Toward a Pattern Language for a Allocation View in SOA

Mohammadreza Shahlaei, Seyyed Mohsen Hashemi

Abstract– Using pattern as proved solution helps to achieve SOA strategic goals. Maturity models are parental framework for SOA roadmaps, and Architecture is an important dimension for maturity models. Architecture has different views, Usually Reference architecture and pattern languages present runtime view of architecture dimension. This paper presents pattern language that creates allocation view for service oriented architecture. This view shows mapping between services and enterprise parts.

Keywords- service oriented architecture, pattern language, reference architecture, allocation view

I. INTRODUCTION

In recent years different computing models such as cloud computing, grid computing and service computing have been proposed. These computing models can be use for improvement in IT resource usage. Business and IT alignment, cost reduction, organization agility and workload reduction are some of strategic goals in service oriented computing.

SOA acceptance is a complex process and there are different roadmaps for it. SOA maturity models with the aim of clarifying the SOA confusion, offers a parental framework for SOA roadmaps [1]. OSIMM is a maturity model that is standard and independent from any vendor and relatively has coverage for all aspects of SOA adaption process. Maturity levels in this model are based on service integration. The seven dimensions of this model are: Business, governance, method, architecture, application, information and infrastructure [2]. A set of activities in maturity model dimension should be done to achieve SOA complete solution. OSIMM is a tool to evaluate solution in service oriented computing.

In SOA acceptance process there are various challenges. Inspecting root of this challenges, has determined that service computing strategic goals in first level and in continue design principals in next level are root of this challenges. With implementing these design principals in method dimension, other challenges appear and other dimensions are affected by method dimension. These Challenges are in different granularity. Service identification, place service in logical domain, service orchestration and routing, service deployment, service Meta data management, latency in execution for loosely coupling and conform to several standards, are some of challenges in service computing.

Proved solutions must be used to solve these challenges. Patterns are set for this purpose. Pattern is a documented, proved and recurrent experience. Pattern is abstraction from proved, repeatable solution that solves one problem in specific context [3]. Pattern catalog is a set of patterns that solve set of challenges and pattern language is a set of related pattern that used for solve set of related problems. Pattern language shows the correct meaning and usage of patterns.

Patterns and pattern languages can be used to solve challenges in SOA but they have narrow coverage in SOA challenges and creating a complete solution for SOA based on them is difficult and maybe impossible.

Architecture is one of the main dimensions in OSIMM that affect other dimensions. Main decisions in service oriented design must be considered in architecture dimension. Architecture in SOA documented using views that are like software architecture. Patterns and pattern language can be used for building views in the architecture.

One important view in architecture is allocation view that is inspired from allocation view in software architecture. This view show mapping between software element, and non software element [4]. Allocation view can be used in SOA. In this paper allocation view is constructed with pattern language and shows the mapping between service as software element and enterprise parts as non software element.

Second section reviews the set of pattern catalog and pattern languages. Third section reviews architecture dimension and some of specification. In fourth section pattern language is presented and last section is for conclusion.

II. HISTORY OF PATTERNS IN SOA

The goal is to use patterns and pattern languages to solve SOA challenges. Different pattern languages have been presented for SOA, but some of them are based on specific technology and some of them are about very fine grain challenges that are not discussed in this section.

The pattern language that published by Arsanjani (2005) [5] is about component integration using service encapsulation. This pattern language is based on specific methodology in service oriented computing. Loosely coupled service integration is a main purpose in this pattern language. The patterns in this pattern language place in method dimension or architecture dimension and there is no care about governance and business dimension.

Another instance of pattern language is one that provided by Haritha Kilaru (2006) [6]. This pattern language is based on software stability models. Service oriented concepts in it are general and independent from any special implementation. Software stability model comes in three levels. Service oriented design principals are in first level. In next level set of patterns are presented to realize the principals. Finally in the third level there is ability to implement special instance for SOA. Service oriented system that is based on this patterns, has an effective design and architecture. This pattern language like previous pattern language doesn't cover business and governance dimensions. And architecture, method and application are coverage dimensions.

Pattern catalog presented by Thomas Erl [7] pays the basic concepts of service orientation.

Manuscript received September 02, 2012.

Mohammadreza Shahlaei, Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran.

Seyyed Mohsen Hashemi, Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran.

This catalog is classified with four levels of architecture. Patterns presented in this catalog, each of them are based on a series of design principles. The main dimension is method but furthermore there are patterns for challenges in architecture, information and governance dimensions.

By respecting to OSIMM, pattern languages usually are related to IT dimension of service orientation. Most of the pattern languages focus on the methodology and architecture dimensions and there is low coverage for business and governance dimension. Some of dimension in OSIMM doesn't have special documented pattern or solutions are not in pattern format. Patterns don't have coverage to all dimensions and levels in maturity model or even don't have full coverage for a special dimension. The patterns and pattern languages that represented for service oriented computing have narrow coverage for complete solution.

In the first years of the emergence of service orientation, that is restricted to service oriented architecture and was considered as an architectural style. The main goals in these years are system integration and reusability. But in recent year the goals are changed. The business and IT alignment, and full coverage for organization requirements are strategic goals for service orientation.

By considering patterns, it is clear that architecture dimension is still an important dimension. Challenges in this dimension are inspired from design principals and strategic goals in service orientation and Main decisions in service oriented design must be considered in architecture dimension. For importance of architecture dimension, Next section is about architecture dimension.

III. ARCHITECTURE DIMENSION

In this section, first part is about architecture definition and related concepts. In second part some of reference architectures are reviewed and last part specifies patterns role in architecture dimension.

A. Architecture definition

Service oriented computing is a software paradigm and SOA has root in software architecture. Therefore this part discusses about software architecture foundations.

Software architecture is composed of elements, connections or relations among them, and usually some other aspect or aspects, such as configuration, constraints or semantics, analyses or properties, or rationale, requirements, or stakeholders' needs. Architecture serves as the basis for system analysis and construction. It also is a primary vehicle for communication among stakeholders and is a mean of education [4].

Many systems cannot be understood by a simple decomposition into parts and subsystems. There are plenty of interactions between parts. They are eco systems. From a holistic perspective, a SOA-based system is an eco system. With a large system, it is clear that nobody is really "in control" or "in charge" of the whole ecosystem, although there are definite stakeholders involved, each of whom has some control and influence over the community [9].

The view concept is emerged from this complexity. The view is a representation of the whole system from the perspective of a related set of concerns. Documenting architecture is a matter of documenting the relevant views and then adding documentation that applies to more than one view. In the year 2000, the IEEE adopted a standard [10]. This standard advocates creating your own views that serve the stakeholders and their concerns and associated with your system.

Architecture styles represent observed architecture approaches. A style description does not generally include detailed problem and context information as Architecture patterns do. Styles and patterns create views in architecture.

B. Reference architecture

Reference architectures are specifications that related to architecture dimension and are discussed in this part. They help us to have better knowledge about pattern role in architecture.

Reference architectures are specifications that act as indicator for architecture maturity assessment for SOA [2].

Reference architectures are relatively complete solution. Reference architecture models the abstract architectural elements in the domain independent of the technologies, protocols, and products that are used to implement the domain.

OASIS presented a Reference architecture that is conceptual and determines view and concepts that must be considered in SOA architecture documentation [9].

Reference architecture that presented by open group is a generic specification that have layers. In these layers there are building blocks and patterns used for abstraction of the various relationships between building blocks. Open group claims that the Reference architecture presented by them have end to end coverage in SOA IT aspect [8].

Reference architectures usually are in high abstraction levels and about runtime view of SOA. However some of them present different view for system.

C. Pattern role in architecture dimensions

For importance of architecture dimension, usually pattern languages goal is to achieve a complete solution in architecture dimension.

On the other hand there is a wide gap between patterns and architecture dimension complete solution. Patterns and pattern languages have narrow coverage for SOA solution. Even By restricting pattern language to architecture dimension, it cannot present complete solution for this dimension.

But Pattern languages can create a view for architecture documentation in SOA. Views that constructed with pattern languages are concrete. These pattern languages and views that created by them can be a complementary for reference architectures.

Next section presents a special view. This view is constructed with a pattern language.

IV. PATTERN LANGUAGE FOR ALLOCATION VIEW

Allocation view is a view that presented in [4]. This view is about interaction between software and non software elements.

For example deployment style describes the mapping between the software's components and connectors and the hardware of the computing platform on which the software executes.

This style result view that is useful for analyzing performance, availability, reliability, and security.

Allocation view can be useful for SOA. Pattern language that presented in this section creates an allocation view. Patterns of this pattern language describe the mapping between organization parts as non software element and service as software elements.

The pattern language creates an allocation view that can be used for analyzing redundancy, reusability, manageability and business service coverage.

From SOA pattern catalog that is presented with Thomas erl [8], four patterns are selected. Figure 1 shows pattern language of these patterns for allocation view in SOA.

First pattern is Enterprise inventory. Root challenge is that Delivering services independently via different project teams across an enterprise establishes a constant risk of producing inconsistent service and architecture implementations, compromising Recomposition opportunities. In this condition, Services for multiple solutions can be designed for delivery within a standardized, enterprise-wide inventory architecture wherein they can be freely and repeatedly recomposed.

This pattern allocates all service to one organization in the inventory template. By using this pattern and further normalization in service identification, there is maximum reusability and minimum redundancy in the organization services.

But establishing a single enterprise service inventory may be unmanageable for some enterprises, and attempts to do so may jeopardize the success of an SOA adoption as a whole. Therefore, Services can be grouped into manageable, domain-specific service inventories, each of which can be independently standardized, governed, and owned. The pattern is named domain inventory.

This pattern is necessary pattern for allocation view and shows the mapping between organization parts as non software element and service as software elements. Using this pattern, increase redundancy and reduce reusability but increase manageability.

However Standardization disparity between domain service inventories imposes transformation requirements and reduces the overall benefit potential of the SOA adoption.

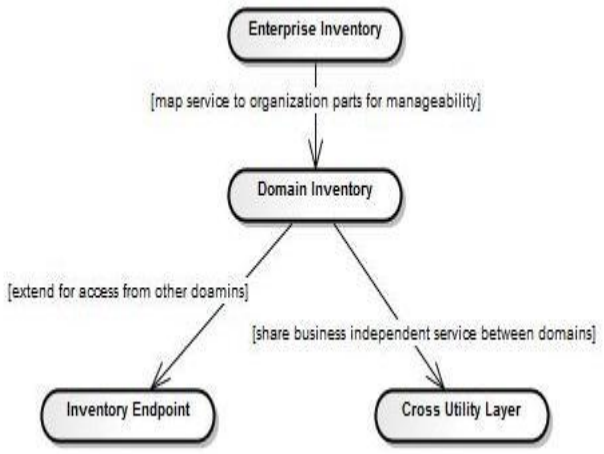

Fig1. Pattern language for allocation view in SOA

The Third pattern is inventory endpoint. A group of services that delivered for a specific inventory may provide capabilities that are useful to services outside of that inventory. However, for security and governance reasons, it may not be desirable to expose all services or all service capabilities to external consumers. In this condition, abstract the relevant capabilities into an endpoint service that acts as an official inventory entry point, dedicated to a specific set of external consumers.

This pattern show special relationship between patterns and organization parts. Using this pattern increase reusability and reduce redundancy of service in SOA.

The last pattern is Cross-Domain Utility Layer. While domain service inventories may be required for independent business governance, they can impose unnecessary redundancy within utility service layers. Solution is to use a common utility service layer can be established, spanning two or more domain service inventories.

This pattern presents another special relationship between service and organization parts. And this pattern increases reusability and reduce redundancy of service in SOA.

The relation between services and organization parts is like to message passing mechanisms. Inventory endpoint is multicast one service to multiple organization part. In contrast, Cross utility layer any cast one service for all organization parts.

Based on these patterns there are three types of relations between services and organization parts.

- Services that restrict to special domain
- Services that are share between all domains
- Service within one domain that other domain can use it

V. CONCLUSION

Architecture is an important dimension in capability maturity model of SOA. Main decisions in service oriented design must be considered in architecture dimension. Using pattern languages creates confident in system design. Therefore pattern language can improve architecture decision quality.

Architecture is documented in multiple views. One useful view for architecture is allocation view.

Pattern language that presented in this paper creates allocation view. This view shows mapping between services and organization parts.

Analyzing redundancy, reusability and manageability are benefit of this view. Furthermore this view is basis of establishing SOA in organizations.

REFERENCES

1. F.Meier, Service Oriented Architecture Maturity Models: A guide to SOA Adoption, University of Skovde Sweden, 2006
2. The Open Group Service Integration Maturity Model (OSIMM), Version 2, The Open Group, 2011
3. F. Buschmann, R. Meunier, H. Rohnert, P.Sommerlad and M.Stal. Pattern-Oriented Software Architecture, John Wiley&Son, 1996
4. P.Clements, F.Bachmann, L.Bass, D.Garlan, J.Ivers, R.Little, P. Merson, R.Nord & A. Stafford, Documenting Software Architectures: Views and Beyond, Second Edition. Addison- Wesley Professional, 2010
5. A.Arsanjani, Toward a pattern language for Service Oriented Architecture and Integration, Part 1: Build a service eco-system,2005
6. H.Kilaru , A pattern language for service-oriented architecture, San Jose State University,2006
7. Thomas Erl, SOA Design Patterns, Prentice Hall, 2009
8. The Open Group SOA Reference Architecture, Technical Standard, October 2010 (C104), available at:

9. www.opengroup.org/bookstore/catalog/c104.htm.
10. The OASIS Reference Architecture for SOA, Version 1.0, OASIS Standard, 23 April 2008, available at:
11. http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf.
12. Recommended Practice for Architectural Description of Software-Intensive Systems, 2000, IEEE Product No.: SH94869-TBR.: IEEE Standard No. 1471-2000

AUTHORS PROFILE

Mohammadreza Shahlaei received master degree in Software Engineering from Science and Research Branch of Islamic Azad University of Tehran. His current research interest includes Service Oriented Computing, software engineering Process, Software Methodology and software architecture.

Seyyed Mohsen Hashemi received the M.S degree in Computer Science from AmirKabir University of technology (Tehran Polytechnic University) in 2003, and the PhD degree in Computer Science from the Azad University in 2009, Moreover he is currently Dean of the Software Engineering and Artificial Intelligence Department Assistant Professor at Science and Research Branch, Islamic Azad University, Tehran, IRAN. His current research interest includes software intensive system, E-X system (E-Commerce, E-Government, E-Business, and so on), Global Village Service, Grid Computing, IBM SSME, Business Modeling, Agile Enterprise Architecting through ISRUP, and Globalization Governance through IT/IS services.