SOA GOVERNANCE MODEL

Matjaz B. Juric
University of Ljubljana, Slovenia
matjaz.juric@fri.uni-lj.si

Eva Zupancic
University of Ljubljana, Slovenia

Abstract:
Service Oriented Architecture (SOA) has become a foundation for service development and for Software-as-a-Service. Essential part of SOA is SOA Governance. This article focuses on the definition of a comprehensive SOA governance model. The SOA governance model addresses service governance, business document governance, service migration, service portfolio management, service ownership, service version management, and service funding model. It provides detailed description of requires activities, principles and objectives. The SOA governance model has been verified on real-world projects, where it has fulfilled the governance objectives.

Keywords: SOA, governance, services

1. INTRODUCTION

Service Oriented Architecture (SOA) has become a foundation for service development and for Software-as-a-Service (SaaS) development in Cloud Computing. SOA is based on several principles, such as loose-coupling, reuse, explicit boundaries, shared contracts and schemas, and meta-model. However, praxis has proven that transition to SOA is complex and often fails to achieve the major SOA objectives (Erl, 2009). To avoid failure, comprehensive and well-defined governance is needed (Jurič, Jennings, Sarang, & Loganathan, 2007). Today, it is well understood that SOA Governance is one of the most important keys to successful transition to SOA (Marks, 2008). However, there is a lack of comprehensive, well-defined SOA Governance models (Erl, 2007). This article addresses the most important part of SOA Governance – service governance and presents a comprehensive, praxis-proven service governance model, which defines a complete set of necessary practices for governance of services.

The article is structured in five sections. In section 2, we give a brief overview of SOA governance. In section 3, we provide a detailed description of our SOA governance model. In section 4, we provide the analysis of related work and the discussion and in section 5, we give concluding remarks.

2. BRIEF SOA GOVERNANCE OVERVIEW

The role of SOA governance is to define the guidelines, define the controls that the guidelines are followed, provide supervision of overall architecture and achieve alignment of services to the architecture (Jurič, Jennings, Sarang, & Loganathan, 2007). Successful governance is often the key for a successful project, as it improves decision making, promotes value-based innovation, effectively manages change, align business objectives and IT investment, fosters enterprise-wide collaboration, increases the likelihood of success, tracks IT investments and returns, links vision and strategy, and enhances ecosystem visibility (Erl, 2009).

The following SOA governance goals apply (Marks, 2008): increased agility, emphasis on Enterprise Architecture, rationalization, reuse, retirement of legacy assets, achieving of organizational effectiveness and accountability, and leveraging of shared services. In SOA Governance, one of the most important topics is service governance. Services are the main building blocks of the SOA architecture (Jurič, Jennings, Sarang, & Loganathan, 2007). Service governance has to assure, that the guidelines are followed. For each service a Service Profile has to be defined (Erl, 2007). Service Profile defines important information for service design, development, and governance. Each service implements a set of capabilities (functionalities), which are represented through operations (interface). Service capabilities are defined using Capability Profiles.

3. SOA GOVERNANCE MODEL

The SOA governance model, as defined in this paper, consists of the following parts: Service governance, Business document governance, Service migration, Service portfolio management, Service ownership, Service version management, Service funding model. In the next sessions we will describe these parts.

3.1 Service governance

In SOA Governance Model, one of the most important topics is service governance. Services are the main building blocks of the SOA architecture. Service governance has to assure, that the guidelines are followed, including: service design, service development, service classification, service reuse, service versioning, and service testing. For each service a Service Profile has to be defined. Service Profile defines important information for service design, development, and governance. Each service implements a set of capabilities (functionalities), which are represented through operations (interface). Service capabilities are defined using capability profiles.
Service governance includes the following activities:
Identification and reuse of services. Services are identified in two possible ways, depending on the development approach used for services: top-down, bottom-up. For the top-down development approach, services are identified through the business process modelling. Each distinct automated activity of a business process represents a candidate capability for a business service, or a data/entity service. Business process activities should not map directly to services. The following procedure should

be applied: business process activities should be grouped, business capabilities should be derived, and business capabilities should be grouped and assigned to services. For the bottom-up development, services are identified based on the functional specification of the service.

Business Services should be non-agnostic, have business focus and generalized enough to enable reuse. Technical services should be agnostic or non-agnostic, and should enable reuse through other Technical and through Business Services. Data/Entity Services should have business focus and be reusable at all levels. It is not advisable to design two or more similar (or almost identical) services. Similarity of services should be assessed from semantic perspective, not syntax perspective. Discover and reuse activities should be defined within the design/development process of a service.

*Definition of Service Profil*e. For a new service, which is about to be developed, the Service Profile should be defined. A detailed definition of the Service Profile is required in this stage of the development. Service Profile should define all relevant aspects of a service. A Service Profile should be approved by the SOA Governance board before the service in being designed. The governance approval for the Service Profile is required to assure compliance of all aspects of service profile definition.

Interface design. Service interfaces for SOA are expressed using WSDL. WSDL for the services should be designed according to the activities, defined by the software development methodology (such as RUP/SOMA). The following steps should be followed: 1. WSDL interface design should be split into abstract and concrete interface design. Abstract interfaces includes: operations, messages, types. Concrete interface includes: bindings, and service endpoint. 2. Design of the operations and their signatures in accordance with MEPs (Message Exchange Patterns): synchronous, asynchronous, one-way, request/response. 3. Design of fault signalling. 4. Definition of messages. 5. Definition of types: Always, external definition of types (XML Schemas) should be used. 6. Definition of bindings: SOAP 1.2 over HTTP should be used for interoperability. 7. Definition of service endpoint. 8. Verification of the interface design. 9. Coding of the WSDL. 10. Internal validation of the WSDL.

Design of service QoS aspects. Design of service QoS aspects should address those QoS aspects, which are defined in the Service Profile specification. For the design of QoS aspects, the following guidelines should be followed: 1. Usage of protocols: Service bindings should be provided for all required protocols. A standard binding for all services is SOAP 1.2 over HTTP. For performance reasons, non-standard bindings can be added (such as SOAP over JMS). 2. Security: All services, which require security, should be secured using WS-Security. Based on the depth of the security, which is required, the following possibilities should be used: WS-Security with Username token for level 1 security. WS-Security with X.509 certificate for level 2 security. All services, which require security, should pass security testing procedures (including pen-test). End-to-end security should be used whenever possible. 3. Transactions: For short-running transactions, WS-Atomic Transaction specification should be used for transaction support. For long-running transaction, Compensating transaction model should be used, in compliance with WS-Business Activity. 4. Adherence to standards: Each service should pass the WS-I Basic Profile compliance. Each service, that requires security, should pass the WS-I Basic Security Profile compliance.

Design of the service implementation. A strict separation of service interface and service implementation should be followed at all times. For the design of service, activities defined by the software development methodology (such as RUP/SOMA) should be followed. Activities for the design of service implementation: Detailed design, Identification of implementation approach, Design of components/classes, Design of dynamical behaviour. Internal verification should be made. Documentation should be produced. After this phase, the results should be governed.

Implementation of the service. Based on the decision for the implementation approach, the following applies: Java development process and coding standards should be followed. Internal verification should be made. Documentation should be produced. After this phase, the results should be governed.
Publishing of service in registry and repository (with approval process). Services are important reusable artefacts in the SOA development. To ensure accessibility, discoverability, and reusability, it is essential, that the service artefacts are published in the registry and repository. The service artifacts should be published to the Publication Registry and to the Repository. For the publication, the data from the Service Profile, such as: Name, Keywords, Taxonomy classification. At least the following service artefacts should be published: WSDL, Related schemas, Functional description, Design documentation.

After the services are published to the Publication Registry and to the Repository, the Approval Process takes place. In the approval process, the SOA Governance board verifies the service design and implementation (separately, first design, then implementation), and either approves or rejects it. If the service is rejected, specific comments should exactly describe, what has to be changed. The design and development steps have to be reiterated (repeated), and the comments have to be taken into account.

3.2 Business document governance

In SOA Governance Model, special attention has to be placed on the governance of business documents. Each business document is represented by the corresponding XML document for SOA development and execution. Each XML document has to be specified by the corresponding XML Schema definition (XSD). For each business document a Business Document Profile has to be defined. XML Schema (Business Document) governance includes the following activities:

Identification and reuse of business documents. Business documents are identified in two possible ways. For the top-down development approach, business documents are identified through the business process modelling. For each business process, business documents are identified. Specifically, each business process has to deal with the following types of business documents: incoming business documents (input for business process), outgoing business processes (output for business process), and business processes that are used within the business process. For the bottom-up development, business documents are identified as distinct artefacts, required in services (particularly Business Services) as input or output parameters of service operations.

Definition of Business Document Profile for XML Schema. For a new XML Schema, which is about to be developed, the Business Document Profile should be defined. A detailed definition of the Business Document Profile is required in this stage of the development. Business Document Profile should define all relevant aspects of a XML Schema for the business document. Particularly important are the name, XML namespace usage, version of the business document, and security restrictions. A Business Document Profile should be approved by the SOA Governance board before the XML Schema in being designed. The governance approval for the Business Document Profile is required to assure compliance of all aspects of business document profile definition.

Design of XML Schemas for business documents. XML Schemas for the business documents should be designed. The following steps should be followed: Design of the data elements, including complex types, and simple types. Structuring of the data elements. Verification of the structure of the data elements. Definition of the XML Schema. Internal validation of the XML Schema. XML Schemas should adhere to the following guidelines: The structure and the hierarchy of the XML Schema should be well thought of. It should allow future extensions and modifications. Adequate number of levels should be introduced (no flat schemas). XML Schemas should always be external to the WSDLs. XML Namespace naming conventions should be strictly followed. Naming conventions regarding the names of elements and attributes, complex types and simple types, should be followed. Only those elements should exist on the first level of the XML Schema, which should be validated as positive for the XLM documents.

Publishing of XML Schemas in registry and repository (with approval process). XML Schemas are important reusable artefacts in the SOA development. To ensure accessibility, discoverability, and reusability, it is essential, that the XML Schemas are published in the registry and repository. The XML Schemas should be published to the Publication Registry and to the Repository. After the XML Schemas are published to the Publication Registry and to the Repository, the Approval Process takes place. In the approval process, the SOA Governance board verifies the XML Schema design, and either approves or rejects it. If the XML Schema is rejected, specific comments should exactly describe, what has to be changed. The design and development steps have to be reiterated (repeated), and the comments have to be taken into account.

3.3 Service migration

Service migration is responsible for defining the principles for service migration between different environments. As SOA solutions are modular, their transition through various environments requires special attention and care. Typical configuration of environments: development, test/staging, production. Service migration has to assure, that: Services are migrated between different environments. That appropriate versions of services are deployed on each environment. That no dependencies between environments exist. That endpoint locations are changed in respect to each environment. Service migration governance activities include:

Service deployment guidelines. Service deployment activities should be automated to enable efficient deployment. Automated service deployment also minimizes the risk of errors or mistakes in the deployment. For each deployment, a deployment plan should be prepared. A deployment plan should define: What should be deployed, Which version should be deployed, What is the target environment, What dependencies exist within the deployment package, What dependencies exist outside the deployment package, What test should be executed in order to verify and validate the deployment success.

Service consumer notification. There is a need for notification of service consumers before the deployments. All service consumers of a service should be notified of a new deployment before the deployment, if the deployment is a major version change (backward incompatible). The notification should include the information regarding: The changes in the new major version, The new version endpoint location, The old version endpoint location (if changed), The guidelines for transition to a new version. For minor version changes (backward compatible), a notification of service consumers is optional.

Service lookup and endpoint management. Service lookup is always done through service registry (Lookup Registry). It is not permissible, that two services, or a process and a service, are connected directly using the URL endpoint reference. Each service consumer should bind to a service though the service registry (Lookup Registry) lookup. This way the service registry becomes a central place for service endpoint management. Additionally, virtual service endpoints on the ESB are used in cases, when it is necessary to provide a specific endpoint location, which should be virtualized to enable flexibility.

3.4 Service portfolio management

Service portfolio management is the process by which services and other SOA assets are leveraged to increase reuse, repeatability of delivery, and related cost savings from software maintenance and reduced application development. The objective is to maximize the reuse of services and to optimize the number of services in the portfolio. Service portfolio management should thus provide an optimal number of services and should by all means prevent the service explosion, where too many services would emerge; or "über" service approach, where just a few huge super-services would be developed. The objectives of service portfolio management are (Marks, 2008): managing services as products, maximizing service reuse and enterprise consumption of services, prioritizing and triggering services work orders based on new requirements and demand, representing service portfolio stakeholders through requirements and demand management, enforcement of reuse, and retiring of services based on enterprise consumption, optimizing service consumption, minimizing service costs and delivering maximum total return services to the organization.

3.5 Service ownership

As services become reusable assets, service ownership plays an important role, particularly related to the service life-cycle. Service ownership model should define the ownership of all service artefacts, including service interfaces, business documents, service implementation, test-cases, and deployment procedures. For each artefact it should be clear who is responsible for changes and modifications. Service ownership has the following responsibilities (Marks, 2008): define and manage service portfolio for areas of the service taxonomy assigned to your organization by business or technical domain, manage services with regular release cycles, manage service requirements and demand management for services within your portfolio, clarify services support responsibilities across the service lifecycle from requirements through development, consumption and run time, maintenance and support, ensure appropriate services management tools are in place.

3.6 Service version management

Service version management is particularly important in SOA, as it stretches beyond the development-time versioning. In SOA, run-time versioning is a very important topic. SOA applications are compositions of services and each service can have its own lifecycle, meaning that service versions can emerge independently of the composite application. Service version management requires the definition of major and minor versioning approaches, how service versions will be introduced, and how communication with service consumers will be managed. Service version management includes the following aspects (Jurič, Brumen, & Rozman, 2009; Bliske, 2008): versioning schema, version compatibility rules, versioning of namespaces, versioning of endpoints, and notification of service consumers about version changes. When a new version is deployed and registered in the service registry, all subscribed service consumers should be notified about the new version. The notification should include the information regarding: the changes in the new version, the new version endpoint location, the old version endpoint location, the guidelines for transition to a new version.

3.7 Service funding model

Service funding model is related to the service ownership and should be well defined, otherwise the governance model will not have the necessary means to achieve service life-cycle conformance. SOA funding model should be explicit and should map to the overall governance strategy. The funding model should be agile and allow that the budgets are shifted with the promotion and demotion of services from enterprise to business unit based on consumption demand and enterprise value. Clear ownership of SOA assets should be defined. The SOA infrastructure should be centrally funded.

4. RELATED WORK

SOA Governance is a relatively new topic and a model comparable to the one presented in this paper has not been developed yet. Klischewski et al. (Klischewski & Askar, 2012) have presented a case analysis on practical and strategic level leads to proposing success factors related to linking the methodology of developing G2G services to the overall effort of interoperability governance. In contrast to our work, they have focused on interoperability only and have not defined a full-scale SOA governance model. Hassanzadeh et al. (Hassanzadeh, Namdarian, & Elahi, 2011) have presented a framework for evaluating SOA governance. They used a questionnaire and in order to test the proposed framework, a sample of 16 experts in the field of SOA. The results confirmed the need to SOA governance model. They also proposed a framework, which in contrast to our work does not define the concrete activities. Trana et al. (Trana, Zduna, Holmesb, Oberortnerb, Mulob, & Dustdar, 2012) have presented a model-driven and view-based approach for addressing problems related to compliance concerns. There work has shown explicit links between compliance sources and the corresponding implementations, reports, and documents for conducting many important tasks such as root cause analysis, auditing, and governance. In contrast to our work this article emphasises the importance of SOA governance. We provide a comprehensive model.

5. CONCLUSION

The SOA governance model, presented in this article defines the complete set of activities related to governance in SOA. The model consists of service governance, business document governance, service migration, service portfolio management, service ownership, service version management, and service funding model. For each aspect it provides detailed description of requires activities, principles and objectives. The SOA governance model, as presented in this article, has been verified on 25 large-scale real-world projects, where SOA has been introduced to enterprises. In all cases, the SOA governance model has been able to fulfil the governance objectives and integrate with the IT governance procedures.

REFERENCE LIST

1. Biske, T. (2008). SOA Governance, Packt Publishing.
2. Erl, T. (2007). SOA Principles of Service Design. Prentice Hall.
3. Erl, T. (2009). SOA Principles, An Introduction to the Service-Orientation Paradigm. Retrieved from http://soaprinciples.com

4. Hassanzadeh, A., Namdarian, L., & Elahi, S. (2011) Developing a framework for evaluating service oriented architecture governance (SOAG). Knowledge-Based Systems, *24*(5), 716-730.
5. Jurič, M.B., Brumen, B., & Rozman, I. (2009). WSDL and UDDI Extensions for Version Support in Web Services, Journal of Systems and Software, *82*(8), 1326-1343, doi:10.1016/j.jss.2009.03.0001.
6. Jurič, M.B., Jennings, F., Sarang, P., & Loganathan, R. (2007). SOA Approach to Integration, Packt Publishing.
7. Klischewski, R. & Askar, E. (2012) Linking service development methods to interoperability governance: The case of Egypt. Government Information Quarterly, *29*, 22-31.
8. Marks, E.A. (2008). Service-Oriented Architecture (Soa) Governance for the Services Driven Enterprise. John Wiley & Sons.
9. Trana, H., Zduna, U., Holmesb, T., Oberortnerb, E., Mulob, E., & Dustdar, S. (2012). Compliance in service-oriented architectures: A model-driven and view-based approach. Information and Software Technology, *54*(6), 531-552.