Combining and skipping are options for adjusting the detail level of the resulting meta model.

Middleware

- Interface
- Application
- Business object
- Technology
- Middleware
- Application Component

IT-business alignment

- SOA service
- Service contract
- Service realization
- Business interface
- Service interface
- Service registry
- Service repository

Disclosure of functionality

- Graphical user interface
- Service integration adapter

Orchestration

- Business process step
- Orchestration engine
- Orchestration
- Role

Complex event processing

- Complex event processor
- Event Dispatcher
- Event Correlator
- Correlation rules
- Event
- Complex event

Business process monitoring

- Business process
- Business process monitor
- Performance indicator

The complex event processor with its dispatcher, correlator, and correlation rules is regarded as one concept, because it usually is a single application built these components. The only difference between events and complex events is their origin. Complex events are created by the correlation engine. As they do not differ in their appearance, complex events and events are regarded as the same concept.

The SOA service consists of a business interface and a service realization. The realization is given implicitly through its relation to other concepts, e.g. service interfaces. The business interface is regarded as the SOA service itself. That means these two concepts are merged to the SOA service.

The concept middleware is not taken explicitly here. A middleware will be regarded as a technology for interfaces. Often the middleware cannot be named clearly as there

are several technologies used in practice, but only the one with the highest prevalence or the one favoured in the IT-strategy is considered as official middleware. For this reason, middleware is regarded as interface technology. Application components are also not regarded in the desired detail level. An application owns all the interfaces the different components have.

With these changes the following list of concepts remains:

Middleware

- Interface
- Application
- Business object
- Technology
- Middleware

IT-business alignment

- SOA service
- Service interface
- Service registry
- Service repository

Disclosure of functionality

- Graphical user interface
- Service integration adapter

Orchestration

- Business process step
- Orchestration engine
- Orchestration
- Role

Complex event processing

- Complex event processor
- Event

Business process monitoring

- Business process
- Business process monitor
- Performance indicator

In this subsection, the concepts for an SOA meta model have been derived. In the next chapter, a meta model with its relations between the concepts is formed out of these concepts.

5.1.3 Deriving a Meta Model from Identified Concepts

Having identified the relevant concepts for an SOA meta model, the creation of the meta model may be completed. The complete meta model will be elaborated in the remainder of this section.

There are no associations between the concepts yet. In the following, the associations are elaborated by discussing the identified concepts. Relevant

associations are noted after each paragraph in the following form. The arrows indicate the reading direction of the association.

Concept A → association → Concept B

The concept interface has several subtypes. These are the graphical user interface (GUI), the service interface and the service integration adapter. The interface itself is regarded as a general type of interface that is used to describe interfaces of legacy applications.

Interface → is generalization of → GUI
Interface → is generalization of → Service interface
Interface → is generalization of → Service integration adapter

The interface always deals with business objects. These can be read or stored by an interface.

Interface → has write access → Business object
Interface → has read access → Business object

An SOA service can require other SOA services or applications via interfaces. SOA services are used via service interface or they require service integration adapters. Service integration adapters adapt legacy interfaces to the current middleware technology so that they can be used by SOA services, too. SOA services also provide their functionality with at least one interface. Any interface must be implemented in at least one kind of technology.

SOA service → require → Interface
SOA service → provide → Interface
Interface → implemented in → Technology

Interfaces are also provided and used by applications. Among the applications are the legacy applications and some special SOA related applications. These are the orchestration engine, the business process monitor, the complex event processor the service registry and the service repository.

Application → is generalization of → Orchestration engine
Application → is generalization of → Business process monitor

Application → is generalization of → Service registry
 Application → is generalization of → Complex event processor
 Application → is generalization of → Service repository

The service repository always contains a service registry and each SOA service should be registered in a service registry. An orchestration engine can execute orchestrations (like BPEL models) This means that SOA services are used by the orchestration in a certain order. The purpose of the business process monitor is to observe business processes.

Service registry → is part of → Service repository
 SOA service → is registered in → Service registry
 Orchestration engine → can execute → Orchestration
 Orchestration → use → SOA service
 Orchestration → cover → Business process step
 Business process monitor → observe → Business process

An application is realized in one or more technologies and can receive and fire events. Furthermore, an application may provide or require interfaces. An application can support a business process.

Application → implemented in → Technology
 Application → can fire → Event
 Application → can receive → Event
 Application → provide → Interface
 Application → require → Interface
 Application → support → Business process step

SOA services can realize business process steps and can fire events that can be required by performance indicators. Business processes have performance indicators for evaluating their quality and consist of business process steps. Orchestrations can realize business process steps.

SOA service → can fire → Event
 SOA service → realize → Business process step
 Performance indicator → require → Event
 Business process → have → Performance indicator

Business process	→ consist of	→ Business process step
Orchestration	→ realize	→ Business process step

A business process step and an SOA service can also be realized by a role. Often this role uses a GUI to realize the process step.

Role	→ realize	→ Business process step
Role	→ realize	→ SOA service
Role	→ use	→ GUI

Finally yet importantly, the concepts have to become meta classes. In addition, they need attributes specifying them. Instances will have a name and a description. The realization of this is given as abstract meta class “Named Element”. Every other meta class inherits from this abstract one. The diagram does not show this, because this would result in a confusingly huge number of arrows providing only little information. The resulting meta model is depicted in Fig. 5-7.

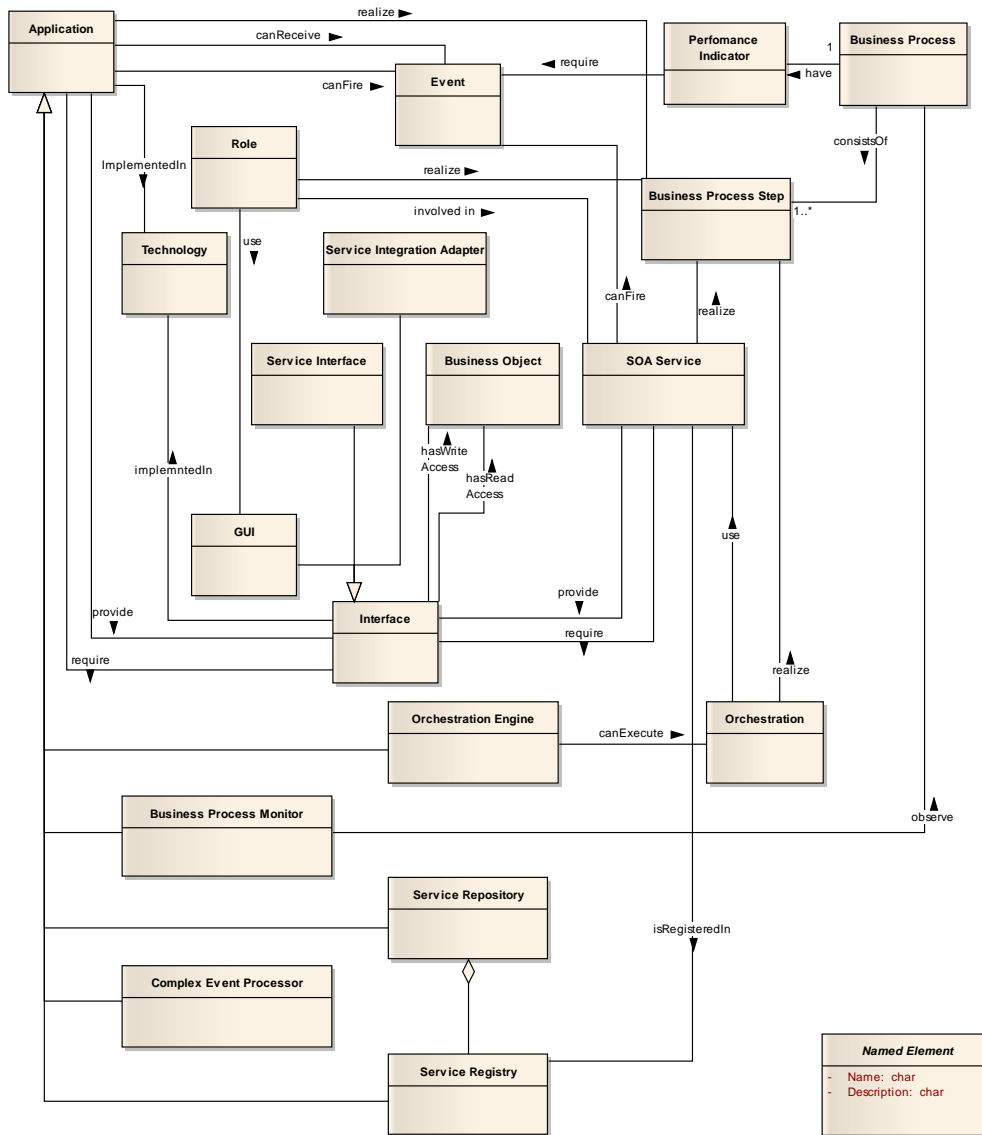


Fig. 5-7: Diagram with SOA meta model

In this section, the SOA meta model has been defined. The derivation of an enterprise architecture meta model is described in the next section. This is done in order to be able to integrate both models to an SOEA meta model.

5.2 Deriving an Enterprise Architecture Meta Model

With the SOA meta model at hand an enterprise architecture meta model is required in order to fulfil the requirement of a formal and individual EA meta model within the approach of this thesis (R5 and R6). Furthermore, the EA meta model is needed for the derivation of the SOEA meta model that formalizes the language containing SOA and EA concepts. In section 4.2 an EA definition frame was given. This frame is used to derive a minimal EA meta model. Afterwards, an approach to define an individual EA meta model is described.

5.2.1 Deriving a minimal EA meta model

The minimal EA meta model will have some commonalities with the SOA meta model. An intersection of the two meta models ensures that the resulting merged meta model will be a connected graph. An unconnected graph would lead to the situation that R8 “Holistic EA Modelling” is not fulfilled.

The minimal EA meta model restricts the individual meta model in the way that all the concepts from the minimal EA meta model have to be present in the individual EA meta model. Extensions of the minimal EA meta model are allowed.

In Fig. 5-8 the previously defined frame of the EA definition is depicted. The concerned elements are the process architecture, the information architecture, the software architecture and from this one especially the applications. In the following the main concepts concerning these elements are identified and brought into relation so that the minimal EA meta model can be derived. The notation of concepts and associations is the same as in the previous section. If possible, then meta elements from the SOA meta model are used.

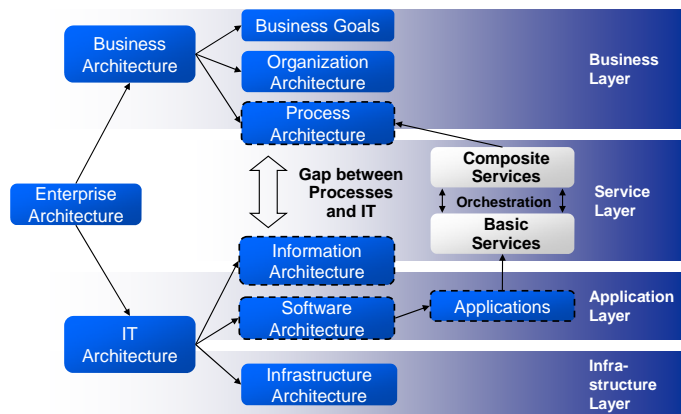


Fig. 5-8: Essential parts of an enterprise architecture (compare [Assman08])

The process architecture has the *business processes* as a main element. A *business process* consists of *business process steps*, which are central elements in the minimal meta model.

Business process → consist of → Business process step

Moreover, *business process steps* are realized by *applications*. These *applications* usually offer *interfaces* and may also require *interfaces* for their function.

Application → supports → Business process step

Application → offer → Interface

Application → require → Interface

This concludes the derivation of the minimal EA meta model that is finally depicted in Fig. 5-9. The red coloured concepts represent the application layer and the blue coloured concepts represent the business layer. Leaving out any of these concepts would lead to underrepresentation of a layer. Therefore, this is not suggested. As desired, there are structures in the minimal EA meta model that are overlapping with the SOA meta model.

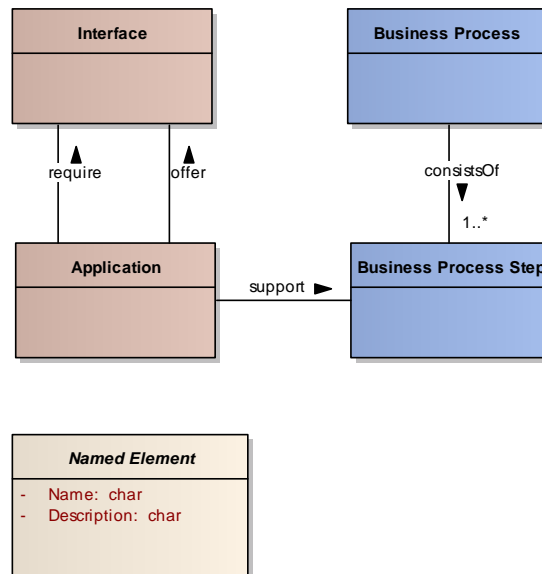


Fig. 5-9: Diagram with minimal EA meta model

The minimal EA meta model that was constructed in this subsection is the premise for the merging of the SOA and an individual EA meta model. Any individual EA meta model has to be an extension of this meta model. This means meta classes, attributes, and associations can be added as desired. Leaving out or changing existing meta classes, attributes, or associations is not allowed. Otherwise, the merging of the two meta models is complicated and the focus on enterprise architecture may get lost.

Probably, an EA meta model that already exists in an enterprise does not include exactly the given minimal EA meta model. In this case, at least matching meta classes have to be found. Because of the coarse granularity and the essential meaning of the meta classes in an EA meta model, this is regarded as possible. The matching meta classes have to be integrated in the individual EA meta model. Their meta associations should also find matching partners, but if none is found, they can just be added. Of course and from then on, these added meta associations have to be minded in the modelling process.

If it is not possible to find at least matching classes, then the individual EA meta model is regarded as inadequate for an EA description and with this also inadequate for further use in this method.

In this section a minimal EA meta model was defined that leaves enough freedom to use an own and existing EA meta model but also determines the smallest set of requirements for an EA meta model. For users that do not have an own EA model at hand, the definition process for an own EA meta model is illustrated shortly in the next subsection.

5.2.2 Defining an individual EA meta model

Unlike the definition process of the SOA meta model, the definition process of the EA meta model shall be a very individual step. There is related work in the field of EA meta models like in [Braun05], [Engels08] and in [Butler07]. These can be chosen as long as they are compliant to the minimal EA meta model.

Enterprises often have very individual structures that are different from structures suggested in existing EA meta models. One reason to insist on an individual meta model is that an existing meta model is already used in another context. Furthermore, the nomenclature can be very different and people refused to adapt to the new terms. The learning effort for the new nomenclature can thus be too high.

Moreover, there can be concepts within an existing meta model that are out of the desired focus. Some are disregarded because the information retrieval would be so expensive that the costs would outweigh the benefit. Others could be simply not of interest for the EA management.

Furthermore, there could be concepts that are not comprised by an existing meta model. As these concepts cannot be omitted without significantly lowering the usefulness for the users, they have to become part of an individual EA meta model.

For these reasons, a way to create a completely individual EA meta model is described in this section. Every reader of this thesis is free to choose an existing meta model, adapt it, or to create a new one.

For this thesis, many discussions with employees were lead for defining an individual enterprise architecture meta model. During these discussions, the relevant concepts and relations were taken down and a first meta model, being consistent with the minimal EA meta model, was depicted. After a review phase, the final meta model was created.

In the following, the concepts found in the interviews are given as a list. Afterwards, their relations are described and the associations for the meta model are pointed out.

- Business goal
- Business application
- Application
- Interface
- Key performance indicator
- Graphical user interface
- Operating system
- Application server
- Data base
- Workflow management tool
- Technology
- Business process
- Business process step
- Deployment component
- Business event
- Business event message
- Business object
- Business service
- Sub service
- Organizational unit
- Role
- Service provider
- Contract

A business goal is part of the business strategy. It may be formulated vaguely, but should be as precise as possible. An example for a vague goal is to become technology leader in a certain sector, because technology leadership might not be clearly defined. The goal to generate 200 million € of volume is more precise. In every case, a business goal should be supported by processes. Otherwise, its fulfilment will probably fail.

Business process → support → Business goal

A business process is a complex sequence consisting of several process steps or even other business processes. Its purpose can be to realize a business service. Furthermore, the business process may have performance indicators that make the quality of the process measurable.

Business process → consist of → Business process step
Business process → is part of → Business process
Business process → realize → Business service
Business process → have → Key performance indicator

A business service is a kind of product or service that the enterprise sells to its clients. In the simplest case, it is a piece of hardware, like an automaton. It may also

be a service that guarantees the functionality of an automaton. In this case, there may be several sub services that the business service consists of. A set of business services is bundled and sold to a client, which is escrowed in a contract. The business service is the smallest element a contract may provide. A contract is an agreement with a client on a set of defined business services to certain conditions.

Contract	→ provide	→ Business service
Business service	→ consist of	→ Sub service

A sub service is the smallest part of a business service that can be fulfilled by a single service provider. The service provider may be any kind of legal person that is able to fulfil a sub service.

Service provider	→ deliver	→ Sub service
------------------	-----------	---------------

There is exactly one responsible organizational unit for a business process step. If an organizational unit acts as a service provider for a sub service, it will be responsible for the business process steps realizing the sub service. Business process steps also work on business objects, which can be accessed in a reading or writing way. During the execution of a process step business events may occur. These business events may be triggered by an application supporting the execution of the business process step, a role (employee) that acts in the process step, or any external system or actor.

Business process step	→ realize	→ Sub service
Business process step	→ hasReadAccess	→ Business object
Business process step	→ hasWriteAccess	→ Business object
Business event	→ occur in	→ Business process step

An organizational unit holds responsibilities. Firstly, the responsibility for process steps. The organizational unit has to ensure that the process step can be executed in the context of any business process. Secondly, an organizational unit hosts applications. That means it has to take care that the application is working correctly all the time. Thirdly, roles have to be provided by them. Roles stand for human actors that have certain skills and act in business processes.

Organizational unit	→ responsible for	→ Business process step
Organizational unit	→ host	→ Application
Organizational unit	→ provides	→ Role

Role → act in → Business process step

A business object is the input or output for a process step, for example an automaton or an invoice. The business objects do not have to exist in material form; they can also be seen as a piece of information. A business object can have different attributes, like the invoice that has a sender and a receiver. As the receiver of an invoice is a business object itself, business objects can contain other business objects.

Business object → is part of → Business object

An application is a piece of software that fulfils a certain purpose in a process step. This reaches from infrastructure software to business applications like a customer relationship management system. The concept infrastructure is not treated as a single concept here, although this would be possible. Instead, several application types are regarded as attribute of an application. The different application types represent the infrastructure needed here. The different application types concerned here are operating system, database, application server, and workflow management tool and business application. Business applications are all applications that do not fit into another category.

Application → support → Business process step
Application → is generalization of → Business application
Application → is generalization of → Application server
Application → is generalization of → Operating system
Application → is generalization of → Data base
Application → is generalization of → Workflow management tool

Applications can be packaged in deployment components. That means that this set of applications is always deployed together. This helps to define approved combinations of applications and eases their version management.

Application → is part of → Deployment component

Each application offers interfaces allowing the invocation of operations. When communicating with other applications, an application requires interfaces that are offered by the other applications used.

Application → offer → Interface
 Application → require → Interface

An interface can generally have two different types – a graphical user interface and a general interface. The graphical user interface allows employees (roles) to interact with the application. The general interface allows applications to communicate with each other.

Interface → is generalization of → Graphical user interface

Technologies are implemented by interfaces and applications. For interfaces, these are protocols like JMS (compare [SunMic00]), CORBA (compare [OMGCOR92]), or Web Services. For applications, these are the programming languages and technical frameworks.

Application → implement → Technology
 Interface → implement → Technology

Business event messages are the electronic equivalent of intangible business events. They contain execution time, context, and content data of a business event. Furthermore, they are created by interfaces. They can be consumed to compute key performance indicators that are monitored in real time.

Interface → can create → Business event message
 Key performance indicator → need → Business event message

Key performance indicators are measurement instruments to monitor the quality of business processes. Each process can have several of these indicators defined.

Business process → have → Key performance indicator

At this point, all concepts and their relationships have been described. From these the meta model in Fig. 5-10 is derived. Just like in the SOA meta model, the abstract class “Named Element” has been added. All other classes are sub classes, so that their instances have minimum set of attributes for identification and description. The colours of the meta classes indicate their layer affiliation.

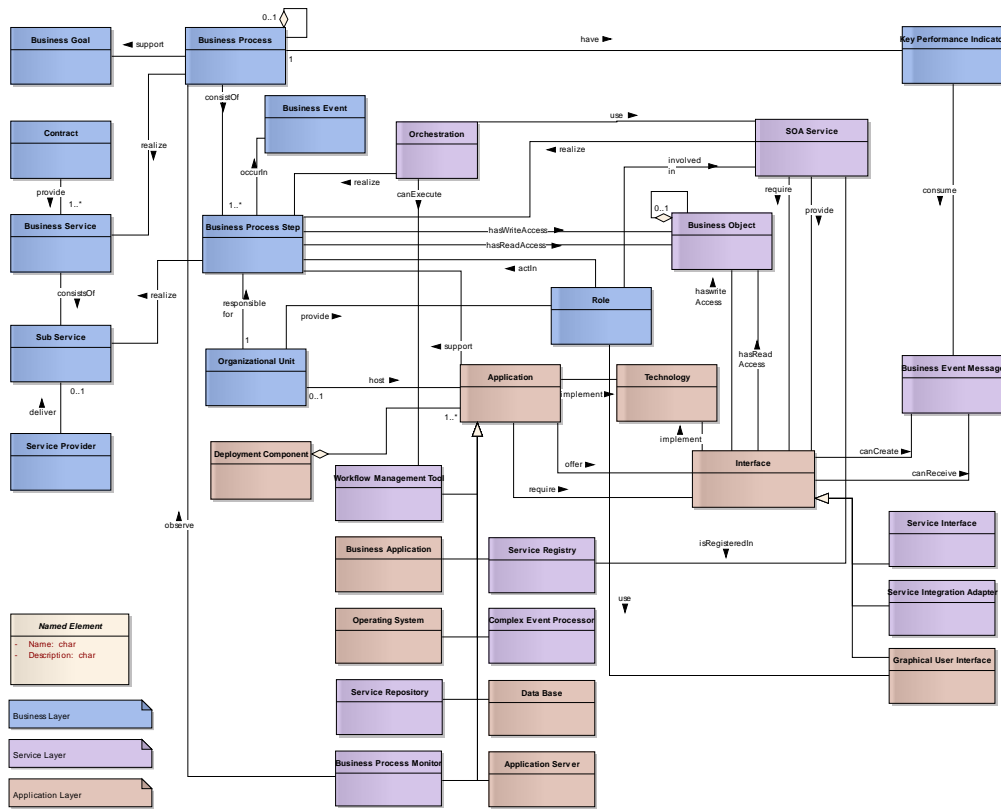


Fig. 5-10: Diagram with EA meta model

The way to find an individual EA meta model was illustrated in this section. Together with the SOA meta model, the union of the two meta models can be worked out. By this, the foundation for the planning of the Service-Oriented Enterprise Architecture is created.

5.3 Union of the SOA and the EA Meta Model

This section describes the merging process of the SOA and the EA meta models. The merging will exemplarily be shown with the parts of the meta models derived in the previous sections. The complete meta models would be too big as examples. Their merged version is shown after the description of the merging process.

The merging of the meta models is necessary, because of several requirements. At first, R8 demands holistic modelling, which forbids the usage of two different meta models. Moreover, R4 demands an integrated language for SOA and EA. This

language also has to be formal (R5) and the EA part has to be individual (R6). Merging the individual EA meta model with the SOA meta model will fulfil these requirements.

There are already approaches on how to merge meta models. These have emerged in the fields of very large databases (VLDB, compare [VLDBOR09]) and the semantic web (compare [Berners01]), mainly. Some approaches like [Madhav02] make use of existing instances of meta models and apply learning algorithms. These are not suited for the problem here, because there is probably at most one EA meta model per enterprise (and probably none for the SOA meta model). Learning algorithms are not applicable with such a small number of instances.

Other approaches, like the model management approach in [Bernst03] work on the meta models only. The approach of [Bernst03] is picked up and used for the meta model merging in this thesis.

In [Bernst03] generic operators working on meta models are described. The ones being useful in this context are match and merge. As the term merge has already been used for the whole process of delivering a unified meta model, merging in the sense of [Bernst03] is referred to as joining. Match takes two models and returns a mapping between them. Join takes two models A and B and a mapping between them and returns the union C of A and B. Hence, the merging problem is split into the two sub problems matching and joining.

Unfortunately, in [Bernst03] only the semantics but not the implementation of the operators is given. For the implementation of the matching and the joining operator, two different references are used. The matching follows the approach of [Lagers08], which is based on the main idea that two concepts match if concepts in their neighbourhood match. The joining follows the ideas given in [Borona07], which describes possible joining conflicts and suggests resolution strategies. Both do not work on a whole model, as suggested in [Bernst03], but on single concepts of a meta model.

Before starting the manipulation of the meta models, these will be transformed in a representation form that is easier to process as the graph form. As already mentioned, merging algorithms originate from the field of databases. That is why the simple table is chosen as representation form.

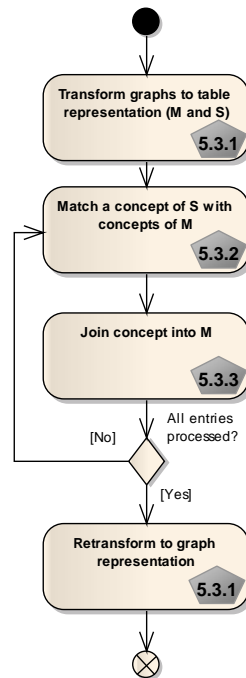


Fig. 5-11: Simplified meta model merging process

In Fig. 5-11 a first simplified form of the merging process is depicted. It contains the main operations used in the algorithm, which are described in the next subsections. At first, the graphs of the meta models are transformed into table representations, namely the master and the slave table (M and S), which is described in subsection 5.3.1. This is followed by describing the matching operator in subsection 5.3.2. Afterwards, the joining operator for entries having been matched is elicited in subsection 5.3.3. When the point is reached that all entries have been matched and joined, then the resulting table can be transformed back to a meta model graph. This is simply the reversion of the previously applied transformation process and described in subsection 5.3.1.

With the understanding of the single operations, the complete algorithm will be described in detail in subsection 5.3.4. The result of its appliance on the SOA and the EA meta model is given in 5.3.5.

5.3.1 Transformation to Table Representation

This subsection describes how the meta models are transformed into table representations. There are two reasons why the meta model graphs are transformed to tables. The first reason is to be able to process the elements of the meta model with ease. The table representation allows sorting, marking and enumerating entries with a simple spreadsheet program. Using the graphical representation form could easily lead to confusion of the user, because too many graphical elements have to be processed at once. In table form, this can be done entry for entry. Secondly, to abstract from the syntactical structures the meta model defines. This is done because it may happen that a concept in one meta model is expressed as a meta class, and in another meta model it is just represented by an attribute of a meta class. For this reason, attributes, concepts, and inherited concepts are transformed into the same representation form. In the following, these will be referenced as (meta modelling) artefacts.

In Fig. 5-12 a generic example with all the elements that are concerned in the transformation is given. Exactly one entry is generated for each artefact (concepts, inherited concepts, and attributes). This allows that each of them potentially is a concept, inherited concept, or attribute in the target meta model.

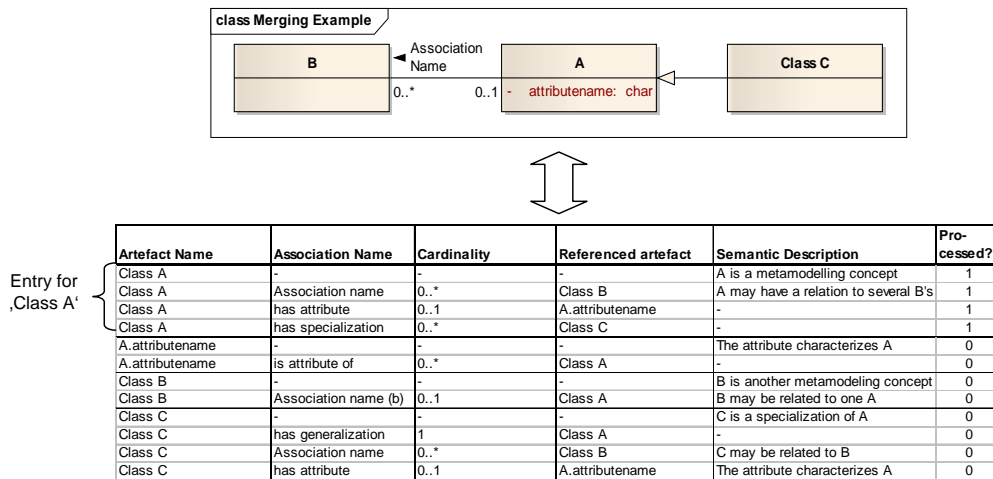


Fig. 5-12: Generic example for table transformation

An entry can consist of several rows, like the entry for Class A. The first row represents the artefact itself and the following rows represent its associations. Any undirected association is split into two directed associations. Only the directed associations originating in the concerned artefact are considered in the entry for the

artefact. Each row of an entry has six fields (columns). These are the names of the concerned artefact, the name of an association of the concerned artefact, the cardinality of the association, the referenced artefact, the semantic description, and a boolean flag that is not important now. The second to fourth fields are always empty for the first row of an entry.

There are special association types for attributes and inheritances defined for the second field. If an artefact has an attribute or inheritance relation, then this results in an extra row. In this case, the second field is filled with “has attribute”, “is attribute of”, “has specialization”, or “has generalization”. Of course, normal associations also occur in the opposite direction in the case of bidirectional associations. For normal associations the name is concatenated with “(b)” for backwards.

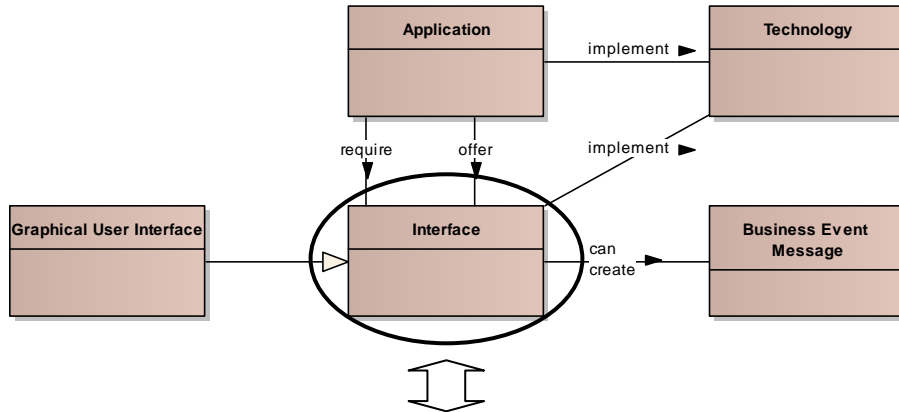
The third field describes the cardinality of the association to the referred object. If there are cardinalities denoted in the graph, then these have to be chosen. As the cardinality field may not be left empty, default values are defined. The default values depend on the association types:

Normal association	0..*
Normal association (b)	0..*
Has attribute	0..1
Is attribute of	0..*
Has specialization	0..*
Has generalization	1

The third field contains the information about the artefact that is referenced by the association. In the case of attributes or inheritance relations, the corresponding attribute or class is inserted.

When applying these transformation rules on the graph given in Fig. 5-12, the table below the graph is generated. With this method, both meta model graphs are to be transformed into a table.

In the following, a running example is introduced. The artefact interface from the EA meta model is considered from now on. In Fig. 5-13 Interface and its associated concepts are shown. The table below is the result of the transformation for just interface. The neighbouring classes are only shown, because they are referenced by associations of interface.



Artefact	Association Name	Cardinality	Referenced Artefact	Semantic Description	Processed?
Interface				Connects applications	0
Interface	has specialization	0..*	Graphical User Interface		0
Interface	implement	0..*	Technology	An interface is implemented in a ...	0
Interface	can create	0..*	Business Event Message		0
Interface	require (b)	0..*	Application		0
Interface	offer (b)	0..*	Application		0

Fig. 5-13: Transformation of 'Interface' to table representation

After the generation of the two tables for the EA and the SOA meta model, a decision on the master and the slave table has to be met. The reason for this originates in the conflict resolution strategy given in subsection 5.3.3. The slave table will later be inserted in the master table. The result will slightly be influenced by the choice. That means if there are similar or equal artefacts it is more likely that the artefact characteristics from slave table artefacts will be discarded.

The reversion of the transformation is achieved by applying the transformation rules the other way round. At first, only the artefacts without their associations are processed. That means only the first row of an entry and the "is attribute of" rows are processed in the first run. When all concepts are transformed, the associations can be transformed as well. Having the meta models transformed to tables, the matching of artefacts can be elicited in the next subsection.

5.3.2 Matching Artefacts

The matching of artefacts is required to be able to decide which artefacts are similar enough to be united during the join operation. The matching operator will be defined for two artefacts resulting in a number indicating similarity. The similarity measure is given as a ratio between zero and one. One means that they are equal in their semantics. Its signature reads:

$$\text{artefact} \times \text{artefact} \rightarrow [0,1]$$

The matching technique presented here is based on the idea of Bayesian networks (compare [Jensen08]) that are used as suggested in [Lagers08]. The approach allows determining the grade of similarity of two meta classes or meta attributes in an automated way, because it was designed to automate the merging of larger meta models. As the scenario of merging an EA and an SOA meta model is still doable by hand and complete automation is not needed because of the relatively rare usage, the approach of [Lagers08] is adapted here. Every evaluation step can be influenced by an expert. Even in [Bernst03] this had been suggested for the complex matching operation. The use of the Bayesian network approach is to be seen as decision support only.

In the rest of the subsection, Bayesian networks are introduced and the way of their usage in the matching operation is elaborated. Afterwards, a small example from the SOA and EA meta model is given.

A Bayesian network B consists of a directed acyclic graph G and a conditional probability distribution P over the nodes of the graph, shortly $B = (G,P)$. The graph consists of a set of vertexes V and a set of edges E . That means $G = (V,E)$, where $E = V \times V$. Each vertex represents a variable $V_i \in V$. The variables concern the similarity of a single aspect of two joining candidates and have a value v_i in the finite set of $\text{val}(V_i)$. Usually, these variables are random variables and their distribution is given through probability tables, but here their distribution is given by the instances of the artefacts to be joined and formulas on how to compute the resulting probabilities.

Edges denote causal dependencies between the nodes. All nodes that are connected to V_i by incoming edges build the set of parent nodes $\text{pa}(V_i)$. The outgoing edges of V_i build the set of child nodes $\text{ch}(V_i)$. That means, a source node of an edge is a parent

and a target node is a child node. P describes how the variable values from the nodes are distributed.

In Fig. 5-14 the Bayesian networks for artefact and association matching are given with example values. The artefacts that were compared there cannot be seen in this diagram, but only the result of the comparison. The child nodes of the association node all have simple set of values, which is either {equal, unequal} or {equal, similar, unequal}. Each of these values can be either one or zero but the sum within each node has to be exactly one.

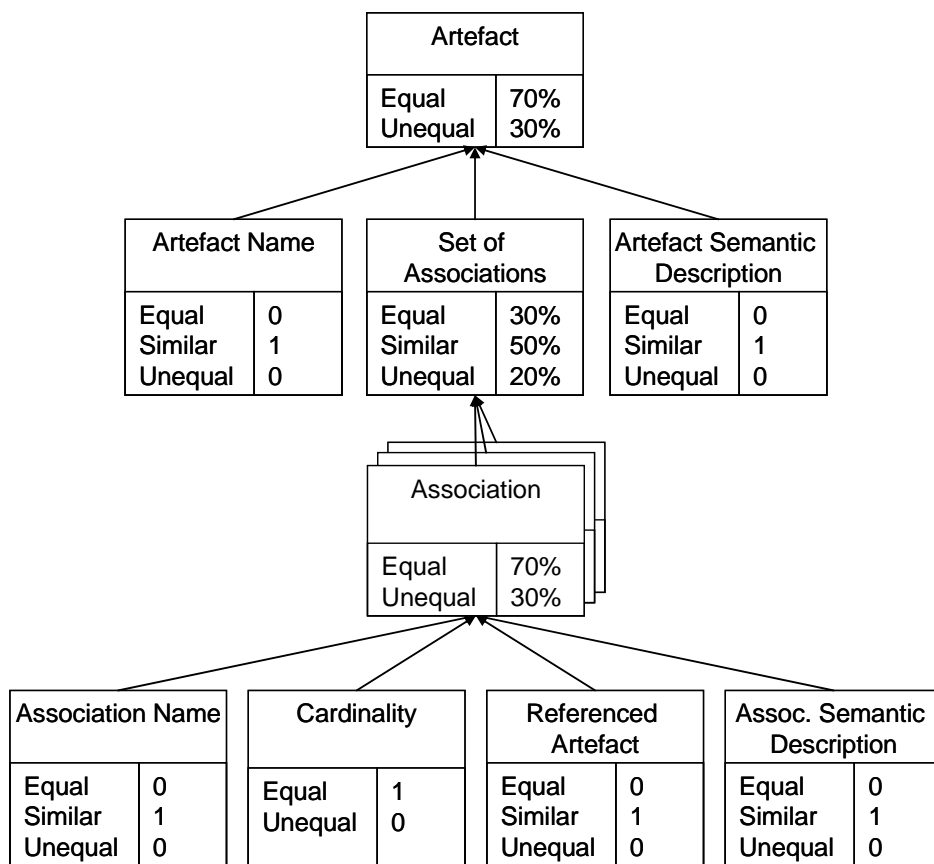


Fig. 5-14: Bayesian network for artefact matching

Depending on the values of the child nodes, the value for the parent node “association match” is computed. Weights for the nodes are given, where equal has weight 1, similar has weight 0.75, and unequal has weight 0. In the beginning, it was mentioned that expert opinions can influence the method. If the value of the “semantic description” node is set on equal, then the result value is overridden by that. That

means an expert has stated that these associations are definitely equal. The value of $\text{Association}_{\text{Unequal}}$ is the complement to one of $\text{Association}_{\text{Equal}}$. The computation is done with the following formula:

$$\text{Association}_{\text{Equal}} = \max \left(\frac{\sum_{i \in \text{ch}(\text{Association})} i_{\text{Equal}} + 0.75i_{\text{Similar}}}{|\text{ch}(\text{Association})|}, \text{Semantic Description}_{\text{Equal}} \right)$$

Only with the association match values computed, the artefact match values can be computed. The “set of associations” node is computed with the values of the of the association nodes of the artefacts that are compared. At first, the artefact with the smaller set of associations is identified. This smaller set of associations is named X_S . The larger set of associations from the other artefact is called X_L .

For each association in X_S the equality value for the best fitting association in X_L has to be found. If the value is at or above 75%, the association of X_S is categorized as equal. If the value is at or above 50% but below 75% the association is categorized as similar, else it is categorized as unequal. After that, the associations are summed up according to their categories. The sums are afterwards normalized to one, so that a valid distribution for the values of the node “set of associations” is created.

$$\text{Set of Associations}_{\text{Equal}} = \frac{|\{i \mid i \in X_S \wedge i_{\text{Equal}} \geq 0.75\}|}{|X_S|}$$

$$\text{Set of Associations}_{\text{Similar}} = \frac{|\{i \mid i \in X_S \wedge i_{\text{Equal}} < 0.75 \wedge i_{\text{Equal}} \geq 0.5\}|}{|X_S|}$$

$$\text{Set of Associations}_{\text{Unequal}} = \frac{|\{i \mid i \in X_S \wedge i_{\text{Equal}} < 0.5\}|}{|X_S|}$$

Now, as the value for the “set of associations” is available, the value for the artefact comparison can be computed. The formula is similar to the formula for the computation of the “Association” node.

$$\text{Artefact}_{\text{Equal}} = \max \left(\frac{\sum_{i \in \{\text{Artefact Name, Semantic Description}\}} i_{\text{Equal}} + 0.75i_{\text{Similar}} + |X_S| \sum_{i \in X_S} i_{\text{Equal}} + 0.75i_{\text{Similar}}}{2 + |X_S|}, \text{Semantic Description}_{\text{Equal}} \right)$$

For the case that an expert is sure to have recognized the same artefacts, the value of the semantic description overrides the other values. According to the value of $\text{Artefact}_{\text{Equal}}$, the two artefacts are regarded as semantically similar. At values above or equal 75% the two artefacts are recommended to be joined.

At this point, the running example is picked up again. The interface from the EA meta model will be matched with the Interface from the SOA meta model. The comparison leads to a result of 78.6% equality. Thusly, the artefacts are regarded as equal enough to be. They will be joined later on, as long as no better matching candidate is found.

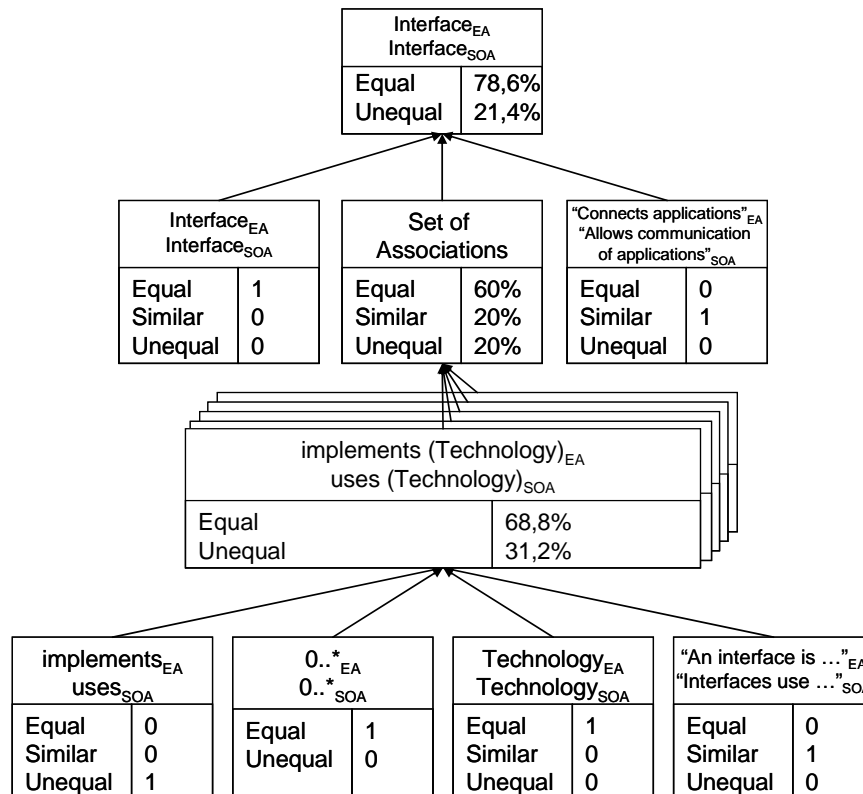


Fig. 5-15: Matching example

For now, it should be clear how the matching operator works. When having a look back on Fig. 5-11, where the merging process is described, the next major step in order is to join artefacts, which will be described in the next section.

5.3.3 Joining Artefacts

This section describes the join operator, which decides how artefacts are joined or inserted into the resulting model. The signature of the join operator reads:

$$\text{artefact}_S \times \text{model}_S \times \text{artefact}_M \times \text{model}_M \rightarrow \text{model}_{S'} \times \text{model}_{M'}$$

The first item in the signature, artefact_S , is any artefact from the slave table that has been compared to the existing artefacts in the master table. model_S represents the slave table, from where artefact_S comes from. The second artefact, artefact_M is the best matching artefact from the master table. If there is no artefact matching to 75% or more, then this parameter is empty. The second input model, model_M , is the master table without the artefact_S . The first output model, $\text{model}_{S'}$ is the slave table without artefact_S . On the contrary, $\text{model}_{M'}$ now contains artefact_S , either completely or in a form joined with artefact_M .

Joining an artefact into a model, two cases may occur. The first case occurs if there is no matching artefact. Then, the artefact is just added to the master table. The resulting model will always be connected as long as there is at least one joined artefact. Due to the minimal EA meta model and its intersection with the SOA meta model, this will always be the case. The second case that may appear when joining an artefact into a model is that there is a nearly equal artefact in both models that can be joined. This requires two steps, first, resolving joining conflicts if a matching is not exactly equal. Secondly, inserting the new artefact including updating the associations of neighbouring artefacts. For bidirectional associations, both ends of the association have to be updated.

For the case that two nearly equal artefacts have been identified, their joining conflicts have to be solved. That means for each conflict type a strategy, on which artefact brings in the dominating property, has to be formulated. For this case, the master table had to be identified. Whenever there is a joining conflict, the property of the master table artefact will preferably be chosen.

The conflict types that may occur are identified in [Borona07]. They have a high similarity with the child nodes from Fig. 5-14. Thusly, the following joining conflicts may appear:

- Name conflict
- Semantic description conflict
- Association name conflict
- Association semantic description conflict
- Cardinality conflict

In general, the master table holds the dominating artefacts. Names and descriptions are usually chosen from the master table. The “Artefact Name” and “Association Name” will be chosen from the master table as it means less learning effort for the employees of the enterprise. The “Artefact Semantic Description” and the “Association semantic description” can be extended by the one of the slave table artefact if there is a point worth adding.

Concerning cardinalities, two strategies make sense. Firstly, to choose a cardinality that is at least as lax as the two joined cardinalities, or secondly, choosing the cardinality from the master table. If there are two different cardinalities given for the association to be joined, then the union of these cardinalities should be given to the new association. For example, the cardinalities 2..5 and 4..* are joined to 2..*.

If the joining conflicts have been resolved, the insertion of the artefact may begin. If it is a joined artefact, the existing entry in the master table is extended by associations of artefacts. If not, a new entry is created and the associations of the artefacts are inserted line per line.

For the case of joined artefacts, the bidirectional associations have to be updated. Otherwise, inconsistencies would occur. Every association has a referenced artefact. If the referenced artefact of an association has been joined, so that its name has changed, then the referenced artefact does not exist anymore with its former name. For this reason, all associations that reference a joined artefact have to be updated. This goes for artefacts in the slave table, too.

The update is realised by replacing the discarded name throughout both tables. This happens if the artefact is referenced by an association. To prevent that two different

artefacts, by chance sharing the same name, are both renamed, the slave table names were marked with a preceding underscore.

The associations between artefacts in the master and the slave table will always be kept during the process. This is wanted, because it ensures the connectivity of the resulting model. All associations between the master and the slave model will be cleared during the whole process, because all the artefacts from the slave model will be either inserted or joined into the master model. With an empty slave model, no such model crossing association can occur. When the slave table is empty, the retransformation of the table back to a graph can be initiated.

After the textual explanation of the joining operation, the running example is picked up again. The joining of the two interface artefacts is illustrated in the example. In Fig. 5-16, the dashed arrows indicate the matching entries that have been found. The joinings are executed by resolving eventual conflicts and inserting the new values into the master table. The entry in the slave table is deleted. The result of the joining can be seen in Fig. 5-17.

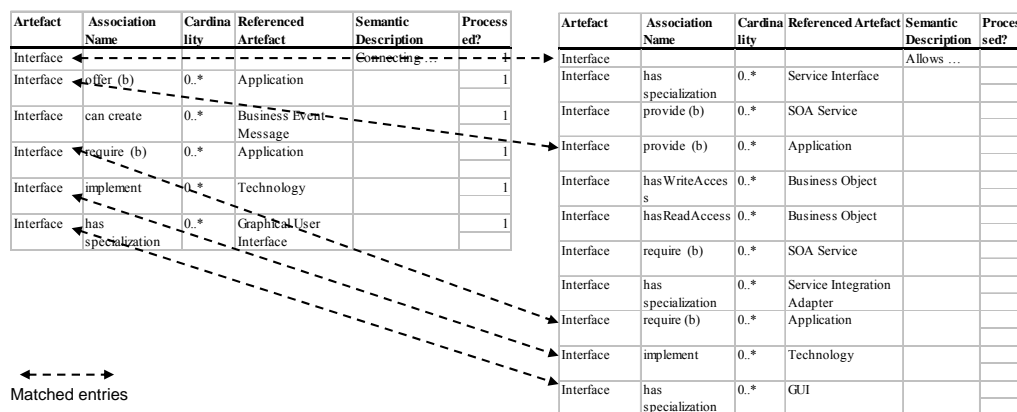


Fig. 5-16: Joining example before joining

Artefact	Association Name	Cardinality	Referenced Artefact	Semantic Description	Processed?
Interface				Allows ...	0
Interface	has specialization	0..*	Service Interface		0
Interface	provide (b)	0..*	SOA Service		0
Interface	offer (b)	0..*	Application		0
Interface	hasStoreAccess	0..*	Business Object		0
Interface	hasRadAccess	0..*	Business Object		0
Interface	require (b)	0..*	SOA Service		0
Interface	has specialization	0..*	Service Integration Adapter		0
Interface	require (b)	0..*	Application		0
Interface	implement	0..*	Technology		0
Interface	can create	0..*	Business Event Message		0
Interface	has specialization	0..*	GUI		0

Fig. 5-17: Joining example after joining

5.3.4 Algorithmic Concerns

So far, the single steps of the merging process have been described. The algorithm presented in this section orchestrates these steps in the way that the result will be a graph representing the merged meta model.

In Fig. 5-18 a detailed diagram for the algorithm is given. The algorithm starts with the transformation of the two graphs into the slave and the master table. This step was explained in sub section 5.3.1. When done so, the comparison of entries may begin. Always the first unmarked entry of the slave table is compared with each entry of the master table. If exactly one matching entry is found (guard [Equal to exactly one]) in the master table, then the two entries are joined (which includes updating the other entries) and the next unmarked slave table entry is compared.

Within the comparison step, a slave table entry is marked when all its comparison computations with master table entries have been done, but no clear matching entry has been found (guard [Else]). As long as this is the case and not all entries are marked, the entry remains in the slave table. The entry could change later because another entry is joined into the master table. If that happens, the processed mark is deleted and the computation is repeated in one of the next comparison iterations.

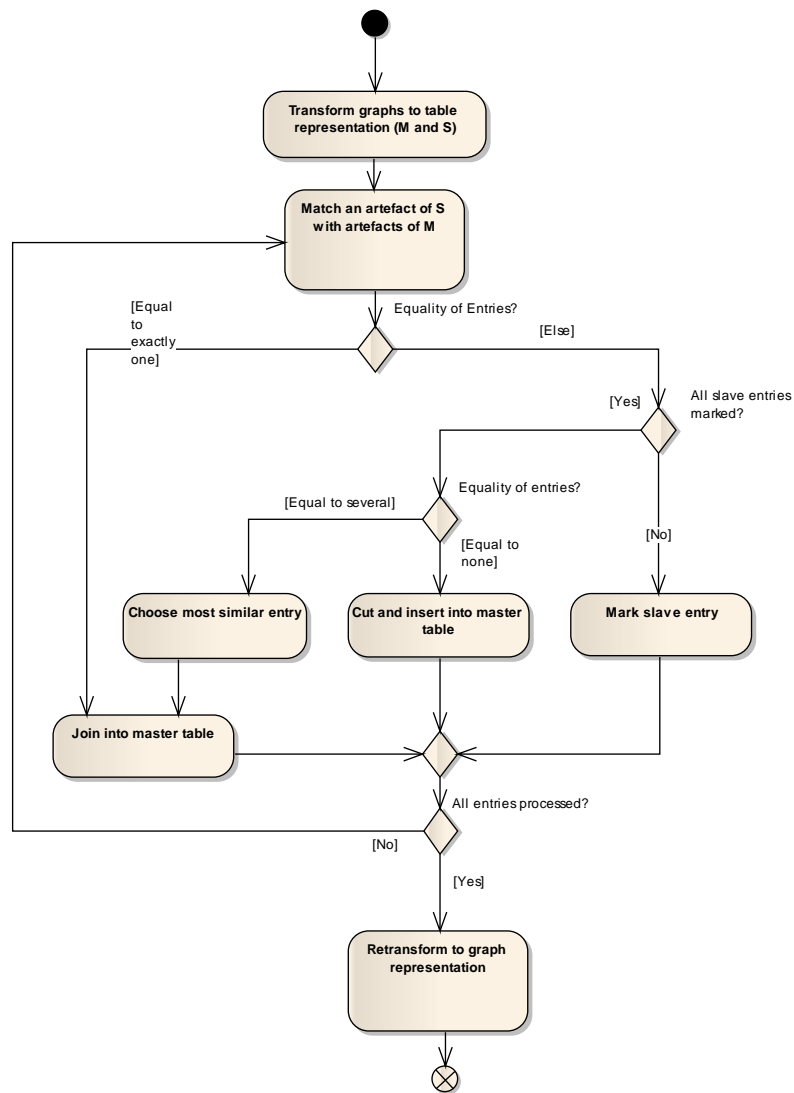


Fig. 5-18: Algorithm for meta model merging

At the point of time where all slave table entries are marked, these entries are forced to be joined or inserted into the master table. If there is a slave table entry that equals (equality value >75%) more than one master table entry (guard [Equal to several]), then the entry with highest computed equality value is chosen. If there is no master table entry with a high equality value for a slave table entry, then the entry is inserted in the master table. Entries that were inserted or joined are taken out of the slave table.

The algorithm leads to an empty slave table, which means that all entries have been processed. If so, the resulting master table is retransformed into the graph representation.

This sub section has shown how the single steps of the previous sub sections are applied in an algorithm that merges two meta models. The next step is to apply it on the given meta models.

5.3.5 Application of the Merging Algorithm

With the meta model merging algorithm given, the SOA and the EA meta model can be merged to an SOEA meta model. Only a brief overview and the results will be shown in this subsection. Some intermediate results of the merging process are given in appendix A.

The decision on master and slave table is taken to the favour of the EA meta model. That means it will be transformed to the master table and the SOA meta model will be transformed in the slave table. This is done because the EA meta model is more specific to the enterprise and employees are more familiar with the old labels.

In Fig. 5-19 the merged SOEA meta model is depicted. The concepts are coloured according to the layers they belong to. In most cases, the joined objects kept the names from the EA meta model. Their names will be better known to the users in an enterprise.

With an SOEA meta model at hand, the enterprise architect has a basis for planning the next steps of the transformation of the enterprise. In this chapter, an SOA meta model and an EA meta model were defined. Moreover, a methodical approach on how to merge them to an SOEA meta model is given. Through the SOA meta model the requirement R2 “SOA formalization” has been fulfilled. Through the EA meta model that may be designed individually by the enterprise architect, the requirements R5

“EA Formalization” and R6 “Individual EA meta model” were satisfied. The result of the meta model merging process, the SOEA meta model, leads to the fulfilment of the requirements R4 “Integrated language for EA and SOA” and R8 “Holistic EA Modelling”. R13 “Methodical approach” has to be covered by all realization chapters. Therefore, it is only partly fulfilled here. At least concerning the SOEA meta model definition, a methodical approach has been realized in this chapter.

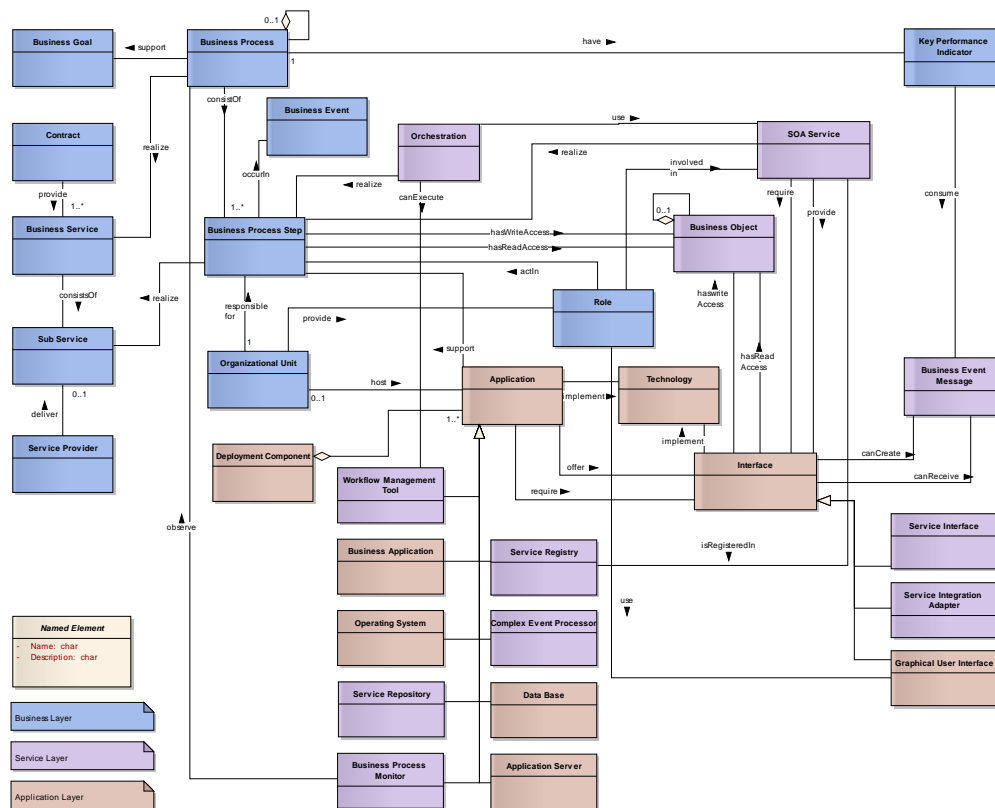


Fig. 5-19: Integrated meta-model of EA and SOA concepts

To be able to realize the management cycle from Fig. 2-17, as well as the requirements R3 “SOA conformance criteria” and R11 “Automation of criteria checks”, not only planning has to be supported. A more crucial point is the measuring that is not covered by the SOEA meta model itself. The next chapter will define everything that is needed to realize the measuring concerning service orientation.

6 Defining Quality Criteria and their Metrics

Quality is the overall goal of planning and design activities and an enterprise architecture planning is not an exception. Thus, mechanisms that allow the evaluation of the design and planning are desirable. Especially quantitative measuring mechanisms should play a role, as they are more precise than qualitative ones (compare [Sommer01]). The quantitative measurement results should then be used to derive qualitative statements.

The derivation of metrics from quality criteria, which are derived from quality properties, will deliver the demanded quantitative measuring mechanisms. The qualitative statements are then determined by indicators (treated in the next chapter). Fig. 6-1 shows the meaning of properties, criteria, metrics, and indicators in the context of the basic solution concept. This chapter focuses only on the definition of quality criteria of an SOA-like EA and their metrics, not their indicators.

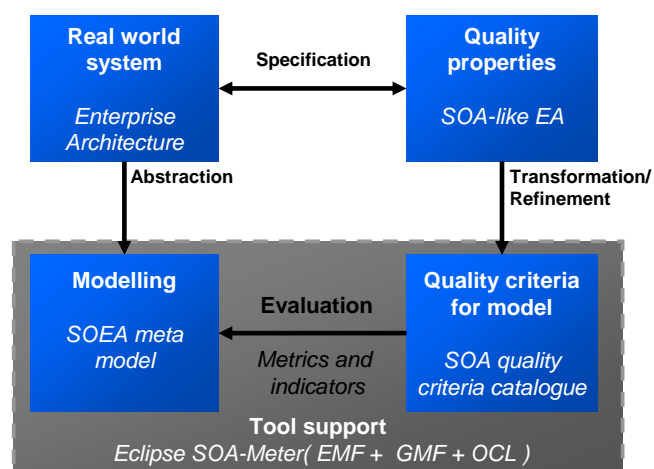


Fig. 6-1: Realization of basic solution concept

The quality concerning an SOA-like EA is regarded as the grade of service orientation of the enterprise architecture. Thus, the here measured quality concerns the conformance of the EA to the SOA definition given in chapter 4. There are also other quality aspects of an enterprise architecture that could be measured. However, this would go beyond the scope of the thesis. However, the evaluation method and tool support could be easily used to implement other quality checks.

The criteria for the evaluation of service orientation are grouped in two categories to cover task fields determined in section 1.2. Firstly, the structural criteria that can mostly be evaluated on the basis of the meta model. The structural criteria focus on the conformance of the enterprise architecture to the SOA reference architecture described in section 4.1.1. The criteria vary from “Is there a service registry?” to “How high is the ratio of monitored processes to unmonitored processes?”.

Secondly, the SOA service quality criteria. They evaluate whether the services within the Service-Oriented Enterprise Architecture are designed well or not. Quality criteria for well-designed services are defined and checked for this purpose.

In Fig. 6-2 the parts of the overview diagram covered by this chapter are shown in the dashed frame. The creation of an instance of the SOEA meta model is task of the architect. It is assumed that the architect is able to describe such an instance. Furthermore, he has to take care that the information in the model is held up to date. This is a general modelling problem. The typical way should be to change the model and then to change the real world system accordingly. However, often the real world system is changed and then the model has to be updated. This is also part of the governance tasks not being covered here.

The major subject of this chapter is the SOA quality criteria catalogue fulfilling the requirement R3 “SOA conformance criteria”. It consists of two parts, the structural quality criteria and the service quality criteria. Subsection 6.1.1 focuses on structural quality, which means the conformance of the EA to the SOA reference architecture. Subsection 6.1.2 focuses on the quality of SOA services. Afterwards in section 6.2, the metrics required to execute quantified measurements for the whole set of quality criteria are elaborated. The metrics support the fulfilment of R7 “As-Is & Target Modelling” and R11 “Automation of criteria checks”.

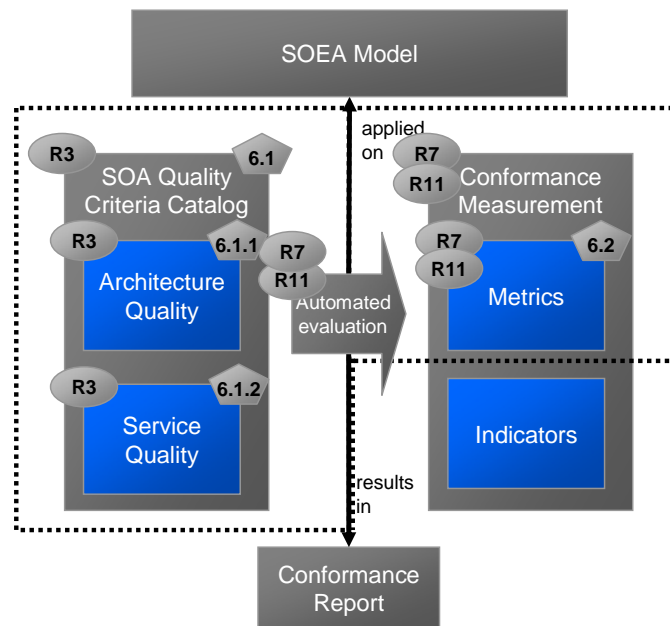


Fig. 6-2: Contribution of chapter 6

Metrics are instructions on how to derive values for measures and their use realizes the measuring step from Fig. 2-17. Their measures stand in relation to the aimed quality criteria. The measure itself does not say whether the quality criterion it concerns has been reached or not. Indicators are required to interpret the measures. The indicator specifies which values of a measure are desirable and which not. The interpretation of the results of the metrics is described in chapter 7 .

The approach for the evaluation of quality criteria is originally based on the Goal Question Metric (GQM) approach described in [Basili92]. The questions in the GQM-approach are synonym with the quality criteria in this approach. For the realization in this thesis, the quality evaluation concept from [VoigtH09] has been adapted. The concepts used are illustrated in Fig. 6-3. A quality property is derived in one or more quality criteria. For every quality criterion there is exactly one indicator. An indicator is a mapping between the values of a metric and the possible values of the indicator. Every indicator has the co-domain $[0,1]$, where zero is the least and 1 is the most desirable value. Furthermore, the indicator can possess a calculation rule, if it uses a compound metric. A metric can be compound of other metrics that are either base metrics or other compound metrics. In contrast to metrics, the indicators are dependent from individual preferences. For this reason, the indicators are elaborated in chapter 7.

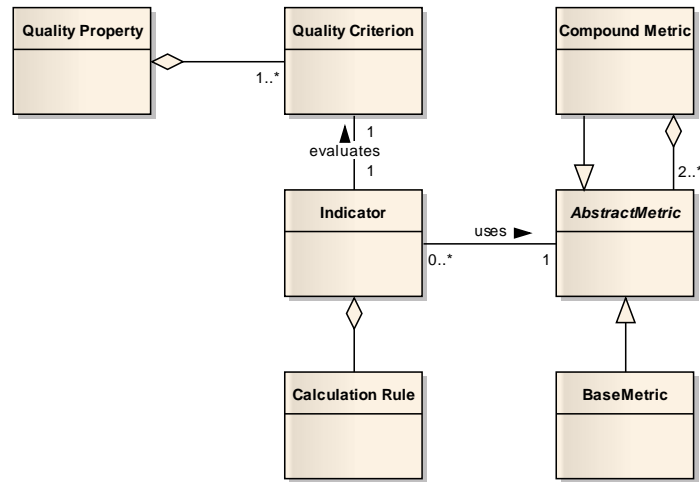


Fig. 6-3: Relation between quality criteria, metrics, and indicators

In the remainder of this chapter, the definition of quality criteria catalogue is given and the corresponding metrics are elaborated.

6.1 The Quality Criteria Catalogue

This section is dedicated to the quality criteria catalogue. The first half of the catalogue consists of the part for the structural quality criteria and is presented in subsection 6.1.1. The second half of the catalogue consists of the part for service quality criteria and is presented in subsection 6.1.2.

6.1.1 Defining Structural Quality Criteria

The structural quality criteria are refined from the SOA definition of this thesis. The main artefact examined for this is the reference architecture depicted again in Fig. 6-4. In addition, the rest of the SOA definition from subsection 4.1.1 is used for the refinement step.

The quality properties according to Fig. 6-1 are the main concepts of service orientation. These have already been pointed out in section 4.1. They are:

- P1 Middleware
- P2 Disclosure of functionality

- P3 Complex event processing
- P4 IT-business alignment
- P5 Orchestration
- P6 Business process monitoring

The structural criteria are categorized concerning these properties. Each criterion is formulated as a question, which is usual in the GQM approach. The quality of a Service-Oriented Enterprise Architecture is tending to be high if the question can be answered positively or a high value within the value range is achieved. That means with all the questions answered in a positive way the examined enterprise architecture is assumed to be conform to the reference architecture.

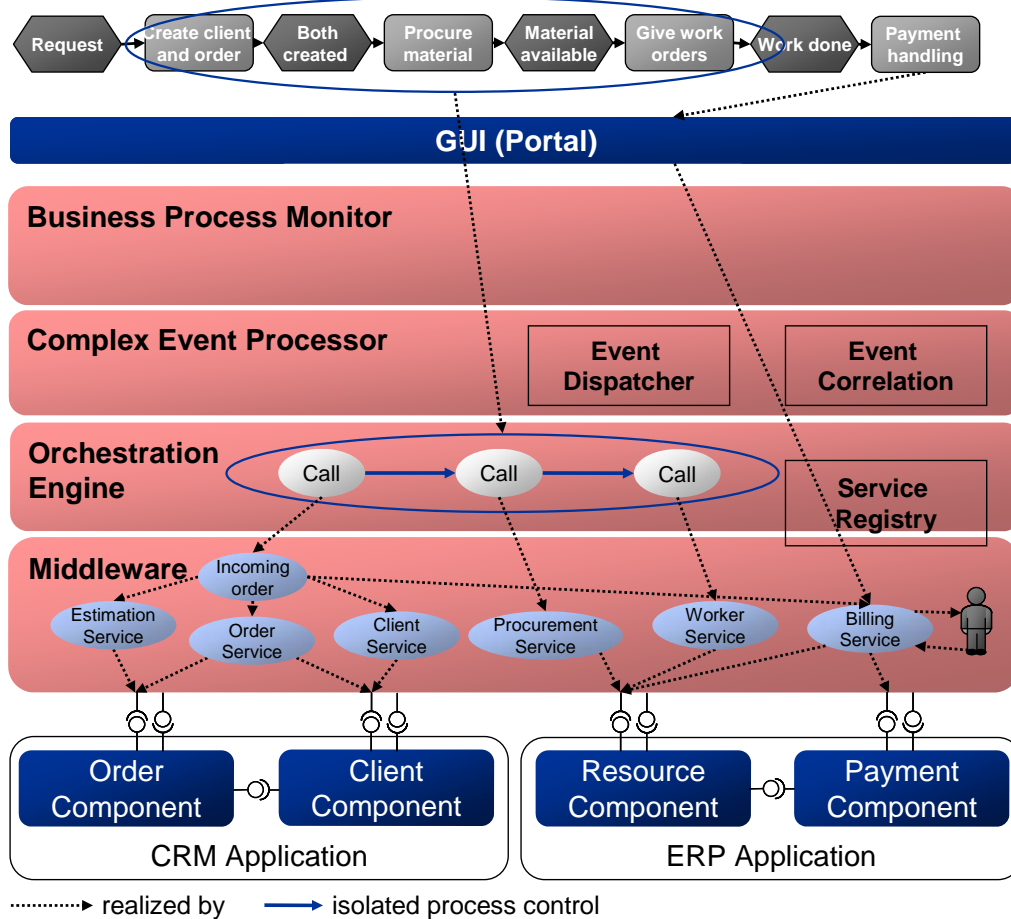


Fig. 6-4: SOA reference architecture

There are 22 questions refining the six quality properties of the reference architecture. The questions are elicited in the remainder of this subsection.

P1 Middleware

Middleware is the first examined area of the reference architecture. There should be a service registry in every service-oriented enterprise architecture. This belongs to the category middleware, because the service registry is essential for the communication initiation. It does not matter whether the registry is implemented in a distributed form or in a centralized form as long as it can be accessed as one logical registry from where all services can be discovered.

Q01 Is there a central service registry?

A service repository usually includes a registry. Hence, it is a suitable alternative for a registry.

Q02 Is there a central service repository?

On the one hand, SOA services shall abstract from technology. For this reason, they have a business interface. On the other hand, they have to be versatile in their field of application. The creation of multi-channel services supports this versatility. Multi-channel services have different technical implementations for the same business interface.

Q03 Are there multi-channel SOA services?

The reuse of SOA services is one of the major aims of an SOA. For this reason, it has to be ensured that this is realized in the architecture.

Q04 Are SOA services reused?

A fully-fledged SOA should cover wide parts of the enterprise architecture. Not only newly developed systems should be built SOA-conform, but also existing legacy applications shall be used within this architecture. The adaptation of legacy systems is the way to integrate them.

Q05 Are legacy systems integrated in the SOA by adapting them?

The common middleware is often hard to achieve within an enterprise architecture that has grown over years. Multi-channel interfaces are suitable when cooperating with other stakeholders, but internally the use of a common middleware is to be achieved.

Q06 Is there a common middleware?

The middleware shall not only be widely spread within the enterprise, but also widely spread among users. By using established standards, the development and integration effort can be reduced in the long term.

Q07 Are exclusively standards used for communication protocols?

P2 Disclosure of functionality

A quality criterion of an SOA as shown in Fig. 6-4 is the disclosure of functionality. Every graphical user interface should use other interfaces that are available for other software components. In every case, the existence of user interfaces offering the only way to invoke a specific software functionality is to prevent.

Q08 Are visualization and functionality separated from each other?

Interfaces should not demand the usage of a certain implementation. At least SOA services should be designed in this way. Applications and SOA services having exchangeable implementations are desirable.

Q09 Are application and SOA service implementations exchangeable?

P3 Complex event processing

The realization of a Service-Oriented Enterprise Architecture demands an event processor to dispatch events and correlate them to complex events.

Q10 Is there an event processor?

Event processing is dependent on event sources. For this reason, SOA services and applications should be event enabled. Event-enabled means that an event is fired for each business relevant operation.

Q11 Are SOA services and applications event-enabled?

The change of a business object state is such a business relevant operation. Therefore, an event should be fired on the change of a business object.

Q12 Are business objects observed with events?

The processing of a process step is another source for a business relevant operation for which an event should be fired.

Q13 Are process steps observed with the help of events?

The flexibility of an enterprise architecture is increased if events can lead to the automated execution of other processes.

Q14 Can events lead to the execution of applications, SOA services, or orchestrations?

P4 IT-business alignment

The concept of SOA services was originally designed to close the gap between the business and the IT. To achieve this SOA services should provide business functions that match well to process steps.

Q15 Do SOA services provide business functions that match well to process steps?

Repetitive human work shall be integrated in SOA services. That means not the employee invokes the next process step, but the orchestration engine. The employee only reports the results of his work to the orchestration engine. If the employee invokes the next process step, the electronically controlled process control flow is interrupted.

Q16 How high is the ratio of human work that is supported by SOA services?

The processes that are executed in a company should be realized by SOA services, so that the gap between processes and IT is narrowed.

Q17 Are process steps realized through SOA services?

P5 Orchestration

An orchestration engine is responsible for the automated execution of SOA service orchestrations. Out of the architectural view, it does not make a difference if there are several instances of an orchestration engine, as long as they use the same orchestration language.

Q18 Is there an orchestration engine that executes SOA service orchestrations?

The automation of business processes is one purpose of service orchestration. For the realization, the business processes have to be modelled and executed in an orchestration language.

Q19 How high is the ratio of processes that are modelled and executed in an interpretable orchestration language?

The second purpose of service orchestration is to render the process control flow explicit. This will greatly improve flexibility as business process implemented with orchestrations can be changed with less effort.

Q20 How much process control flow is hidden in applications?

P6 Business process monitoring

The monitoring of processes has to be supported by a dedicated application. Otherwise, the information retrieval and transformation of all the process relevant data would hardly be manageable.

Q21 Is there a BPM application?

Furthermore, the existing business processes should be monitored with this application by making use of automatically generated business events.

Q22 How high is the ratio of processes that are monitored automatically?

The quality criteria for SOA services have not been defined yet, but this will be done in the following subsection.

6.1.2 Defining SOA Service Quality Criteria

In this subsection, the quality properties and the corresponding criteria of SOA services are named and explained. The quality properties differ from the structural properties because they only concern the quality of SOA services and not the environment they are used in. It is strongly recommended to have the SOA service definition (section 4.1.2) in mind, when reading this section.

The quality properties were gathered from several references. These are [Krafzi06], [OASIS05], [ErlTho06], [Dostal05], [Engels08], [Josutt08], and [UecanE08]. The gathered collection has had a plethora of properties with substantial overlaps. The overlaps were filtered out and the following set of properties was refined. Just as in the subsection before, questions concerning the quality criteria are formulated.

- P7 Reusability
- P8 Granularity
- P9 Technology Independence
- P10 Orchestration
- P11 Statelessness
- P12 Loose Coupling
- P13 Functional compactness
- P14 Compensation
- P15 Service Contract
- P16 Discoverability

P7 Reusability

Generally, this is nothing new in software development. It aims at the usage of code in different applications instead of writing similar or even the same code in every application. Common libraries with mathematical functions are an example for this principle. When designing an SOA service, the architect will have a certain scenario

with specific users for it in mind. However, simple usage by other users shall be supported.

Reusability brings a new facet into the service-oriented world. Compared with reuse in object-oriented programming there is a difference. An object has a blueprint and with this information, it can be instantiated. When reusing an object the blueprint (the code) is used in another context and other instances of the object are created on another platform. An SOA service also has a blueprint (the contract) that leaves out implementation details and execution platform. Therefore, an SOA service is shared for all users in the same way.

Q23 Are the SOA services reusable?

The reusability question will be answered with the help of other quality properties. These are P8 Granularity, P9 Technology Independence, P11 Statelessness, P12 Loose Coupling, P15 Service Contract and P16 Discoverability.

P8 Granularity

Granularity relates to the functional extent services are offering. If services are too fine-grained, usability decreases because it is hard to find the right one among all the others and it is laborious to orchestrate them. If they are too coarse-grained, they will not fit to the demands of the user, thus reducing reusability. Because of this, an adequate number of SOA services and service operations per service should be found.

Q24 How many service operations do SOA services have?

Too many operations within an SOA service decrease the usability. A SOA service with too many operations is similarly negative as a god class for object orientation.

Q25 How many business objects are covered by an SOA service?

Business objects are closely related to services. The smaller the number of write and read accessed objects per service, the better the IT-business alignment.

P9 Technology Independence

Technology independence aims at the transparent implementation of SOA services. It is not of interest how an SOA service does something but what it does. For example, if there is an SOA service storing an order the service requester does (and should) not know whether there is a MySQL database or an employee with a sheet of paper storing the information. If the user does not know where the information is stored, then how should he know how to retrieve later on? The answer comes with the appropriate service that retrieves the order data (for a certain order number). Again, it is not of interest whether the service searches a database or causes an employee to look it up on his sheet. Technology independence decouples function from technology.

Q26 Do SOA services enforce the use of concrete technologies?

A SOA service should be described transparent fro technology. For this reason, it should not enforce the use of any concrete technology.

P10 Orchestration

Orchestration means to combine several SOA service calls, which leads to the creation of a new high level SOA service. SOA services should be designed for using each other. Otherwise, the hard-wired orchestration (as described in section 4.1) will be hard to realize.

Q27 Do SOA services provide interfaces being adequate for orchestration tools?

SOA services must provide interfaces that can be used by orchestration tools.

P11 Statelessness

Statelessness in general means that services do not have an internal state affecting the execution of operations. In other words, this means that the repeated execution of an operation does not lead to a different result than the single execution. This is hard to achieve in every case. Getting closer to this ideal state is of advantage for the predictability and therefore the quality and test effort of the system.

Q28 Are SOA service operations idempotent?

Calling the same operation with the same input parameter twice should result in the same state. This is called idempotency.

P12 Loose Coupling

Loosely coupled SOA services are only weakly dependent from each other. Their implementation may change without affecting other SOA services. If a service A premises the availability of service B then A is strongly coupled to B. If B additionally premises A, they are strongly coupled to each other (compare [Sieder07]). Loose coupling increases the maintainability of software systems and is a necessity for binding of services at runtime that provides a new degree of freedom.

Q29 Do SOA services communicate with business event messages?

The communication via business event messages is regarded as a form of very loose coupling.

P13 Functional compactness

Functional compactness aims at the use of a preferably little number of services in a process. It is regarded as positive if several process steps are covered by the operations of a single service. This generally eases the work with the services, because the user has to read less documentation and a better working experience.

Functional compactness also means that an SOA service has only a small amount of dependencies to other services, as this allows covering a process or task domain with a small number of SOA services. Therefore, SOA services having a low adherence (dependencies to other services) and a high coherence (dependencies among own service operations) have a high usability.

Q30 Do SOA services have a compact process context?

SOA services should hold as much as possible of the functionality required in a certain process context to increase the IT-business alignment.

P14 Compensation

Compensation is the counterpart of transaction rollback in the context of service orientation. The transaction concept is not embedded in the concept of service orientation. However, transactions can be parts of SOA service operations. The execution of service orchestrations can be lasting over a long time, so that a transaction locking all used resources is not practicable. Instead, the relevant service operations have to be compensated by other service operations. This means a sequence of service operations will lead the system to a state that is (at least nearly) equal to the state it had before the execution of the first operation. Of course, this is not always possible, but at least desirable.

Q31 Are the SOA services compensable?

The service operations of an SOA service should be compensable by the operations of the same service at least with operations from other services.

P15 Service Contract

An SOA service contract holds all the information that could be relevant to the requester. On the one hand, the contract must contain sufficient information allowing the requester to decide whether it fits to its needs. On the other hand, the contract may not include any implementation details. Otherwise, the contract must be changed when the implementation is changed. The form of these descriptions is not fixed at all and does not have to be formal, though machine-readable descriptions like IDL or WSDL for interfaces may have great benefits. According to the definition from Fig. 6-5, the contract has to provide information about:

- Description
 - Interface
 - Functionality
 - Usage
 - Lifecycle Status
 - Responsibility
- Objection Level Agreement (OLA)
 - Constraints
 - Availability
 - Accessibility

- Visibility
- Security
- Monitoring and Reporting
 - Business event messages
 - Predefined reports

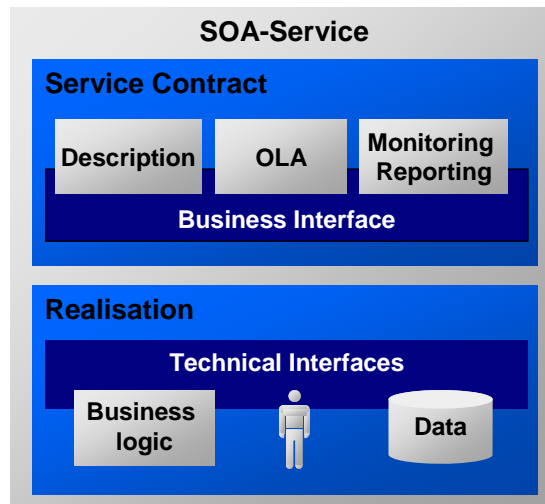


Fig. 6-5: Structure of an SOA service

Q32 Are the contracts of SOA services complete?

A complete contract is inevitable for the adequate use of SOA services.

P16 Discoverability

Discoverability is strongly dependent on adequate service contracts. A requester that has a certain task to be done must be able to find the service that fits to his needs. Otherwise, a reusability benefit can hardly be drawn. There are several possibilities to ensure this characteristic and generally, the service registry is responsible for this task. However, a service registry does not have to be a high performing software system. In the beginning, a simple text document or Wiki can be sufficient.

Q33 Can SOA service be discovered easily within the enterprise?

SOA services should be registered in a service registry. Furthermore, the registry should offer sophisticated search mechanisms.

This concludes the definition of service quality criteria. In the following section, metrics are defined on the basis of the questions to be able to measure the realization of the quality criteria catalogue.

6.2 Metrics for Quality Criteria

In this section, the metrics delivering quantifiable results for the quality criteria are described. These metrics should be as formal as possible and their information should be retrievable from the SOEA model if possible. As it is not possible to answer all questions completely with formal metrics, there is a distinction between formal metrics and subjective metrics. These have to be determined by one or more experts. However, they are tried to be avoided as the requirement R11 “Automation of criteria checks” will suffer from these.

Other approaches using metrics for quality evaluation already exist. Some of them have been examined for potential reuse in this thesis. However, metrics for quality are rare. Many related approaches exist in the field of software development. A metrics suite for object-oriented design has been defined in [Chidam94]. There are six metrics measuring properties of a class diagram. One example for a metric is CBO. It counts the number of classes associated to a given class. The kind of metrics given in [Chidam94] is completely independent of the model context. Unfortunately, this makes the metrics useless for the evaluation of service-oriented enterprise architectures. Not the number of relations is of interest for the evaluation of the SOEA model, but it is of interest if the right kinds of objects are related to some object. The metrics presented in [Santan03] are an extension of the approach in [Chidam94]. The old metrics are adapted for aspect-oriented system designs. However, the approach remains completely context independent, which renders it inadequate for evaluating SOEAs.

A more interesting suite of metrics is proposed in [Vascon07]. The approach presents a simple meta model for the information system architecture (ISA, compare [Kremer05] and section 2.1.1). The ISA meta model is depicted in Fig. 6-6. The measuring points of the metrics are defined on the basis of the meta model. The major goal of the metrics is to assist the architect previewing the impact of his ISA design choices on the non-functional qualities of the enterprise information system.

This is already very similar to the approach needed for the evaluation of an SOEA. As first reason, the metrics are designed for evaluating quality criteria of enterprise

architecture (called information system architecture in [Krcmar05]). That means a similar context is given. Secondly, the basis for the measuring points is built by a meta model.

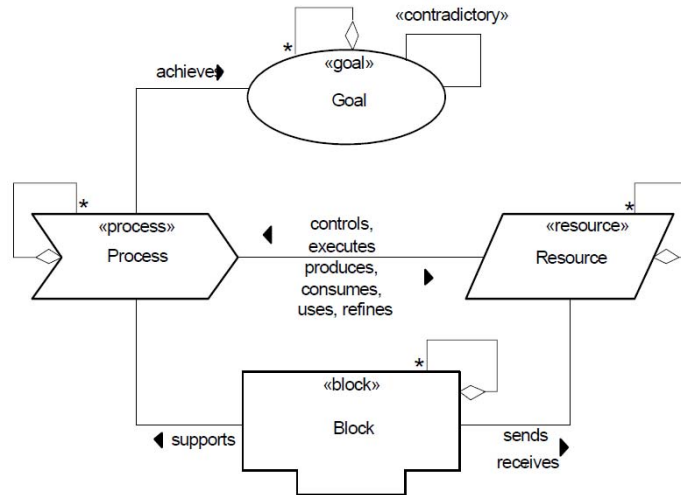


Fig. 6-6: ISA meta model as in [Vascon07]

However, the SOEA meta model is quite different from the ISA meta model. Furthermore, the quality criteria are different, too. In [Vascon07], the quality criteria are not further specified than on the level of usability, maintainability, reliability and so on. For this reason, the intersection of the metrics from [Vascon07] with metrics usable in this thesis is very small. Only two of the metrics could be adapted to service quality criteria. Adaptation means, that the measuring points of the ISA meta model are changed to measuring points in the SOEA meta model. In the following, the adapted metrics are named and the relation to the respective quality criteria from the previous section is discussed.

ISA metric name	LCOIS - Average Lack of Cohesion in IS Blocks.
ISA metric description	This metric measures the correlation between application blocks and the information entities used in that application block.
Corresponding quality criteria	P8 Granularity
Changes for adaptation	This metric is adapted as the metric M46 Stored Business Objects. Information entities are regarded as business objects and application blocks are regarded as service interfaces.

Fig. 6-7: Adaptation of LCOIS metric

ISA metric name	NOIS - Average Number of Operations in IS Blocks.
ISA metric description	The Average Number of Operations in IS Blocks is computed counting the number of operations on each IS Block divided by the number of IS Blocks
Corresponding quality criteria	P8 Granularity
Changes for adaptation	This metric is adapted as the metric M48 Operation quantity. Information entities are regarded as business objects and application blocks are regarded as service interfaces.

Fig. 6-8: Adaptation of NOIS metric

In the remainder of the section, the metrics for the quality criteria are elaborated. The adaptations mentioned above will already be included.

6.2.1 Metrics for Structural Quality Criteria

In Fig. 6-9 the template for a metric description is given. It is based on the metric definition of the ISO standard [ISO/IEC01]. It contains a name of the metric, a short description and shows whether it is a compound or base metric. Furthermore, there are three types of metrics. A metric can be automatable, objective, subjective, or have a mix of these properties. An automatable metric is always objective. Additionally, it can be computed with the help of the SOEA model. On an objective metric can be clearly decided with the help of facts. For example, the fact if a registry has a key word search or not, can be decided objectively. A subjective metric has to be decided by an expert,

because there are no clear facts or the information gathering would be too complicated. In the latter case, the expert estimates the value. Compound metrics can have a mix of these metric properties.

The co-domain of a metric informs about the range of results that the metric may have. The interpretation gives a rough indication about what is a “good” or “bad” result within the co-domain. If the co-domains of two base metrics that are used in a compound metric cannot be merged, then the co-domain is defined as their Cartesian product. The operator for the Cartesian product is “x”. A compound metric consists of several base metrics. Their measures are stored as a tuple, but not calculated with each other. Only indicators apply calculations on measure tuples.

The co-domains of the metrics are often the sets of natural or real numbers denoted by N and R respectively. N includes the zero element. The notation N_6 stands for the natural numbers from zero to six. R_1 is used for relative amounts, which means it embraces the interval $[0,1]$.

Acronym, Name	<i>{Base / Compound}</i>	<i>{Automatable/ Objective/ Subjective/ Mixed}</i>
Short Description		
Calculation Rule or Measuring Point		
Co-domain	Interpretation	

Fig. 6-9: Template for metric description

In the rest of this subsection, the metrics for the quality criteria are named and described. As already mentioned, the indicators for the metrics are given later in the results chapter.

Q01 Is there a central service registry?

A service registry holds information on SOA services. It acts as the mediator between service requestor and provider. Without a service registry, the reuse of SOA services is improbable, because it will be hard to find the suiting SOA services. The knowledge should be stored centrally and not distributed over several registries (from the logical point of view). The realization of the service registry may be distributed over several systems. However, the main property of a logical registry is that every SOA service has to be discoverable by using any of the physical service registries.

The metric suggested for this question is a compound metric that measures the existence and the main features of the registry.

M1 Registry features	<i>Base</i>	<i>Objective</i>
Evaluates the feature richness of the service registry		
<p>The search and automation features are examined for the measurement. There should (also) be machine-readable descriptions like WSDL files for services. Furthermore, the search mechanism is distinguished into key word search, meta information search and search mechanisms that support behavioural descriptions like visual contracts, sequence diagrams or similar.</p> <p>The registry feature measure has the result 0 but its result is increased by one for machine-readable descriptions. Additionally, either zero, one or two points are added, for the case that key word search and/or meta information search or behavioural description based search are implemented. If there are several logical registries, the rounded average value is taken into account. A maximum of three points can be reached.</p>		
N ₃	The higher the number, the better the registry will support the visibility of the SOA services.	

M2 Registry existence	<i>Base</i>	<i>Automatable</i>
Counts the number of existing logical service registries		
The instances of the class “service registry” are counted		
N	No registry is worst, one is best because it is central then. Beyond this, the more logical registries the worse.	

M3 Service registry	<i>Compound</i>	<i>Mixed</i>
Combines M1 and M2		
M1 x M2		
N ₃ x N	The better the singular results the better the result of the compound metric.	

Q02 Is there a central service repository?

The service repository usually allows the user to define its own meta model of a domain. For example, a domain with services, service orchestrations, processes, users, solutions, and the according relations between these object types. It is used by many stakeholders that get in contact with any of the object types and need to be informed about status changes of the objects. A repository can also support the lifecycles of these objects, e.g. the development phases of a service that is designed by architects, implemented by developers, and tested by testers. The repository usually includes the functionality of a service registry. Developing SOA services, a repository is helpful.

M4 Repository features	<i>Base</i>	<i>Objective</i>
Evaluates the feature richness of the service repository		
For the measurement, features are examined. Important features of a repository are:		
<ul style="list-style-type: none"> ▪ Free definable object meta model and taxonomy support ▪ Role model support for users/stakeholders ▪ Versioning of objects and attached documents ▪ Lifecycle support of arbitrary object types 		
Each supported feature increases the result by one. If several repositories exist, the rounded average value is taken into account.		
N ₃	The higher the number, the better the repository.	

M5 Repository existence	<i>Base</i>	<i>Automatable</i>
Counts the number of existing service repositories		
The instances of “service repository” are counted		
N	No repository is worst, one is best because it is central then. Beyond this, the more repositories the worse.	


M6 Service repository	<i>Compound</i>	<i>Mixed</i>
Combines M4 and M5		
M4 and M5		
N ₃ x N	The better the singular results the better the result of the compound metric	

Q03 Are there multi-channel SOA services?

Multi-channel services offer different technical interfaces for the same functionality. The advantage is that different service users in different situations (firewall, technology preference, etc.) may use the service. A multi-channel service is recognized by the technologies its interfaces are implemented in.

M7 Multi-channel services	<i>Base</i>	<i>Automatable</i>
Observes the different interface technologies a service interface is implemented in.		
Each SOA service is examined for its Service interfaces and the technologies they are implemented in. Within the meta model, the items shown below are used as measuring points. The associations each SOA service holds to a technology over the given path are counted and the average is built over these numbers.		
<pre> graph LR Tech[Technology] SI[Service Interface] SS[SOA Service] SI -- implement --> Tech SS -- provide --> SI </pre>		
<i>Fig. 6-10: Measuring points for multi-channel services</i>		
R	Up to a certain point, the higher the number the better. Bigger numbers of interfaces become worse again.	

Q04 Are SOA services reused?

M8 SOA service reuse	<i>Base</i>	<i>Automatable</i>
Observes reuse of SOA services by examining the number of process steps they are used in.		
Within the meta model, SOA services and process steps are directly connected as depicted. The number of process steps per SOA service is counted and an average is computed.		
 <pre> graph LR SOA[SOA Service] -- realize --> BPS[Business Process Step] </pre>		
<i>Fig. 6-11: Measuring points for SOA service reuse</i>		
R	The higher the number the higher the reuse and therefore the better.	

Q05 Are legacy systems integrated in the SOA by adapting them?

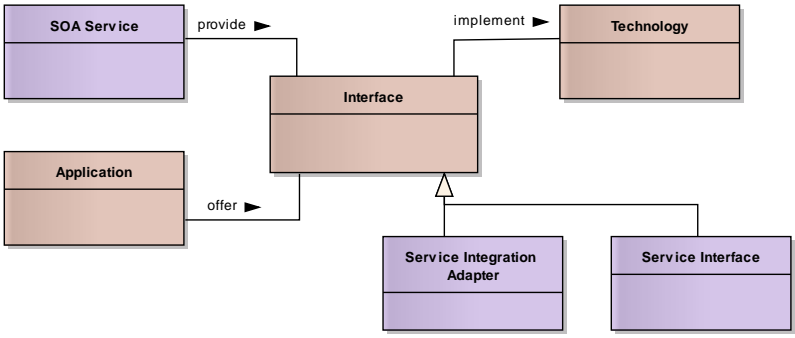
Legacy systems shall be integrated in a service-oriented landscape. For this purpose, they have to be used by SOA services. If there is no adequate interface provided by a legacy application, then a service integration adapter is written. It just offers the functionality in any suitable form to make it available for the SOA service and is therefore not comparable to a service interface.

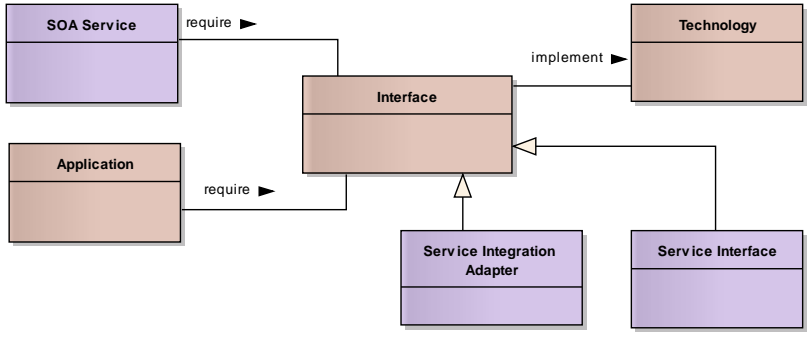
M9 Legacy adaptation	<i>Base</i>	<i>Automatable</i>
SOA services are examined for the number of interfaces they use and that are provided by applications.		
As shown in the figure below, the relations from SOA services to applications via interfaces are followed to determine the application used. As result, the ratio between applications used by SOA services to the number of all applications is computed.		
<pre> classDiagram class SOA_Service[SOA Service] class Interface class Service_Integration_Adapter[Service Integration Adapter] class Application SOA_Service --> Interface : require Service_Integration_Adapter -- > Interface Application -- > Interface : offer </pre>		
<i>Fig. 6-12: Measuring points for legacy adaptation</i>		
R_1	The higher the number the higher the adaptation ratio and therefore the better.	

Q06 Is there a common middleware?

A common middleware shall be implemented by all interfaces (except the graphical ones) so that integration is eased on the technological level. On the one hand, the offering of such a technology is important, on the other its usage degree also states something about the quality of the middleware.

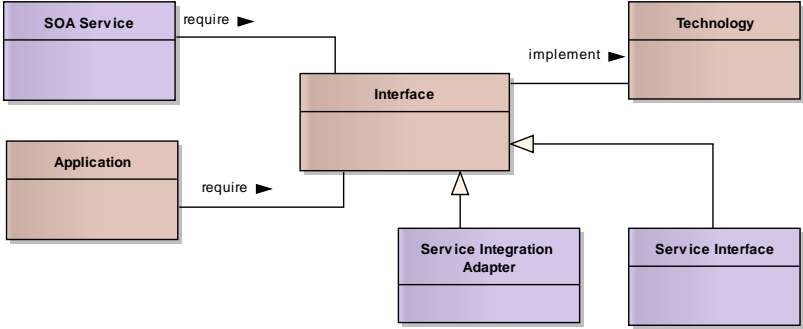
M10 Common middleware	<i>Compound</i>	<i>Automatable</i>
The enterprise architecture is examined for a middleware, or in other words interface technology that is widely used and widely spread. Combines M11 and M12.		
M11 x M12		
$R_1 \times R_1$	The higher the numbers the higher the middleware prevalence and therefore the better.	

M11 Middleware saturation	<i>Base</i>	<i>Automatable</i>
<p>The enterprise architecture is examined for an interface technology that is widely spread.</p>		
<p>There are applications and SOA services that share the middleware interface technology via a service integration adapter or a service interface. The ratio to service integration adapters and service interfaces that do not share the middleware technology is computed. The middleware technology has to be determined by the architect.</p>		
 <p>The diagram illustrates the relationships between various components in a service-oriented architecture. It features six main components: SOA Service, Application, Interface, Technology, Service Integration Adapter, and Service Interface. SOA Service and Application are shown as boxes with a top section and a bottom section. Interface is a central box with a top section and a bottom section. Technology is a box with a top section and a bottom section. Service Integration Adapter and Service Interface are boxes with a top section and a bottom section. SOA Service has a directed association labeled 'provide' pointing to Interface. Application has a directed association labeled 'offer' pointing to Interface. Interface has a directed association labeled 'implement' pointing to Technology. Both Service Integration Adapter and Service Interface have inheritance relationships with Interface, indicated by solid lines with hollow triangle heads pointing to the Interface box.</p>		
<p><i>Fig. 6-13: Measuring points for middleware saturation</i></p>		
<p>R_1</p>	<p>The higher the number the higher the saturation ratio and therefore the better.</p>	

M12 Middleware usage	<i>Base</i>	<i>Automatable</i>
<p>The enterprise architecture is examined for an interface technology that is widely used.</p>		
<p>For applications or SOA services that share the same interface technology via a service integration adapter or a service interface, the ratio of usage (required for applications respectively) is computed.</p>		
 <pre> classDiagram class SOA_Service[SOA Service] class Application class Interface class Technology class Service_Integration_Adapter[Service Integration Adapter] class Service_Interface[Service Interface] SOA_Service --> Interface : require Application --> Interface : require Interface --> Technology : implement Interface < -- Service_Integration_Adapter Interface < -- Service_Interface </pre>		
<p><i>Fig. 6-14: Measuring points for middleware usage</i></p>		
<p>R₁</p>	<p>The higher the number the higher the usage ratio and therefore the better.</p>	

Q07 Are exclusively standards used for communication protocols?

There should be a limited set of technologies used for communication. The ideal solution would be that there is only the middleware technology that is used, but this is probably utopian. Therefore, also the number of the different technologies used is examined. According to how many different technologies are agreed as supportable standard in the enterprise, the value can be used to find out whether more are used or not.

M13 Standard communication	<i>Base</i>	<i>Automatable</i>
<p>The enterprise architecture is examined for the number of interface technologies that are currently in use.</p>		
<p>For applications and SOA services that require a service integration adapter or a service interface, the number of implemented technologies is counted.</p>		
 <p style="text-align: center;"><i>Fig. 6-15: Measuring points for standard communication</i></p>		
R	<p>The higher the number of technologies the worse.</p>	

Q08 Are visualization and functionality separated from each other?

The graphical user interface of an application should not be the only possibility to invoke functionalities. There should always be interfaces that are also machine-processable in order to orchestrate the functionality in a service-oriented way. Completely determining the functionality of an interface would require behaviour-based descriptions of interfaces. These are rather rare and often not comparable. Therefore, the metric is a subjective metric.

M14 Functionality separation	<i>Base</i>	<i>Subjective</i>
<p>The ratio of functionality that is provided by graphical user interfaces only, is measured with this metric.</p>		
<p>An expert knowing the applications and their interfaces has to estimate this value.</p>		
R ₁	<p>The lower the ratio the better.</p>	

Q09 Are application and SOA service implementations exchangeable?

An exchangeable implementation allows replacing the implementation of functionality without affecting its users – as long as the functionality stays the same. Especially SOA services are designed for this characteristic. But, also applications may interface facades that allow unnoticeable changes. Whether applications of interfaces have this possibility, has to be estimated by an expert. However, SOA services, when build correctly, have this characteristic from scratch.

The measured values can only give hints on the characteristic. As the functional extent of an applications and SOA services cannot be measured here, instead the number of instances is taken into account.

M15 Implementation exchangeability	<i>Compound</i>	<i>Mixed</i>
This metric combines M16, M17 and M18 and measures the exchangeability of implementations.		
$(M16*1 + M17*M18) / (M16*M17)$		
R ₁	The higher the ratio the better.	

M16 SOA services	<i>Base</i>	<i>Automatable</i>
Simply counts the number of SOA services.		
Within the SOEA model, the number of instances of SOA services is counted.		
N	No interpretation suggested.	

M17 Applications	<i>Base</i>	<i>Automatable</i>
Simply counts the number of applications (without operating systems and databases).		
Within the SOEA model, the number of instances of applications except of operating systems and databases is counted.		
N	No interpretation suggested.	

M18 Application exchangeability	<i>Base</i>	<i>Subjective</i>
Determines the ratio of implementation of functionality of applications that is easily exchangeable.		
An expert estimates the value.		
R ₁	The higher the better.	

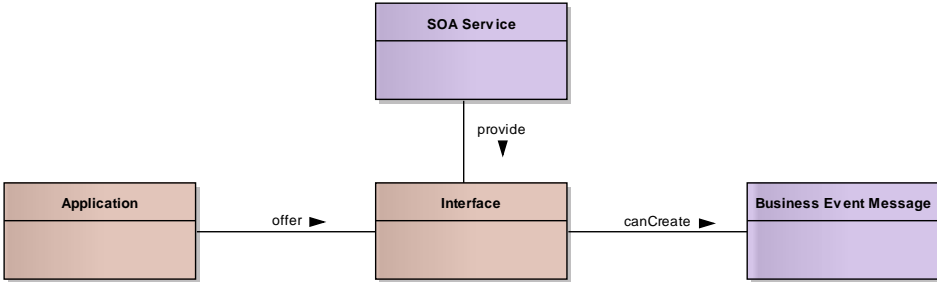
Q10 Is there an event processor?

M19 Event processor	<i>Base</i>	<i>Objective</i>
Measures the existence and feature richness of event processors. The features complex event processing and event distribution are examined. Complex event processing concerns the possibility to correlate events over time. Event distribution allows to distribute events on the manner of the publish-subscribe approach. There may be several instances of the event processor.		
If there are no event processors the result is zero else at least one. For each feature that is owned by at least 50% of the event processors an additional point is granted.		
N ₃	The higher the number, the better event processors will be able to support the complex event processing.	

Q11 Are SOA services and applications event-enabled?

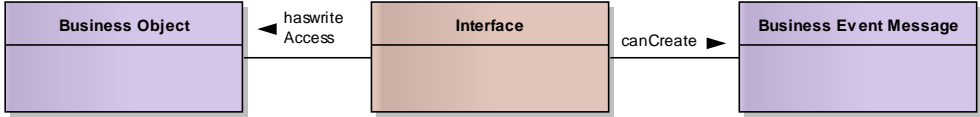
If the interfaces that are provided by an application or SOA service create business event messages, they are regarded as event-enabled. There is no specific value for an interface how many business event messages it should be able to create, because this depends on the implemented functionality. The minimum number of events should be three for start, stop, and abort messages of one piece of functionality.

The target value is somewhere near one but not exactly one, as not every kind of interface has to send event-messages in order to be able to monitor the actions in an IT landscape.

M20 Event enablement	<i>Base</i>	<i>Automatable</i>
Measures the ratio of event-enabled services and applications		
The ratio of interfaces provided by applications and SOA services probable of creating at least three event messages is measured within the SOEA model.		
 <pre> graph TD Application[Application] -- offer --> Interface[Interface] SOAService[SOA Service] -- provide --> Interface Interface -- canCreate --> BusinessEventMessage[Business Event Message] </pre> <p style="text-align: center;"><i>Fig. 6-16: Measuring points for event enablement</i></p>		
R ₁	The higher the ratio, the better the business layer will be supported by CEP.	

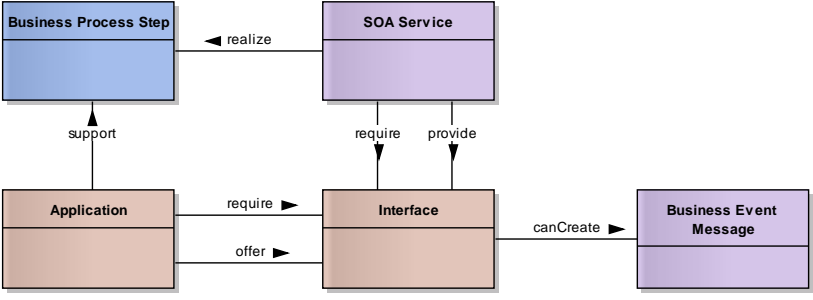
Q12 Are business objects observed with events?

A Service-Oriented Architecture concerns business objects in several ways. SOA services are often designed with the help of business objects. The number of business objects that are observable with events is also a hint on the service enabling of an enterprise architecture.

M21 Business object events	<i>Base</i>	<i>Automatable</i>
Measures the ratio of business objects that are treated with event-enabled interfaces.		
The ratio of business objects that are treated with event-enabled interfaces is taken from the SOEA model.		
 <pre> classDiagram class BusinessObject class Interface class BusinessEventMessage BusinessObject --> Interface : haswrite Access Interface --> BusinessEventMessage : canCreate </pre>		
<i>Fig. 6-17: Measuring points for business object events</i>		
R ₁	The higher the ratio, the better the business layer will be supported by CEP.	

Q13 Are process steps observed with the help of events?

Business event messages should not only be potentially created by applications and SOA service but also be used in process steps. Therefore, the ratio of process steps that are realized with event enabled SOA services and applications are of interest.

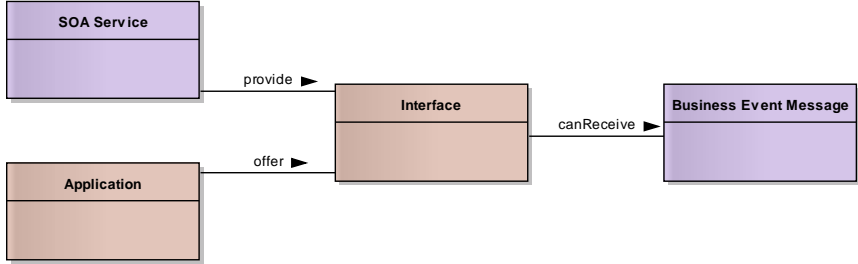
M22 Process events	<i>Base</i>	<i>Objective</i>
Measures the ratio of process steps that are treated with event-enabled applications or SOA services.		
The ratio of process steps that are treated with event-enabled applications or SOA services is taken from the SOEA model.		
 <pre> graph TD SOA[SOA Service] -- realize --> BPS[Business Process Step] App[Application] -- support --> BPS App -- require --> Int[Interface] App -- offer --> Int SOA -- require --> Int Int -- provide --> SOA Int -- canCreate --> BEM[Business Event Message] </pre> <p style="text-align: center;"><i>Fig. 6-18: Measuring points for process events</i></p>		
R_1	The higher the ratio, the better will the business layer be supported by CEP.	

Q14 Can events lead to the execution of applications, SOA services, or orchestrations?

The main objects to the answer of this question are the workflow management tool, the applications, and the SOA services. Only workflow management tools are able to execute orchestrations. Therefore, they offer the mightiest actions to react on events. Applications and SOA services can (theoretically) invoke each other. Most often SOA services will invoke applications and not vice versa.

M23 Invocations	<i>Compound</i>	<i>Automatable</i>
The ratio of possible invocations upon events is measured with the help of the metrics M24 and M25. They measure the business event messages upon those orchestrations and applications respectively SOA services can be invoked.		
M24 x M25		
$R_1 \times R_1$	The higher the ratio, the better CEP will be supported.	

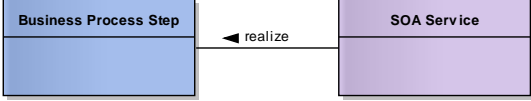
M24 Orchestration invocation	<i>Base</i>	<i>Automatable</i>
<p>This metric measures the ratio of events that are receivable by a workflow management tool.</p>		
<p>If the workflow management tool has interfaces that are able to receive business event messages, it can initiate the execution of orchestrations. The ratio of receivable events to non-receivable events per workflow management tool is determined with the SOEA model concepts shown below.</p> <div data-bbox="389 725 1158 981" data-label="Diagram"> <pre> classDiagram class Application class Interface class BusinessEventMessage["Business Event Message"] class WorkflowManagementTool["Workflow Management Tool"] Application -- > WorkflowManagementTool Application --> Interface : offer Interface --> BusinessEventMessage : canReceive </pre> </div> <p style="text-align: center;"><i>Fig. 6-19: Measuring points for orchestration invocation</i></p>		
R ₁	The higher the ratio, the better CEP will be supported.	

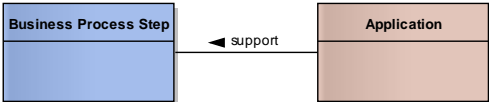
M25 Application invocation	<i>Base</i>	<i>Automatable</i>
<p>This metric measures the ratio of events that are receivable by applications and SOA services.</p>		
<p>If applications and SOA services have interfaces that are able to receive business event messages, they can invoke other services (provided an existing middleware). The ratio of receivable events to non-receivable events per application or SOA service is measured and the average ratio is computed from the SOEA model.</p>		
<div style="text-align: center;">  <pre> graph LR SS[SOA Service] -- provide --> I[Interface] A[Application] -- offer --> I I -- canReceive --> BEM[Business Event Message] </pre> </div> <p style="text-align: center;"><i>Fig. 6-20: Measuring points for application invocation</i></p>		
R ₁	The higher the ratio, the better CEP will be supported.	

Q15 Do SOA services provide business functions that match well to process steps?

As SOA services shall be tailored with alignment to the business, a process step should be implementable with the least possible number of SOA services (which still may have several operations). Applications used for the realization of the process step will be interpreted negatively.

M26 Process step matching	<i>Compound</i>	<i>Automatable</i>
<p>This metric measures the average number of SOA services and the average number of applications that are involved in a process step. Therefore, it combines M27 and M28.</p>		
M27 x M28		
R ₁ x R ₁	The higher the ratio, the better the IT-business alignment will be supported.	

M27 SOA service matching	<i>Base</i>	<i>Automatable</i>
This metric measures the average number of SOA that are involved in a process step.		
Measuring points can be taken from the SOEA model as shown below.		
 <p data-bbox="448 763 1098 797"><i>Fig. 6-21: Measuring points for SOA service matching</i></p>		
R	The higher the ratio, the better the IT-business alignment will be supported.	

M28 Application matching	<i>Base</i>	<i>Automatable</i>
This metric measures the average number of applications that are involved in a process step.		
Measuring points can be taken from the SOEA model as shown below.		
 <p data-bbox="448 1337 1098 1370"><i>Fig. 6-22: Measuring points for application invocation</i></p>		
R	The higher the ratio, the better the IT-business alignment will be supported.	

Q16 How high is the ratio of human work that is supported by SOA services?

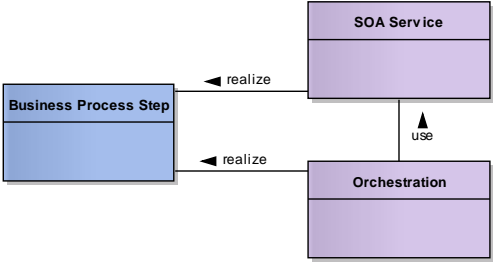
Process steps humans act in should be covered by SOA services embracing the human interaction. Otherwise, the process control flow is given from a workflow management tool to a human, which is unreliable because humans are forgetful.

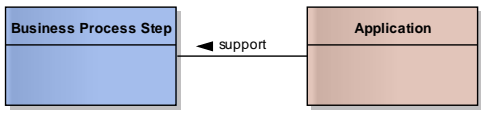
M29 Human support	<i>Base</i>	<i>Automatable</i>
This metric measures how extensively the human work that is done in process steps is covered by SOA services.		
For measurement, the process steps that have human actors are concerned. If there are SOA service realizing the process step where the same roles are involved that act in the process step, then the human work counts as covered by SOA services. The metric returns the ratio between supported to unsupported human work.		
<pre> graph TD Role[Role] -- actIn --> BPS[Business Process Step] SOA[SOA Service] -- realize --> BPS Role -- involved in --> SOA </pre>		
<i>Fig. 6-23: Measuring points for human support</i>		
R ₁	The higher the ratio, the better the IT-business alignment will be supported.	

Q17 Are process steps realized through SOA services?

Processes shall be implemented by services and not by applications directly. Therefore, the process steps that are realized by service or orchestrations are observed. Process steps that are supported directly by applications are usually not desired in a Service-Oriented Enterprise Architecture.

M30 Process realization	<i>Compound</i>	<i>Automatable</i>
This metric combines M31 and M32 and measures process realization with SOA services and orchestrations. Direct application usage is punished.		
$(M31 + 1) / (M32 + 1)$		
R	The higher the value the better.	

M31 Service realization	<i>Base</i>	<i>Automatable</i>
<p>The number of process steps that are directly realized by SOA services or orchestrations are computed with this metric.</p>		
<p>The number of process steps that have at least one association to SOA service or orchestration is set into ratio with the total number of process steps.</p>		
<div style="text-align: center;">  <pre> graph LR SOA[SOA Service] -- realize --> BPS[Business Process Step] Orch[Orchestration] -- realize --> BPS Orch -- use --> SOA </pre> </div> <p style="text-align: center;"><i>Fig. 6-24: Measuring points for service realization</i></p>		
R ₁	The lower the ratio the better.	

M32 Application realization	<i>Base</i>	<i>Automatable</i>
<p>The number of process steps that are directly supported by applications is computed with this metric.</p>		
<p>The number of process steps that have at least one association to application is set into ratio with the total number of process steps.</p>		
<div style="text-align: center;">  <pre> graph LR App[Application] -- support --> BPS[Business Process Step] </pre> </div> <p style="text-align: center;"><i>Fig. 6-25: Measuring points for application realization</i></p>		
R ₁	The lower the ratio the better.	

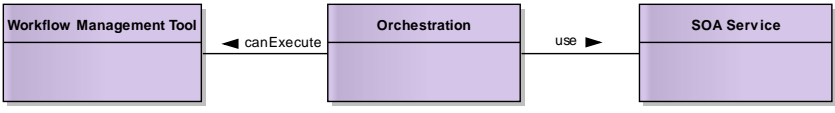
Q18 Is there an orchestration engine that executes SOA service orchestrations?

An orchestration engine is a workflow management tool that is able to interpret service orchestrations that are given in a specific language like BPEL or XPD (XML)

Process Definition Language, compare [WFMCXP05]). The process control flow can be hold by such an engine, so that the process flow will not stop because of human failure. Therefore, it is desirable to have most of the services used in orchestrations.

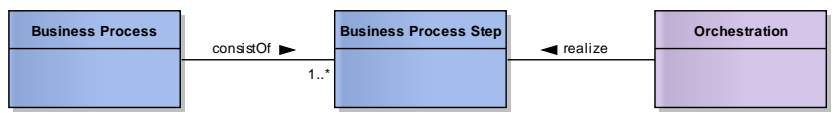
M33 Orchestration engine	<i>Compound</i>	<i>Mixed</i>
This metric measures the existence of an orchestration engine and the ratio of SOA services that is used in service orchestrations. For this reason it combines M34 and M35		
M34 x M35		
$N \times R_1$	The higher the number the better, the higher the ratio the better,	

M34 Orchestration engine existence	<i>Base</i>	<i>Subjective</i>
This metric measures the existence of an orchestration engine being able to interpret SOA service orchestrations.		
The number of instances of workflow management tools that have a technology adequate for SOA service orchestration. These technologies are for example BPEL or XPDL. An expert has to add new technologies that come up for service orchestration.		
<pre> classDiagram class Technology class Application class WorkflowManagementTool["Workflow Management Tool"] Application -- > WorkflowManagementTool Application --> Technology : implement </pre>		
<i>Fig. 6-26: Measuring points for Orchestration engine existence</i>		
N	The higher the better.	

M35 Orchestrated SOA Services	<i>Base</i>	<i>Automatable</i>
<p>This metric measures the ratio of SOA services that is used in service orchestrations.</p>		
<p>The ratio of SOA services that are part of an orchestration and are executed by an arbitrary workflow management tool is measured with the SOEA model as shown below.</p>		
 <pre> classDiagram class WorkflowManagementTool class Orchestration class SOAService Orchestration --> WorkflowManagementTool : canExecute Orchestration --> SOAService : use </pre>		
<p><i>Fig. 6-27: Measuring points for orchestrated SOA services</i></p>		
R ₁	The higher the ratio the better.	

Q19 How high is the ratio of processes that are modelled and executed in an interpretable orchestration language?

The number of processes realized by SOA service orchestrations is a sign for service orientation in an enterprise.

M36 Process orchestrations	<i>Base</i>	<i>Objective</i>
<p>This metric measures the ratio of processes that are realized with SOA service orchestrations.</p>		
<p>The ratio of processes that is realized by SOA service orchestrations is determined by the number of process steps that are covered by an orchestration in relation to the total number of process steps.</p>		
 <pre> classDiagram class BusinessProcess class BusinessProcessStep class Orchestration BusinessProcess --> BusinessProcessStep : consistOf 1..* Orchestration --> BusinessProcessStep : realize </pre>		
<p><i>Fig. 6-28: Measuring points for orchestrated processes</i></p>		
R ₁	The higher the ratio the better.	

Q20 How much process control flow is hidden in applications?

The process control flow shall be isolated in a Service-Oriented Architecture, and shall be modelled in executable SOA service orchestrations. Otherwise, the flexibility to change processes can hardly be realized. Applications often have this process control flow built-in so that the applications have to be changed if the process logic is changed.

M37 Process logic	<i>Base</i>	<i>Subjective</i>
This metric measures the ratio of process control flow that is located in applications.		
As the implementation details of applications are not visible in the SOEA model but rather hidden in the program code, the process control flow ratio of applications has to be guessed by experts knowing the application landscape.		
R ₁	The lower the ratio the better.	

Q21 Is there a BPM application?

A business process monitoring application has to observe processes concerning their key performance indicators.

M38 BPM usage	<i>Compound</i>	<i>Objective</i>
This metric measures the ratio of processes that are monitored with the help of a BPM application and have sufficient key performance indicators. Therefore, it combines M40 and M41.		
M40 x M41		
R x R ₁	The higher the values the better.	

M39 BPM observation	<i>Base</i>	<i>Objective</i>
This metric measures the ratio of processes that are monitored with the help of a BPM application.		

The measuring point is found in the SOEA model where the ratio of processes that are monitored to processes that are not monitored is computed.



Fig. 6-29: Measuring points for BPM application

R ₁	The higher the ratio the better.
----------------	----------------------------------

M40 KPI usage	Base	Objective
This metric measures the number of KPIs per process.		
The measuring point is determined by the number of instances of key performance indicators per business process. The information is retrieved from the SOEA model as depicted below.		
<pre> graph LR BP[Business Process] -- have --> KPI[Key Performance Indicator] BP --- KPI </pre>		
Fig. 6-30: Measuring points for KPI usage		
R	The higher the number the better.	

Q22 How high is the ratio of processes that are monitored automatically?

A process should be monitored with automated IT support so that business event messages are used.

M41 BPM automation	Base	Objective
This metric measures the support of process monitoring by business event messages. Therefore, it combines M40 and M42.		
M40 x M42.		
R x R	The higher the values the better.	

M42 KPI messages	<i>Base</i>	<i>Objective</i>
This metric measures the number of consumed business event messages per key performance indicator.		
<p>Measuring points are found in the SOEA model as depicted below. The metric computes the number of consumed messages per key performance indicator.</p> <div data-bbox="571 645 1059 748" data-label="Diagram"> <pre> graph LR KPI[Key Performance Indicator] -- consume --> BEM[Business Event Message] </pre> </div> <p style="text-align: center;"><i>Fig. 6-31: Measuring points for KPI messages</i></p>		
R	The higher the value the better.	

6.2.2 Metrics for Service Criteria

The second part of the quality criteria catalogue is given in this section. It concerns the quality attributes of SOA services that are observed out of their enterprise architecture context. The first subsection defines the quality criteria and the following one defines appropriate metrics for these criteria. The indicators for the metrics are given in chapter 7. The topic of SOA service quality has been treated in the supervised diploma thesis [UecanE08]. The results of the work are integrated in this section.

Now, metrics for the quality criteria of SOA services are introduced. These metrics can be based on the SOEA meta model but are often based on other information sources like the service contract. Metrics for SOA services are given in the same format as the metrics for the structural quality criteria.

Most metrics are applied on single SOA services, allowing improving services in a well-directed way. In order to evaluate the whole set of services within an enterprise, the arithmetic average or the median has to be computed.

P7 Reusability

Q23 Are the SOA services reusable?

Reusability is the most diversified quality criteria for SOA services. The reason for diversity is that most of the other quality criteria are integrated in this one. Only the sum of the other concepts allows a judgment over the reusability of services. The reusability depends on the P8 Granularity, P9 Technology Independence, P11 Statelessness, P12 Loose Coupling, P15 Service Contract and P16 Discoverability.

M43 Reusability	<i>Compound</i>	<i>Mixed</i>
Reusability is dependent from many other metrics for service criteria. The metrics M44 , M49 , M51 , M52 , M58 and M59 are taken into account.		
M44 x M49 x M51 x M52 x M58 x M59		
$(N \times N \times N) \times N \times N_1 \times (N \times N) \times R_1 \times N_1 \times N_3 \times N$	The better the single metrics the better this one. Refer to interpretations of sub metrics.	

P8 Granularity

Q24 How many service operations do SOA services have?

Granularity metrics concern the number of service operations per service and the number of services itself. It is difficult to identify a useful number of services within an organization. This heavily depends on the range of processes and functionality the enterprise provides.

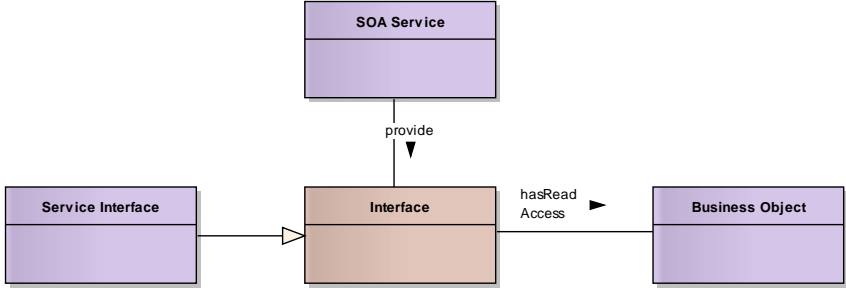
Q25 How many business objects are covered by an SOA service?

A possible way to evaluate the appropriate number of SOA services is to compare it with the number of business objects that are covered with each SOA service. Of course, this value depends on the granularity of business objects, but still offers a lead for service granularity.

M44 Granularity	<i>Compound</i>	<i>Automatable</i>
The granularity of an SOA service is based on the number of services and the number of operations per service. Therefore, M45 and M48 are combined in this metric		
M45 x M48		
$(N \times N) \times N$	The values should not be too high and not too low.	

M45 Business object coverage	<i>Compound</i>	<i>Automatable</i>
The granularity of an SOA service is also based on the number of services. Absolute numbers are hard to define. Therefore, the number of business objects that are involved in the SOA service is determined as a reference value.		
M46 x M47		
N x N	The values should not be too high and not too low.	

M46 Stored Business Objects	<i>Base</i>	<i>Automatable</i>
The granularity of an SOA service is also based on the number of services. Absolute numbers are hard to define. Therefore, the number of business objects that are stored by the SOA service is determined as a reference value.		
The provided service interface of an SOA service stores business objects that are counted per service.		
<pre> classDiagram class SOA_Service[SOA Service] class Service_Interface[Service Interface] class Interface class Business_Object[Business Object] SOA_Service --> Interface : provide Service_Interface -- > Interface Interface --> Business_Object : haswrite Access </pre>		
<i>Fig. 6-32: Measuring points for stored business objects</i>		
N	The value should not be too high and not too low.	

M47 Accessed Business Objects	<i>Base</i>	<i>Objective</i>
<p>The granularity of an SOA service is also based on the number of services. Absolute numbers are hard to define. Therefore, the number of business objects that are accessed by the SOA service is determined as a reference value.</p>		
<p>The provided service interface of an SOA service accesses business objects that are counted per service.</p>  <pre> classDiagram class SOA_Service[SOA Service] class Service_Interface[Service Interface] class Interface class Business_Object[Business Object] SOA_Service --> Interface : provide Service_Interface -- > Interface Interface --> Business_Object : hasRead Access </pre> <p><i>Fig. 6-33: Measuring points for accessed business objects</i></p>		
N	The value should not be too high and not too low.	

M48 Operation quantity	<i>Base</i>	<i>Objective</i>
<p>The granularity of an SOA service is based on the number of services and the number of operations per service.</p>		
<p>The number of service operations has to be found in the service description of an SOA service or can be determined with any technical interface description like a WSDL file.</p>		
N	The value should not be too high and not too low.	

P9 Technology Independence

Q26 Do SOA services enforce the use of concrete technologies?

Technology independence is given if the service does not enforce the use of any technology or application within its description. In addition, the description has to be

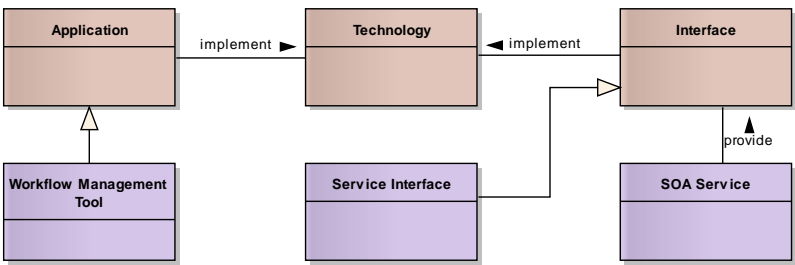
complete. If there is no reference to an application or technology within the service description, the service is evaluated positively for this criterion. The interface technology with that the service is implemented is not regarded.

M49 Technology transparency	<i>Base</i>	<i>Subjective</i>
An SOA service should have no references to concrete technologies, aside from the interface technology, nor should describe how something is done.		
The complete service description is scanned for implementation restrictions. Concrete technologies or restrictions on the (manual) implementation of the service are counted. The technologies from the SOEA model can be taken as help. Interface technologies must not be concerned.		
N	The value should be as low as possible.	

P10 Orchestration

Q27 Do SOA services provide interfaces being adequate for orchestration tools?

An electronic interface is the necessary requirement for orchestration. It should be given by default. However, an interface implemented in a technology that is adequate for the use in the existing orchestration engines (workflow management tool) has to be present.

M50 Orchestration	<i>Base</i>	<i>Objective</i>
An SOA service should have an interface technology that is also used by the majority of orchestration engines.		
For the measurement, interface technologies and workflow management tool technologies are compared. This can be done with the help of the SOEA model. If there is no workflow management tool, the value is zero. If the technology of any service interface of a service is covered by a workflow management tool, the value increases by 1/(number of workflow management tools).		
 <p>The diagram illustrates the relationships between various components in a service-oriented architecture. It features six boxes: 'Application' (top left), 'Technology' (top middle), 'Interface' (top right), 'Workflow Management Tool' (bottom left), 'Service Interface' (bottom middle), and 'SOA Service' (bottom right). 'Application' and 'Technology' are connected by a horizontal arrow labeled 'implement' pointing from Application to Technology. 'Technology' and 'Interface' are connected by a horizontal arrow labeled 'implement' pointing from Interface to Technology. 'Workflow Management Tool' has a vertical arrow pointing up to 'Application'. 'Service Interface' has a vertical arrow pointing up to 'Interface'. 'SOA Service' has a vertical arrow pointing up to 'Interface' labeled 'provide'. Additionally, there is a curved arrow pointing from 'Service Interface' to 'Interface'.</p>		
<i>Fig. 6-34: Measuring points for orchestration</i>		
R ₁	The higher the value the better.	

P11 Statelessness

Q28 Are SOA service operations idempotent?

Statelessness is measured by the ratio of idempotent service operations within a service. An idempotent service operation is regarded as stateless. Idempotency is mathematically defined as $f(x) = f(f(x))$. This means that multiple execution of a service operation (with identical input parameters) always leads to the same result (output and post conditions).

M51 Statelessness	<i>Base</i>	<i>Subjective</i>
A service is regarded as stateless if his service operations are idempotent.		
Testing all operations with all inputs concerning their result in case of repeated execution is not justifiable. Therefore, an expert known to the SOA service has to estimate whether an operation is idempotent or not. That means the repeated execution of an operation with the same input parameters leads to the same output and post conditions.		
R ₁	The higher the value the better.	

P12 Loose Coupling

Q29 Do SOA services communicate with business event messages?

Loose coupling concerns the premise of other systems. Sending events is regarded as a very loose form of coupling, because the sender does not know its recipients and the recipient does not have to know the sender.

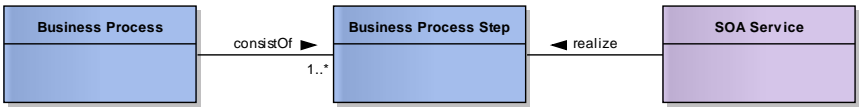
M52 Loose Coupling	<i>Base</i>	<i>Objective</i>
SOA services should not be dependent on other services or applications. Therefore, their communication should be processed with events.		
For the measurement, the number of business event messages sent by an SOA service is concerned. The number of different business event messages events has to be seen in relation with the number of service operations. For this reason, the ratio the number of events per service operation is measured.		
<pre> classDiagram class BusinessEventMessage["Business Event Message"] class Interface class SOAService["SOA Service"] class ServiceInterface["Service Interface"] BusinessEventMessage <--> Interface : canCreate SOAService <--> Interface : provide ServiceInterface -- > Interface </pre>		
R	The higher the value the better.	

Fig. 6-35: Measuring points for loose coupling

P13 Functional compactness

Q30 Do SOA services have a compact process context?

Functional compactness is measured with help of the processes that are implemented by the SOA services. As a positive criterion, the number of process steps within a process that are realized by an SOA service is observed. Within a process, the process steps that are realized by an SOA service are counted and the ratio to the total number of steps is computed. It does not play a role which service operation is used to implement the process step. The higher the number of steps realized within a service the more compact the service seems to be. The value is averaged over the number of processes the SOA service is involved.

M53 Functional compactness	<i>Base</i>	<i>Objective</i>
SOA services should hold as much as possible of the functionality required in a certain context. This is measured with the help of the business processes the SOA service is used in. The more process steps the SOA service is involved in, the better the functional compactness.		
The measurement is done with the help of the SOEA model. For an SOA service, the average number of process steps it realizes within a process is counted.		
 <pre> classDiagram class BusinessProcess class BusinessProcessStep class SOAService BusinessProcess "1" -- "*" BusinessProcessStep : consistOf SOAService -- BusinessProcessStep : realize </pre>		
<p style="text-align: center;"><i>Fig. 6-36: Measuring points for functional compactness</i></p>		
R	The higher the value the better.	

P14 Compensation

Q31 Are the SOA services compensable?

Compensation is considered as given if a service operation can generally be undone with a sequence of service operations from the same service. Of course, read only services should not be regarded. If the service operation can only be undone with the help of service operations from other services the compensation attribute is given in a weaker form. The worst case occurs if no compensation is possible at all.

M54 Compensation	<i>Compound</i>	<i>Subjective</i>
The service operations of an SOA service should be compensable by the operations of the same service or at least with operations from other services. This metric combines M55, M56 and M57		
M55 x M56 x M57		
$R_1 \times R_1 \times R_1$	The higher the third value the better. Apart from that, the higher the second value the better. The lower the first value the better.	

M55 No Compensation	<i>Base</i>	<i>Subjective</i>
Compensation can hardly be determined automatically. For this reason, an expert has to decide which of the operations of a service are not compensable at all.		
The expert evaluates the compensability of the service operations of an SOA service. Read-only service operations are not concerned. Not compensable means that there is no way to compensate the results of a service operation. The ratio of these service operations to other service operations of the same SOA service is measured.		
R_1	The lower the value the better.	

M56 External Compensation	<i>Base</i>	<i>Subjective</i>
Compensation can hardly be determined automatically. For this reason, an expert has to decide which of the operations of a service are externally compensable.		
The expert evaluates the compensability of the service operations of an SOA service. Read-only service operations are not concerned. Externally compensable means that there is the possibility to compensate an operation with the help of other SOA services. The ratio of such service operations to other service operations of the same SOA service is measured.		
R_1	The higher the value the better.	

M57 Internal Compensation	<i>Base</i>	<i>Subjective</i>
Compensation can hardly be determined automatically. For this reason, an expert has to decide which of the operations of a service are internally compensable.		
The expert evaluates the compensability of the service operations of an SOA service. Read-only service operations are not concerned. Internally compensable means that there is the possibility to compensate with the help of operations from the same SOA service. The ratio of such service operations to other service operations of the same SOA service is measured.		
R ₁	The higher the value the better.	

P15 Service Contract

Q32 Are the contracts of SOA services complete?

Service Contracts are inevitable for SOA services. Their quality is measured by the completeness of their descriptions.


M58 Service contract	<i>Base</i>	<i>Subjective</i>
<p>The SOA service should have a description containing the following information:</p> <ul style="list-style-type: none"> ▪ Description <ul style="list-style-type: none"> ▪ Interface ▪ Functionality ▪ Usage ▪ Lifecycle Status ▪ Responsibility ▪ Objection Level Agreement (OLA) <ul style="list-style-type: none"> ▪ Constraints ▪ Availability ▪ Accessibility ▪ Visibility ▪ Security ▪ Monitoring and Reporting <ul style="list-style-type: none"> ▪ Business event messages ▪ Predefined reports 		
<p>The measurement is supposed to be done by an expert. Each criterion should be checked and if one of them misses, the result is decreased by 0.1, starting with the value one. Negative results are not possible.</p>		
R ₁	The higher the value the better.	

P16 Discoverability

Q33 Can SOA service be discovered easily within the enterprise?

Discoverability is achieved by providing means to discover the service without special previous knowledge. Within a Service-Oriented Architecture, this is achieved by registering the service in a registry. Of course, sheer registration is no guarantee for discoverability. However, from the service point of view it is a precondition that the service registered and that is described in detail. The latter is covered by the service contracts.

M59 Discoverability	<i>Compound</i>	<i>Mixed</i>
A service is regarded as discoverable if it is registered in a service registry or repository determined by M60 . The kind of registry has influence on the evaluation of this criterion and is already covered with M1 .		
M60 x M1		
$N_1 \times N_3 \times N$	The higher the values the better.	

M60 Service registration	<i>Base</i>	<i>Automatable</i>
A service is regarded as discoverable if it is registered in a service registry or repository. This metric determines whether the SOA service is registered in a registry or not.		
The measurement can be done with the SOEA model, where the registration of SOA services in registries can be examined.		
 <pre> graph LR A[SOA Service] -- isRegisteredIn --> B[Service Registry] </pre>		
<i>Fig. 6-37: Measuring points for service registration</i>		
N_1	The higher the value the better.	

The definitions of the service criteria catalogue and the corresponding metrics have been presented in this chapter. Using the metrics for an evaluation requires indicators that are given in the following chapter.

7 Interpretation of Measures

The metrics that were described in the previous chapter do not yet allow a statement about the fulfilment of the given quality criteria. For this reason, this chapter covers the definition of indicators for the defined metrics. The metrics and their indicators are treated separately in section 7.1, because the indicators are, other than the metrics, dependent on the context of usage. With the indicators defined, a complete report on service orientation of an enterprise architecture can be given. The form of representation is described in section 7.2. If the quality criteria are not fulfilled, actions that lead to improvement of the situation have to be defined. In section 7.3 the improvement suggestions are covered.

7.1 Indicators for Service Orientation Metrics

Indicators are dependent from their context that means there are no fixed values for the metrics that have to be stuck to. Dependent on the user of this framework, who considers a quality criterion as more or less important, the preferred indicator values may differ. Here, every metric is treated as equally important and the indicator values are to be seen as a suggestion or default setting.

Fig. 7-1 depicts the structure of chapter 7 and the requirements that are covered. The indicators for metrics of structural criteria are described in subsection 7.1.1. Afterwards, the indicators for service criteria are given in subsection 7.1.2. Both sections support the fulfilment of the requirements R7 “As-Is & Target Modelling” and R11 “Automation of criteria checks”. Section 7.2 presents a how the information of indicators can be condensed and formatted. The indicators allow evaluating the current situation of an enterprise architecture (concerning service orientation) but an improvement of the situation requires actions to be taken. The way to find and prioritize those actions is given in section 7.3. With this, R12 “Improvement suggestions” will be covered.

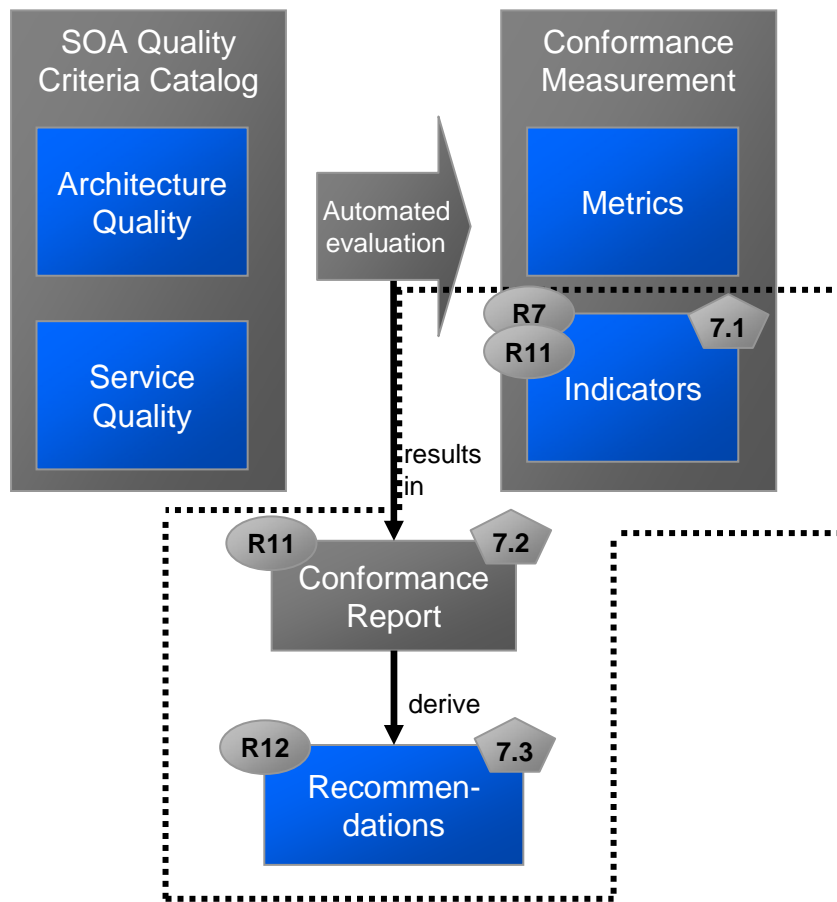


Fig. 7-1: Contribution and structure of chapter 7

The indicators for structural criteria and for services are given in the format depicted in Fig. 7-2. Numbering of the indicators is consistent with the numbering of the metrics. That means IM1 corresponds to M1, IM2 to M2 and so on. If there is a compound metric for a quality criterion, then there is no indicator for its base metrics.

The domain of an indicator is equal to the co-domain of the corresponding metric. The co-domain of the indicator is always the interval $[0,1]$. Discrete sets like a traffic light system or a grade point system are not chosen here because these can be defined after the needs of the user afterwards. The value of the indicator is always the better the higher it is. That means 0 is the worst and 1 is the best result.

Acronym, Name of the corresponding metric	Domain	Calculation Rule (if compound)
Interpretation Rule		
Indication function		
Conditions for parameter integrity	Suggested values for adjustable parameters	

Fig. 7-2: Template for indicator definition

Within the indication function, the determined value of a base metric is referenced with its acronym, e.g. M1. Compound metrics have tuple values. The single tuple elements are referenced with the corresponding base metric acronyms.

Adjustable parameters are often part of the indication function. If the interpretation of a metric is adjustable, then these parameters are the adjustable screws to concentrate on. Adjustable parameters are named like the indicator acronym plus an additional index, e.g. IM1₁. Suggested values for the parameters are given in the lower right corner of the table. Changing parameters requires taking care of the resulting co-domain of the function. Therefore, conditions for parameter integrity are defined. These ensure that the co-domain remains [0,1]. If it is not [0,1] afterwards, the changed parameter values are invalid. The redefinition of the indicator function could solve this but is not suggested.

A traffic light system could easily be applied on these indicators by defining that values in the interval [0, 1/3[correspond to red, values in [1/3, 2/3[correspond to orange and values between [2/3, 1] correspond to green. Other values are possible as well.

7.1.1 Indicators for Structural Criteria

This subsection covers the definition of indicators for the structural criteria metrics.

IM3 Service registry	$N_3 \times N$	$M1 \times M2$
The higher the value for M1 the better. M5 equal zero means no registry exists, which is worst. One is best because it is central then. Beyond this, the more logical registries the worse.		
$IM1 := M1 / 3$ $IM2 :=$ If $M2 = 0$ then 0 if $M2 = 1$ then 1 else $1/M2$		
$IM3 := IM1 * IM1_1 + IM2 * IM2_1$		
$IM1_1 + IM2_1 = 1$	$IM1_1 = 1/2$	$IM2_1 = 1/2$

IM6 Service repository	$N_3 \times N$	$M4 \times M5$
The higher the value for M4 the better, as more functions are implemented. M5 equal zero means no repository exists, which is worst. One is best because it is central then. Beyond this, the more repositories the worse.		
$IM4 := M4 / 3$ $IM5 :=$ If $M5 = 0$ then 0 if $M5 = 1$ then 1 else $1/M5$		
$IM6 := IM4 * IM4_1 + IM5 * IM5_1$		
$IM4_1 + IM5_1 = 1$	$IM4_1 = 1/2$	$IM5_1 = 1/2$

IM7 Multi-channel services	R	M7
A plethora of service interface technologies is helpful but not too much interface technologies should be supported, as the maintenance of too many interface technologies will outweigh their benefit.		
$IM7 := \begin{cases} \text{If } M7 < IM7_1 & \text{then } M7 / IM7_1 \\ \text{if } M7 = IM7_1 & \text{then } 1 \\ & \text{else } 1/(M7 - IM7_1) \end{cases}$		
$IM7_1 \geq 1$	$IM7_1 = 4$	

IM8 SOA service reuse	R	M8
The higher the number of process steps used by a service the better. Theoretically, an unlimited number of reuses is possible. To keep the values of the indicator in a reasonable high range, a maximum reference value ($IM8_1$) is set.		
$IM8 := \begin{cases} \text{If } M8 \leq IM8_1 & \text{then } M8 / IM8_1 \\ & \text{else } 1 \end{cases}$		
$IM8_1 \geq 1$	$IM8_1 = 20$	

IM9 Legacy adaptation	R_1	M9
The higher the ratio of applications used by SOA services the better.		
$IM9 := M9$		
-	-	

IM10 Common middleware	$R_1 \times R_1$	$M11 \times M12$
The higher the numbers the higher the middleware prevalence and thus the better.		
$IM10 := M11 * IM11_1 + M12 * IM12_1$		
$IM11_1 + IM12_1 = 1$	$IM11_1 = 2/3$ $IM12_1 = 1/3$	

IM13 Standard communication	R	M13
The lower the number of interface technologies the better, except of a small number of common interface technologies (IM13 ₁). To keep the indicator value in reasonable borders a maximum reference value (IM13 ₁) is set.		
$\text{IM13} := \begin{cases} \text{If } M13 \leq M13_1 & \text{then } 1 \text{ else} \\ \text{if } M13_1 \leq M13 \leq M13_2 & \text{then } 1 - (M13 - \text{IM13}_1) / (M13_2 - \text{IM13}_1) \\ & \text{else } 0 \end{cases}$		
IM13 ₁ ≥ 1	$\begin{aligned} \text{IM13}_1 &= 4 \\ \text{IM13}_2 &= 20 \end{aligned}$	

IM14 Functionality separation	R	M14
The ratio of functionality that is only provided by graphical user interfaces is measured with M14. As high values are not acceptable at all, a ratio of IM14 ₁ is already considered as worst possible result.		
$\text{IM14} := \begin{cases} \text{If } M14 > \text{IM14}_1 & \text{then } 0 \\ & \text{else } 1 - (M14 / \text{IM14}_1) \end{cases}$		
0 < IM14 ₁ ≤ 1	IM14 ₁ = 1/3	

IM15 Implementation exchangeability	R ₁	$(M16 + M17 * M18) / (M16 + M17)$
The measurement of M18 has to be done by an expert, whereas the ratio for SOA services is regarded as given. The higher the overall exchangeability the better.		
IM15 := M15		
-	-	

IM19 Event processor	N_3	M19
The higher the value for M19 the better the feature richness of the event processor.		
IM19 := $M19 / 3$		
-	-	-

IM20 Event enablement	R_1	M20
The higher the value for M20 the better the EDA support.		
IM20 := $M20$		
-	-	-

IM21 Business object events	R_1	M21
The higher the value for M21 the better the EDA support.		
IM21 := $M21$		
-	-	-

IM22 Process events	R_1	M22
The higher the value for M22 the better the EDA support.		
IM21 := $M21$		
-	-	-

IM23 Invocations	$R_1 \times R_1$	M24 x M25
The higher the value for M23 the better the EDA support.		
IM23:= $M24 * IM24_1 + M25 * IM25_1$		
$IM24_1 + IM25_1 = 1$	$IM24_1 = \frac{1}{2}$ $IM25_1 = \frac{1}{2}$	

IM26 Process step matching	R x R	M27 x M28
The higher the values the better the support for IT-business alignment. Theoretically, the number of SOA services and applications is unlimited. For getting reasonable indicator values, a maximum reference value of $IM26_1$ is set. As services are considered coarser grained than applications their weight is increased.		
$IM26 := \text{If } M27 * IM27_1 + M28 * IM28_1 > IM26_1 \text{ then } 1$ $\text{else } (M27 * M27_1 + M28 * M28_1) / IM26_1$		
$IM26_1 > 1$ $IM27_1 + IM28_1 = 1$	$IM26_1 = 20$ $IM27_1 = 3/5$ $IM28_1 = 2/5$	

IM29 Human support	R_1	M29
The higher the value for M22 the better the IT-business alignment.		
$IM29 := M29$		
-	-	

IM30 Process realization	R	$M30 = (M31 + 1)/(M32 + 1)$
The higher the value the better for the IT-business alignment.		
$IM30 := (M30 - 0.5) * 0.75$		
-	-	

IM33 Orchestration engine	$N \times R_1$	$M34 \times M35$
The higher the value for orchestration engine existence and the ratio of orchestrated services the better. A maximum reference value for M34 has to be set ($IM34_1$), because its co-domain is infinite.		

$\text{IM33} := \begin{cases} \text{If } M34 > \text{IM33}_1 & \text{then } 1 * \text{IM34}_1 + M35 * \text{IM35}_1 \\ \text{else } & (M34 * \text{IM34}_1) / \text{IM33}_1 + M35 * \text{IM35}_1 \end{cases}$	
$\text{IM33}_1 \geq 1$	$\text{IM33}_1 = 30$
$\text{IM34}_1 + \text{IM35}_1 = 1$	$\text{IM34}_1 = \frac{1}{4}$ $\text{IM35}_1 = \frac{3}{4}$

IM36 Process orchestrations	R_1	M36
The higher the ratio of processes realized by orchestrations the better.		
$\text{IM36} := M36$		
-	-	-

IM37 Process logic	R_1	M37
The lower the ratio of process control flow hidden in applications the better.		
$\text{IM37} := 1/M37$		
-	-	-

IM38 BPM usage	$R_1 \times R$	$M39 \times M40$
The higher the value for BPM observation of processes and the usage of KPIs the better. A maximum reference value for M40 has to be set (IM38_1), because its co-domain is infinite.		
$\text{IM38} := \begin{cases} \text{If } M40 > \text{IM38}_1 & \text{then } M39 * \text{IM39}_1 + 1 * \text{IM40}_1 \\ \text{else } & M39 * \text{IM39}_1 + (M40 * \text{IM40}_1) / \text{IM38}_1 \end{cases}$		
$\text{IM38}_1 \geq 1$	$\text{IM38}_1 = 5$	
$\text{IM39}_1 + \text{IM40}_1 = 1$	$\text{IM39}_1 = \frac{1}{2}$ $\text{IM40}_1 = \frac{1}{2}$	

IM41 BPM automation	R x R	M40 x M42
The higher the value for the usage of business event messages for KPIs and the usage of KPIs the better. Maximum reference values for M40 and M42 have to be set (IM41 ₁ , IM41 ₂), because their co-domains are infinite.		
IM40 :=	If M40 > IM41 ₁ then 1 else M40	
IM42 :=	If M42 > IM41 ₂ then 1 else M42	
IM41 :=	IM40 * IM40 ₂ + IM42 * IM42 ₁	
IM41 ₁ >= 1	IM41 ₁ = 5	
IM41 ₂ >= 1	IM41 ₂ = 5	
IM40 ₂ + IM42 ₁ = 1	IM40 ₂ = ½ IM42 ₁ = ½	

7.1.2 Indicators for Service Criteria

Indicators for service criteria are given in exactly the same way as the indicators for structural criteria. Metrics for service criteria carry the same name as their service quality criterion and as their indicator. The acronym only differs in the first letters, which are M, S, and IM respectively.

IM43 Reusability	(N x N x N) x N x N ₁ x (N x N) x R ₁ x N ₁ x N ₃ x N	M44 x M49 x M51 x M52 x M58 x M59
The better the single metrics the better the reusability.		
IM43 :=	IM43 ₁ *IM44 + IM43 ₂ *IM49 + IM43 ₃ *IM51 + IM43 ₄ *IM52 + IM43 ₅ *IM58 + IM43 ₆ *IM59	
IM43 ₁ + IM43 ₂ + IM43 ₃ + IM43 ₄ + IM43 ₅ + IM43 ₆ = 1	IM43 ₁ = 1/6 IM43 ₄ = 1/6 IM43 ₂ = 1/6 IM43 ₅ = 1/6 IM43 ₃ = 1/6 IM43 ₆ = 1/6	

IM44 Granularity	(N x N) x N	M45x M48
The values should not be too high and not too low in order to have SOA services of balanced size.		
<p>IM44 := IM44₁*IM45 + IM44₂*IM48</p> <p>IM45 := IM45₁*IM46 + IM45₂*IM47</p> <p>IM46 := If M46 >= IM46₁ then 0 else If M46 <= IM46₂ then 1 else (IM46₁ - M46)/(IM46₁ - IM46₂)</p> <p>IM47 := If M47 >= IM47₁ then 0 else If M47 <= IM47₂ then 1 else (IM47₁ - M47)/(IM47₁ - IM47₂)</p> <p>IM48 := If M48 >= IM48₁ then 0 else If IM48₁M48 <= IM47₂ then 1 else (IM47₁ - M47)/(IM47₁ - IM47₂)</p>		
<p>IM44₁ + IM44₂ = 1</p> <p>IM45₁ + IM45₂ = 1</p> <p>IM46₁ > IM46₂</p> <p>IM47₁ > IM47₂</p>	<p>IM44₁ = ½</p> <p>IM44₂ = ½</p> <p>IM45₁ = ½</p> <p>IM45₂ = ½</p> <p>IM46₁ = 10</p> <p>IM46₂ = 2</p> <p>IM47₁ = 20</p> <p>IM47₂ = 5</p>	

IM49 Technology transparency	R_1	M49
Any technology restriction within the SOA service description leads to a reduction of the value. Three restrictions are taken as a reference value for a completely inadequate service description.		
IM49 := If M49 \geq IM49 ₁ then 0 else else (IM49 ₁ - M49)/IM49 ₁		
IM49 ₁ \geq 1	IM49 ₁ = 3	

IM50 Orchestration	R_1	M50
The more SOA services there are that can be orchestrated in the present workflow engines (with any service interface technology provided) the better.		
IM50 := M50		
-	-	

IM51 Statelessness	R_1	M51
The higher the ratio of stateless SOA services the better. However, 1 is an ideal value. This is why the value IM51 ₁ is already regarded as optimal.		
IM51 := If M51 \geq IM51 ₁ then 1 else M51 / IM51 ₁		
IM51 ₁ \leq 1	IM51 ₁ = 0.8	

IM52 Loose Coupling	R	M52
The more events per operation are created by an event the better. A value of IM52 ₁ events per operation is regarded as sufficient.		
IM52 := If M52 \geq IM52 ₁ then 1 else M52 / IM52 ₁		
IM52 ₁ \leq 1	IM52 ₁ = 5	

IM60 Service registration	N_1	M60
The higher the values the better the discoverability of the SOA services.		
IM60 :=	M60	
-	-	

Now, all the indicators for the metrics have been defined. With this, a report on service orientation can be created. The structure of this report is given in the next section.

7.2 Report on Service Orientation of an EA

Having the results of the metrics and indicators at hand, a user-friendly form of representation is needed. This section proposes a clearly arranged way to represent the results. The enterprise architect shall be able to draw his conclusions from the information with the overview on the metric and indicator results. For this reason, this section describes the layout of a report on service orientation of an enterprise architecture.

The retrieval of the measures has not yet been described, but will be done in the next chapter. This is done, because the tool support shall retrieve the lion's share of the required metrics.

A report on service orientation should include the following information. Firstly, it should list the quality criteria and their categorization. Secondly, it should include the indicator values for each criterion. Thirdly, the metric values used for the computation of indicators should be contained as well. Of course, a clear arrangement should also include an aggregated graphical representation of the information. To have an overview on the information, the table and diagram views from Fig. 7-3 and Fig. 7-4 are proposed.

Model-Based Evaluation of Service-Oriented Enterprise Architectures

Category	Criterion Acronym	Criterion Question	Indicator Acronym	Indicator Value	Metric Value
Middleware	Q01	Is there a central service registry?	IM3	1	1
	Q07	Are exclusively standards used for communication protocols?	IM13	0,25	17
Disclosure of Interfaces	Q08	Are visualization and functionality separated from each other?	IM14	0,3	0,23
	Q09	Are application and SOA-service implementations exchangeable?	IM15	0,5	0,5
Complex Event Processing	Q10	Is there an event processor?	IM19	0,33	1
	Q14	Can events lead to the execution of applications, SOA-services?	IM23	0,5	0,6 x 0,4
IT-Business Alignment	Q15	Do SOA-services provide business functions that match with business processes?	IM26	0,7	10 x 20
	Q17	Are process steps realized through SOA services?	IM30	0,5	1,16
Orchestration	Q18	Is there an orchestration engine that executes SOA services?	IM33	0,3	9 x 0,3
	Q20	How much process control flow is hidden in applications?	IM36	0,2	0,2
Business Process Monitoring	Q21	Is there a BPM application?	IM37	0,7	1,42
	Q22	How high is the ratio of processes that are monitored automatically?	IM41	0,5	2 x 3
SOA Services	Q23	Are the SOA services reusable?	IM43	0,6	0,7 x 0,4 x 0,9 x 0,2 x 0,6 x 0,8
	Q33	Can SOA service be discovered easily within the enterprise?	IM60	1	1

Fig. 7-3: Table view of report on service orientation

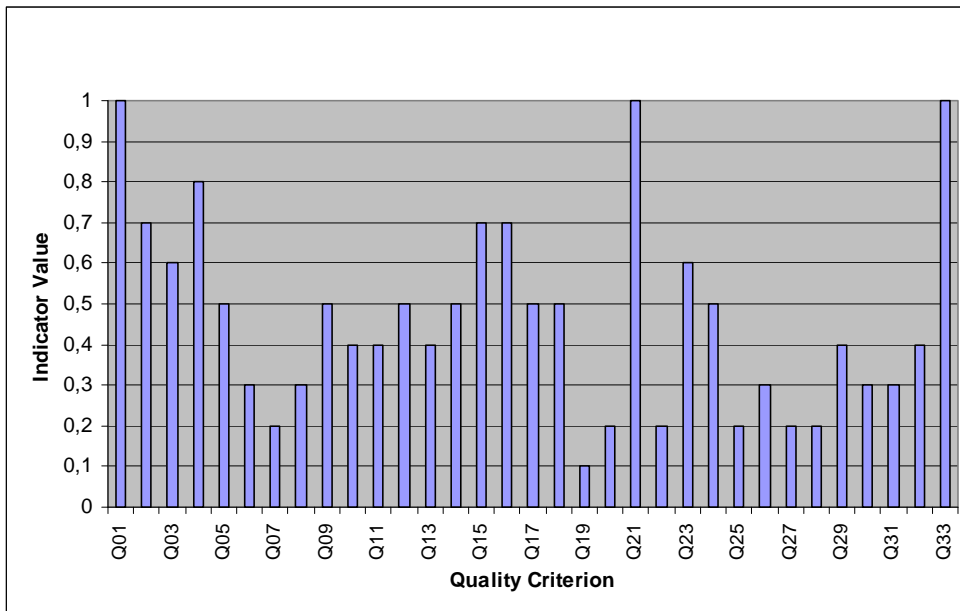


Fig. 7-4: Bar diagram of report on service orientation

The kiviati diagram showing categories of the quality criteria is a more condensed form of the report. It can be used as summary, e.g. in presentations.

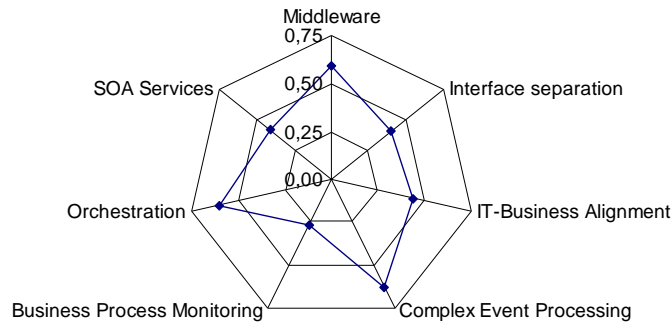


Fig. 7-5: Diagram of report on service orientation

This section has provides a way to present the results of the SOEA evaluation in a clearly layouted form. By that, the weaknesses of the EA concerning service orientation can be revealed. Of course, consequences should be drawn from these insights. The next chapter will propose remedial actions for potential weaknesses.

7.3 Recommendation of Improvements

Knowing the weaknesses of the enterprise architecture concerning service orientation is only the premise for improving the situation. Actions for improvements have to be derived from the gathered information. This section describes an approach on how to derive these actions. The result will consist of the actions to be taken and a priority order in that the steps shall be executed. The strategy for prioritizing actions is presented in 7.3.2 and the strategy for defining actions is given in 7.3.1.

7.3.1 Defining Remedial Actions

A remedial action is a task or project that increases the quality of an enterprise architecture towards service orientation. The determination of these actions is dependent from the indicator values that have been identified. Any low indicator value points out room for improvement.

Up to a certain point, the way for determining an action corresponding to a low indicator is similar in each case. This similar process will be described as a strategy. The part beyond the strategy is dependent from the individual case and has to be covered by the enterprise architect.

A low indicator value is the result of the measures it is calculated from. Increasing an indicator value premises the alteration of its measures. The metrics included in the calculation of the indicator can be tracked in section 7.1 containing the indicator definitions. The metrics listed there have to be examined concerning their measures. If the measuring points lie within the model, then a model design and the according real world system changes have to be brought up. This changes the measures in a way increasing the targeted indicator value.

For example, if IM3 is very low, then the metrics M1 and M2 have to be influenced. M1 measures the features of the service registry and M2 its existence. If M2 is low then a registry should be added to the model and the possible consequences should be checked. Afterwards, a task to set up a registry has to be created and realized. Which registry to choose and how to install it, is task of the enterprise architect then.

7.3.2 Strategy for Prioritizing Actions

Concerning the priority ordering, a top down approach is favoured here. It starts with the examination of the quality properties for service orientation, specifies a priority order, and then examines the possible improvements for single quality properties (P1 to P6). P7 to P16 are regarded here as one group of SOA service quality properties.

A precedence relation graph for the quality properties is shown in Fig. 7-6. The higher a property is depicted the higher its priority. The higher the priority of a property the sooner the actions to improve these properties should be taken.

The first property to be concerned is the middleware. A middleware strategy for the realization of SOA services should be decided first. Tailoring SOA services is closely related to establishing an IT-business alignment and the disclosure of functionality. An adequate method for the tailoring of SOA services is given in [UecanE08]. When at least some SOA services exist, their orchestration can be concerned. The correlation of events and the initiation of compensating workflows upon critical events should be tackled when the basis for the orchestration of SOA services has been established. The

last property in the hierarchy is business process monitoring. It is strongly dependent on the complex event processing and should be realized afterwards.

There are different quality criteria within each property. In general, their prioritization can be decided as follows. Quality criteria that demand the existence something are to be preferred over criteria that demand ratios of certain items. For example, the existence of a registry (IM3) is more important than the ratio of service reuses per service (IM8). It does not make sense to aspire high a reuse ratio of 20 before realizing other steps of the property orchestration (like establishing an orchestration engine). As a rule of thumb, an averaged indicator value of a property as shown in Fig. 7-5 should be higher as the values for properties of lower precedence.

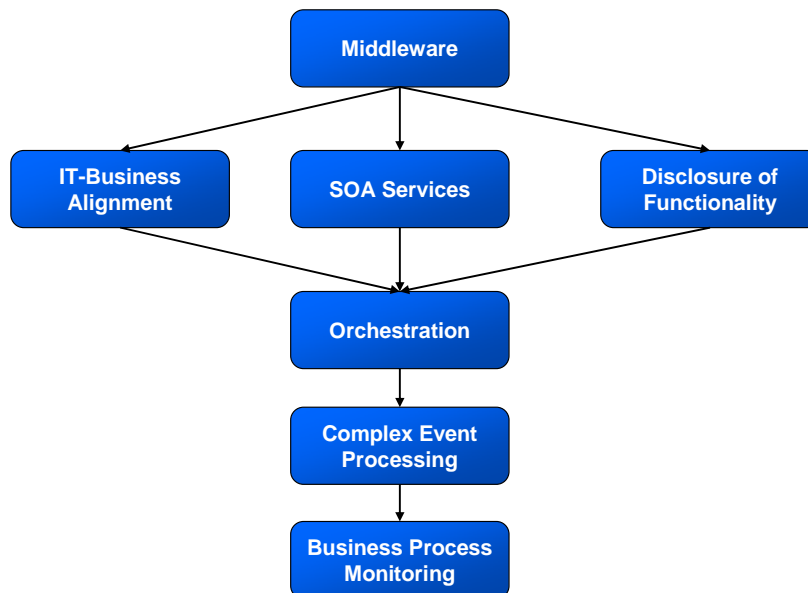


Fig. 7-6: Precedence graph for quality properties

The strategy for the determination of a priority order concludes this chapter. The chapter has covered the interpretation of measurements by defining indicators and suggesting a presentation form. Furthermore, the derivation of actions has been covered here. The prototypical implementation of a customized modelling tool allowing the automated evaluation of quality criteria will be described in the next chapter.

8 Tool support with an Eclipse-Based Prototype

This chapter is dedicated to the description of the tool support for the evaluation method described in the previous chapters. It is a guide for the enterprise architect on how to create the SOAMeter tool that fits to his individual SOEA meta model. Every tool is individual because of the individual SOEA meta model that is allowed in the method. Other parts of the editor, like the graphical editor, are dependent from the individual SOEA meta model and their creation has to be described generically as well. The tool is based on the work of the master's thesis [Doroci09].

The construction of the SOAMeter tool is split into three parts. Section 8.1 covers the implementation of the SOEA meta model and a primitive tree syntax editor for its instances – the SOEA models. Section 8.2 describes how to implement a graphical editor for SOEA models. Afterwards, the metrics based on the SOEA model will be implemented. The main requirements fulfilled by the SOAMeter tool are R10 “Tool support” and R11 “Automation of criteria checks” with the stress on automation. Many other requirements are covered by the tool. However, they are only the implementation of what has been elaborated in the previous chapters.

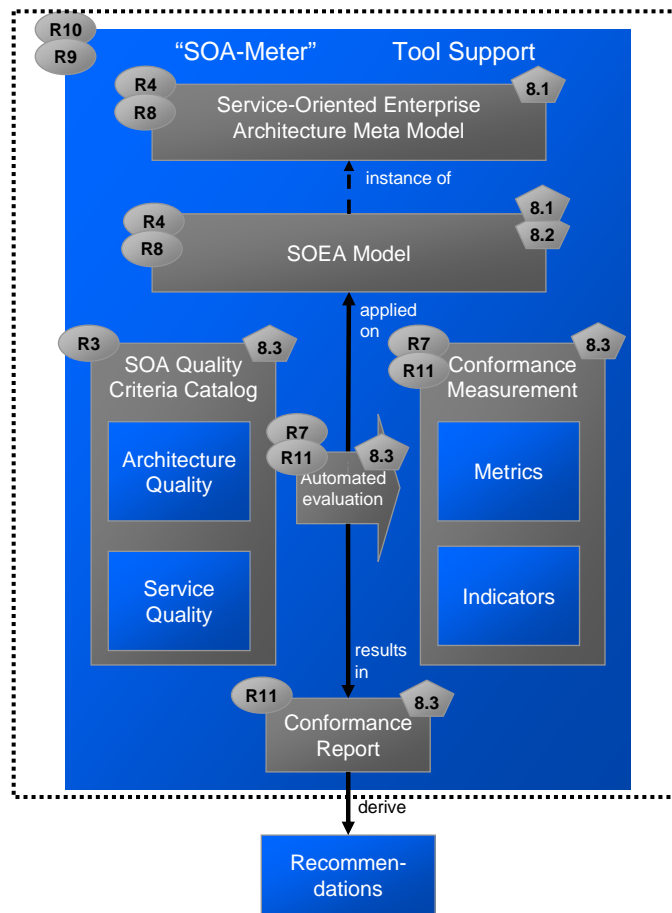


Fig. 8-1: Contribution of chapter 8

The eclipse platform has been chosen for the prototypical implementation of the tool. The decision for eclipse is based on the plethora of functionalities that are already implemented by existing framework components. The Eclipse Modelling Framework (EMF, compare [Eclips03]) supports the creation of an SOEA meta model and offers a tree syntax editor for models that are instances of the pre-defined SOEA meta model. The graphical modelling framework (GMF, compare [Eclips06]) offers the possibility to create a graphical editor for SOEA models. Finally, the EMF contains an OCL library (compare Elips07) offering the possibility to implement statements of the object constraint language (OCL, compare [OCLspe06]) for models of the EMF. OCL is a formal language used to describe expressions on UML models. These expressions typically specify invariant conditions that must hold for the system being modelled.

The following table specifies the versions of the eclipse frameworks used for the implementation:

Eclipse Platform	3.4.2
Eclipse Modelling Framework	2.4.2
Eclipse Graphical Modelling Framework	1.1.3
OCL 2.0 Parser/Interpreter	1.2.3
J2SE	1.5

Fig. 8-2: Versions of used technologies

Based on this configuration, the steps of the creation of a SOAMeter tool for the modelling and evaluation of enterprise architectures will be described in the next sections.

8.1 SOEA Meta Model Implementation with EMF

The first step after setting up a new workspace is to implement the SOEA meta model with the Eclipse Modelling Framework. The second is to create a new GMF project. The GMF offers a very comfortable tutorial in form of a cheat sheet and a dashboard. The cheat sheet guides the user step for step when creating a graphical meta model editor. The dashboard does the same but with a graphical illustration of the steps.

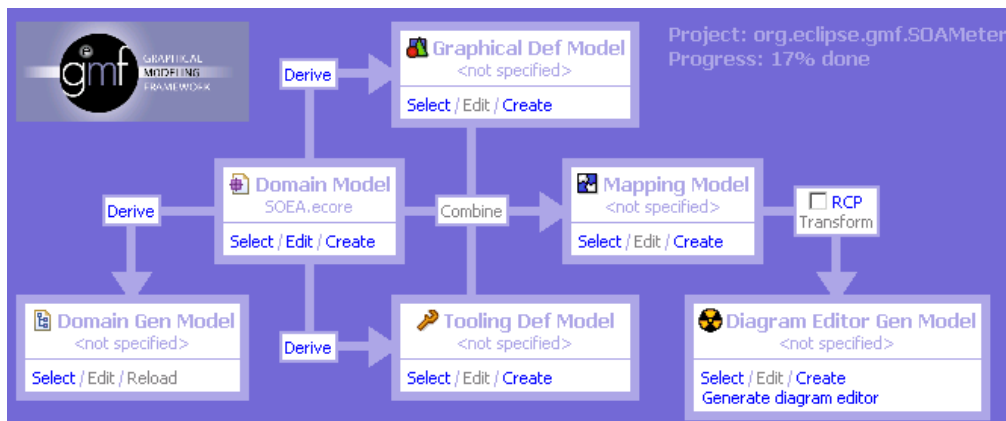


Fig. 8-3: GMF dashboard after first step

In Fig. 8-3, the GMF dashboard is depicted after the initial creation of an empty Ecore Model. Ecore is the eclipse implementation of the Meta Object Facility and

allows the creation of a description of the SOEA meta model. The Ecore Model syntax tree with some basic elements is shown in Fig. 8-4. The elements shown there should be added to every SOEA Ecore model, because they will ease the implementation of the metrics (in form OCL constraints) afterwards. The first element is the eclass “Enterprise_Architecture” having a “consists of” reference to “__Named_Element”. Furthermore, “__Named_Element” is a supertype for all SOEA meta model classes. The “__Metric” is the last additional class.

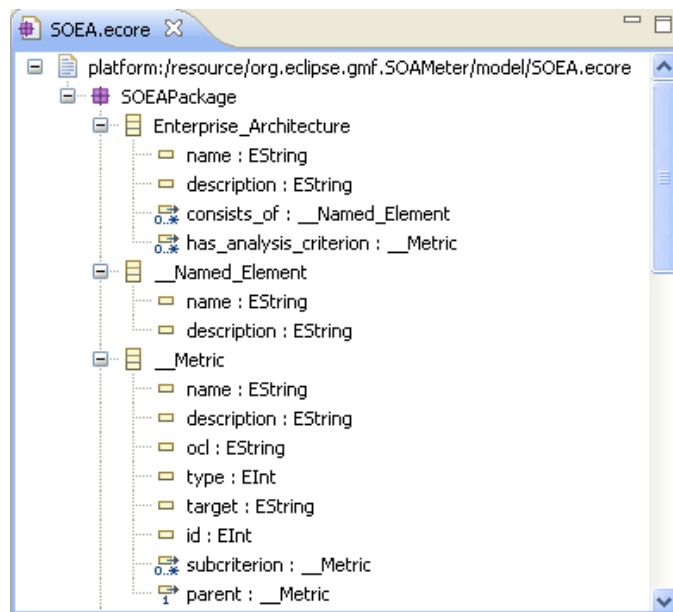


Fig. 8-4: Ecore model with basic elements

In the following, the meta model elements have to be added to this Ecore model. This can be done in a graphical representation of the Ecore model – the Ecore diagram. Changes in the diagram are automatically updated in the Ecore model and vice versa. The diagram is created by right clicking the SOEA.ecore file in the package explorer and choosing the option “Initialize Ecore Diagram File...”.

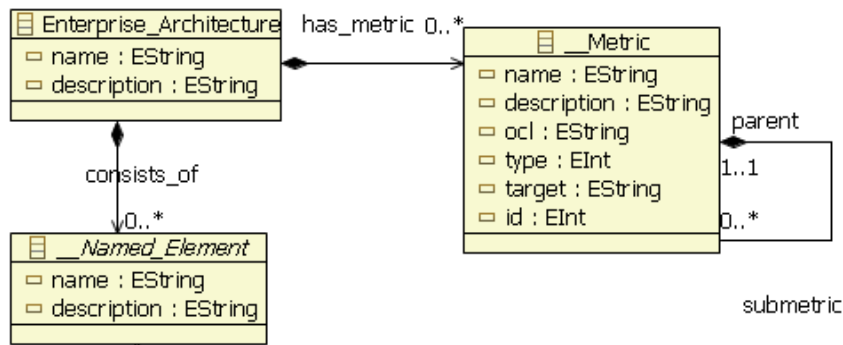


Fig. 8-5: Ecore model diagram with basic elements

All classes of the SOEA meta model have to be added as subclass of “__Named_Element”. There is a noteworthy about the bidirectional associations (or references) in the Ecore model. To create a bidirectional reference, two directed edges have to be created and their attribute “EOpposite” has to be set to the edge in the other direction. When all elements from the individual SOEA meta model have been added, the next dashboard step can be taken. The Ecore model will be transformed to the domain gen model. This model is used for generating code from the Ecore model. After the wizard dialog, the genmodel will be created.

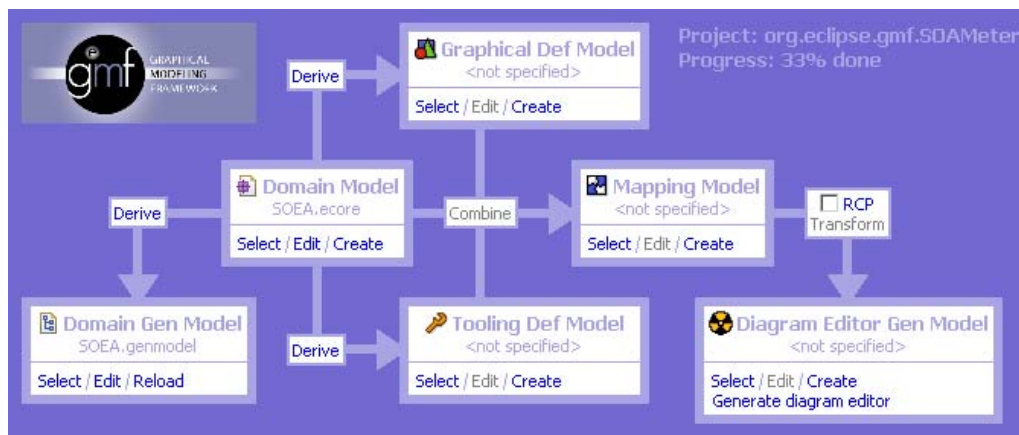


Fig. 8-6: GMF dashboard after second step

The editor with tree syntax can be generated by right clicking the genmodel package and selecting “Generate all”. Three new projects are generated. Now, the new project with the extension .editor has to be run in an eclipse application. When the new eclipse has started a new project via File→New→Other→Example EMF Model

Creation Wizards→SOEAPackage has to be created. For the use of the editor, refer to the EMF documentation (compare [Eclips03]). Fig. 8-7 shows the tree syntax editor with some sample objects.

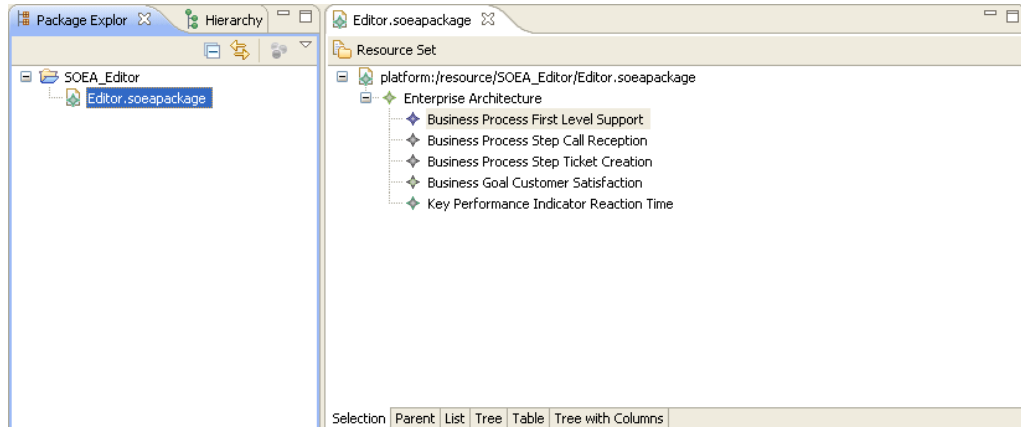


Fig. 8-7: Tree syntax editor generated from “Domain Gen Model”

The steps taken so far have resulted in a tree syntax editor for SOEA models. In the following, a more comfortable graphical editor will be created with the help of the GMF.

8.2 Creation of a Graphical Editor

This section proceeds with the tool creation for the SOEA meta model. The tree syntax editor is not very user friendly, because it is quite complicated to create bigger models and following a path of references is tedious. For these reasons, a simple graphical editor will be created.

The remaining steps from the GMF dashboard are required to create the graphical editor. Therefore, the derivation of the “Graphical Def Model” is processed now. The Graphical Def Model determines the graphical representation of the Ecore model elements. The second dialogue window in the “Graphical Def Model” wizard demands to determine a domain element. In this case, it has to be “Enterprise Architecture”. The third and last dialog window allows specifying elements that should not be represented in a graphical way. Everything except of “__Named_Element”, “Enterprise_Architecture”, and “__Metric” has to be selected in this step. Afterwards, the eclipse workspace should look similar as in Fig. 8-8.

Model-Based Evaluation of Service-Oriented Enterprise Architectures

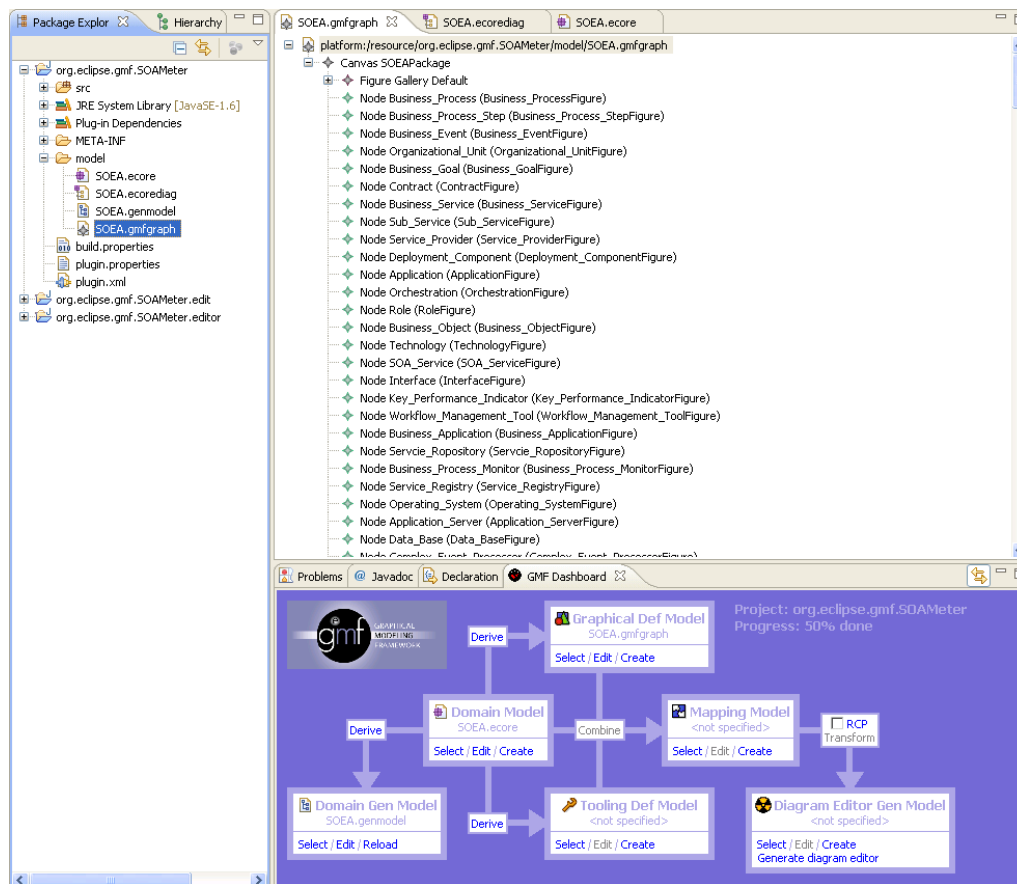


Fig. 8-8: Eclipse workspace with “Graphical Def Model”

The next dashboard step creates the “Tooling Def Model”. This model determines for which Ecore model elements creation tools will be generated. Creation tools are the pushbuttons (including functionality) in an editor, which create a new model element. Again, a wizard guides through the creation process, which is similar to the previous one. In the third dialogue window of the wizard, the elements for which a creation tool should be generated have to be selected. Everything except of “__Named_Element”, “Enterprise_Architecture”, and “__Metric” should be selected. The result is depicted in Fig. 8-9.

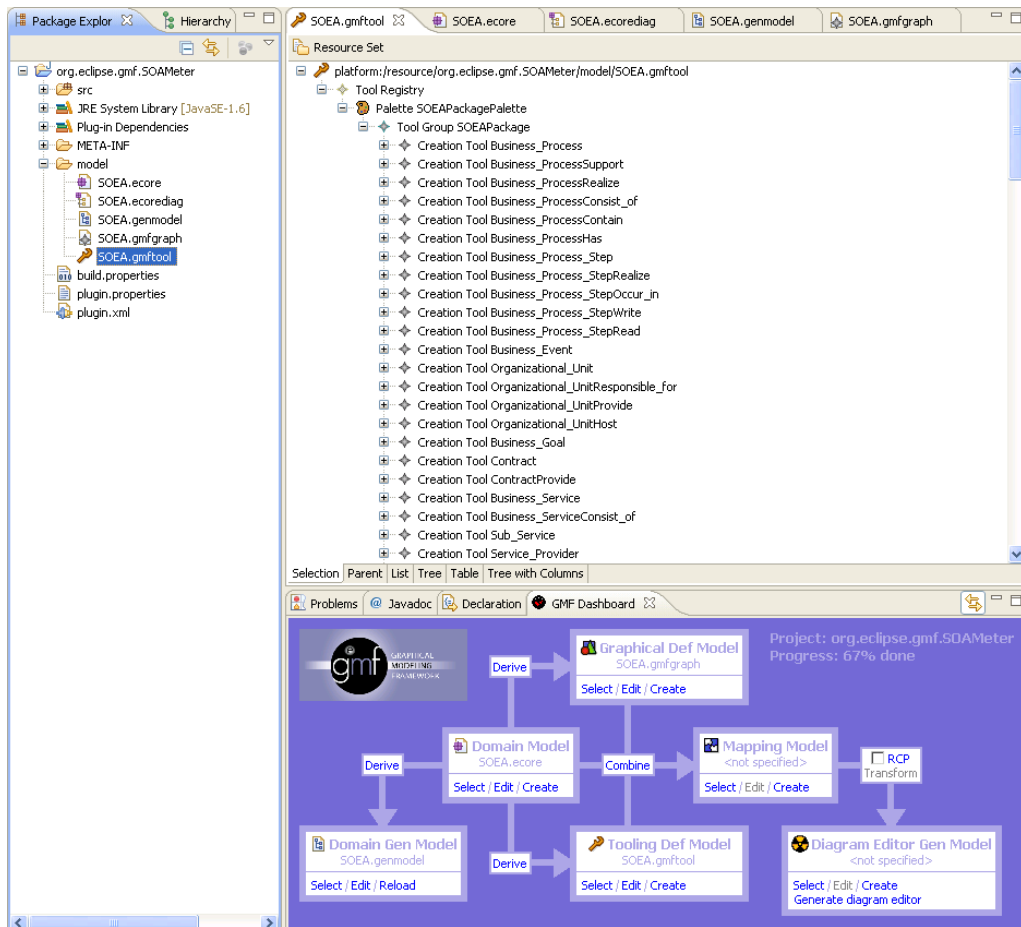


Fig. 8-9: Eclipse workspace with “Tooling Def Model”

The next step is the most complicated for the creation of the graphical editor. It combines the “Domain Model” with the “Graphical Def Model” and the “Tooling Def Model”. The resulting mapping model defines a mapping between a meta model element from the “Domain Model”, a display variant from the “Graphical Def Model” and a creation tool from the “Tooling Def Model”. The combine button from the dashboard calls a wizard dialogue. Firstly, the existing three models to be mapped have to be specified. Afterwards, the nodes and links for the editor have to be selected. The dialogue is given in Fig. 8-10. All nodes to appear in the graphical editor have to be selected there. As before, “Enterprise_Architecture”, “__Named_Element” and “__Metric” have to be left out.

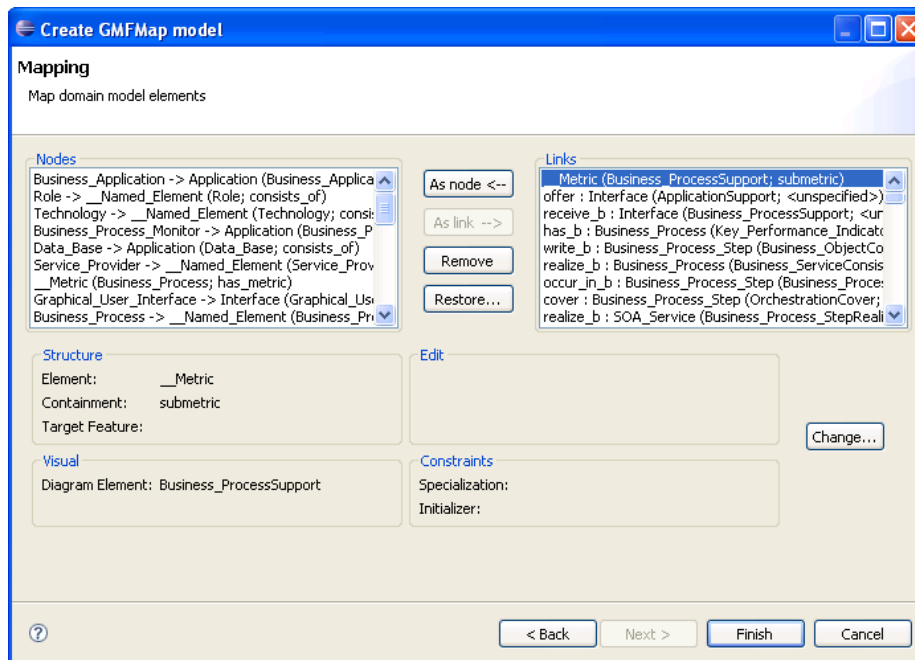


Fig. 8-10: Creation of the mapping model

Unfortunately, the labels for the nodes are not filled in an automated way. Therefore, the label mappings for each node have to be set. The label for a node will be filled with the name attribute of the corresponding node (compare Fig. 8-11).

In addition, it may happen that the creation tools and diagram links have been interchanged by the framework. For this reason, the properties of each “Node Link” have to be checked. The entries “Diagram Link” and “Tool” have to be examined on correctness. Both entries must match to the “Target feature” that can also be seen in the property window in Fig. 8-12.

Afterwards, the generation of the editor can be started by pressing the transform button on the GMF dashboard. This initiates the creation of the “Diagram Editor Gen Model”. The last step is to generate the editor code by clicking on the “Generate diagram editor” link in the dashboard. Starting the graphical editor is similar to the syntax tree editor. The code has to be run as new eclipse application by selecting File→New→Other→Examples→SOEA Diagram. A screenshot of the graphical editor is shown in Fig. 8-13. The graphical editor allows the generation and manipulation of instances of the SOEA meta model that was previously defined as “Domain Model” (Ecore Model). The editor prevents the manipulations of the model, which would result in non-conformance to the SOEA meta model.

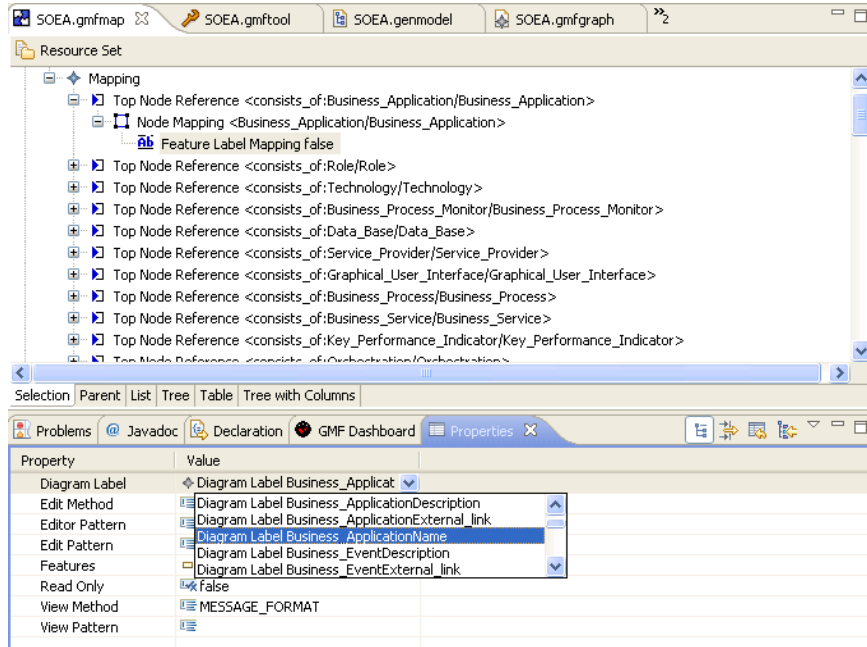


Fig. 8-11: Defining node labels

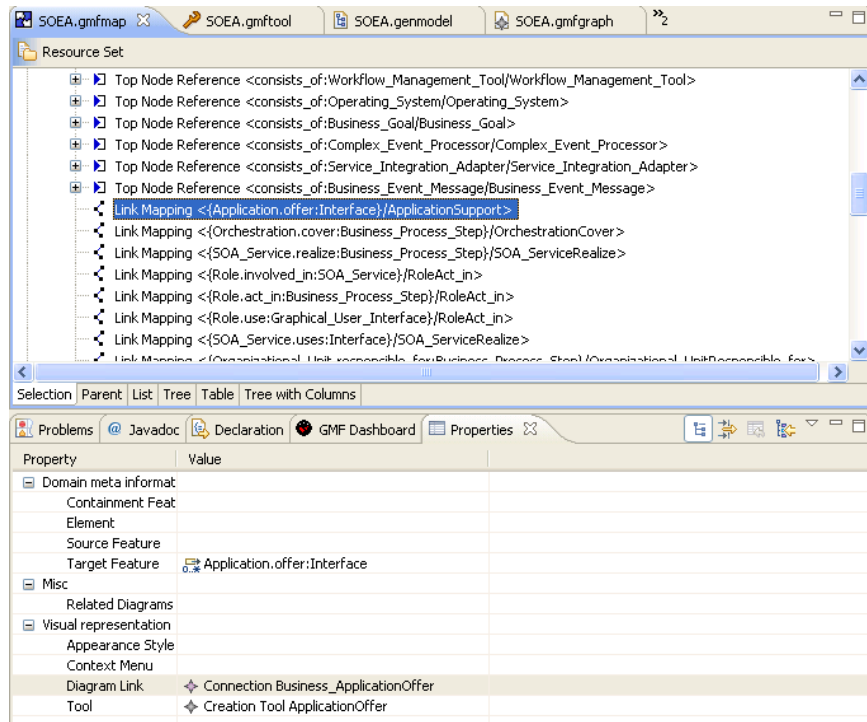


Fig. 8-12: Correcting interchanged mappings

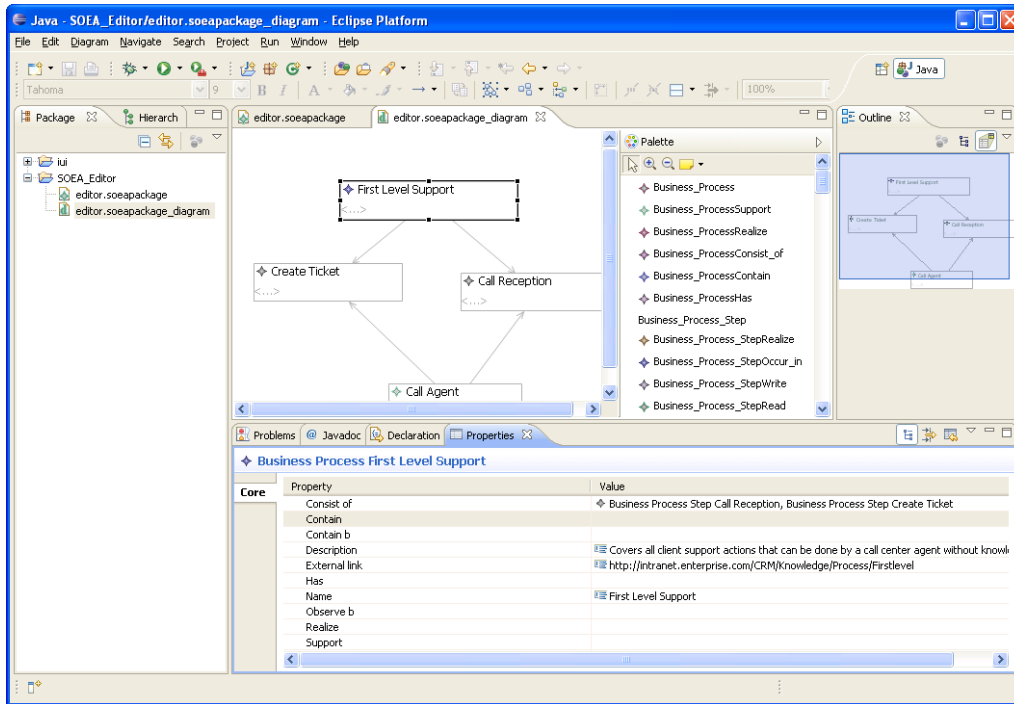


Fig. 8-13: Screenshot of the graphical editor

The creation of an editor for SOEA models with an individual SOEA meta model has been described so far. As next step, the editor will be extended by a plugin allowing formulating OCL constraints on the model.

8.3 Implementing Service Orientation Metrics

The metrics for service orientation that can be evaluated in an automated way will be implemented with the help of the Object Constraint Language (OCL, compare [OCLspe06]). OCL constraints are suited for this task because they allow checking the structure of graphs and executing simple calculations. This is sufficient for the metrics having their measuring points within the SOEA model.

The implementation is realized as an eclipse-plugin for the existing SOAMeter tool. This section will only describe the usage of the plugin. The source code creation is not of interest for the user because he is able to use the plugin with small adaptations. These adaptations consist of changing labels of meta classes within the OCL constraints. An OCL constraint is formulated on the level of the meta model. Changes in the meta model may occur if the merging process has changed the label of a meta

class, e.g. from “Orchestration” to “Service_Orchestration”. In this case, the OCL constraints have to be changed. How this can be done is explained later in this section.

The source code needs not to be changed as long as the basic Ecore elements from Fig. 8-4 remain the same. As they are used for implementation reasons and do not belong to SOEA meta model used in the editor, there should be no reason for changing them.

The plugin is reached via the entry “SOEA Metrics” in the menu bar (see Fig. 8-14). Only the options “Calculate measures” and “Manage metrics” are of interest here. At first, manage metrics will be examined.

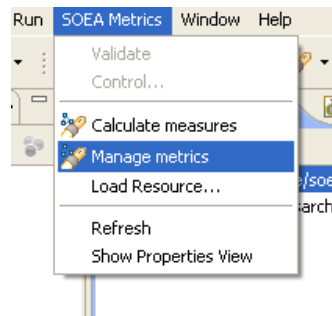


Fig. 8-14: Menu bar entry of the OCL plugin

The metric management window (see Fig. 8-15) shows an overview of the metrics that have already been created. In addition, it allows creating, changing, and deleting metrics.

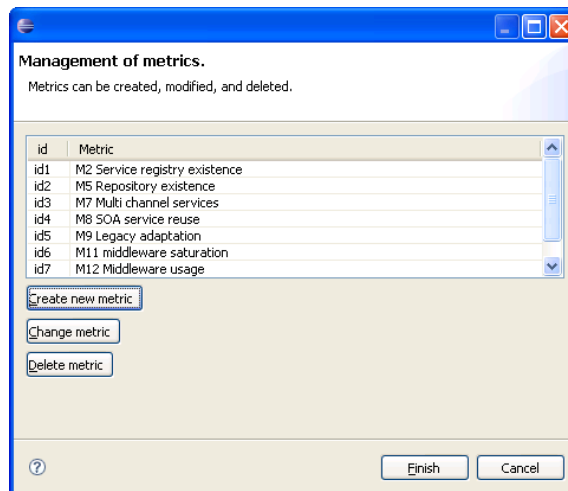


Fig. 8-15: Metric management window

Changing and creating a metric is nearly the same step. In the case of changing a metric, the metric creation window is filled with the data of the existing metric. Fig. 8-16 shows a metric creation window with the metric M2 Service registry existence. Next to a name and a description, a target value can be declared here. The target value can be used as the aim that is considered as ideal for this value.

The OCL constraint has to be typed in the respective field. The OCL specification [OCLspe06] describes how valid constraints have to be arranged. In addition, the result type has to be declared. If all mandatory fields have been filled, then the progress can be saved by clicking the “Finish” button.

Creation of a metric.
Please, fill out the following fields:

Name:
M2 Service registry existence

Metric type:
Absolut

OCL constraint
self.contains_of->
select(oclIsTypeOf(Service_Registry))->size()

Target value (optional)
1

Comprehensive description (optional)
Counts the number of occurrences of service registries.

Finish Cancel

Fig. 8-16: Metric creation window

Calculation of measures.
The following measures were calculated:

id	Metric	Result	Target value
id1	M2 Service registry existence	1	1
id2	M5 Repository existence	Error	1
id3	M7 Multi channel services	2.1	4
id4	M8 SOA service reuse	3	10
id5	M9 Legacy adaptation	75,00 %	1
id6	M11 middleware saturation	30,00 %	1
id7	M12 Middleware usage	40,00 %	1

Finish Cancel

Fig. 8-17: Measure calculation window

After having defined the metrics, their calculation can be executed by the eclipse plugin. The window in Fig. 8-17 shows the window that appears when selecting the entry “Calculate measures” from the menu bar. It shows the result values of the OCL constraints and their target values. If an entered OCL constraint is not well-formed, then the error is denoted in the result field.

Large parts of the report on service orientation can be filled with the information of the plugin because the objective measures are calculated with the OCL constraints. However, the remaining subjective measures have to be retrieved by experts and the indicators have to be applied afterwards.

This chapter has described how to implement a SOAMeter tool supporting the planning of enterprise architectures and supporting the automated calculation of metrics. The implementation is to be seen as a proof of concept and not as a tool ready for production. The chapter on the tool support finalizes the contribution of this thesis. Therefore, the conclusion is given in the next chapter.

9 Summary, Conclusion and Outlook

The model-based support for the transformation of an enterprise architecture to a Service-Oriented Enterprise Architecture is the topic of this thesis. The here presented SOAMeter approach tackling this task will be summarized in section 9.1. The novelty of the approach and the differences to existing approaches concerning the requirements stated in section 2.4 will be pointed out in the conclusion in section 9.2. Section 9.3 finalizes this thesis by providing an outlook that names the open challenges and the tasks for possible future work.

9.1 Summary

This section will give an overview on the contribution of this thesis. In one sentence, this thesis provides a model-based approach allowing enterprise architecture modelling paired with the evaluation of service orientation of the modelled enterprise architecture. In order to realize this, a definition of SOA including a meta model, a meta model merging algorithm leading to an SOEA meta model, an SOA quality criteria catalogue, as well as corresponding metrics and indicators have been elaborated. In the following, the interplay of these solution items is summarized and the value of the SOAMeter approach will be accentuated.

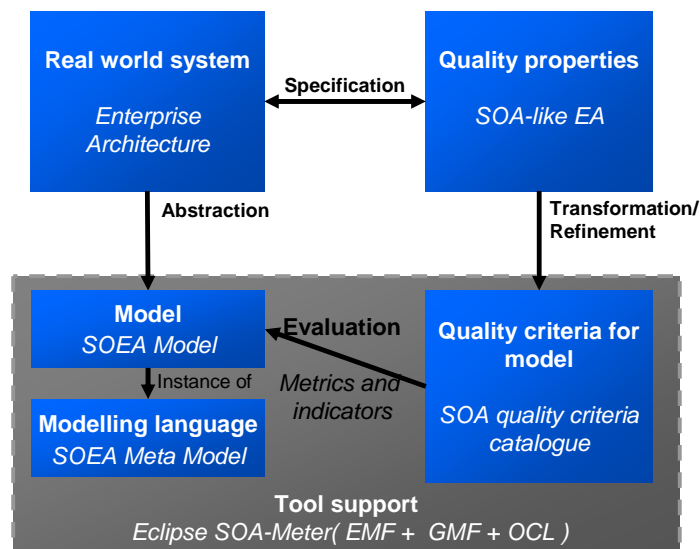


Fig. 9-1: Basic solution concept

The basic solution concept as depicted in Fig. 9-1 depicts the relation of the solution items. Quality properties in form of an SOA definition have been specified for the enterprise architecture being a real world system. The SOEA meta model and the corresponding SOA quality criteria catalogue were developed in order to be able to evaluate the quality properties of an existing EA. The SOEA meta model can express EAs that are service-oriented as well as those that are not service-oriented. An SOEA model that is not service-oriented yet, can be step-wise transformed to a service-oriented EA by applying changes that will raise the value of the evaluation results and therefore guide the transformation process.

The modelling of SOEAs and the evaluation of the SOA quality criteria has been prototypically implemented in an eclipse-based tool. This SOAMeter tool allows the creation of SOEA meta models and their instantiation. This feature is based on the EMF and GMF. Furthermore, the calculation of metrics is supported by an OCL plugin for this tool.

The elaborated approach allows an enterprise architect to model and plan the elements of the enterprise architecture. Furthermore, he can control the conformance of his EA to a fully-fledged SOA with the evaluation system of metrics and indicators. By this, the transformation of an enterprise architecture to an SOA-like EA is supported. Hence, the risk of failure when introducing an SOA is decreased.

9.2 Conclusion

In the conclusion, the approach from this thesis is compared with the existing similar approaches. For this reason, the requirements from section 2.4 are picked up again. At first, their fulfilment is elicited. Afterwards, the table overview shows the rating of other approaches in direct comparison with this approach.

- **R1 SOA definition**

This thesis provides a comprehensive SOA definition that includes a reference architecture for SOA and a service definition. The reference architecture comprises six main concepts of service-orientation. These are middleware, IT-business alignment, disclosure of functionality, orchestration, complex event processing, and business process monitoring. R1 is fulfilled.

- **R2 SOA formalization**

A meta model for Service-Oriented Architectures has been derived from the SOA definition of this thesis. R2 is fulfilled.

- **R3 SOA conformance criteria**

The SOA conformance criteria are formulated in the form of the SOA quality criteria catalogue. The catalogue concerns structural criteria (concerning the architecture) and service criteria (concerning the quality of SOA services). R3 is fulfilled.

- **R4 Integrated language for EA and SOA**

The SOEA meta model is the integrated modelling language for EA and SOA. It has been derived by merging the SOA meta model and an individual EA meta model. R4 is fulfilled.

- **R5 EA Formalization**

The exemplary derivation of an EA meta model has been shown. Furthermore, the defined minimal EA meta model serves as a lead for individual EA meta models. R5 is fulfilled.

- **R6 Individual EA meta model**

Any individual EA meta model that contains the minimal EA meta model can be used, because the meta model merging method from section 5.3 has been provided. It allows the creation of an integrated modelling language for SOEAs. R6 is fulfilled.

- **R7 As-Is & Target**

The as-is and target modelling is realized by the checking of the conformance criteria. That means that there is only one model at a time. An extra target model with the completely SOA-like EA (perfect in the sense of service-orientation) is not needed because all changes that lead to the fulfilment of the conformance criteria will lead the EA model towards the 'perfect' SOA-like EA. An extra model with the ideal architecture vision could be helpful, but has not been implemented. For this reason, R7 is only partly fulfilled.

- **R8 Holistic EA Modelling**

Firstly, the EA meta model is connected. For this reason, consequences of changes can be foreseen easily. Secondly, depending on the individual EA meta model, the SOEA meta model encloses the business, service and application layer of an EA. R8 is fulfilled.

- **R9 Individual views**

The individual views are part of the tool support and were not covered, because the created tool serves as a prototype only. Creating different views is a technical challenge, not as a conceptual one. R9 is not fulfilled.

- **R10 Tool support**

The tool prototype for the creation of SOEA models and automated checking of quality metrics has been implemented. Furthermore, a description on how to create such a tool with respect to the individual EA meta model is provided. The functionality of the tool is still on a premature level. R10 is partly fulfilled.

- **R11 Automation of criteria checks**

A complete set of metrics has been created for quality criteria catalogue. As far as possible, the calculation of the metrics has been automated. The interpretation rules for metric results have been defined but are not implemented in the prototype. As the implementation belongs to the tool support requirement, R11 is still regarded as fulfilled.

- **R12 Improvement suggestions**

Only strategies for improvement suggestions have been formulated. The approach is not able to make improvement suggestions like change the technology of interface A from X to Y. For this reason, R12 is only partly fulfilled.

- **R13 Methodical approach**

From the enterprise architects view, the steps from creating an EA meta model to merging the meta models to building an individual tool prototype have been described in a methodical way. No method has been defined for the regular use and maintenance of the tool. For this reason, R13 is only partly fulfilled.

	Information System Arch.	Zachman Framework	SEI approach	TOGAF	Quasar Enterprise	SOA-Meter
R1	○	○	●	◐	●	●
R2	○	○	○	◐	◐	●
R3	○	○	●	○	○	●
R4	○	○	○	◐	●	●
R5	○	○	○	○	◐	●
R6	◐	◐	◐	○	○	●
R7	◐	◐	○	◐	●	◐
R8	◐	○	○	●	●	●
R9	◐	●	○	◐	◐	○
R10	○	◐	○	◐	◐	◐
R11	○	○	○	○	○	●
R12	○	○	◐	○	○	◐
R13	◐	◐	◐	●	●	◐

Fig. 9-2: Final table showing the fulfilment of requirements

The table overview in Fig. 9-2 shows the table from Fig. 2-31 extended by the column for the approach of this thesis. The suggested approach cannot fulfil all the demanded requirements. However, it clearly has advantages over the existing approaches. The most important point hereby is R11. The combination of a modelling approach and the automated checking of quality criteria has not been realized for the domain of service-oriented enterprise architectures. In addition, the strengths of the presented work lie in the SOA definition and its combination with enterprise architectures.

An unfulfilled requirement is the demand for individual views on the SOEA model. These are not implemented yet, but this could be done within the prototype tool. Furthermore, a second instance of the SOEA model as target model is not realized in the approach. Due to the conformance checks, this is dispensable but it would ease the SOEA planning further.

The tool support has only been developed to a prototype level. Therefore it lacks the support for displaying and managing indicators. In addition, the improvement suggestions could be further automated, but it is even hard to make a concept for such an approach.

Finally, the presented approach is not complete or perfect, but it provides clear improvements in comparison to the existing approaches. It also shows up a possible path for the further development of this and similar approaches. The possible improvements for this approach are elicited in the following and final section.

9.3 Outlook

The SOAMeter approach has been designed to support the transformation of an EA to an SOA-like EA. This task is related to many of the artefacts and processes within an enterprise. Hence, there is a plethora of possibilities to enhance the approach. The possible options are categorized in conceptual and tooling improvements.

Some conceptual improvements can be derived from the missed requirements. So, the missing explicit target model that embodies the vision of a completely SOA-like EA can extend the approach. With this target model, the improvement suggestions could be derived more easily. Comparing the concrete SOEA target model to the as-is SOEA model, change steps could be generated that transform the as-is model in the direction of the target model, while increasing the quality measures of the model.

The explicit target model would be one way to improve the improvement suggestion approach. Another way could be a catalogue of patterns. If a certain pattern within the SOEA model is recognized and certain quality measures are not fulfilled, then such a pattern could be suggested to improve the current model.

If a suitable SOA meta model arises to a standard, then it could be adapted as meta model for the SOAMeter. With high effort, the OASIS SOA reference architecture [OASISR08] could be reduced to an adequate granularity level and afterwards be extended by the missing concepts of business process monitoring and complex event processing. However, there is no official standard of an SOA meta model yet.

A justified point of criticism of the SOAMeter concerns the data import and data retrieval for the SOEA model. If there is an existing EA model in an enterprise, then there should be a possibility to transfer or steadily access the available data. Furthermore, the changes in the real world system have to be updated by hand in the SOAMeter tool. Means of automation for this process could save a lot of tedious and error-prone work.

The basic solution concept from Fig. 9-1 can be extended for further quality properties that concern an enterprise architecture. The new quality properties have to be transformed into quality criteria that can be evaluated with the help of metrics related the existing SOEA model. If necessary, the meta model has to be extended by concepts that are needed to evaluate the new quality criteria. An example for the extension could be the need for compliance to the ISO 27001 [ISO05]. The required information security management system has to plan, implement, check, and optimize the usage of current security technologies. The comprehensive usage of such technologies could be checked by an extension of the SOAMeter.

SOA has been described as the latest evolution step of enterprise architectures. This evolution will surely not stop at the level of the SOA described in this thesis. At the point of time, when a new trend becomes apparent, the SOAMeter should be extended by this new trend. Of course, this entails changes of the quality properties and criteria, the metrics and indicators, as well as the SOEA meta model.

The tool prototype can be improved with several features. First of all the missing views on the model were helpful for planning and impact analysis. In addition, an explicit target model and a difference function comparing the as-is and target model could be supported by the tool. Rather easy to implement is the display of indicator values. For now, only the metric values are supported by the SOAMeter tool. The realization of data import and retrieval will be more challenging as the implementation has to integrate several data sources within the enterprise. Finally yet importantly, automated improvement suggestions would increase the value of the tool.

This probably non-exhaustive list of possible improvements and extensions of the SOAMeter approach finalizes this thesis.

References

- [Apache03] Apache Software Foundation. *Apache Active MQ (Version 5.2.0)*, The Apache Software Foundation: Forest Hill, MD, USA (2003) <http://activemq.apache.org/> (accessed 12.04.09).
- [Assman08] Assmann, Martin; Engels, Gregor. *Transition to Service-Oriented Enterprise Architecture*. In Proceedings of the Second European Conference on Software Architecture 2008. Morrison, R., Balasubramaniam, D., Falkner, K., Eds., pp 346–349. Springer: Berlin, (2008).
- [Assmanb09] Assmann, Martin; Haack, Markus; Scheider, Hendrik; vom Hagen, Nico; Zacharias, Roger. *SOA Business case*. In *Transform IT: Optimale Geschäftsprozesse durch eine transformierende IT*. Frank Keuper, Kiumars Hamidian, Eric Verwaayen, Torsten Kalinowski, Eds. Gabler: Wiesbaden, Deutschland (2009).
- [Assmanc09] Martin Assmann; Gregor Engels; Thomas von der Maßen; Andreas Wübbeke. *Identifying Software Product Line Component Services*. In Proceedings of the 4th International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE. Stefan Jablonski and Leszek Maciaszek, Ed., pp 45–56. INSTICC Press: Setubal (2009).
- [Assmann08] Assmann, Martin; Engels, Gregor. *Service-Oriented Enterprise Architectures: Evolution of Concepts and Methods*. In Proceedings of the 12th International IEEE Enterprise Distributed Object Computing Conference, pp xxxiv--xlili. IEEE Computer Society: Washington, DC, USA (2008).
- [Banden09] Bandener, Nils. Diploma thesis: *Visual Interpreter and Debugger for Dynamic Models Based on the Eclipse Platform*. University of Paderborn (2009).
- [Baresi03] Baresi, Luciano; Heckel, Reiko; Thöne, Sebastian; Varro, Daniel. *Modeling and Validation of Service-Oriented Architectures: Application vs. Style*. In Proceedings of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2003), Helsinki (Finland), pp 68-77. ACM Press (2003).

-
- [Basili92] Basili, Victor R. *Software modelling and measurement: the Goal/Question/Metric paradigm*. College Park, MD, USA (1992).
- [Berner01] Berners-Lee, Tim; Hendler, James; Lassila, Ora. *The Semantic Web*. In *Scientific American*. http://www.sciam.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21 (accessed 10.02.08).
- [Bernst03] Bernstein, Philip. *Applying Model Management to Classical Meta Data Problems*. In *Online Proceedings of the First Biennial Conference on Innovative Data Systems Research*. Asilomar, CA, USA (2003)
- [Bianco07] Bianco, Phil; Kotermanski, Rick; Merson, Paulo. *Evaluating a Service-Oriented Architecture*. Carnegie Mellon - Software Engineering Institute (2007) <http://www.sei.cmu.edu/publications/documents/07.reports/07tr015.html> (accessed Mar 12, 2009).
- [Bieman94] Bieman, J. M.; Ott, L. M. *Measuring functional cohesion*. In *IEEE Transactions on Software Engineering*. 20 (8) 644–657, (1994)
- [Borona07] Boronat, Artur; José; Ramos, Isidro; Letelier, Patricio. *Formal Model Merging Applied to Class Diagram Integration*. In *Electron. Notes Theor. Comput. Sci. Jg. 166, S. 5–26* (2007).
- [Braun05] Braun, C.; Winter, R. *A Comprehensive Enterprise Architecture Metamodel and Its Implementation Using a Metamodelling Platform*. In *Enterprise Modelling and Information Systems Architectures (EMISA), Proceedings of the Workshop in Klagenfurt, October 24-25*. J. Desel, U. Frank, Eds. LNI, pp 64–79. Gesellschaft für Informatik (GI) (2005).
- [Butler07] Butler, John C.; Hubby, Ravi; Koethe, Manfred R.; Melo, Walcelio L. *EAML: A MOF-Based, Common Enterprise Architecture Meta-model*. OMG Whitepaper (2007) http://www.omg.org/news/whitepapers/isp_ea_paper.pdf (accessed May 20, 2009).
- [Capgem01] Capgemini Inc. *The Integrated Architecture Framework*. www.capgemini.com/iaf (accessed Feb 23, 2009).
- [CBDISA08] CBDI Service Architecture & Engineering. *CBDI SAE Meta Model for SOA version 2.0*. (2008) http://www.cbdiforum.com/public/meta_model_v2.php (accessed May 10, 2009).

- [Chidam94] Chidamber, S. R.; Kemerer, C. F. *A Metrics Suite for Object Oriented Design*. IEEE Transactions on Software Engineering, 20 (6) 476–493 (1994).
- [Christ01] Christensen, Erik; Curbera, Francisco; Meredith, Greg; Weerawarana, Sanjiva. *Web Service Definition Language (WSDL)* W3C (2001). <http://www.w3.org/TR/wsdl> (accessed Jun 12, 2008).
- [Decker07] Decker, Gero; Barros, Alistair. *Interaction Modelling Using BPMN*. In Business Process Management Workshops. Arthur H. M. ter Hofstede, Boualem Benatallah, Hye-Young Paik, Eds. Lecture Notes in Computer Science, pp 208-219. Springer (2007).
- [DernGe03] Dern, Gernot. *Management von IT-Architekturen: Leitlinien für die Ausrichtung, Planung und Gestaltung von Informationssystemen*. 2nd ed., Friedr. Vieweg & Sohn Verlag, Wiesbaden (2003).
- [Dmytro07] Dmytro Rud. *Granularitätsmetriken für serviceorientierte Architekturen*. Vortrag zum DASMA Metrik Kongress. Magdeburg (2007).
- [Doroci09] Dorociak, Rafal. Masters thesis: *Erstellung und Evaluierung von serviceorientierten Unternehmensarchitekturmodellen*. University of Paderborn, Germany (2009).
- [Dostal05] Dostal, Wolfgang. *Service-orientierte Architekturen mit Web Services: Konzepte - Standards – Praxis*. 1st ed., Elsevier Spektrum Akad. Verlag, München (2005).
- [Eclips01] Eclipse Foundation. *The Eclipse Platform*. <http://www.eclipse.org/> (accessed Sep 12, 2009).
- [Eclips03] The Eclipse Foundation. *The Eclipse Modeling Framework Project*. <http://www.eclipse.org/modeling/emf/?project=emf> (accessed Sep 12, 2009).
- [Eclips06] Eclipse Foundation. *The Eclipse Graphical Modeling Framework*. <http://www.eclipse.org/modeling/gmf/> (accessed Sep 12, 2009).
- [Eclips07] Eclipse Foundation. *The Eclipse Modeling Development Tools*. <http://www.eclipse.org/modeling/mdt/?project=ocl> (accessed Sep 12, 2009).
- [Engels08] Engels, Gregor; Hess, Andreas; Humm, Bernhard; Juwig, Oliver; Lohmann, Marc; Richter, Jan-Peter; Voß, Markus; Willkomm,

-
- Johannes. *Quasar enterprise: Anwendungslandschaften serviceorientiert gestalten*. 1st ed., dpunkt.verlag, Heidelberg (2008).
- [ErlTho06] Erl, Thomas. *Service-Oriented Architecture: Concepts, technology, and design*. 6th ed., Prentice-Hall, Upper Saddle River, NJ (2006).
- [ErlTh09] Erl, Thomas. *SOA design pattern*. 1st ed., Prentice Hall, Upper Saddle River, NJ (2009).
- [Frohn09] Frohnhoff, Stephan. *Dissertation: Use Case Points 3.0*. University of Paderborn, Germany (2009).
- [Goethe96] Goethert, Wolfhart B.; Park, Robert E.; Florac, William A. *Goal-Driven Software Measurement - A Guidebook*. CMU/SEI-96-HB-002, Carnegie Mellon - Software Engineering Institute (1996).
- [Hafner08] Hafner, Martin; Winter, Robert. *Processes for Enterprise Application Architecture Management*. In HICSS '08: Proceedings of the Proceedings of the 41st Annual Hawaii International Conference on System Sciences, p 396. IEEE Computer Society: Washington, DC, USA (2008).
- [Hausma05] Hausmann, Jan Hendrik. *Dynamic Meta Modeling: A Semantics Description Technique for Visual Modeling Languages*. PhD Thesis, University of Paderborn (2005).
- [Haren07] Haren, Vann. *TOGAF 2007 Edition: The Open Group Architecture Framework (Incorporating 8.1.1)*. 8th ed.; TOGAF series, Van Haren Publishing, Zaltbommel (2007).
- [HarenV09] Haren, Vann. *TOGAF Version 9: A Manual*. 9th ed.; TOGAF series, Van Haren Publishing, Zaltbommel (2009).
- [HessHu07] Hess, Andreas; Humm, Bernhard; Voss, Markus; Engels, Gregor. *Structuring Software Cities A Multidimensional Approach*. In EDOC '07: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference, pp 122–129. IEEE Computer Society: Washington, DC, USA (2007).
- [Hofmei08] Hofmeister, Helge; Wirtz, Guido. *Supporting Service-Oriented Design with Metrics*. In EDOC '08: Proceedings of the 2008 12th International IEEE Enterprise Distributed Object Computing Conference, pp 191-200. IEEE Computer Society: Washington, DC, USA (2008).

- [HuhnsM05] Huhns, Michael N.; Singh, Munindar P. *Service-Oriented Computing: Key Concepts and Principles*. IEEE Internet Computing, 9 (1) 75–81 . (2005).
- [IBMGov07] IBM. *SOA Governance and Service Lifecycle Management*. <http://www-01.ibm.com/software/solutions/soa/gov/lifecycle/> (accessed Jul 22, 2009).
- [IBMSOA06] IBM. *Effective SOA governance* (2006) <ftp://ftp.software.ibm.com/software/rational/web/whitepapers/soagov-mgmt.pdf> (accessed Jul 22, 2009).
- [ISO99] The International Standards Organization. *Industrial automation systems - Requirements for enterprise-reference architectures and methodologies*. (1999). <http://www.mel.nist.gov/sc5wg1/gerastd/15704fds.htm> (accessed 04.04.09).
- [ISO05] The International Standards Organization. *The ISO 27001 Information Security Management System Specification*. (2005) <http://www.standards.bz/iso-27001.html> (accessed 02.10.09).
- [Jensen01] Jensen, Finn V. *Bayesian networks and decision graphs*. 1st ed., Springer, New York, NY (2001).
- [Josutt08] Josuttis, Nicolai. *SOA in der Praxis: System-Design für verteilte Geschäftsprozesse*. 1st ed., dpunkt.verlag, Heidelberg (2008).
- [KaibMi04] Kaib, Michael. *Enterprise Application Integration: Grundlagen, Integrationsprodukte, Anwendungsbeispiele*. 1st ed., Deutscher Universitäts-Verlag, Wiesbaden (2004).
- [Kaisle05] Kaisler, Stephen H.; Armour, Frank; Valivullah, Michael. *Enterprise Architecting: Critical Problems*. Hawaii International Conference on System Sciences (2005).
- [Kazman00] Kazman, Rick; Klein, Mark; Clements, Paul. *ATAM: Method for Architecture Evaluation*. CMU/SEI (2000) <http://www.sei.cmu.edu/publications/documents/00.reports/00tr004.html> (accessed Mar 12, 2009).
- [Krafzi06] Krafzig, Dirk; Banke, Karl; Slama, Dirk. *Enterprise SOA: Service-Oriented Architecture best practices*. 6th ed., Prentice-Hall, Upper Saddle River, NJ (2006).

-
- [Krcmar05] Krcmar, Helmut. *Informationsmanagement*, 4th ed., Springer, Berlin (2005).
- [Lagers08] Lagerstrom, Robert; Chenine, Moustafa; Johnson, Pontus; Franke, Ulrik. *Probabilistic Metamodel Merging*. In CAiSE Forum. Zohra Bellahsène, Carson Woo, Ela Hunt, Xavier Franch, Remi Coletta, Eds. CEUR Workshop Proceedings, pp 25–28. CEUR-WS (2008).
- [Lankes05] Lankes, Josef; Matthes, Florian; Wittenburg, Andre. *Softwarekartographie: Systematische Darstellung von Anwendungslandschaften*. In Wirtschaftsinformatik 2005. eEconomy, eGovernment, eSociety: 7. Internationale Tagung Wirtschaftsinformatik (WI 2005) Bamberg (23.-25.02.2005). Ferstl, O. K., Sinz, E. J., Eckert, S., Eds., pp 1443-1462. Physica-Verl.: Heidelberg (2005).
- [Madhav02] Madhavan, Jayant; Doan, AnHai; Domingos, Pedro; Halevy, Alon. *Learning to map between ontologies on the semantic web*. In WWW '02: Proceedings of the 11th international conference on World Wide Web, pp 662-673. ACM: New York, NY, USA (2002).
- [MDAspe03] OMG, editor. *Model Driven Architecture Specification*. Object Management Group Inc. Framingham, MA (2003) <http://www.omg.org/mda/specs.htm> (accessed 23.05.08)
- [Masak05] Masak, Dieter. *Moderne Enterprise Architekturen*. Springer-Verlag, Berlin (2005).
- [Matthe08] Florian Matthes, Sabine Buckl Alexander M. Ernst Josef Lankes. *Enterprise Architecture Management Pattern Catalog*. TU München. sebis: München (2008).
- [Minoli08] Minoli, Daniel. *Enterprise architecture A to Z: Frameworks, business process modelling, SOA, and infrastructure technology*. CRC Press/Taylor & Francis, Boca Raton, Fla. (2008).
- [MyersG73] Myers, G. J. *Composite Design: The Design of Modular Programs*. IBM. Poughkeepsie, New York (1973).
- [Nieman05] Niemann, Klaus D. *Von der Unternehmensarchitektur zur IT-Governance: Bausteine für ein wirksames IT-Management*. 1. Aufl.; Edition CIO, Vieweg, Wiesbaden (2005).

- [ISOIEC01] International Organization for Standardization (ISO) / International Electrotechnical Commission. *ISO/IEC Standard No. 9126: Software engineering – Product quality; Parts 1–4*. Geneva, Switzerland (2001).
- [O’Brien05] O’Brien, Liam; Bass, Len; Merson, Paulo. *Quality Attributes and Service-Oriented Architectures*. Software Engineering Institute of Carnegie Mellon University (2005).
- [Merson07] O’Brien, Liam; Merson, Paulo; Bass, Len. *Quality Attributes for Service-Oriented Architectures*. In *SDSOA ’07: Proceedings of the International Workshop on Systems Development in SOA Environments*, p 3. IEEE Computer Society: Washington, DC, USA (2007).
- [OASIS05] OASIS. *OASIS Web Services Business Process Execution Language (WSBPEL) TC*. (2005) http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel (accessed Mar 26, 2009).
- [OASISR06] OASIS. *Reference Model for Service Oriented Architecture v1.0*. (2006) <http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.html> (accessed Apr 05, 2006).
- [OASISP05] Organization for the Advancement of Structured Information Standards (OASIS). *OASIS WS-BPEL Extension for People (BPEL4People) Version 1.0*. (2005) http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=bpel4people (accessed Apr 01, 2009).
- [OCLspe06] OMG, editor. *Object Constraint Language Specification - Version 2.0*. Object Management Group Inc. Framingham, MA (2006) <http://www.omg.org/docs/formal/06-05-01.pdf> (accessed Apr 28, 2009).
- [OMGCOR92] OMG, editor. *CORBA Specification Catalog (Version 1.0)*. Object Management Group Inc.: Framingham, MA (1992) http://www.omg.org/technology/documents/corba_spec_catalog.htm (accessed 17.06.09).
- [RielAr05] Riel, Arthur J. *Object-oriented design heuristics*. 11th ed.; Addison-Wesley: Reading, Mass. (2005)
- [Santan03] Sant’anna, Claudio; Garcia, Alessandro; Chavez, Christina; Lucena, Carlos; Staa, Arndt von v. *On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework*. In *Proceedings of the XVII Brazilian Symposium on Software Engineering*. (2003).

-
- [Scheer98] Scheer, August-Wilhelm. *ARIS: Modellierungsmethoden, Metamodelle, Anwendungen*. 3rd ed., Springer, Berlin (1998).
- [Scheerb02] Scheer, August-Wilhelm. *ARIS: Vom Geschäftsprozess zum Anwendungssystem*. 4th ed., Springer, Berlin (2002).
- [Schelp07] Schelp, Joachim; Stutz, Matthias. *SOA-Governance*. HMD - Praxis Wirtschaftsinformatik, 253 (2007).
- [Sieder07] Siedersleben, Johannes. *SOA revisited. Komponentenorientierung bei Systemlandschaften*. Wirtschaftsinformatik. (49) 110–117 (2007).
- [Sommer01] Sommerville, Ian. *Software Engineering*. 6th ed.; Pearson Studium: Munich, (2003).
- [Steven74] Stevens, Wayne P.; Myers, Glenford J.; Constantine, Larry L. *Structured Design*. IBM Systems Journal. 13 (2) 115–139 (1974).
- [SunMic00] Sun Microsystems. *Java Messaging Service*. <http://java.sun.com/products/jms/> (accessed Apr 08, 2008).
- [SunMi03] Sun Microsystems. *Java RMI over IIOP. Technology Documentation Home Page*. <http://java.sun.com/j2se/1.4.2/docs/guide/rmi-iiop/> (accessed 08.04.08).
- [UecanE08] Esref Ücan. Diploma thesis: *Serviceidentifikation anhand erweiterter Geschäftsprozessmodelle*. University of Paderborn, Germany (2008).
- [Uschol95] Uschold, Mike; King, Martin; Moralee, Stuart; Zorgios, Yannis. *The Enterprise Ontology. The Knowledge Engineering Review*. 13 (1) pp. 31–89 (1995).
- [Vascon07] Vasconcelos, A.; Sousa, P.; Tribolet, J. *Information System Architecture Metrics: An Enterprise Engineering Evaluation Approach*. The Electronic Journal Information Systems Evaluation, 10 (1) 91–122 (2007).
- [Vinosk05] Vinoski, Steve. *Old Measures for New Services*. IEEE Internet Computing, 9 (6) 72–74 (2005).
- [VoigtH09] Voigt, Hendrik. *Kontextsensitive Qualitätsplanung von Softwaremodellen. Dissertation*. University of Paderborn (2009), to be published.
- [VLDBOR09] Very Large Data Base Endowment Inc. *VLDB.org*. <http://www.vldb.org/> (last accessed 04.02.2009).

- [Weiser91] Weiser, Mark. *Program slicing*. In ICSE '81: Proceedings of the 5th international conference on Software engineering, pp 439–449. IEEE Press: Piscataway, NJ, USA (1981).
- [WFMCXP05] WFMC. *Workflow Management Coalition Workflow Standard: Process Definition Interface - XML Process Definition Language*. Workflow Management Coalition (2005) http://www.wfmc.org/standards/docs/TC-1025_xpdl_2_2005-10-03.pdf. (accessed 17.07.09)
- [Winter08] Winter, Robert; Aier, Stephan; Scholl, Ulrich; Discher, Stefan. *Economic Justification of Service-Oriented Architecture*. SAP AG and University of St. Gallen (2008) <http://www.sap.com/community/showdetail.epx?ItemID=16041> (accessed Mar 25, 2009).
- [WoodsD06] Woods, Dan; Mattern, Thomas. *Enterprise SOA designing IT for business innovation*. 1st ed., O'Reilly, Beijing (2006).
- [Zachma87] Zachman, John. *A framework for information systems architecture*. IBM Systems Journal, 26 (3) 276-292 (1987).
- [Zelevs99] Zelewski, Stephan. *Ontologien zur Strukturierung von Domänenwissen*. University of Essen (1999) <http://www.pim.uni-essen.de/mitarbeiter/person.cfm?name=pimstze>, (accessed 12.06.2008).

Appendix A

In this part of the appendix, the application of the merging algorithm described in chapter 5 shall be demonstrated in more detail. Therefore, the table representations of the meta models are given and some chosen options are documented.

The meta models were merged with the knowledge of an expert. That means the “Semantic Description” column was used to determine artefacts that are regarded as equal by the expert. For example, “Business Process Step” from the SOA meta model was set equal with the “Process Step” from the EA meta model. To do so the semantic description was set to an exact equal string for both meta models. This has also been done for “Named Element”/“Named Element”, “Performance Indicator”/“Key Performance Indicator”, “Business Process”/ “Business Process”, “Interface”/“Interface”, “Application”/ “Application”, “Role”/“Role and “Technology”/“Technology”. This decreases the effort for applying the algorithm drastically.

Artefact	Association Name	Cardinality	Referenced Artefact	Semantic Description
Application				Application
Application	has specialization	0..*	Business Process Monitor	
Application	has specialization	0..*	Orchestration Engine	
Application	provide	0..*	Interface	
Application	has specialization	0..*	Service Repository	
Application	canFire	0..*	Event	
Application	has specialization	0..*	Service Registry	
Application	canReceive	0..*	Event	
Application	has specialization	0..*	Complex Event Processor	
Application	realize	0..*	Business Process Step	
Application	require	0..*	Interface	
Application	ImplementedIn	0..*	Technology	

Business Object			
Business Object	hasRead Access (b)	0..*	Interface
Business Object	hasWrite Access (b)	0..*	Interface
Business Process			Business Process
Business Process	consistsOf	1..*	Business Process Step
Business Process	have	0..*	Performance Indicator
Business Process	observe	0..*	Business Process
Business Process Monitor			
Business Process Monitor	has generalization	1	Application
Business Process Monitor	observe	0..*	Business Process
Business Process Step			Process Step
Business Process Step	consistsOf (b)	0..*	Business Process
Business Process Step	realize (b)	0..*	Orchestration
Business Process Step	realize (b)	0..*	Role
Business Process Step	realize (b)	0..*	SOA Service
Business Process Step	realize (b)	0..*	Application
Complex Event Processor			
Complex Event Processor	has generalization	1	Application
Event			
Event	require (b)	0..*	Performance Indicator
Event	canFire (b)	0..*	Application
Event	canReceive (b)	0..*	Application

Model-Based Evaluation of Service-Oriented Enterprise Architectures

Event	canFire	0..*	SOA Service	
GUI				
GUI	use (b)	0..*	Role	
GUI	has generalization	1	Interface	
Interface				Interface
Interface	provide (b)	0..*	SOA Service	
Interface	provide (b)	0..*	Application	
Interface	has specialization	0..*	Service Interface	
Interface	hasRead Access	0..*	Business Object	
Interface	hasWrite Access	0..*	Business Object	
Interface	implemntedIn	0..*	Technology	
Interface	require (b)	0..*	SOA Service	
Interface	has specialization	0..*	Service Integration Adapter	
Interface	has specialization	0..*	GUI	
Interface	require (b)	0..*	Application	
Named Element.Name	is attribute of	0..*	Named Element	
Named Element.Description	is attribute of	0..*	Named Element	
Named Element				Named Element
Named Element	has attribute	0..1	Name	
Named Element	has attribute	0..1	Description	
Orchestration				
Orchestration	realize	0..*	Business Process Step	

Orchestration	use	0..*	SOA Service	
Orchestration	canExecute	0..*	Orchestration	
Orchestration Engine				
Orchestration Engine	has generalization	1	Application	
Orchestration Engine	canExecute	0..*	Orchestration	
Performance Indicator				Key Performance Indicator
Performance Indicator	have (b)	1	Business Process	
Performance Indicator	require	0..*	Event	
Role				Role
Role	involved in	0..*	SOA Service	
Role	realize	0..*	Business Process Step	
Role	use	0..*	GUI	
SOA Service				
SOA Service	provide	0..*	Interface	
SOA Service	involved in (b)	0..*	Role	
SOA Service	require	0..*	Interface	
SOA Service	isRegisteredIn	0..*	Service Registry	
SOA Service	canFire (b)	0..*	Event	
SOA Service	realize	0..*	Business Process Step	
SOA Service	use (b)	0..*	Orchestration	
Service Integration Adapter				
Service Integration Adapter	has generalization	1	Interface	
Service Interface				
Service Interface	has	1	Interface	

Model-Based Evaluation of Service-Oriented Enterprise Architectures

generalization			
Service Registry	Aggregation	0..*	Service Repository
Service Registry	has generalization	1	Application
Service Registry	isRegisteredIn (b)	0..*	SOA Service
Service Repository	has generalization	1	Application
Service Repository	Aggregation (b)	0..*	Service Registry
Technology	implementsIn (b)	0..*	Interface
Technology	ImplementedIn (b)	0..*	Application
Technology	uses (b)	0..*	Interface

Table representation of the SOA meta model

Artefact	Association Name	Cardinality	Referenced Artefact	Semantic Description
Application	Aggregation	0..*	Deployment Component	Application
Application	support	0..*	Business Process Step	
Application	has specialization	0..*	Application Server	
Application	has specialization	0..*	Business Application	
Application	host (b)	1	0..Organizational Unit	
Application	has specialization	0..*	Data Base	
Application	has specialization	0..*	Operating System	
Application	implement	0..*	Technology	
Application	require	0..*	Interface	
Application	offer	0..*	Interface	

Application	has specialization	0..*	Workflow Management Tool	
Application Server				Application Server
Application Server	has generalization	1	Application	
Business Application				Business Application
Business Application	has generalization	1	Application	
Business Event				Business Event
Business Event	occurIn	0..*	Business Process Step	
Business Event Message				
Business Event Message	need (b)	0..*	Key Performance Indicator	
Business Event Message	can create (b)	0..*	Interface	
Business Goal				
Business Goal	support (b)	0..*	Business Process	
Business Object				
Business Object	hasWriteAccess (b)	0..*	Business Process Step	
Business Object	Aggregation	1	0..Business Object	
Business Object	hasReadAccess (b)	0..*	Business Process Step	
Business Process				Business Process
Business Process	support	0..*	Business Goal	
Business Process	realize	0..*	Business Service	
Business Process	has generalization	1	Business Process	
Business Process	consistOf	1..*	Business Process Step	
Business Process	have	0..*	Key Performance Indicator	

Model-Based Evaluation of Service-Oriented Enterprise Architectures

Business Process	Aggregation	1	0..Business Process
Business Process Step			Process Step
Business Process Step	hasWriteAccess	0..*	Business Object
Business Process Step	support (b)	0..*	Application
Business Process Step	occurIn (b)	0..*	Business Event
Business Process Step	realize	0..*	Sub Service
Business Process Step	hasReadAccess	0..*	Business Object
Business Process Step	responsible for (b)	1	Organizational Unit
Business Process Step	consistOf (b)	0..*	Business Process
Business Process Step	actIn (b)	0..*	Role
Business Service			
Business Service	consistsOf	0..*	Sub Service
Business Service	realize (b)	0..*	Business Process
Business Service	provide (b)	0..*	Contract
Contract			
Contract	provide	1..*	Business Service
Data Base			
Data Base	has generalization	1	Application
Deployment Component			
Deployment Component	Aggregation (b)	1..*	Application
Graphical User Interface			
Graphical User	has	1	Interface

Interface	generalization			
Interface				Interface
Interface	implement	0..*	Technology	
Interface	has specialization	0..*	Graphical User Interface	
Interface	can create	0..*	Business Event Message	
Interface	require (b)	0..*	Application	
Interface	offer (b)	0..*	Application	
Key Performance Indicator				Key Performance Indicator
Key Performance Indicator	need	0..*	Business Event Message	
Key Performance Indicator	has generalization	1	Key Performance Indicator	
Key Performance Indicator	have (b)	1	Business Process	
Named Element.Name	is attribute of	0..*	Named Element	
Named Element.Description	is attribute of	0..*	Named Element	
Named Element				Named Element
Named Element	has attribute	0..1	Name	
Named Element	has attribute	0..1	Description	
Operating System				
Operating System	has generalization	1	Application	
Organizational Unit				
Organizational Unit	responsible for	0..*	Business Process Step	
Organizational Unit	provide	0..*	Role	
Organizational Unit	host	0..*	Application	
Role				Role
Role	provide (b)	0..*	Organizational Unit	
Role	actIn	0..*	Business Process Step	

Model-Based Evaluation of Service-Oriented Enterprise Architectures

Service Provider				
Service Provider	deliver	1	0..Sub Service	
Sub Service				
Sub Service	deliver (b)	0..*	Service Provider	
Sub Service	consistsOf (b)	0..*	Business Service	
Sub Service	realize (b)	0..*	Business Process Step	
Technology				Technology
Technology	implement (b)	0..*	Interface	
Technology	implement (b)	0..*	Application	
Workflow Management Tool				
Workflow Management Tool	has generalization	1	Application	

Table representation of the EA meta model

In the following table the merged concepts and associations are listed. The boldly marked concepts were the dominating ones, so that their name was kept in the resulting SOEA meta model.

EA Meta Model	Association	Ref. Artefact	SOA Meta Model	Association	Ref. Artefact
Application	support	Business Process Step	Application	realize	Business Process Step
Application	implement	Technology	Application	Implemented In	Technology
Application	require	Interface	Application	require	Interface
Application	offer	Interface	Application	provide	Interface
Application	has specialization	Workflow Management Tool	Application	has specialization	Orchestration Engine
Business Object			Business Object		
Business Process			Business Process		
Business Process	consistOf	Business Process Step	Business Process	consistsOf	Business Process Step
Business Process	have	Key Performance Indicator	Business Process	have	Performance Indicator

Business Process Step			Business Process Step		
Graphical User Interface			GUI		
Graphical User Interface	has generalization	Interface	GUI	has generalization	Interface
Interface			Interface		
Interface	implement	Technology	Interface	implemented In	Technology
Key Performance Indicator			Performance Indicator		
Key Performance Indicator	need	Business Event Message	Performance Indicator	require	Event
Named Element			Named Element		
Named Element.Name	is attribute of	Named Element	Named Element.Name	is attribute of	Named Element
Named Element.Description	is attribute of	Named Element	Named Element.Description	is attribute of	Named Element
Role			Role		
Role	actIn	Business Process Step	Role	realize	Business Process Step
Workflow Management Tool			Orchestration Engine		
Workflow Management Tool	has generalization	Application	Orchestration Engine	has generalization	Application