A Pattern Language for Architecture Assessments of Service-oriented Enterprise Systems

Alfred Zimmermann
Faculty of Informatics
Software Architecture Research Group
Reutlingen University, Germany
alfred.zimmermann@reutlingen-university.de

Fritz Laux
Faculty of Informatics
Data Management Research Group
Reutlingen University, Germany
fritz.laux@reutlingen-university.de

René Reiners
Fraunhofer FIT
Schloss Birlinghoven
Sankt Augustin, Germany
rene.reiners@fit.fraunhofer.de

***Abstract* – The SOA Innovation Lab – an innovation and research network of industry leaders in Germany and Europe - investigates the practical use of service-oriented enterprise systems. Current state of practice approaches for assessing maturity of service-oriented enterprise software architectures were intuitively developed, having sparse metamodel or pattern foundation and being rarely validated. This is a real problem for practical architecture assessments in repeatable cyclic evaluations of base architectures of service-oriented systems, which are based on recurring patterns for analysis of continuously growing services over the time. In our research we have developed an original pattern language for supporting architecture assessments and optimization of enterprise systems, leveraging and extending base frameworks like the Capability Maturity Model Integration (CMMI) and The Open Group Architecture Framework (TOGAF), and considering additionally similar related work models of architecture maturity. We have deduced our original architecture quality assessment and optimization pattern language from our special designed architecture maturity framework. We have applied our architecture assessment pattern language in consecutive cyclic assessment workshops with global vendors of service-oriented platforms.**

Keywords – service-oriented architecture; architecture assessment; pattern language; metamodel; maturity framework.

I. INTRODUCTION

Innovation oriented companies have introduced service-oriented architectures (SOA) to assist in closing the business - IT gap. Our approach investigates the SOA ability of heterogeneous enterprise systems as in [1] and integrates elements from convergent architecture methods, technologies and related software patterns, as in [2], [3], and [6] with evaluation methods for service-oriented enterprise systems as in [4].

The hypothesis of our research is as follows:

1) The CMMI [5] is well known as a suitable framework to assess software processes; nevertheless the metamodel of CMMI can be extended to enable quality assessments of service-oriented software architectures.

2) The idea of software patterns could be applied consistently in architecture assessments for both capability assessments and improvements of service-oriented architectures for enterprise systems.

Software architecture assessment patterns are based on the seminal work of software patterns originated from the work of [7]. A pattern records solution decisions taken by many builders in many places over many years in order to resolve a particular problem. Patterns are human readable structures of text and graphics showing a standardized and repeatable way to derive a solution from a specified problem in a specific context. Patterns describe usually sophisticated and hidden solutions for given design problems. But we can also use patterns like testing patterns for the assessment of software architectures and other software artifacts. We call a collection of patterns, which are organized in a directed acyclic graph structure, a *pattern language*. Elements of a pattern language are navigable sequences of patterns.

Our original and validated pattern language approach for assessment patterns for quality assessments of enterprise software architectures relies on related work (Section II) and on a specific maturity framework (see Section III) for assessing architecture capabilities and maturity levels of service-oriented enterprise systems. We derived an associated architecture assessment pattern language from our previous work on an architecture maturity framework and our basic architecture quality pattern catalog, which was also previously developed. The new introduced architecture quality assessment language extends and sequences the basic architecture assessment patterns to 43 integrated patterns (Section IV). These patterns are used to identify quality indicators for different architectural aspects and specific structures of service-oriented software systems. Finally, we sketch main evaluation results (Section V) and draw conclusions and future directions (Section VI).

II. RELATED WORK

The Open Group Architecture Framework (TOGAF) [8] as the current standard for enterprise architecture provides the basic blueprint and structure for our enterprise software architecture domains of service-oriented enterprise systems like *Architecture Strategy and Management*, *Business Architecture*, *Information Architecture*, *Application Architecture*, *Technology Architecture*, *Service & Operation Architecture*, and *Architecture Realization*.

SOA is the computing paradigm that utilizes services as fundamental flexible and interoperable building blocks for both structuring the business and for developing applications. SOA promotes a business-oriented architecture style as promoted in [9] and [3]), based on best of breed technology of context agnostic business services that are delivered by applications in a business focused granularity. To provide dynamic composition of services within a worldwide environment SOA uses a set of XML-based standards. A main innovation introduced by SOA is that business processes are not only modeled, but services are executed from orchestrated services, too.

To transform CMMI into a specific framework for architecture assessments of service-oriented enterprise systems we have combined CMMI with current SOA frameworks and maturity models. We use TOGAF [8] and ideas related to the business and information architecture from [10] as a basic structure for enterprise architecture spanning all relevant levels of service-oriented enterprise systems.

The Architecture Capability Maturity Model (ACMM) [11] framework, which is included in TOGAF, was originally developed by the US Department of Commerce. The goal of ACMM assessments is to enhance enterprise architectures by identifying quantitative weak areas and to show an improvement path for the identified gaps of the assessed architecture. The ACMM framework consists of six maturity levels and nine specific architecture elements ranked for each maturity level - deviant from CMMI.

The SOA Maturity Model of Inaganti and Aravamudan [12] considers the following multidimensional aspects of a SOA: scope of SOA adoption, SOA maturity level to express architecture capabilities, SOA expansion stages, SOA return on investment, and SOA cost effectiveness and feasibility. The scope of SOA adoption in an enterprise is differentiated by the following levels: intra department or ad hoc adoption, inter departmental adoption on business unit level, cross business unit adoption, and the enterprise level, including the SOA adoption within the entire supply chain. The SOA maturity levels are related to CMMI, but used differently , using five ascending levels to express enhanced architectural capabilities: level 1 for initial services, level 2 for architected services, level 3 for business services, level 4 for measured business services, and level 5 for optimized business services.

The SOA Maturity Model from Sonic [13] distinguishes five maturity levels of a SOA, and associates them in analogy to a simplified metamodel of CMMI with key goals and key practices. Key goals and key practices are the reference points in the SOA maturity assessment.

The SOA Maturity Model of ORACLE [14] characterizes in a loose correlation with CMMI five different maturity levels: opportunistic, systematic, enterprise, measured, industrialized and associates them with strategic goals and tactical plans for implementing SOA. Additionally the following capabilities of a SOA are referenced with each maturity level: Infrastructure, Architecture, Information & Analytics, Operations, Project Execution, Finance & Portfolios, People & Organization, and Governance.

III. Assessment Framework

Our idea and contribution is to extend existing service-oriented architecture (SOA) maturity frameworks to accord with a sound metamodel approach. Our metamodel for architecture evaluation enlarges the standardized CMMI, which is originally used to assess the quality of software processes and not the quality of software architectures.

The aim of our SOAMMI – SOA Maturity Model Integration - framework [15] is to provide a holistic framework to assess architectures of service-oriented enterprise systems. We have analyzed and systematically integrated evaluation criteria, maturity domains, architecture capabilities, and level rankings from state of the art SOA maturity and evaluation models as described in [11], [13], and [14]. In addition we have adapted architecture assessment elements from [4] and [15], and extended singular architecture patterns from our previous work [16] to our new assessment pattern language (Section IV).

The SOAMMI architecture maturity framework introduces original architecture areas and organizes them within extended architecture domains, which are mainly based on TOGAF. Our intention was to leave most structural parts e.g. *Maturity Levels, Capability Levels, Specific Goals and Practices, Generic Goals and Practices* - of the original CMMI metamodel as untouched concepts. We extend these concepts of the metamodel by reclusively connected architecture patterns, as navigable architecture quality patterns of a pattern language, and enlarge these by other architecture specific structures and contents. The metamodel of SOAMMI in Figure 1 has similarities with the CMMI metamodel. We leaved the understanding of Maturity Levels and Capability Levels the same like in CMMI. Additionally we added the following concepts: Architecture Domain, Architecture Area, Architecture Pattern, and replaced all the contents of related Specific Goals, Specific Practices, and the Generic Practices, to fit for our architecture evaluation purpose. We used multiplicity indicators for class relations to add a basic metamodel semantic. Not indicated multiplicities corresponds to the default 1 cardinality or a 1..1 multiplicity.

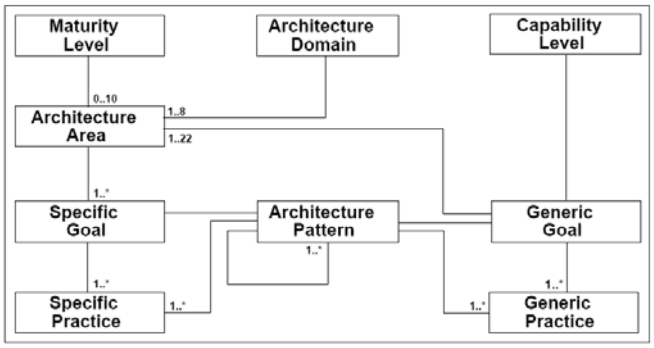

Figure 1. SOAMMI Metamodel – Main Concepts

In terms of requirements from customer oriented domain-models and reference use scenarios, our model has introduced in [15] the following maturity levels, which define architecture assessment criteria for service-oriented enterprise systems and help to measure the architecture maturity: The semantic of these maturity levels as in [15] was adapted from [5] to conform with the architecture assessment scope for service-oriented enterprise systems:

1. *Maturity Level: Initial Architecture,*
2. *Maturity Level: Managed Architecture,*
3. *Maturity Level: Defined Architecture,*
4. *Maturity Level: Quantitatively Managed Architecture,*
5. *Maturity Level: Optimizing Architecture.*

We have derived the architecture domains mainly from TOGAF [8], where they are used as specific architecture subtypes and corresponding phases of the TOGAF-ADM (Architecture Development Method). Architecture areas cover assessable architecture artifacts and are correspondent, but very different, parts of process areas from CMMI. To fit our architecture assessment scope, we have defined 22 original architecture areas of the SOAMMI framework [15], linked them to our architecture maturity levels and ordered them in line with our specific enterprise and software architecture domains. Each of the delimited architecture area is accurately described in a catalog including *name* of architecture area, *short identification* of architecture area and a *detailed description*.

SOAMMI supports both the staged and continuous representations. The same staging rules as in CMMI apply to SOAMMI and should therefore enable the flexible adoption of both model representations: *continuous* for assessing single architecture areas and *staged* for assessing the whole architecture maturity. The assessment of capability levels could be applied to iterate specific architecture areas or to assess or improve a focused innovation aspect, involving one ore more architecture areas. To verify and support persistent institutionalizations of architecture areas we introduce architecture related generic goals and practices. All architecture areas are affected by the same generic goals and associated generic practices. In the following, two example architecture areas together with their goals and practices are presented.

A. Examples of Architecture Areas

1. Business Products & Services

Purpose: Structure, design, model, and represent business products and associated business services, which are necessary to support modeled products.

Maturity Level: 3

Specific Goals (SG) and Specific Practices (SP):

SG 1: Model Business Products as Origin of Business Processes

SP 1.1 Structure business products within product lines

SP 1.2 Design business products by defining product structures and product rules

SP 1.3 Model and represent business products

SG 2: Model Business Services associated with Business Products

SP 2.1 Structure business services according to product types

SP 2.2 Design business services by defining service structures and service levels

SP 2.3 Model and represent business services

The second example extends contents from the first example and provides a base for our pattern language scenario:

2. Business Processes & Rules

Purpose: Structure, design, model, and represent business value chains and business processes to support business capabilities.

Maturity Level: 2

Specific Goals (SG) and Specific Practices (SP):

SG 1: Model Business Value Chains as Root of Business Capabilities and Business Processes

SP 1.1 Identify business value for business operations

SP 1.2 Structure value chains

SP 1.3 Optimize business considering customer channels and supplier networks

SG 2: Model and Optimize Business Processes

SP 2.1 Identify business activities for business processes: system activities, user interaction activities, manual activities

SP 2.2 Structure business processes for business roles and organizational units

SP 2.3 Define business workflows and business process rules

SP 2.4 Model and represent business processes

SG 3: Model and Represent Business Control Information

SP 3.1 Identify and represent control information for product monitoring

SP 3.2 Identify and represent control information for process monitoring.

IV. Assessment Pattern Language

Our pattern language for architecture assessments of service-oriented enterprise systems provides a procedural method framework for architecture assessment processes and questionnaire design. This method framework of our new introduced pattern language was inspired from [7], and derived from the structures of the metamodel of SOAMMI as well as from our seminal pattern catalog from previous research [16]. We note that our architecture patterns are basically assessment process patterns for enterprise architecture management and are therefore not fine granular classical design patterns.

We have linked each specific and each generic goal within our assessment framework to a distinctive pattern of our pattern language. We organize and represent our architecture assessment patterns according to the following structures: *Architecture Domains*, *Architecture Areas*, *Problem Descriptions* - associated with *Specific Goals*, *Solution Elements* - are connected to *Specific Practices*, *Related Patterns* - are connections to next applicable patterns of the pattern language.

Linking solution elements to specific practices of the SOAMMI framework enables concrete solutions for architecture assessments and improvements of service-oriented enterprise systems. This assessment and improvement knowledge is both verification and design knowledge, which is a procedural knowledge based on standards, best practices, and assessment experience for architecture assessments of service-oriented enterprise systems. It is therefore both concrete and specific for setting the status of service-oriented enterprise architectures, and helps to establish an improvement path for change. Patterns of our language show what to assess. Our patterns aim to represent verification and improvement knowledge to support cooperative assessments synchronizing people in cyclic architecture assessments.

Associated with our architecture assessment pattern language we have set up an assessment process to show how to assess architecture capabilities. This process is based on a questionnaire for architecture assessment workshops providing concrete questions as in [4], answer types, and helping to direct and standardize the related assessment process. Additionally, we have included process methods for workshops, result evaluations, improvement path information for technology vendors and for application organizations, as well as change support and innovation monitoring instruments.

Based on the two examples of architecture areas from Section III - Business Products & Services Architecture and Business Process & Rules, we sketch a pattern language scenario (as a typical small example):

1. Model and represent Business Products
2. Model Business Services for Business Products
3. Model Business Value Chains as Root of Business Processes
4. Model and Optimize Business Processes
5. Model and Represent Business Control Information.

We are representing the core causalities of our architecture assessment and improvement patterns in the reduced canonical form, which we have adapted from [2] and [6]. Our pattern form denominates consciously the problem and the solution part as basic elements of our patterns. This canonical form is extendable in further work by additional parts like contexts, forces, examples, explanations, and linked patterns. The following examples show a concrete extract of 5 related patterns, which derives from the sketched architecture areas from Section III:

1. *Pattern Example: Business Product*
   Problem: How can we structure, design, model, and represent each business product as an origin for modeling business processes?
   Solution:
   - Structure business products for product lines
   - Design business products by defining product structures and product rules
   - Model and represent business products

   Related Patterns: Business Services, Value Chain, Business Process, Business Control Information
2. *Pattern Example: Business Service*
   Problem: How can we structure, model, and represent each business service needed to support business products?
   Solution:
   - Structure business services for product types
   - Design business services by defining service structures and service levels
   - Model and represent business services

   Related Patterns: Value Chain, Business Process, Business Control Information
3. *Pattern Example: Value Chain*
   Problem: How can we structure, optimize and represent business value chains as roots for business process modeling?
   Solution:
   - Identify business value for business operations
   - Structure value chains
   - Optimize business considering customer channels and supplier networks

   Related Patterns: Business Process
4. *Pattern Example: Business Process*
   Problem: How can we structure, model and optimize business processes, related workflows, and business process rules?
   Solution:
   - Identify business activities for business processes: system activities, user interaction activities, manual activities
   - Structure business processes for business roles and organizational units
   - Define business workflows and business process rules
   - Model and represent business processes

   Related Patterns: Business Control Information
5. *Pattern Example: Business Control Information*
   Problem: How can we model and represent business monitoring and control information?
   Solution:
   - Identify and represent control information for product monitoring
   - Identify and represent control information for process monitoring

   Related Patterns: None

This scenario of pattern interactions is specified by the graph of architecture patterns (Figure 2), indicating the navigation direction (to the next applicable patterns).

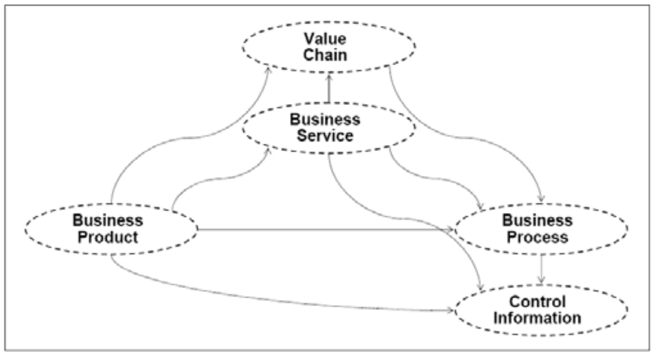

Figure 2. Architecture Pattern Language Scenario

In Figure 2, we indicated selective navigation paths to the next applicable patterns. Value Chain is for instance only a constructive artifact and not a real observable concept: Therefore the Value Chain Pattern has no associated Business Control Information Pattern. We have identified and distinguish a set of 43 patterns as parts of a new researched and introduced pattern language in the context of 7 Architecture Domains and 22 Architecture Areas. Even though our architecture quality patterns accords to the Specific Goals, the Specific Practices and the Generic Goals from the SOAMMI framework, they extend these structures by navigable patterns as part of an architecture assessment language. Only this pattern structure enables architecture quality assessors to navigate easily in two directions to support the diagnostics and optimization process,, and to provide a clear link to questionnaire and the related answer and result concepts. The full collection of patterns of the architecture assessment pattern language was derived from the SOAMMI framework in (Section III):

1. Architecture Domain: Architecture Strategy and Management

Architecture Area: ***EAM E****nterprise* ***A****rchitecture* ***M****anagement*

1. Pattern: Architecture Strategy

Architecture Area: ***GOV A****rchitecture* ***Go****vernance*

2. Pattern: Architecture Management
3. Pattern: Architecture Governance

Architecture Area: ***OPM O****rganizational* ***P****erformance* ***M****onitoring*

4. Pattern: Performance Baselines for Architecture

Architecture Area: ***QAM Q****uantitative* ***A****rchitecture* ***M****anagement*

5. Pattern: Manage Architecture Quantitatively
6. Pattern: Manage Architecture Agility

Architecture Area: ***AID A****rchitecture* ***I****nnovation and* ***D****eployment*

7. Pattern: Architecture Innovation Management

Architecture Area: ***CAR C****ausal* ***A****nalysis and* ***R****esolution*

8. Pattern: Causes of Architecture Defects
9. Pattern: Resolution of Architecture Defects

Architecture Area: ***ARM A****rchitecture* ***R****equirements* ***M****anagement*

10. Pattern: Architecture requirements Management

Architecture Area: ***ARD A****rchitecture* ***R****equirements* ***D****evelopment*

11. Pattern: Customer Requirements
12. Pattern: Architecture Requirements
13. Pattern: Validate Architecture Requirements

2. Architecture Domain: Business Architecture

Architecture Area: ***BDC B****usiness* ***D****omains &* ***C****apabilities*

14. Pattern: Business Domain
15. Pattern: Domain Granularity & Coupling

Architecture Area: ***BCS B****usiness* ***C****apabilities &* ***S****ervices*

16. Pattern: Business Capabilities
17. Pattern: Business Services
18. Pattern: Service Agility

Architecture Area: ***BPS B****usiness* ***P****roducts &* ***S****ervices*

19. Pattern: Business Product
20. Pattern: Business Service

Architecture Area: ***BPR B****usiness* ***P****rocesses &* ***R****ules*

21. Pattern: Value Chain
22. Pattern: Business Process
23. Pattern: Business Control Information

3. Architecture Domain: Information Architecture

Architecture Area: ***DEC D****ata* ***E****ntities &* ***C****omponents*

24. Pattern: Entity Service

Architecture Area: ***BOB B****usiness* ***O****bjects*

25. Pattern: Business Object
26. Pattern: Business-IT-Alignment

4. Architecture Domain: Application Architecture

Architecture Area: **SDO S**ystem ***Do****mains*

27. Pattern: System Domain Mapping

Architecture Area: ***SSC S****ystem* ***S****ervices &* ***C****apabilities*

28. Pattern: Application Service Design
29. Pattern: Application Vendor Services

5. Architecture Domain: Technology Architecture

Architecture Area: ***PFS P****lat****f****orm* ***S****ervices*

30. Pattern: Platform Service Design
31. Pattern: Platform Vendor Services

Architecture Area: ***TSC T****echnology* ***S****ervices &* ***C****apabilities*

32. Pattern: Technology Service Design
33. Pattern: Technology Vendor Services

6. Architecture Domain: Service & Operation Architecture

Architecture Area: ***SDT S****ervice* ***D****esign &* ***T****ransition*

34. Pattern: Support Service Design
35. Pattern: Service Management

7. Architecture Domain: Architecture Realization

Architecture Area: ***ASC A****rchitecture* ***S****tandards &* ***C****ompliance*

36. Pattern: Architecture Standards Management
37. Pattern: Architecture Standards Definition

Architecture Area: ***ACO A****rchitecture* ***Co****ntracts*

38. Pattern: Architecture Contracts.

Architecture Area: ***AIN A****rchitecture* ***In****stitutionalization*

39. Pattern: Base Architecture Practices
40. Pattern: Managed Architecture
41. Pattern: Defined Architecture
42. Pattern: Quantitatively Managed Architecture
43. Pattern: Optimizing Architecture.

V. EVALUATION AND FINDINGS

The practical benefits of our pattern language were demonstrated by the successful use as guideline for the questionnaire design in four major capability assessments of service-oriented vendor technology architectures. Architecture assessments need to address key challenges for companies during the built-up and management of service-oriented architectures.

SOAMMI seems to be complex in practice. Therefore a pragmatic simplification of the SOAMMI framework was particularly required in counting assessment results. Additionally we have considered for our assessments specific user requirements from companies using and providing service-oriented enterprise systems.

Following these ideas, the basic structure of our questionnaire [15] was taken from the SOAMMI architecture areas with one or more questions per Specific Goal. User requirements have been consolidated and mapped against specific goals. Wherever no user requirements could be mapped, Specific Practices have been used to generate questions on the level of specific goals. Through this procedure each Specific Goal could be related to at least one concrete question.

The assessment process takes about 3 months in total to complete for each software technology provider. The first step is a pre-workshop (2-3 hours) to make sure that the architecture provider can identify the appropriate experts for the assessment workshop itself. Then the actual assessment workshop (4- 6 hours) is held a few weeks later, so that the provider has enough time to identify the experts that should participate and prepare answers. Finally, a series of follow up workshops for specific questions (3-4 hours each) are arranged with the system technology provider.

VI. Conclusion and future Work

We have introduced an original pattern language for assessing capabilities and architecture maturity of service-oriented enterprise systems. In this paper we have motivated the necessity to extend existing SOA maturity models to accord to a clear metamodel approach due to the verified CMMI model. Based on the related work to CMMI, which is an assessment and improvement model for software processes but not for architectures, we have developed suitable models for assessments of service-oriented enterprise systems. Our specific architecture assessment approach of the SOAMMI framework was founded on current architecture standards like TOGAF and architecture assessment criteria from related work approaches. Additionally a dashboard was developed to support practical assessment processes, which were aligned both with the process for CMMI and with empirical questionnaire and interview methods. The presented SOAMMI framework was validated in consecutive assessment workshops with four global vendors of service-oriented platforms and has provided transparent results for subsequent changes of service oriented product architectures and related processes. Our empirical validation and optimization of the presented maturity framework is an ongoing process, which has to be synchronized with future cyclic evaluations of SOA platforms and their growing number of services. Extended validations of customers of service oriented technologies are planned for the next phase of our framework research and development. Future work additionally has to consider conceptual work on both static and dynamic architecture complexity, and in connecting architecture assessment procedures with prognostic processes on architecture maturity with simulations of enterprise and software architectures. Additional improvement ideas include patterns for visualization of architecture artifacts and architecture control information to be operable on an architecture management cockpit. We are working at extending our pattern language to a full canonical form in order to support full standardized cyclic architecture assessments for service-oriented products and solutions.

References

[1] H. Buckow, H.-J. Groß, G. Piller, K. Prott, J. Willkomm, and A. Zimmermann, "*Method for Service-Oriented EAM with Standard Platforms in Heterogeneous IT Landscapes*", Proc. 2nd European Workshop on Patterns for Enterprise Architecture Management (PEAM2010), Paderborn, Germany, GI-Edition - Lecture Notes in Informatics (LNI), P-160, 2010, pp. 219-230.

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "*Design Patterns*", Addison Wesley, 1994.

[3] T. Erl, "*SOA Design Patterns*", Prentice Hall. 2009.

[4] P. Bianco, R. Kotermanski, and O. Merson, "*Evaluating a Service-Oriented Architecture*", SEI-2007-TR-015, Carnegie Mellon University, Software Engineering Institute, 2007.

[5] CMMI-DEV-1.3 2010 "*CMMI for Development, Version 1.3*", Carnegie Mellon University, Software Engineering Institute, SEI-2010-TR-033, 2010.

[6] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, „*Pattern-oriented Software Architecture*", Wiley, 1996.

[7] C. Alexander, S. Ishikawa, M. Silverstein, M. Jacobson., I. Fiksdahl-King, and S. Angel, „*A Pattern Language*", Oxford University Press, 1977.

[8] TOGAF "*The Open Group Architecture Framework*" Version-9, The Open Group, 2009.

[9] D. Krafzig, K. Banke, and D. Slama, „*Enterprise SOA*", Prentice Hall, 2005.

[10] *Essential Architecture Project*, http://www.enterprise-architecture.org, last access: June, 19th, 2011

[11] ACMM, "*Architecture Capability Maturity Model*", in TOGAF Version 9, The Open Group Architecture Framework, The Open Group, 2009, pp. 685-688.

[12] S. Inaganti and S. Aravamudan, "*SOA Maturity Model*", BP Trends, April 2007, 2007, pp. 1-23.

[13] Sonic: "*A new Service-oriented Architecture (SOA) Maturity Model*", http://soa.omg.org/Uploaded%20Docs/SOA/SOA_Maturity.pdf, last access: June, 19th, 2011.

[14] Oracle: "*SOA Maturity Model*", http://www.scribd.com/doc/2890015/oraclesoamaturitymodelcheatsheet, last access: June, 19th, 2011.

[15] H. Buckow, H.-J. Groß, G. Piller, K. Prott, J. Willkomm, and A. Zimmermann, "*Analyzing the SOA-ability of Standard Software Packages with a dedicated Architecture Maturity Framework*", EMISA 2010: October 7– 8, 2010 - Karlsruhe, Germany, GI-Edition - Lecture Notes in Informatics (LNI), P-172, 2010, pp. 131-143.

[16] A. Zimmermann, E. Ammann, and F. Laux, „*Pattern Catalog for Capability Assessments and Maturity Evaluation of Service-oriented Enterprise Architectures*", PATTERNS 2010 - The Second International Conferences on Pervasive Patterns and Applications, November 21-26, 2010 - Lisbon, Portugal, IARIA Proceedings of the PATTERNS 2010 Conference, 2010, pp. 13-19.