Pattern Innovation for Architecture Diagnostics in Services Computing

Alfred Zimmermann
Reutlingen University, Faculty of Informatics
Architecture Reference Lab of the
SOA Innovation Lab, Germany
alfred.zimmermann@reutlingen-university.de

René Reiners
Fraunhofer FIT
User-Centered Ubiquitous Computing
Sankt Augustin, Germany
rene.reiners@fit.fraunhofer.de

***Abstract* – Assessing the maturity of service-oriented enterprise software architectures is a problem since the current practice has been developed rather intuitively, providing only a sparse and rarely validated metamodel. In preliminary research, we have developed an original pattern language for supporting repetitive enterprise system architecture assessments. The aim is the evaluation and optimization of these kinds of architectures. For this purpose, we extended base frameworks like the Capability Maturity Model Integration and The Open Group Architecture Framework. Since we apply a pattern catalogue for the assessment of enterprise system architectures, we see ourselves confronted with the problem that patterns are traditionally derived after long experience by an expert group of pattern authors. In our view, this may lead to a decelerated reuse of available knowledge. Our approach intends to integrate available knowledge from services computing and software architects directly from the beginning of the pattern development process. Over time, these ideas are iteratively developed towards validated patterns by feeding back the insights of pattern applications. This allows the early integration of new findings and concepts into the pattern catalogue at an early stage whereas already existing patterns are continuously refined. In this work, we propose both, a clear maturity framework background for the developed architecture assessment patterns, and an early integration of new ideas as pattern candidates within a pattern innovation and evolution process.**

Keywords – service-oriented systems; architecture maturity framework; assessment patterns; pattern evolution.

I. INTRODUCTION

Innovation oriented companies have introduced services computing systems to assist in closing the gap between business and information technology and thus enabling business opportunities for service and emerging cloud computing paradigms in the context of novel enterprise architecture management approaches. One of the main problems is that until today the transparency of this innovation change to system architectures based on services and cloud computing in information technology is blurred. Our approach investigates the ability of heterogeneous enterprise services systems [1] and integrates system architecture elements from convergent architecture methods, technologies and related software patterns, as in [2], [3], and [4] with evaluation methods for service-oriented enterprise systems [5].

The SOA Innovation Lab - an innovation network of industry leaders in Germany and Europe - investigates the practical use of vendor platforms in a service-oriented and cloud-computing context. For this purpose we have researched a suitable set of architecture assessment instruments for services computing, leveraging and extending the Capability Maturity Model Integration (CMMI) [6] and the Open Group Architecture Framework (TOGAF) [7]. This set extends our previous work and consists of ESARC - our Enterprise-Services-Architecture-Reference-Model [8] and [9], an associated ESA-Architecture-Maturity-Framework [1] and [8], and an ESA-Pattern-Language [10] for supporting architecture evaluation and optimization.

Our research explores the novel hypothesis to relevantly support a major effort of software architects during architecture assessments of service-oriented systems:

1. CMMI [6] is well known as a suitable basic maturity framework to assess software processes. Nevertheless the metamodel of CMMI can be transformed to enable quality assessments for software architectures.
2. The idea of software patterns can consistently be applied and extended in service-oriented architecture assessments for capability diagnostics of service-oriented architectures. The collected architecture assessment patterns could be iteratively improved within our original pattern evolution process.

We are reporting in this research paper about our current research step to combine our previous evaluated architecture assessment metamodel with a newly introduced community-oriented pattern evolution process. In Section II we present related and preliminary work concerning service-oriented architectures and frameworks. Additionally, we present current findings on architecture maturity assessment. Section III gives a brief introduction to software patterns as best practices for application and software design also providing background information about the approach, a pattern's intention, structure and the combination of patterns. The application of architecture patterns as test cases for the maturity assessment of software architectures is shown in Section IV together with the derived SOA maturity model integration SOAMMI and results from a first practical validation process. We explain the pattern innovation process that is combined with the currently existing pattern catalogue providing collaborative means for the early integration of new concepts and their evaluation during a project's lifetime. Finally, we conclude in Section V on the current state and provide an outlook on future work and directions.

II. Architecture Maturity And Its Assessment

The Open Group Architecture Framework (TOGAF) [7] as the current standard for enterprise architecture provides the basic blueprint and structure for our enterprise software architecture domains of service-oriented enterprise systems, as in the ESARC reference model [9]: Architecture Governance, Architecture Management, Business & Information Reference Architecture, Information Systems Architecture, Technology Architecture, Operation Architecture, Security Architecture, and Cloud Services Architecture.

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 [11] 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 also combined services are executed from different orchestrated services.

In recent work, we have transformed the Capability Maturity Model Integration into a specific framework for architecture assessments of service-oriented enterprise systems. For this reason, we have combined CMMI with current SOA frameworks and maturity models. We used TOGAF and ideas related to the business and information architecture from [12] as a basic structure for enterprise architecture spanning all relevant levels of service-oriented enterprise systems. In contrast to the Enterprise Architecture Project in [12] we are focussing on standardized structures form TOGAF [7] and extend these in our ESARC Architecture Reference Model, as in [8] and [9], with currently researched new additional architectural views: Operation Architecture, Security Architecture, and Cloud Services Architecture.

The Architecture Capability Maturity Model (ACMM) framework, which is included in TOGAF [7], 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, which are ranked for each maturity level, and are deviant from the understanding of maturity levels in CMMI.

Inaganti and Aravamudan [13] describe the following multidimensional aspects in their SOA Maturity Model: 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, applying 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.

Sonic [14] 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.

ORACLE [15] considers in their SOA Maturity Model 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.

A. *The SOAMMI Framework*

The aim of the SOAMMI – SOA Maturity Model Integration - framework [1] is to provide an integral framework to assess architectures of service-oriented enterprise systems and to accord with a sound metamodel approach. The previously mentioned related work elements where developed following in contrast to SOAMMI only a pragmatic and intuitive approach, having no explicit metamodel and outside of common architecture standards, like TOGAF. The 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 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 is illustrated in Figure 1 also revealing that it has similarities with the original CMMI metamodel; we left the semantics 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.

The semantics of these maturity levels as in [1] were adapted from [6] to conform to the architecture assessment scope for service-oriented enterprise systems. In terms of requirements from customer oriented domain-models and reference use scenarios, our model has introduced in [8] five maturity levels, which define architecture assessment criteria for service-oriented enterprise systems and help to measure the architecture maturity, like Initial Architecture, Managed Architecture, Defined Architecture, Quantitatively Managed Architecture, and Optimizing Architecture.

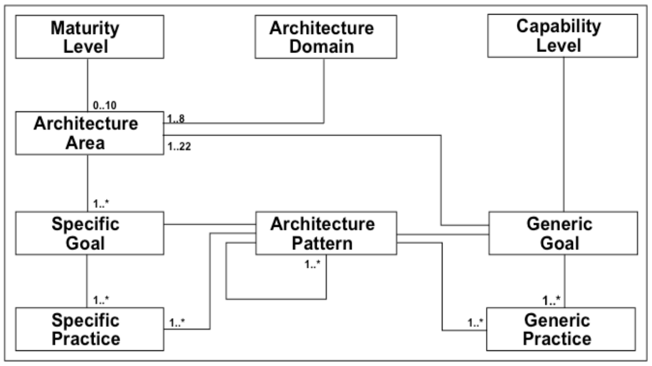

Figure 1: SOAMMI Metamodel – Main Concepts.

We have derived the architecture domains mainly from TOGAF 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 [1] and [8], 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 or 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.

B. *Example of Architecture Area*

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, and 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.

III. PATTERN COLLECTIONS AND LANGUAGES

Design patterns originated as an architectural concept introduced in the seminal book "A Pattern Language" written by Christopher Alexander [16]. He captured his experience gathered over time and structured this knowledge in smaller units as patterns describing good qualities of a real world examples. The level of detail varied from landscape characteristics over areas, quarters up to single parts of houses and even rooms. Alexander structured the different patterns by means of size. This way, patterns explaining concepts of larger areas were explained first, connecting the currently read pattern to descendent patterns with a higher level of detail.

Alexander's intention was to describe best practices and effective design solutions in order to share design knowledge with other people facing similar problems in a related context. The solution proposed by a software pattern should be generic rather than specific, such that it can be implemented in numerous different ways. The benefit of using patterns is that they communicate insights into common design problems and reflect solutions that a community of experts has developed over time.

An important quality of a pattern was the readability by non-experts. Since every pattern was written in prose with a standard vocabulary, the concepts could be understood by a large readership that was not necessarily experts in the domain.

This thought of structuring knowledge in patterns was picked up in many different computer-science domains like software design, human-computer interaction design, website design and many others.

In particular, Gamma et al. extended the notion of patterns into the domain of software engineering, and constructed twenty-three classic software design patterns [2]. Since then, the concept of design patterns also became essential in the domain of Human-Computer-Interaction (HCI), where patterns are commonly used to describe and preserve solutions to recurring user interface design problems. Borchers transferred the pattern concept to human-computer interaction design for interactive exhibits [17]. From his point of view, especially patterns in HCI need to bridge the gap between users with conceptual knowledge and understanding of the problem domain and software engineers who are deeply involved in the technical development.

A given pattern is not always the optimal solution in every case, but tends to work in practice and supports user acceptance for the system. Tidwell describes the influence of patterns in user interface design stating that each implementation of the same pattern differs somehow in its characteristics although it comes from the same origin [18]. Thus, patterns should be seen as description of a problem solution as starting-point and not as fixed design rules.

A similar approach is introduced by Schümmer and Lukosch in the domain of computer-supported collaboration [19]. They structure their pattern language along the level of technical complexity: The more detailed a pattern describes a certain solution, the more technical this description becomes. Up to a certain degree of detail, they consider all patterns as relevant for all stakeholders. Beyond that point, the target group changes to engineers that need to technically implement the design suggestion.

In addition to working solutions, the description of *anti-patterns* is also a valid information source for application and interface designers. They document surprisingly failing approaches that turn out to be ineffective or counter-productive in practice [20]. Other collections, e.g., in UI design, focus on pointing out repetitions of design flaws [21]. Here, concepts that have intruded many designs but actually lead to rejection are discussed and the reasons for design failures are explained.

A collection of patterns, which are organized in a directed acyclic graph structure, is referred to as a *pattern language*. Elements of a pattern language are navigable sequences of patterns. In contrast to pattern language, pattern *collections* provide semi-structured clusters of patterns that are not interconnected in a hierarchy. This is for example the case in [2] who distinguish between structural, creational and behavioral patterns.

IV. Pattern Innovation for Architecture Diagnostics

Although design patterns are mainly used to inform the design of a system, they are also applied as test cases for assessing software. Software architecture assessment patterns are based on the seminal work of software patterns originated from the work of [16].

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 [20], and derived from the structures of the metamodel of SOAMMI as well as from our initial pattern catalog from previous research [10].

We have linked each specific and each generic goal within our assessment framework to a distinct 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* that are connected to *Specific Practices* and *Related Patterns*, which are subsequent connections of applicable patterns within the pattern language.

Linking elements to specific practices of the SOAMMI framework indicate 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 [8], 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.

We have identified and distinguished a set of 43 patterns as parts of a newly 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 (cf. Section II).

A. Evaluation and Findings

The practical benefits of our SOAMMI assessment 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, as in [1] and [8]. 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 in [8] 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.

B. Shortcomings for Updating and Refining

The pattern catalogue that serves as a basis for our assessments is continuously a subject of consideration with regard to pattern refinement, pattern improvement and catalogue extension. In parallel to the assessments, feedback on the state of the patterns that were used during the evaluation is gathered.

This way, we have a chance to update existing patterns or derive variants of them. However, we cannot be sure that a new pattern or derivation is really valid. On the other side, the variant or new formulation can be a promising pattern candidate and later be validated and therefore be integrated into the pattern catalogue in order to use and benefit from it as early as possible.

The current process, however, does not foresee the inclusion of non- or semi-validated patterns. The validation process of a pattern also is a time-consuming process with much iteration. It can partially be combined with additional SOA assessments but then still a subset of new patterns needs to be investigated in more depth.

So, our aim is to gather the feedback, adjust our current findings and preserve knowledge, feedback and new findings within our catalogue.

For this reason, we aim at establishing an evolution process that makes it possible to integrate early results into the existing pattern catalogue. Continuous refinement and therefore the lifelines of the pattern catalogue need to be ensured. The requirements for such a process were already defined in preliminary work [18] and [23]. The process itself is described in the following section.

Traditionally, pattern collections are published after a long period of development and validation where the essences from design experience can be extracted. This is mostly done by a small, closed group of design experts as described in [24]. In the approaches presented in the previous sections, much effort was put into the derivation and evaluation of mature and evaluated patterns.

However, we see the problem that many findings must be regarded earlier, at the state of an idea in order to be able to consider many findings in a flexible pattern set. This holds the chance to start working with patterns very early – even if it not yet fully proven. Our process [25] wants to include new ideas and concepts into the project's lifecycle as early as possible. Over time, the idea, which is directly formulated as a *pattern candidate*, gets refined and evaluated.

In this scenario, it may turn out that the candidate is not a pattern and needs to be rejected. Alternatively, after continuous refinement and evaluation the pattern candidate may become more mature, reaching a new state, e.g., being "under consideration". The counter-result is also possible: A promising pattern idea may also turn out to lead to a bad decision or concept. In this case, we speak of a surprisingly failing solution. In order to avoid similar failures in the future, we formulate this concept as an anti-pattern. This way, the pattern gets a warning character, allowing follow-up to directly cross out this idea and alter considerations.

V. Conclusion and future Work

In this work, 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.

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 current research extends SOAMMI to support architecture diagnostics for complex integrated enterprise systems in the emerging context of services and cloud computing architectures.

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.

The need for iteratively updating our assessment pattern collection motivated us to merge the efforts done for SOA assessment with a flexible and iterative pattern refinement and creation process. After talking about SOA maturity and assessment, we looked at the concept of involving *many* stakeholders into the pattern creation and evolution process and to adapt already available knowledge and findings from the project's domain as early as possible.

Our presented *pattern-lifecycle process* allows for continuously evaluating gathered knowledge during the project's lifetime and makes patterns as well as pattern ideas available during the whole development process. This way, pattern collections can be formulated collaboratively without needing to wait for a closed author group that shares its well-evaluated design knowledge after a longer period of time.

Additional improvement ideas include an architecture pattern and knowledge repository, as well as 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 fully standardized cyclic architecture assessments for service-oriented products and solutions. The pattern evolution process represents a new aspect to the assembly and structuring our patterns and will further explored in the SOA assessment domain.

Acknowledgment

This paper extends ideas from the SOA Innovation Lab, which is a major research and innovation network on Enterprise Architecture Management for Services and Cloud Computing in Germany and Europe.

Parts of the research leading to these results has received funding from the European Union Seventh Framework Program (FP7/2007-2013) under grant agreement n°261817, the BRIDGE project (www.bridgeproject.eu) under the Security Program SEC-2010.4.2-1.

References

[1] 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," in *EMISA*, 2010, pp. 131-143.

[2] E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides, *Design Patterns. Elements of Reusable Object-Oriented Software*, 1st ed. Amsterdam: Addison-Wesley Longman, 1994, p. 416.

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

[4] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. Chichester, UK: Wiley, 1996.

[5] P. Bianco, R. Kotermanski, and P. Merson, "Evaluating a Service-Oriented Architecture," *Engineering*, no. September, pp. 1-91, 2007.

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

[7] TOGAF "*The Open Group Architecture Framework*" Version-9.1, The Open Group, 2011.

[8] A. Zimmermann, H. Buckow, H.-J. Gross, O. F. Nandico, G. Piller, and K. Prott, "Capability Diagnostics of Enterprise Service Architectures Using a Dedicated Software Architecture Reference Model," *Services Computing, IEEE International Conference on*, vol. 0, pp. 592-599, 2011.

[9] A. Zimmermann and G. Zimmermann, "*ESARC - Enterprise Services Architecture Reference Cube for Capability Assessments of Service-oriented Systems*", SERVICE COMPUTATION 2011 - The Third International Conferences on Advanced Service Computing, September 25-30, 2011 Rome, Italy, ISBN 978-1-61208-152-6, IARIA Proceedings of SERVICE COMPUTATION 2011, pp. 63-68.

[10] A. Zimmermann, F. Laux, and R. Reiners, "A Pattern Language for Architecture Assessments of Service-oriented Enterprise Systems," in *PATTERNS 2011, Third International Conferences on Pervasive Patterns and Applications*, 2011, no. c, pp. 7-12.

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

[12] "Essential Architecture Project." [Online]. Available: http://www.enterprise-architecture.org. [Accessed: 11-Mar-2012].

[13] S. Inaganti and S. Aravamudan, "SOA Maturity Model," *BP Trends*, no. April, pp. 1-23, 2007.

[14] Sonic: "*A new Service-oriented Architecture (SOA) Maturity Model*", http://soa.omg.org/Uploaded%20Docs/SOA/SOA_Maturity.pdf, [Accessed: 11-Mar-2012].

[15] ORACLE, "ORACLE: 'SOA Maturity Model'."[Online]. Available: http://www.scribd.com/doc/2890015/oraclesoamaturitymodelcheatsheet. [Accessed: 11-Mar-2012].

[16] C. Alexander, *A Pattern Language: Towns, Buildings, Construction*. New York, New York, USA: Oxford University Press, 1977.

[17] J. Borchers, *A Pattern Approach to Interaction Design*, 1st ed. John Wiley & Sons, 2001, p. 268.

[18] J. Tidwell, *Designing Interfaces*, 1st ed. O'Reilly Media, 2005, p. 352.

[19] T. Schümmer and S. Lukosch, *Patterns for Computer-Mediated Interaction*. Chistester, West Sussex, England: John Wiley & Sons, 2007, p. 600.

[20] R. Reiners, I. Astrova, and A. Zimmermann, "Introducing new Pattern Language Concepts and an Extended Pattern Structure for Ubiquitous Computing Application Design Support," in *PATTERNS 2011, Third International Conferences on Pervasive Patterns and Applications*, 2011, pp. 61-66.

[21] J. Johnson, *GUI bloopers 2.0: Common User Interface Design Don'ts and DOS*, vol. 2, no. October. New York, NY, USA: Morgan Kaufmann, 2007.

[22] T. Grill and M. Blauhut, "Design Patterns Applied in a User Interface Design (UID) Process for Safety Critical Environments (SCEs)," in *HCI and Usability for Education and Work*, vol. 5298, A. Holzinger, Ed. Springer Berlin / Heidelberg, 2008, pp. 459-474.

[23] C. R. Prause, "Reputation-based self-management of software process artifact quality in consortium research projects," in *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, 2011, pp. 380-383.

[24] "The BRIDGE Design Pattern Library." [Online]. Available: http://pattern-library.sec-bridge.eu/. [Accessed: 11-Mar-2012].

[25] R. Reiners, "*A Pattern Evolution Process – From Ideas to Patterns*", Proceeedings Informatiktage 2012 Bonn - Germany, in Lecture Notes in Informatics, Vol. S-11, 2012, pp. 115-118.