

A Conceptual Framework for Unified and Comprehensive SOA Management*

Ingo Müller¹, Jun Han¹, Jean-Guy Schneider¹, and Steven Versteeg²

¹ Swinburne University of Technology, Hawthorn, Victoria 3122, Australia
{imueLLer,jhan,jschneider}@swin.edu.au

² CA Labs, CA (Pacific), Melbourne, Victoria 3004, Australia
steven.versteeg@ca.com

Abstract. Business requirements, such as regulations and laws, corporate policies, quality of service aspects, etc. affect the management of enterprise-scale SOA systems during different life-cycle stages. Such requirements also induce interdependencies across different elements and aspects of an SOA system. Therefore, a systematic approach to SOA management must be in place in order to effectively govern the development and execution of automated business processes throughout the entire SOA life-cycle in compliance with business requirements. Until now, industry and research have focused on specific management aspects rather than unified and comprehensive solutions. This paper addresses this issue by proposing a conceptual framework for SOA management that combines a micro-kernel/plugin architecture with the concept of management workflows. The micro-kernel incorporates a unified registry/repository model, facilitating the extension of specific management capabilities with plug-ins. Management workflows compose the capabilities of multiple plug-ins into comprehensive management processes that can be linked to events in different places in an SOA system.

1 Introduction

Service-oriented architecture (SOA) constitutes a set of design principles and best practices for guiding the implementation and execution of automated business processes in heterogeneous IT environments. SOA is a complex design exercise that intrinsically ties together business requirements, software engineering aspects, and operational characteristics of IT infrastructures. It results in a multitude of concrete architectures and implementations varying from small sets of Web services to complex enterprise-scale mediation systems that automate business processes across different application silos and organisational bodies.

The key to enterprise-scale SOA is business/IT alignment. In order to be successful, SOA-based business process automation must yield maximum value for a business at the lowest possible costs with acceptable risks and in compliance with business requirements. Business requirements, including regulations and

* This work is supported by the Australian Research Council and CA Labs.

laws, corporate policies, quality of service aspects, etc. affect the management of SOA systems during different life-cycle stages. They also induce interdependencies across different elements and aspects of an SOA system. Therefore, a systematic approach to SOA management must be in place to effectively govern the development and execution of automated business processes throughout the entire SOA life-cycle in compliance with business requirements.

According to a recent Gartner press release [1], business/IT alignment is where many SOA projects fail. The study identifies insufficient SOA governance as an “increasingly significant” risk of failure and predicts that by 2010 “less than 25 percent of large companies will have the sufficient technical and organisational skills necessary to deliver enterprise wide SOA.” Yet, industry and research have not focused on comprehensive and unified governance solutions. Existing SOA management products are fragmented, separate the management of different assets and life-cycle stages, and focus primarily on engineering aspects of runtime service management. Research contributions are either focused on abstract design guidelines and best practices or on specific management aspects.

Business/IT alignment is crucial for enterprise-scale SOA, but yet little understood. We argue that an essential part of a systematic investigation of SOA governance and management issues is the development of a conceptual software framework that leverages required levels of business/IT alignment by unifying the management of relevant assets across the *entire* SOA life-cycle. This paper proposes such a framework that is aimed at enabling agile, unified, and comprehensive SOA management by combining a micro-kernel/plugin architecture with the concept of management workflows.

The rest of this paper is organised as follows. Section 2 outlines the fundamental concepts of SOA governance and SOA management as they motivate our framework. Section 3 proposes our conceptual SOA management framework, followed by an outline of the generic principles of management workflows in Section 4. Current trends in industry and research are discussed in Section 5. Finally, Section 6 summarizes this paper and gives an outlook to future work.

2 SOA Governance and Management

The structural openness at the architecture level, the diversity and complexity of enterprise-scale SOA systems, and the mandatory compliance with business requirements require explicit measures for the effective management of SOA systems. These measures are referred to as *SOA governance*. SOA governance can be typified as means to reconcile the business requirements with the characteristics of the existing IT infrastructure that affect an SOA system (Figure 1). This reconciliation or mediation is also denoted as *business/IT alignment*.

The core activities of SOA governance are a *top-down to-be* analysis and a *bottom-up as-is* analysis. The starting point of any SOA initiative must be the clear definition of relevant assets (*e.g.* business services/processes, business policies, etc.) that reflect functional and non-functional business requirements, including aspects such as accountability, ownership, permissions, and compliance to specific regulations, laws, standards, best practices, or quality of service

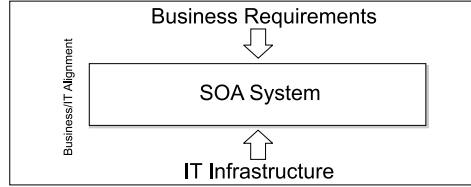


Fig. 1. Illustration of the role of SOA as mediator between business requirements and an existing IT infrastructure

requirements. The identification of business requirements is imperative for SOA because they form the base for the design, implementation, and operation of SOA-based automated business processes. Hence, we assume for the remainder of this paper that all relevant business requirements are specified (in)formally with a business reference model using existing standards such as BPMN or BPEL, and frameworks such as ITIL, COSO, and Cobit.

SOA governance also encompasses an as-is analysis of the static and dynamic properties of the existing IT infrastructure. It is essential to understand and constantly validate the dependencies between business requirements and infrastructure characteristics. Therefore, SOA governance is a continuous process that constantly ‘mediates’ between business requirements and the operational reality in an SOA system. In that sense, SOA governance establishes a control process on top of an SOA system as shown in Figure 2. The SOA control process puts procedures in place for directing the automation, guiding the execution, and controlling the compliance of business processes with business requirements. In case violations are detected, respective procedures define and trigger mitigation strategies.

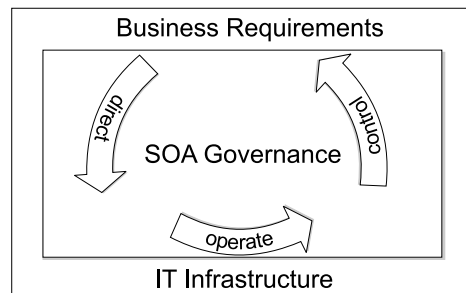


Fig. 2. Illustration of the SOA governance control loop

The implementation and execution of SOA governance procedures with concrete concepts and techniques is denoted as *SOA management*. To ensure effective SOA governance, SOA management must address the following issues:

- *Focus on all relevant assets in an SOA system.* Assets model relevant entities and aspects of an SOA system. For example, business services model

atomic business services and complex business processes. Business actors model existing users and decision makers. Business policies model business requirements. The mentioned assets form only the core set. There are typically more assets in concrete SOA projects.

- *Define and manage the entire life-cycle for every asset.* A simplified SOA life-cycle model contains the four stages: *development*, *deployment*, *execution*, and *evolution*. A life-cycle stage defines a set of artefacts and metadata items that represent an asset. A life-cycle stage also specifies a set of valid operations that can be performed to manipulate the representations of assets. More complex life-cycle models exist. They may vary for different assets.
- *Establish traceability.* The aim of SOA governance is to support business/IT alignment. Therefore, it is essential to manage accountability and compliance of actions that process/manipulate assets in an SOA system (*e.g.* add a service to a repository, enact a business process). Besides the real-time monitoring and validation of such actions and the overall system status, persistent records must be kept in form of logging and audit trails.

Effective SOA management generates large amounts of data (audit trails for tracking actions performed on assets) and includes extensive run-time monitoring (interception and introspection of messages). Therefore, SOA management/governance has a direct impact on the performance of an SOA system, leaving SOA practitioners with a difficult problem. As the Gartner press release [1] points out: “there is no one size fits all” approach to SOA management because “too little or too much governance will kill an SOA project.” Accordingly, an SOA management approach must be customisable to specific business requirements and characteristics of an SOA system.

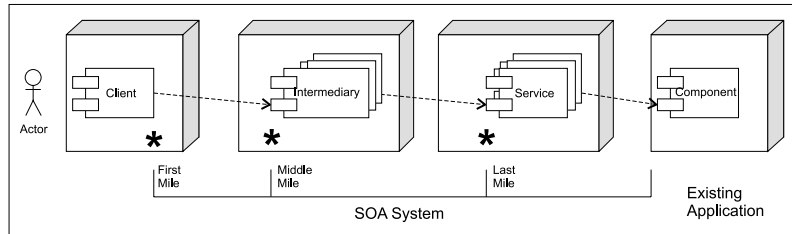


Fig. 3. Simplified UML deployment diagram of a generic enterprise-scale SOA system (The asterisks mark prospective locations for manager components)

Consider a generic enterprise-scale SOA system as depicted in Figure 3. In general, an SOA system can be structured into three sections, denoted as *miles*. The *first mile* spans the section of an SOA system between a client application and a gateway/entry point to the SOA mediation tier. The *middle mile* encompasses a mediation tier of intermediaries (*e.g.* Enterprise Service Bus) that virtualise the location and technology of the actual service components. The *last mile* contains atomic and composite service components that expose and

aggregate the functionality of components of existing applications and software systems. Hence, composite services are an integral part of an SOA system in order to enable the uniform management of atomic services and compositions of services, respectively.

There are a number of requirements for an SOA management approach that can be drawn from the generic layout of an SOA system and the scope of SOA management as outlined before:

1. SOA management is performed in various places in an SOA system. Management facilities are typically incorporated directly at the beginning of every mile of an SOA (as marked with asterisks in Figure 3) to ensure ‘end-to-end’ management of service requests.
2. SOA management is performed by manager components that consist of three logical parts: (i) a *sensor*, a policy enforcement point (PEP) that intercepts messages, (ii) a *decision maker*, a policy decision point (PDP) that analyses messages according to well-defined policies and rules, and (iii) an *actuator* that triggers actions (*e.g.* message manipulation, event notification, mitigation activities) according to the outcome of the decision making.
3. Manager components perform *local* management of ‘what matters and where it matters’ in a lightweight fashion to limit the impact on the performance of a managed SOA system to a minimum.
4. Manager components coordinate and synchronise their activities within and across miles that are affected by service requests. A ‘single form of record’ is required in order to enable manager components to aggregate and share information about assets, their representations, and monitoring/logging data.
5. Manager components can be flexibly adapted to changes of business requirements/the existing IT infrastructure. The separation of business functionality from management functionality and the use of declarative techniques enables non-invasive, extensible, and agile management approaches that reduce the need for extensive coding and recompilation.
6. SOA management approaches are customisable. They can be tailored to the specific scope and requirements of an SOA project. They facilitate the incorporation of various techniques and technologies to solve management aspects in an appropriate manner, *e.g.* utilise management capabilities of the existing IT infrastructure such as federated identity management (to reduce the costs of an SOA project and to capitalise on tested IT infrastructure).

3 Conceptual Management Framework

We propose a generic framework for unified and comprehensive SOA management of enterprise-scale SOA systems based on the requirements outlined above. To meet these requirements, an SOA management system cannot exhibit a monolithic or any other rigid structure that imposes limiting constraints on the target SOA system. Hence, we argue SOA management should be implemented into an SOA system as a sub-system in form of a *hub-spoke architecture* such that the

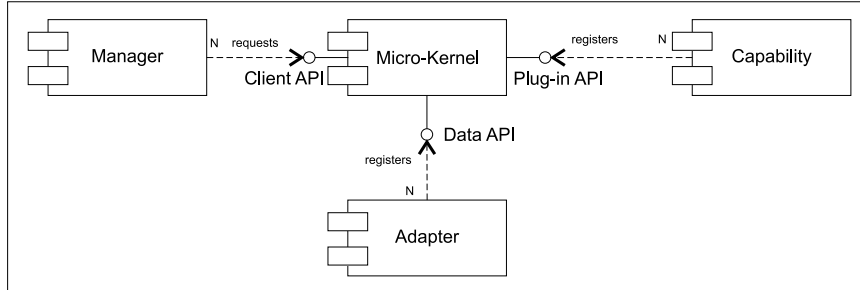


Fig. 4. UML component diagram of the proposed SOA management framework

spokes can be flexibly intertwined with business logic components of the SOA system. Based on the general assertion that service-oriented design principles are good means for developing agile, extensible, and adaptable software systems, we intend to utilise this concept together with a lightweight architectural style in a uniform SOA management framework as shown in Figure 4.

The centrepiece of our SOA management framework is a *micro-kernel* that models a unified *registry/repository* as a ‘single form of record’. The micro-kernel establishes a facade to a shared data space of an enterprise-scale registry/repository. It provides uniform access to information about all relevant assets and their corresponding representations from disparate sources in an SOA system and, for example, enables a comprehensive impact analysis of changes to any relevant asset. Although the micro-kernel represents the central hub component in the hub-spoke architecture, it can be implemented in a distributed manner. The micro-kernel implements functionality (Data API) to (i) register/de-register adapters to access databases/data sources, and their respective data models, (ii) handle discrepancies in the various data models with the help of adapters, and (iii) enable uniform data query and manipulation operations.

The organisation of the registry/repository is oriented on the definition of the ebXML Registry/Repository standard [2] which is aimed at managing any kind of electronic content, that is, XML documents, text files, and arbitrary binary content. An instance of electronic content that represents the whole or parts of an asset (also denoted as *artefact*) is stored in the *repository* section. Moreover, *metadata* (associated with artefacts) is stored in the *registry* section. Accordingly, the repository is a sink for artefacts whereas the registry embodies a cataloguing system that keeps standardised metadata for enabling the effective management and manipulation of the repository content.

The micro-kernel does not provide SOA management functionality other than for querying and manipulating artefacts and metadata items, respectively. Actual management functionality must be implemented as *management capabilities* and must be connected to the micro-kernel as *plug-ins*. Hence, the micro-kernel provides functionality (Plug-in API) to (i) register/de-register plug-ins, (ii) discover and match management capabilities, and (iii) invoke management capabilities.

Management capabilities can access and manipulate artefacts and metadata items in the registry/repository through the Plug-in API of the micro-kernel. The permissions to do so depend on the properties of the actor whose service request triggered a management action, and the affected assets and their state. Permissions are verified by a specific management capability prior to execution. Management capabilities may be of different granularity and complexity ranging from simple event notification, template validation, or the creation of an audit trail record to the integration of external IT management systems (*e.g.* for utilising federated identity management).

The micro-kernel also provides functionality (Client API) for *manager components* to request management capabilities for supporting the monitoring, enforcement, and tracking of the compliance of service provisioning activities with business requirements. Manager components are situated in the SOA system infrastructure next to the service components they manage. They intercept messages to and from the managed service components in a transparent fashion. A manager component comprises of three logical units that provide functionality for different aspects of SOA management: (i) intercepting messages, (ii) performing decision making, and (iii) executing actions on messages and/or in the SOA system. The logical units of a manager component may be distributed depending on particular management aspects, specific business requirements, and the characteristics of the managed SOA system.

Manager components are typically situated in every mile of an SOA system. First mile manager components provide capabilities to client applications *e.g.* to dynamically discover services, negotiate and fix SLAs, validate, manipulate, and transform messages to meet security requirements. They may be embedded into a remote API in order to be installed on client computers outside the actual SOA infrastructure. They are crucial for managing SLAs because they are located closest to the service consumers and, therefore, are most suitable to assess the consumer experience. There is typically a *1-to-1* relationship between client applications and first mile manager components.

Middle mile manager components are located in the mediation tier of an enterprise-scale SOA system which means either at a gateway to the mediation tier or in a container with intermediary components. They are primarily concerned with authentication, authorisation, security, and performance aspects and may be involved in message transformation and routing activities. They may also monitor type and number of specific service requests in order to provide statistical information for optimised routing and utilisation of infrastructure elements. There is typically a *1-to-1* or *1-to-n* relationship between middle mile manager components and intermediary components.

Last mile manager components are placed in container with the service component they monitor. They enforce the enactment of business processes and manage consequential effects (*e.g.* logging statistical data, creating audit trail records, sending event notifications to subscribers) in compliance with relevant business requirements, standards, and technologies that are implied by disparate development teams, existing IT infrastructures, organisational bodies, etc. They

are also responsible for triggering appropriate mitigation actions in case existing software systems and applications that are represented by services malfunction or violate particular business requirements. There is typically a *1-to-1* or *1-to-n* relationship between last mile manager components and service components.

Manager components operate based on *management workflows*. Management workflows represent procedural context-aware knowledge about how to apply and enforce business requirements for the enactment of business services/processes with respect to the properties of the requesting actor. Management workflows can be understood as the glue that ties together relevant business requirements and activities in an SOA system for the implementation and execution of automated business processes. As such, management workflows are the technical means that implement SOA governance procedures in order to ensure the ‘direct-operate-control’ SOA governance control loop. Management workflows utilise the components of the SOA management framework by specifying compositions of management capabilities that define how business requirements are enforced, for example, what information of intercepted messages is required, how it is processed, and what actions are triggered depending on the outcome of decision making processes. The use of declarative techniques for defining management workflows increases the agility of our SOA management approach because it allows the flexible adaptation to changes without coding efforts, re-compilation, or re-starting of affected manager components.

4 Outline of Management Activities

This section outlines the generic principles of how the management workflow concept integrates in an SOA system. For the sake of simplicity we illustrate the intertwined processing of service requests and management workflows with a simplified scenario in which one manager component is assigned to exactly one service component as shown in Figure 5.

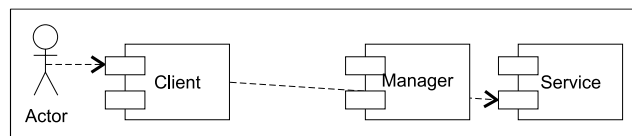


Fig. 5. Example scenario with one client, one manager, and one service component

The management of a request of an atomic service works as follows:

1. An actor triggers a service request message that is sent by the client to an appropriate service component.
2. Before the message arrives at the service component it is intercepted by its associated manager component in a transparent fashion.
3. The manager component inspects header/payload of the message in order to determine its context (properties of actor/requested services). Based on the

Table 1. Sequential management workflow for an incoming service request

Task	Description
1	Retrieve service requester identity and obtain credentials/permissions.
2	Verify requester credentials against security and access policies.
3	Verify request message format based on an XSD template.
4	Validate request message contents according to pre-conditions, e.g. semantics of parameter values and value ranges.

retrieved information, the manager component selects a suitable management workflow from the registry/repository that encompasses management tasks for all relevant business requirements (cf. Table 1 for a simple example).

4. The manager component enacts the management workflow using the message as input. The tasks of the management workflow process data from the message and may alter, add, or remove data from the message. Upon execution, a management task raises a flag in case a policy is violated.
5. If all management tasks have been successfully executed, the message is sent to the service component. If a policy is violated, the manager component does not send the message to the service component but generates and returns an error message to the client.
6. The service component executes the requested business service and produces a response message that is subsequently sent back to the client.
7. The response is also transparently intercepted by the manager component which then performs steps 2 - 4 on the response message according to another management workflow (cf. Table 2 for a simple example). If all tasks of the management workflow have been successfully executed, the manager component passes on the response message to the client. In case of failure, an error message is generated and returned to the client.

Table 2. Sequential management workflow for an outgoing service response

Task	Description
1	Create and store audit trail record.
2	Annotate response message with performance metrics information.
3	Send event notifications to subscribed listeners.

The manipulation of messages with annotations is a powerful means for the manager component to provide additional information or to correct the header/payload of a message to guide the processing of service requests. Message annotations are also a powerful means for exchanging information or coordinating management activities among distributed manager components across and within first/middle/last miles in more complex scenarios.

Hence, message annotations are essential for coordinating the activities between different manager components involved in the management of business processes. Firstly, because we do not assume the existence of additional communication channels between management components. Secondly, because the

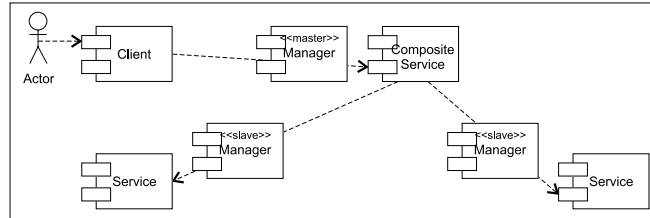


Fig. 6. Relationship between manager components and services in a composite service

relationships between manager components are dynamic depending on specific service requests, message routing, etc. Therefore, the *master* manager component associated with a composite service component coordinates the activities of *slave* manager components associated with component services via message annotations (cf. Figure 6).

5 Trends in Industry and Research

The topic of SOA management and governance marks an active area of development in industry and research in academia. The vendors of SOA infrastructure products have transformed their offerings in the past 5 years from disparate development tools and simple UDDI registries into comprehensive SOA infrastructure and management suites. The key driver of this evolution is the need for products to support close loop management and closer business/IT alignment in order to be beneficial for practitioners. Despite the progress, industry products still expose a number of open issues.

Firstly, full interoperability among and within product suites for truly unified SOA management is not available due to disparate tool sets and standards (vendor lock-in) and growth by acquisition strategies (*e.g.* IBM SOA foundation includes products of the WebSphere, Rational, and Tivoli product families). Various product suites comprise heterogeneous tools of partially overlapping functionality and/or limited interoperability. Thus, manual workarounds are often required that prevent the fully systematic and automated management of assets throughout all life-cycle stages.

Secondly, although a ‘single form of record’ service registry/repository is typically described as the pivotal element for effective SOA management (*e.g.* Strnadl [3]) existing products suites incorporate multiple disconnected local repositories. Moreover, most product suites separate service registry (based on the limiting UDDI standard) from metadata repositories resulting in the poor utilisation of metadata for service discovery and impact analysis.

Thirdly, existing product suites support the aggregation of management capabilities in the form of simple approval/promotion workflows that cannot be substantially extended. As a consequence, different management aspects are typically handled separately.

Fourthly, existing product suites are primarily focused on engineering aspects of service run-time management and policy enforcement. They leverage only limited business/IT alignment in form of business activity monitoring (BAM) that allows the definition and monitoring of business-level key performance indicators (KPI). In general, the systematic mapping between business-level aspects and engineering-level models and methods is still not well understood. The expressiveness of BAM is currently restricted to simple metrics and decision making/decision support. Policy management is concentrated on engineering aspects for implementing security, reliability, and transaction management.

Research has also shown a great deal of interest in SOA management and governance. Papazoglou *et al.* [4] lists the topic of service management and monitoring as one of four key research topics in the area of SOC. And, there already exists a significant body of research results. Firstly, research work investigated specific management capabilities. The interested reader is referred to Papazoglou *et al.* [4] for an overview.

Secondly, research work was conducted on general conceptual frameworks that are aimed at guiding practitioners in designing, implementing, deploying, and operating SOA systems. Although, SOA management is an intrinsic part of these conceptual frameworks, management aspects are not explicitly investigated as discussed by Arsanjani *et al.* [5]. Moreover, these conceptual frameworks typically remain at a high level of abstraction. As an example, consider the business/enterprise architecture level by Schepers *et al.* [6] and Roach *et al.* [7].

Thirdly, a few research activities address the problem of SOA management with the same scope as we do. Arrott *et al.* [8] describes an extended service component concept denoted as *rich service*. A rich service models functional and non-functional aspects of one business service/process. It can be flexibly managed and extended via its internal message bus and plug-in mechanism. However, the focus of this concept on the component level restricts the modelling and coordination of management activities across components. In contrast, our approach is focused on the system level. Siljee *et al.* [9] outlines DySOA which is a management infrastructure that links a set of management components (similar to our manager components) to every service component in an SOA system. DySOA is focused on QoS-driven run-time management of services. Unlike our approach, it does not promote the unified management of all relevant assets throughout the full SOA life-cycle. Belter's work [10] provides a generic high-level framework for service management systems that is centred on a UDDI service registry. The framework is focused on comprehensive management of all relevant SOA assets. However, our framework is more specific. Belter's contribution omits information about how elements of the management system interleave with components of the managed SOA system, how management capabilities are implemented, extended, adapted, and how business policies are enforced.

The latest EU Research Framework Programme (www.cordis.europa.eu/fp7) addresses the current lack of research of the complex problems of business/IT alignment and SOA management/governance by providing significant

funding for a number of large projects including Compass (www.compas-ict.eu), Master (www.master-fp7.eu), and S-Cube (www.s-cube-network.eu).

6 Summary and Future Work

Industry and research have until now only focused on specific management aspects rather than unified SOA management. We argue that a unified and systematic approach to SOA management is essential to effectively govern enterprise-scale SOA systems in compliance with business requirements. Therefore, we have proposed a conceptual framework for fostering unified and comprehensive SOA management as part of a systematic investigation of SOA governance and management issues. Our framework is characterized by its flexible micro-kernel/plugin architecture and the concept of management workflows which enables agile context-based management of assets such as business services/processes, policies, and actors. In future work, we plan to verify our framework by investigating specific management issues and incorporating the results into the framework based on our expertise in configurable adaptive systems [11], software composition [12], and policy-based management of service registries [13] in order to create techniques and tools that support practitioners in their daily work.

References

1. Goasduff, L., Forsling, C.: Bad Technical Implementations and Lack of Governance Increase Risks of Failure in SOA Projects. Online Press Release (June 2007), <http://gartner.com/it/page.jsp?id=508397>
2. Breininger, K., Farrukh Najmi, N.S.: ebXML Registry Services and Protocols, Version 3.0 (2005), <http://docs.oasis-open.org/regrep/regrep-rs/v3.0/regrep-rs-3.0-os.pdf>
3. Strnadl, C.F.: Bridging Architectural Boundaries Design and Implementation of a Semantic BPM and SOA Governance Tool. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 518–529. Springer, Heidelberg (2007)
4. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: a Research Roadmap. *International Journal of Cooperative Information Systems* 17, 223–255 (2008)
5. Arsanjani, A., Zhang, L.J., Ellis, M., Allam, A., Channabasavaiah, K.: S3: A Service-Oriented Reference Architecture. *IEEE IT Professional* 9, 10–17 (2007)
6. Schepers, T., Iacob, M., Van Eck, P.: A Lifecycle Approach to SOA Governance. In: *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC 2008)*, Fortaleza, Brazil, pp. 1055–1061. ACM Press, New York (2008)
7. Roach, T., Low, G., D'Ambra, J.: CAPSICUM - A Conceptual Model for Service Oriented Architecture. In: *Proceedings of the 2008 IEEE Congress on Services - Part 1 (Services)*, Honolulu, USA, pp. 415–422. IEEE Computer Society Press, Los Alamitos (2008)
8. Arrott, M., Demchak, B., Errnagan, V., Farcas, C., Farcas, E., Krüger, I.H., Menarini, M.: Rich Services: The Integration Piece of the SOA Puzzle. In: *Proceedings of the IEEE International Conference on Web Services (ICWS 2007)*, Salt Lake City, USA, pp. 176–183. IEEE Computer Society Press, Los Alamitos (2007)

9. Siljee, J., Bosloper, I., Nijhuis, J., Hammer, D.: DySOA: Making Service Systems Self-adaptive. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSSOC 2005. LNCS, vol. 3826, pp. 255–268. Springer, Heidelberg (2005)
10. Belter, R.: Towards a Service Management System in Virtualized Infrastructures. In: Proceedings of the IEEE International Conference on Services Computing (SCC 2008), Honolulu, USA, pp. 47–51. IEEE Computer Society Press, Los Alamitos (2008)
11. Coleman, A., Han, J.: Coordination systems in role-based adaptive software. In: Jacquet, J.-M., Picco, G.P. (eds.) COORDINATION 2005. LNCS, vol. 3454, pp. 63–79. Springer, Heidelberg (2005)
12. Lumpe, M., Schneider, J.G.: A Form-based Metamodel for Software Composition. *Journal of Science of Computer Programming* 56, 59–78 (2005)
13. Phan, T., Han, J., Schneider, J.G., Ebringer, T., Rogers, T.: Policy-Based Service Registration and Discovery. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 417–426. Springer, Heidelberg (2007)