Characterizing Policies That Govern Service Oriented Systems

by

Priyanka Gupta
B. E., Anna University, 2007

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Supervisory Committee

Characterizing Policies That Govern Service Oriented Systems

by

Priyanka Gupta
B. E., Anna University, 2007

Supervisory Committee

Dr. Hausi A. Müller, (Department of Computer Science)
Supervisor

Dr. Alex Thomo, (Department of Computer Science)
Departmental Member

Abstract

Supervisory Committee
Dr. Hausi A. Müller, Department of Computer Science
Supervisor
Dr. Alex Thomo, Department of Computer Science
Departmental Member

SOA governance not only ensures that the concepts and principles for service orientation and its distributed architecture are managed appropriately and delivered on the stated business goals for services but also controls the evolution of these service-oriented systems. Evolving services must be able to manage their own actions based on high level global business goals and low level local rules. One way to specify such goals is in the form of policies. Policies are operating rules to orchestrate and maintain order, security, and consistency throughout the service lifecycle. In this ubiquitous world of SOA, there are diverse kinds of policies that can be leveraged for governing services. However, these policies are not often documented properly which then leads to redundancy in policy creation and development. To characterize these policies, the thesis first introduces a taxonomy that classifies policies applicable towards the field of SOA governance. This document then identifies the characteristics of policies that are most influential as the organizational maturity evolves. The intended outcome of this thesis is to present the readers with an overall idea of governance policies and their classification as the enterprise system progresses, from being service oriented to virtualized and eventually to a cloud oriented system.

In this thesis, policies that govern service oriented systems are categorized on the basis of their empirically observable behavior and their applicability to phases of the service

lifecycle. This document also recommends policies and their classification, based on enforcement style in the virtualization layer and execution phase in the cloud layer. With this classification, we aim to provide a comprehensive overview of existing policies facilitating policy based governance and evolution in distributed service oriented environments.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I am infinitely thankful to my supervisor, Prof. Hausi A. Müller for his guidance and generosity. I have been very lucky for having pursued my graduate education under his guidance. I can honestly say that he is the best supervisor any student could ever get.

I would like to acknowledge the constant support and guidance I received from my colleagues in the Rigi Lab especially Ron Desmarais, Norha Villegas and Qin Zhu. I would also like to thank my colleagues—Ishita Jain, Qian Yang, Atousa Pahlevan, Xiaochun Li and Sowmya Balasubramanian and faculty members—Dr. Alex Thomo, Dr. Sudhakar Ganti and Dr. Venkatesh Srinivasan for their continuous encouragement.

I am very grateful to my parents and my brother who believed in me and gave me permission to come so far from home to fulfill my dreams. A big hug to all my cousins for showering their unfaltering love and affection on me. I am also very thankful to God for giving me such a wonderful family who has always cared for me and been there for me during thick and thin.

Last but not the least; I am indebted to my spiritual Guru, Sri Sachchidananda Swamiji for having bestowed upon me His grace and blessings.

Dedication

For my family

Chapter 1 Introduction

"If you are moving towards an SOA initiative, if you are embarking on an SOA project, that project will fail without governance."

—Frank Kenney, Gartner

1.1 Research Motivation and Context

The world of computing is now at a tremendous inflection point. The unprecedented level at and speed with which systems, businesses and individuals are able to interact are fuelling tremendous innovation across the globe. The ability to integrate and leverage capabilities from a wide spectrum of resources is extremely important just as the need to align information technology (IT) capabilities with the business for greater speed, agility, and differentiation. Gaining a solid understanding and agreement among stakeholders on how to define, adapt, and maintain business services and processes as well as to create critical links among them is challenging. Managing change and evolution across this ever-increasing web of information, services, and systems is truly an uphill battle. With an effective service oriented architecture (SOA) approach and infrastructure, much common ground can be established in spite of the significant variety of SOA components and applications [Rog08].

SOA is an effective paradigm for organizing a system into services that are running in a distributed execution environment and may be under the control of different ownership domains [IBM07]. SOA is an architecture paradigm that enables business to organize, define, and implement IT projects in such a way that their value can be directly correlated with the business goals and drivers of the enterprise [BLGM08]. Thus, the primary goal

of SOA is to align the business world with the world of IT in a way that makes both domains more effective [HKG]. Moreover, SOA is a way for businesses to model and structure their organizations and operations so that they can efficiently leverage individual capabilities, co-ordinate their efforts to leverage IT profitably, and reduce complexity. With the proliferation of computing devices and enterprise systems, software systems have evolved from software-intensive systems to systems of systems [SEI07], and now to ultra-large-scale systems (ULS) and socio-technical ecosystems [NFG+06]. While today orchestration and composition of services is fairly static, we expect it to become much more dynamic as dynamic service attributes and user contexts become more palatable. Moreover, as environment uncertainty and run-time dynamics become more prevalent in the natural system evolution towards socio-technical ecosystems, flexible and dynamic orchestration will gain in importance.

The distributed nature of services, across various business lines, promises an enterprise seeking SOA transformation great benefits and, thus, forces the chief architects to make SOA governance the overriding concern in system design. Identifying, specifying, creating and deploying services require SOA governance through a very strong and efficient body that can oversee the entire lifecycle of an enterprise's service portfolio [BC06]. Understanding, governing, controlling, and managing environment uncertainty and run-time dynamics of these computing systems—given the onslaught of ever-changing business environment and processes—is a crucial requirement for the survival and success of many businesses [MKS06].

To attain a structured approach to governance, IBM defined the SOA governance paradigm using six important components in implementing governance: governance is (1)

guided by policies, (2) controlled by standards, (3) managed by a responsible party, (4) implemented by some process or procedure, (5) supported by a mechanism or method, and (6) monitored by a set of metrics [BLGM08]. SOA governance is implemented by leveraging, extending, and adapting IT governance mechanisms to ensure that the concepts and principles for service orientation and its distributed architecture are aligned sufficiently, managed appropriately, and able to deliver on the stated business objectives [BC06]. Thus, SOA governance not only affects the quality of business results obtained, but also the effectiveness of the maintenance processes through its deployed SOA infrastructure.

To cope with the ever increasing complexity of managing distributed, heterogeneous computing systems, SOA governance provides the ability to ensure that all of the independent efforts (in the design, development, deployment, or operations of a service) come together to meet the enterprise SOA requirements. Of the six important components of SOA Governance, policies play a key role in enabling governance in any service-oriented environment. By adding policies, points of control and agility for both business and IT are added and SOA is made more consumable, thereby accelerating adoption of SOA solutions. Policies at the highest level can be a simple set of requirements expressed in natural language. So, in general, policies can be considered as requirements that a well-defined set of services must follow or an externally consumable statement of system constraints or capabilities that effect the interaction between a consumer and a provider. As the service lifecycle advances, policies become decomposed from high level goals into low level objectives and machine understandable rules.

Gleason et al. argue that policies are often created in an ad hoc manner and communicated through mechanisms that are incompatible with the underlying service model and architecture [GMP05]. Inspired by this lack of structure, this thesis intends to characterize and categorize the plethora of diverse policies in an enterprise to realize controlled governance in service oriented systems with sustainable benefits to the enterprise.

As the organization adopts the SOA transformation journey, it progressively realizes a need for increasing levels of flexibility, a need to achieve more benefits and higher return on investment, and a need to adapt to a newer paradigm that moves the enterprise to a higher level of effectiveness and agility. Based on the maturity model from the open group, a service oriented system evolves to a virtualized service oriented system and eventually to a cloud oriented system. Research in governance in the virtualization layer and the cloud layer is still evolving; but concepts of SOA governance can be applied directly to the perils of virtualization layer and the cloud as they complement the service oriented architecture. This thesis not only provides an extensive classification of policies in the service oriented environment but also identifies policies and governance requirements in the cloud environment. The shift in focus, observed in the policy taxonomy, as the organization evolves is articulated. Policy taxonomy serves multiple purposes. Effectively implemented, the taxonomy provides the cornerstone of an SOA strategy, supports linkage to other IT infrastructure issues (such as security) and can impart broader enterprise value. In addition to providing policy taxonomy and an extensive classification, we present a case study in Chapter 6. This case study is an extension to IBM's SOA case study [Kee09] by including governance components into

the enterprise's (as in the case study) SOA adoption model. We incorporated governance policies into the case study based on Oracle's leverage points [ACH+07] and our IBM-SGMM [BLGM08] based policy classification. We conducted this case study to test the suitability of our classification in a practical scenario.

To structure and deploy governance, there is a critical need for defining and enforcing a body of policies across every level of an enterprise system. Our case study exhibits the addition of a virtualization layer and a cloud layer into the enterprise's SOA model and the policies that are applicable in those layers based on Chapter 5.

1.2 Research Methodology and Approach

Determining a correct solution to bring about effective governance is dependent on the organization and their business needs [Woo06]. Efforts are laid to make governance more appropriate and effective. The methodology used in this research is quantitative and hence the design of this survey follows the format of a *quantitative method*. The purpose of our research is to achieve control, management and evolution of service oriented system by classifying policies based on service lifecycle. To achieve this classification, information is gathered from secondary sources such as earlier research and mass media.

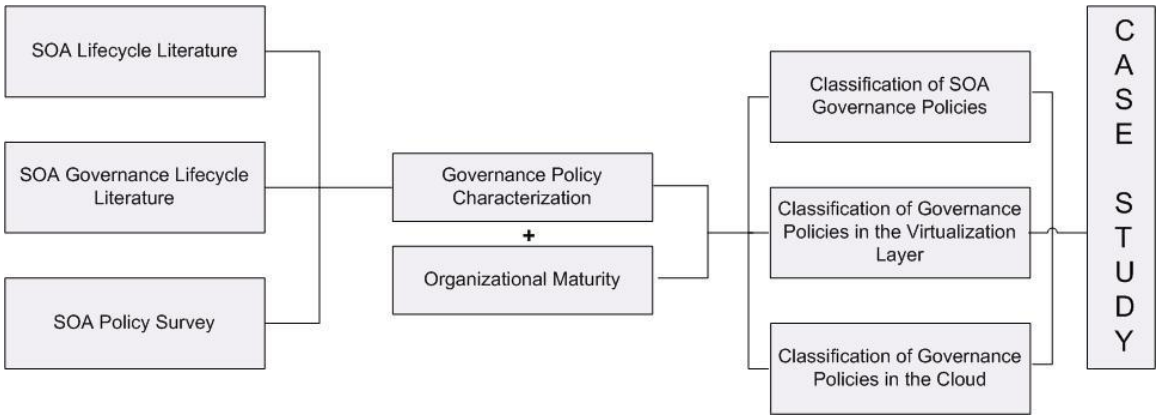

Figure 1: Research Approach

To characterize policies that govern service oriented systems, we collected and analyzed a large collection of papers dealing with governance and policies for service oriented systems. The lifecycle phases used in the classification are based on the IBM-SGMM and other SOA lifecycle models including SOMA, SOAF, SOMA-ME and SODDM. The categorization of policies reveals how different parts contribute to the governance architecture as a whole. The papers are selected on their relevance towards service governance and policies. The policies are then classified based on their common attributes and the phase of service lifecycle they most precisely belong to. The first level of categorization of policies is derived from service lifecycle phases derived from Governance Lifecycle IBM-SGMM [BLGM08]. The other deeper levels of classifications are obtained from both, common characteristics they comprise and other SOA lifecycles as in [Mar08, BLGM08]. The articulation of this survey on the categorization of policies would reveal how the parts (policies in each phase) fit into the whole governance architecture by emphasizing each part separately.

1.3 Contributions

In order to provide an understanding of the existing policies, this thesis has to cover a number of different topics. This thesis identifies the following contributions in an increasing order of significance.

Handbook of policies for governance officials: On the whole, this thesis aims to provide governance officials an extensive set of policies from which they can pick and choose ones that would be most suitable for their governance processes, be in the service environment, abstraction layer or the cloud system.

SOA governance policies survey: to the author's best knowledge—provides the most comprehensive coverage of SOA governance policies in the SOA governance domain to date. Chapter 4 discusses the available policies in the field of SOA governance.

Governance in the cloud and virtualization layer: A list of available policies and governance requirements in both the layers is stated extensively. Research in both the layers is still in its nascent stages and an attempt to identify policies that would be most appropriate for these two layers is stated clearly. Chapter 5 provides in great detail governance requirement and policy categorization in both these layers.

SOA governance policy taxonomy: The policy taxonomy, that forms the basis of policy categorization in the virtualized and cloud layer, is based on the work of other researchers. Chapter 3 discusses this in great detail.

Governance case study: Chapter 6 investigates the application of our governance policy classification on an existing practical implementation of SOA transformation in an enterprise. IBM provides such a case study [Kee09] and we build upon the same by identifying governance policy leverage points as stated by Oracle [ACH+07]. The case study also provides an overview of how governance policies are implemented across the virtualization layer and the cloud layer as the enterprise advances towards adopting the two layers.

1.4 Thesis Outline

We provide the readers with a brief overview about the thesis in Chapter 1. Chapter 2 introduces Service Oriented Architecture (SOA) along with its advantages and limitations. The thesis discusses the purpose of SOA Governance and how SOA can be brought to perfection by implementing governance techniques. This chapter also presents

about two important SOA governance approaches by IBM and Oracle and also about the seven most important components of governance. Chapter 3 describes in greater detail one key governance component—Policy and introduces a new taxonomy to characterize SOA governance policies. Chapter 4 sheds light on the various categories of policies at the SOA level in detail. Chapter 5 discusses organizational maturity and its effect on SOA governance. This chapter also describes how governance and governance policies evolve as the virtualization layer and the cloud infrastructure are adopted by the enterprise. Chapter 6 implements an exploratory case study based on IBM's publication featuring a case study to address the scenario of transforming the company's siloed business model to a service based model. Our case study extends IBM's case study by introducing governance components in their SOA adoption model. This chapter describes the application of policies when the enterprise incrementally adopts all the three layers (i.e., services layer, virtualization layer and cloud layer). Chapter 7 draws some conclusions and renders future work.

Chapter 2 Background

"Without an effective governance model, your dreams of SOA nirvana can turn into nightmares of down systems, high development costs, unmanageable production environments and unhappy customers."

—Mike Kavis, Catalina Marketing

This chapter provides background information on Service Oriented Architecture and the various approaches proposed for developing a service oriented system. Following this, a need for governance and governance approaches are explained in detail. Because SOA spans the entire enterprise, the scope of governance policies employed must pertain to that particular enterprise asset. Hence to determine the scope of governance policies ranging across the entire enterprise, Luckham's model [Luc01] of a layered enterprise system is adopted based on which the policy scope can be narrowed down to the enterprise layer it most aptly belongs to. A brief introduction of Luckham's model [Luc01] of a layered enterprise system is given.

2.1 Service Oriented Architecture and Lifecycle Approaches

SOA is an approach to IT solutions that is driven by the organization's needs. IT solutions are delivered using small modules called "services" that support the explicitly described needs of the organization. These IT services are designed for use by more than one IT solution. Over time new IT solutions can be built from already existing services. Existing applications can expose themselves as a set of IT services, whilst new IT systems may be implemented in smaller modules with IT service interfaces included in the original designs.

Change is a constant in any environment, and being able to respond to change without investing considerable time and effort in modifying IT systems is the primary benefit of SOA. Furthermore, SOA enables the organization to not only transform internal systems to be more service oriented, but also permits collaboration amongst enterprises, other partners as well as third-party provisioning of useful business functions in a seamless, non-disruptive fashion [IMS09].

A number of preliminary methodologies have emerged to address the huge demand for process guidance and proven best practices in SOA projects. Most of the proposed approaches for service oriented development aim to support the full SOA lifecycle, including planning, analysis and design, construction, testing, deployment, and governance activities, while others limit their scope to a subset of these phases, such as analysis and design.

Service-Oriented Analysis and Design (SOAD) [CK07]: SOAD proposes elements that should be part of a service-oriented analysis and design methodology. It also introduces SOA-specific techniques, such as service conceptualization, service categorization and aggregation, policies and aspects, meet-in-the-middle process, semantic brokering, and service harvesting.

IBM Service Oriented Modeling and Architecture (SOMA) [AGA+08]: SOMA is a full-blown modeling methodology by IBM consisting of three steps: identification, specification, and realization of services, flows (business processes), and components realizing services.

Service Oriented Architecture Framework (SOAF) [EAK06]: SOAF consists of five main phases: information elicitation, service identification, service definition, service realization, and roadmap and planning.

Service Oriented Unified Process (SOUP) [Mit05]: Its lifecycle consists of six phases: incept, define, design, construct, deploy, and support.

Papazoglou et al. [PVDH06] examine a service development methodology from the point of view of both providers and consumers, which attempts to cover the full SOA lifecycle. The methodology utilizes an iterative and incremental process that comprises one preparatory and eight distinct main phases.

The lifecycle phases specified in each of the methodologies have their own advantages. For our survey, information on these various approaches is extremely important because the classification follows a lifecycle based approach. Moreover, these methodologies impart finer details on various phases of the service lifecycle. With this information about the different phases, it is easier to identify the phase in which a particular policy will act as a control point.

2.2 Need for Governance

Often, new service-oriented projects are undertaken without explicit SOA governance in place. This works if the scope of the project is small, but, as the architecture grows to include new service providers and consumers, the overall reliability and predictability of the architecture begins to decline; this is typically the result of a lack of governance. For example, a lack of attention to service boundaries may have resulted in duplicating some old service functionality in a new service. Or perhaps the demands placed on a certain service may have increased dramatically because of a new service consumer, and this

may have been unaccounted for—resulting in the inability of the service to meet its Service Level Agreement (SLA). Furthermore, as enterprise services evolve and new service versions are deployed, careful consideration must be taken each time to assess the impact of the changes to other services and to provide backward-compatibility. In order to prevent many of these potential problems from occurring, some amount of formal governance is critical even in a new service-oriented project to ensure delivery of consistent and reliable services [IMS09]. Some of the potential consequences of an ungoverned SOA are: 1) Lack of interoperability, thereby, creating silos of business services; 2) Escalations in support costs and finally; and 3) An overall SOA failure by letting chaos reign and perpetuating a "garbage in, garbage out" environment.

2.3 Governing Dynamics

SOA offers significant advantages, but it puts additional demands on visibility, control, and overall governance. In essence, SOA governance may be viewed as a management architecture: a framework that blends the flexibility of SOA with the control and predictability of a traditional IT architecture. SOA introduces many independent and self-contained moving parts—components that are typically widely reused across the enterprise and are a vital part of mission-critical business processes. SOA has the potential to introduce risk and, without proper governance, can disrupt business processes and create significant inefficiencies. To be successful, a SOA governance strategy must be applied during the initial deployment of SOA and continued until it retires or reforms itself. The goal must be to establish a framework for assuring service quality and engendering trust between service providers and consumers as services progress through their lifecycles [Hyn06].

Business services are expressly designed to align with the business needs. With the advent of business services, enterprises could orchestrate these services into composite applications and implement workflows. While this new set of technologies solves the original problems of proprietary, tightly coupled, fine-grained systems, it introduces a new challenge for the enterprise: a lack of control over change. To cope with the ever increasing complexity of managing distributed, heterogeneous computing systems, systems themselves should be able to manage their own behavior in accordance with high-level objectives from human administrators [Hyn06] and enterprise level business requirements. Identifying, specifying, creating and deploying services require SOA governance through a strong and efficient body that can oversee the entire lifecycle of an enterprise's service portfolio [Hyn06]. Understanding, governing, controlling, and managing environment uncertainty and run-time dynamics of these computing systems—given the onslaught of the ever-changing business environment and processes—is a crucial requirement for the survival and success of many businesses [MKS06].

SOA governance is implemented by leveraging, extending, and adapting IT governance mechanisms to ensure that the concepts and principles for service orientation and its distributed architecture are aligned sufficiently, managed appropriately, and able to deliver on the stated business objectives [BC06]. Thus, SOA governance not only affects the quality of business results obtained, but also the effectiveness of the maintenance processes through its deployed SOA infrastructure. The focus of SOA governance framework is management and control of service lifecycle. It manages the composition of various services and therefore ensures that the composite application performs in the desired manner. Managing the SOA lifecycle is an intrinsic part of SOA governance. It

also includes mechanisms for enforcement and monitoring of various business services. SOA based enterprises usually establish governance mechanisms for effective usage of services and to satisfy the SLAs [MR07].

2.4 Dimensions of SOA Governance

To attain a structured approach to governance, IBM defined the SOA governance paradigm using six important components in implementing governance: governance is (1) guided by policies, (2) controlled by standards, (3) managed by a responsible party, (4) implemented by some process or procedure, (5) supported by a mechanism or method, and (6) monitored by a set of metrics [BLGM08]. These components are used to control, manage, and maintain distributed systems of services and resources to optimize business and evolution objectives. In particular, it involves the run-time monitoring of SOA processes to gather key measures such as execution time, availability, throughput, latency, or resource consumption. It also controls dynamic service selection as well as the evolution of services, metadata, and applications in an organization's service-oriented architecture.

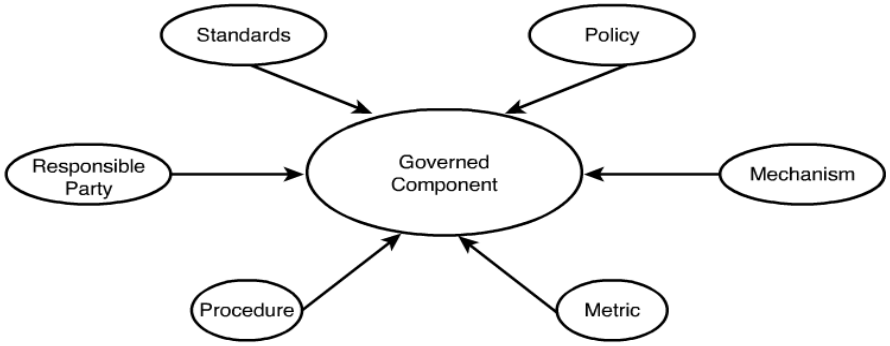

Figure 2: SOA Governance Paradigm [BLGM08]

2.5 Key Approaches for Implementation of Governance

Major SOA solution providers emphasize the SOA governance pillars differently. IBM advocated that implementing SOA governance and management requires the consideration of three pillars: people, process, and technology [MGN+09]. Even though the entire governance lifecycle is usually described, for the purposes of this thesis we consider the IBM SOA publications that lean towards a process-centric view of SOA governance. Oracle purports a more management-centric and policy-centric view. The reviewed Oracle approach is mostly applicable to providing a bridge from the design stage view to the execution view and is particularly useful for validating a SOA governance approach during its installation and deployment from the policy point of view. Both the perspectives constitute useful checklists for the requirements engineering phase of a SOA governance system. Of course, the SOA solution providers concentrate first and foremost on using governance to optimize business objectives.

2.5.1 The IBM Perspective

IBM defined a SOA governance framework as a lifecycle consisting of four phases—plan, define, enable, and measure—as depicted in Figure 3 [Woo06]. This framework controls the SOA lifecycle which itself consists of four phases (i.e., model, assemble, deploy, and manage) [BMT06]. The activities of the four phases that have to be carried out as part of the SOA governance lifecycle are as follows:

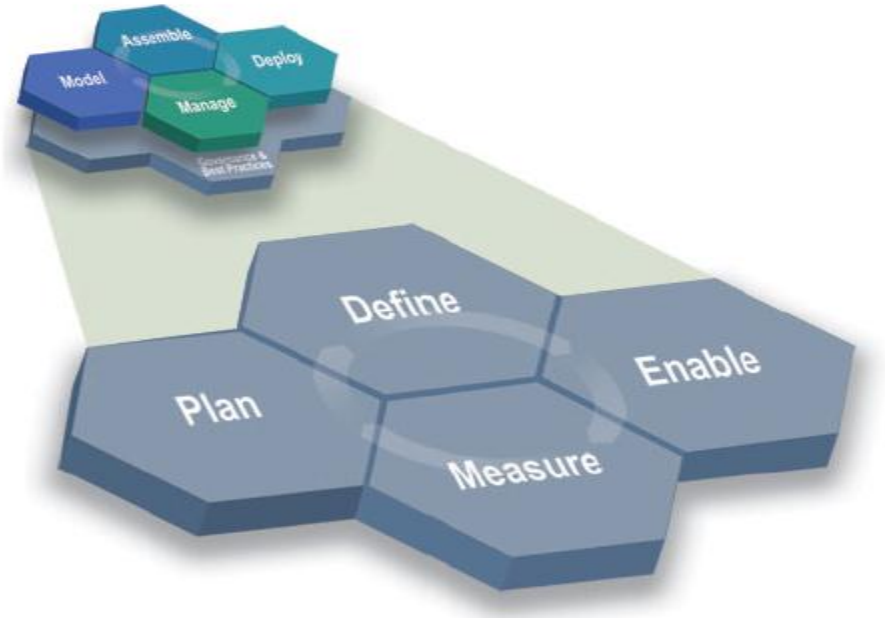

Figure 3: IBM SOA governance lifecycle [Woo06]

Plan: In this phase, understanding the overall scope of governance within the organization is important. Moreover, areas of improvement (i.e., policy and process improvement) are identified in this phase. This phase includes:

a) Defining or refining the overall business vision, strategies, and objectives.

b) Reviewing and revising IT and SOA alignment and capabilities.

Define: After identifying the scope and improvements, this phase deals with defining or modifying the governance arrangements and mechanisms including:

a) Establishing the SOA and SOA governance centre of excellence.

b) Defining policies and mechanisms to guarantee service level agreements.

c) Establishing funding mechanisms.

d) Conducting staff training.

Enable: All the solutions determined in the above two phases are deployed and put into action. One of the most important observations in this phase is the deployment of

SOA with mechanisms to enforce policies. A baseline is established for policy and process improvement. Infrastructure and applications are instrumented to support the *Measure* phase.

Measure: Governance decisions made in the *Define* and *Enable* phases are monitored in this phase. Selected measures of a managed process are fed back to the controller. The controller then tries to optimize business objectives by adapting policies and processes.

This enhances the effectiveness of the governance framework and, thereby, allows for business tuning and process improvement.

IBM's SOA Governance and Management Method (SGMM) presents a framework for the definition, design and implementation of SOA governance and service lifecycle management [BLGM08]. The governance processes of this method are depicted in Figure 4 together with the mechanisms and components that are needed to implement and manage them. The four processes—service strategy, design, transition, and operation—correspond to the four phases of the SOA governance lifecycle.

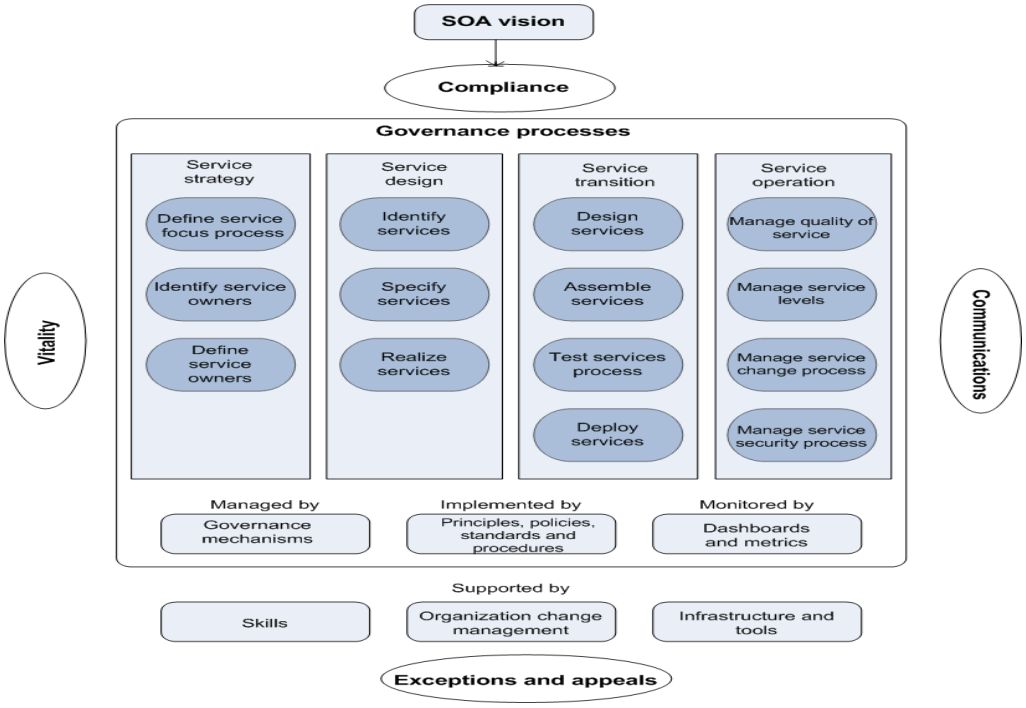

Figure 4: Key components of IBM's SGMM [BLGM08]

2.5.2 The Oracle Perspective

To meet the business goals in the context of enterprise architecture and SOA, Oracle enforces policies across several dimensions including information, people, architecture, technology, infrastructure, finance, portfolios, projects, and operations (cf. Figure 5). For each dimension, Oracle defines key leverage points to be governed. The policies, which are defined in documents and enforced through technology, define the role of governance to achieve the business and IT goals and alignment between them. To facilitate enterprise-wide SOA adoption, SOA governance experts at Oracle recommend organizing policies and processes around these leverage points. If all the appropriate policies are put in place, SOA goals can be governed wisely and effectively [ACH+07].

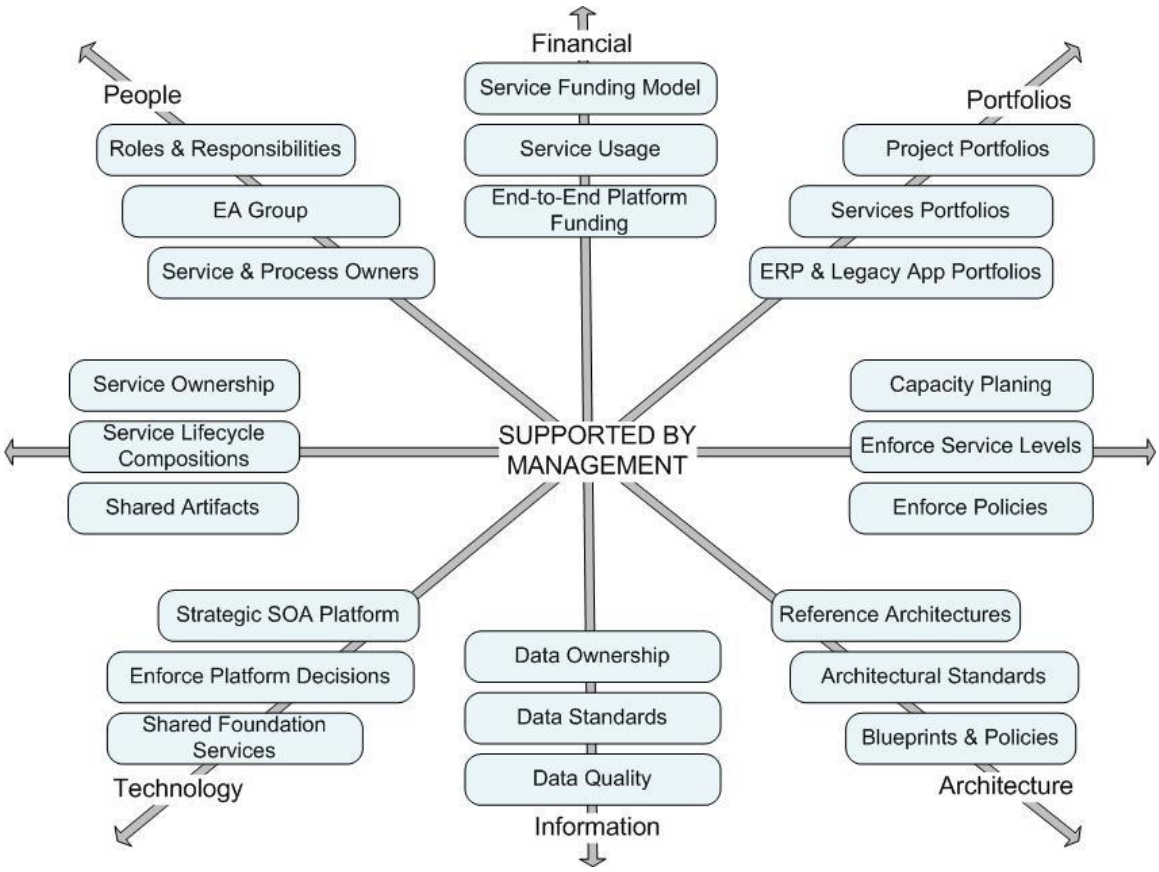

Figure 5: Oracle leverage points for SOA governance policies [ACH+07]

2.6 A Layered Enterprise System—An Introduction

"Enterprise systems feature a set of integrated software modules and a central database that enables data to be shared by many different business processes and functional areas throughout the enterprise."

—Kenneth C.Laudon and Jane P.Laudon

The Core Enterprise System is the set of custom applications and bought-in packages that form the heart of the IT system and that run on distributed servers and/or mainframes. Security and scalability are of major importance here. The Core Enterprise System is protected by firewalls and other security mechanisms. The code that implements the core business logic behind web services, and also manages core data

assets, is here. Enterprise systems are distributed, layered systems designed for controlling complexity [Luc01]. Figure 6 represents the layered architecture of an enterprise system.

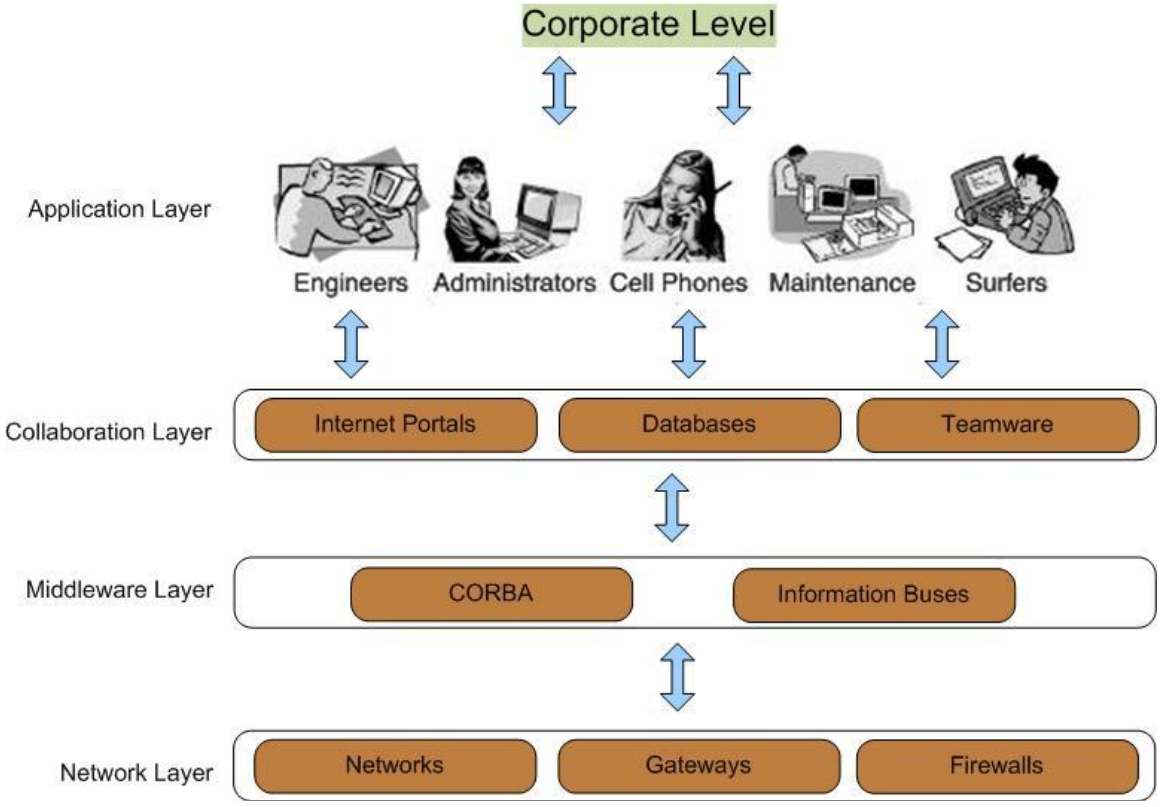

Figure 6: A Layered Enterprise System [Luc01]

The following section presents the four most important layers of an enterprise system.

Application Layer

This layer is where organizational planning takes place. Enterprise planning is the most prominent activity in this phase. This layer comprises of user interfaces to the enterprise's services, planning and accounting services, spreadsheets, calendars, browsers, communication devices, system administration tools, etc. This layer is where human users work and conceptualize their goals. This layer is where high level goals are framed and passed onto the lower layers for realization.

Collaboration Layer

This layer contains components that facilitate in making the applications available to the users. Databases, servers, operating systems (to a certain extent), application service providers, business rules engines, sophisticated collaboration tools are all part of this collaboration layer.

Middleware

This layer enables communication in the collaboration layer. The middleware layer includes Object Resource Brokers, information buses, communication layers on top of basic networks enabling communication between applications and application services.

Network Layer

This layer handles information transfer from within/across an enterprise. Management of this layer includes tools that track resource consumption, enables alerts/warnings if the resources malfunction and tools for security and privacy.

Over the past few decades, enterprise requirements have expanded and traditional enterprise architectures have struggled to meet the new flexibility and interconnectivity requirements, while also being costly to expand and maintain. SOA effectively modernizes the relationship between business and IT infrastructure by implementing standalone services that perform definite business functions and possess the capability to exchange data with other services. SOA imparts tremendous focus on service to traditional IT systems in the enterprise, but it does not abolish the need for enterprise architecture. Differently put, it is still essential to comprehend application and data capabilities and integration requirements between the various IT systems at the enterprise. SOA facilitates the construction of useful services over the latent IT systems

by providing new ways for integrating the various underlying systems in support of a services paradigm. Presence of enterprise systems significantly simplifies the creation of service-oriented systems with its resultant benefits [IMS09].

With SOA prevailing across the enterprise system, SOA governance must be dominant wherever the enterprise assets are acquiring the "serviceable" status. In this thesis, the layered enterprise system has been utilized to determine the scope of governance. This document focuses on governing enterprise transformation through the application of policies; hence the enterprise layers can be utilized most appropriately in the scope determination of governance policies.

2.7 Organizational Maturity and its Influence on Governance

Organizational maturity is a progressive realization in a service-oriented world, moving from tight integration to loose coupling and provisioning of business and information capabilities as a service. The OSIMM has seven dimensions across seven maturity levels (cf. Figure 7). Each maturity level represents a significant increase in the level of maturity necessary to realize service orientation [Ope09]. These multiple dimensions, as shown in Figure 7, along which service integration maturity can be quantified, provides areas of focus for future efforts so that SOA benefits are thoroughly realized. These dimensions are indicative of the various enterprise components that SOA influences. Based on the maturity model, maturity levels 4 and 5 indicates the existence of service oriented behavior in the enterprise system and that composite applications are built from loosely-coupled services. The services interoperate across all of the layers of an enterprise and even across enterprises. These services are usually managed by applying SLA objectives to the most relevant organizational segments. Also, at this level of service

maturity a business process can be compounded from a set of collaborating services through the use of modeling languages thereby facilitating more interaction between developers and the business analysts. However, services at this level are more closely coupled with the infrastructure.

At level 6, the services are provided through a level of indirection. The service consumer invokes the service through the invocation of a virtual service or across a virtualized layer. The virtualization imparts more less coupling from the infrastructure on which the service executes, conceding more opportunities for service composition. Although virtualization can be utilized in systems that are not service oriented, this level extends the concept and advantages of virtualization to services.

At level 7, service assembly may be performed at run-time, either assisted by the business analysts via suitable tooling, or by the system itself. The services are not scoped to an enterprise but can be acquired from third party providers and utilized on a pay as you go basis. For example, during initial stages of organizational maturity, information sharing is restricted to a set of applications with point to point integration. As the organization advances, information is available as a service and is provided to the consumers through a standardized interface [Ope09].

SOA provides enterprise services to a heterogeneous ecosystem of administrators, partnering organizations, researchers, engineers, security and maintenance personnel and software and hardware assets of the enterprise. Because of the changing needs of this ecosystem, most enterprises are obliged to manage a complex heterogeneous environment comprised of new and legacy IT systems. As the number of enterprise services (and supporting IT systems) increase, the overhead of managing their complexity

also increases. Governing these enterprise services over time approach becomes extremely complex, making it difficult for an organization to analyze the impact of replacing an enterprise service, upgrading the IT system that supports it, or opening the service up to new and different users. Organizations are often hesitant to undergo any transformation because of this underlying complexity, so they become less responsive and less able to keep up with the demands of their user ecosystem.

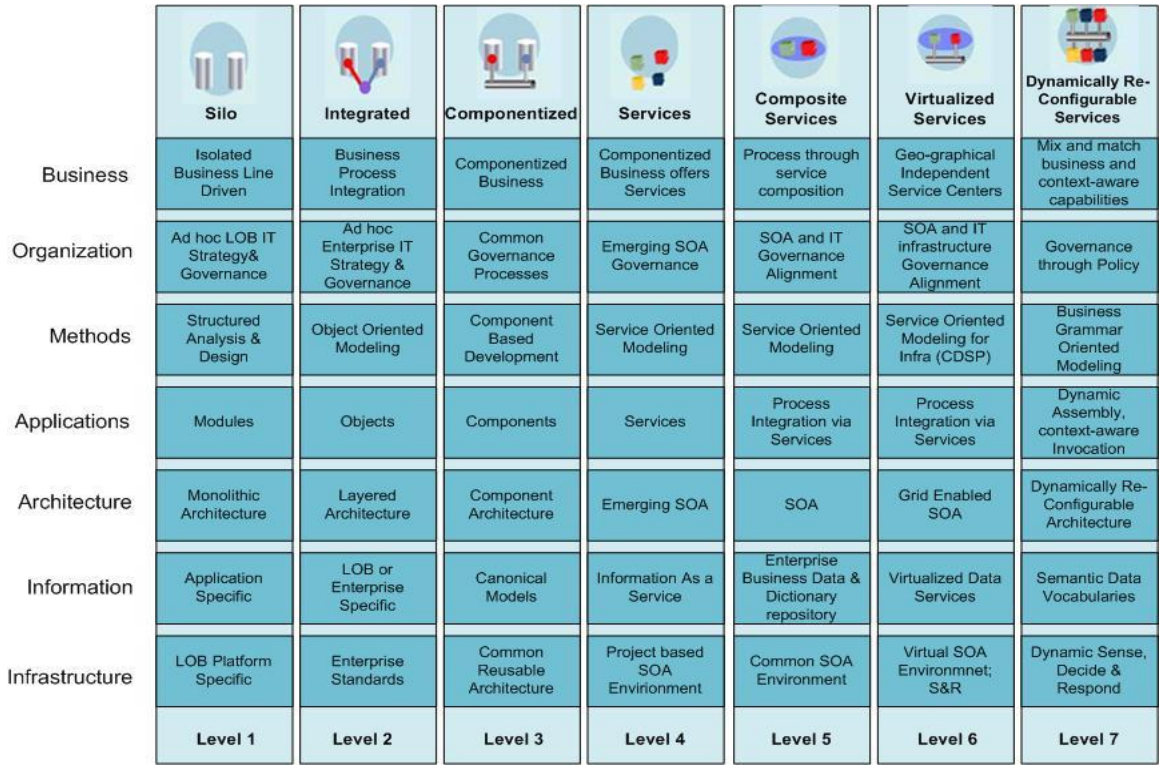

Figure 7: The Open Group's Service Integration Maturity Model (OSIMM) [Ope09]

It is instructive to observe that, in most cases, the enterprise exhibits some form of governance over the enterprise architecture albeit informal. This informal governance is clearly applied inconsistently and proves to be insufficient as the scope of the architecture expands. In order to ensure that the enterprise reaps the benefits of their stated and evolving business goals, governance must evolve appropriately. For example, certain

straightforward situations might require manual intervention to systematically verify for stereotyped issues faced during architectural maturity; otherwise, more extensive governance is usually needed. Ideally, an organization must be able to practice SOA governance to an extent that is indispensable to administer the architecture in its current state and endure flexible growth and smooth operation into the future. Therefore the goal of any enterprise must be to ascend its SOA governance methodologies to be on par with the current architectural state, which is an evolving target [IMS09]. This thesis addresses how governance and its components would grow along with a typical SOA maturity progression and confers the governance requirements and the policies as they become pertinent.

2.8 Summary

We provide a comprehensive view of SOA governance in this chapter. In particular, we explained six important components as defined by IBM. The two most important and widely accepted approaches towards SOA governance are elucidated. One of the approaches by IBM offers a process perspective towards governance whereas Oracle offers a solution that assumes a policy perspective. Finally, we presented an enterprise system and its layers that form the basis of scope determination of policies specified in the following chapters.

Chapter 3 Policies

"By definition, a government has no conscience. Sometimes it has a policy, but nothing more."

—Albert Camus

3.1 Introduction

Policies can be considered as requirements that well-defined set of services must follow or an externally consumable statement of system constraints or capabilities that effect the interaction between a consumer and a provider. Policies can be of three basic types: *Absolute*, *Derived* (from business goals) [Mar08] and *Compositional* (combination of absolute policies or absolute and derived policies). Viewing policies from the perspective of service development phase, policies are broadly classified into *design-time policies* (enforced before the service is deployed) and *run-time policies* (enforced while the service is in operation or in the service registry). The most successful SOA initiatives implement a rigorous SOA governance program—the interaction of policies with mechanism, standards, metrics, people and procedure to deliver SOA benefits, ensure SOA success, and facilitate evolution [ACH+07]. It is the oversight of and the interaction between these components that ensures that SOA provides a powerful framework for matching needs and capabilities and for combining capabilities to address those needs.

Policies play a key role in enabling governance in any service-oriented environment. By adding policies, points of control and agility for both business and IT are added and SOA is made more consumable, thereby accelerating adoption of SOA solutions. Policies at the highest level can be a simple set of requirements expressed in natural language. So, in

general, policies can be considered as requirements that a well-defined set of services must follow or an externally consumable statement of system constraints or capabilities that effect the interaction between a consumer and a provider. As the service lifecycle advances, policies become decomposed from high level goals into low level objectives and machine understandable rules. A hierarchy of policies embodies the decisions that need to be made for effective service management. Higher level policies define the rules for lower level policy adaptations to optimize business and evolution.

3.2 Policy—An Important Component of Governance

The six components defined in the previous section—*policies*, *procedure*, *mechanism*, *standard*, *metrics* and *people* together with the technology pillar—constitute different perspectives on the SOA governance lifecycle from the early planning stages to the steady-state run-time operation. The entities and relationships introduced also constitute design views for SOA governance methodologies. On the one hand, SOA governance ensures that the concepts and principles for service orientation and its distributed architecture are managed appropriately and are able to deliver on the stated business objectives. On the other hand, SOA governance controls the evolution of these service-oriented systems. Naturally, most research dealing with SOA governance methodologies and best practices concentrate on how to deliver and satisfy business objectives and often do not cover the maintenance and evolution challenges. While not readily apparent, closer inspection reveals that the implementations of such governance methodologies embody classic software evolution concepts, such as levels of indirection and feedback loops [MGN+09].

3.2.1 Governance Feedback Loops

To optimize business functions, service-oriented systems must be extensively instrumented to keep track of useful figures such as *execution time*, *availability*, *throughput*, *latency*, or *resource consumption*. SOA governance policies use such figures to identify trends, adjust policies and processes, and manage service levels accordingly [LS08]. In particular, a controller monitors and controls the effects and outcomes of policies and processes using a feedback loop as depicted in Figure 8. Selected results of SOA policies and processes are fed back to a SOA controller which then decides whether there is a need to adapt policies and/or processes in order to optimize outcome.

IBM researchers introduced the notion of an autonomic element as a fundamental building block for designing self-adaptive and self-managing systems such as SOA governance controllers [Mil07, Com04]. An autonomic element consists of an autonomic manager (i.e., controller), a managed element (i.e., process), and two manageability interfaces. The core of an autonomic manager constitutes a feedback loop, referred to as monitor-analyze-plan-execute-knowledge (MAPE-K) loop, as depicted in Figure 8.

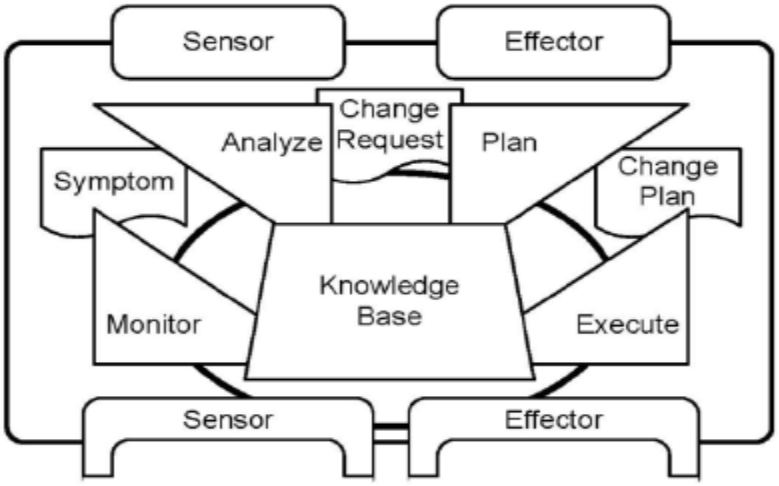

Figure 8: Autonomic Manager

The manager gathers measurements from the managed element as well as information from current and past states from various knowledge sources via a service bus and then adjusts the managed element if necessary through a manageability interface (i.e., the sensors and effectors at the bottom of Figure 8) according to the control objective. Note that an autonomic element itself can be a managed element the sensors and effectors at the top of the autonomic manager in Figure 8 are used to manage the element (i.e., provide measurements through its sensors and receive control input (e.g., rules or policies) through its effectors). If there are no such effectors, then the rules or policies are hard-wired into the MAPE-K loop. Even if there are no effectors at the top of the element, the state of the element is typically still exposed through its top sensors [MGN+09].

Similarly, SOA governance management processes can also be characterized using the four autonomic phases (i.e., monitoring, analyzing, planning, and executing). Note that the autonomic architectural pattern applies to both design-time and run-time governance. Often only half of a MAPE-K loop (i.e., the monitor and analysis phases) is fully automated; the other half is then executed manually. Thus, SOA governance feedback loops can not only be used to optimize its business functions, but also its evolution processes.

3.2.2 Levels of Indirection

"Any problem in computer science can be solved with another layer of indirection."

—David Wheeler

After studying the SOA governance components and selected inter-dependencies, we classified particular entities and relationships according to the time they are most relevant

or most critical—*design-time*, *deployment-time*, or *steady-state run-time*. Design-time SOA governance entities and relationships are often immutable and static and, thus, cumbersome to change and evolve. In contrast, run-time components can be dynamic and are therefore easier to maintain and evolve. The feedback loops discussed in the previous section are eminently useful for run-time evolution, but do not ease the evolution of static components.

However, a classic software engineering strategy—separation of concerns through levels of indirection—can alleviate this problem to a certain extent. Levels of indirection can be introduced for design-time and run-time SOA governance entities and relationships. In particular, levels of indirection can be designed for the key components of IBM's SGMM outlined in Figure 4 as well as Oracle's leverage points for SOA governance policies outlined in Figure 5. The IBM governance process model with its two layered governance lifecycle—plan, define, enable, and measure—constitutes a level of indirection, which facilitates extendibility and evolution of processes [BMT06]. Another elegant architectural solution, which is particularly suitable for evolving governance policies, is to organize the policies into layers where the higher level policies orchestrate lower level policies as in the IBM Autonomic Reference Architecture (ACRA) as specified in [Com04].

It is worth arguing that SOA governance mechanisms can not only be used to control the delivery of the stated service business objectives, but also the evolution of these service-oriented systems. Feedback loops and levels of indirection can be considered as potential SOA governance mechanisms. SOA assures integrity, simplicity, robustness, reuse, ease of maintenance and agility in a typical business setting. A governance model

specifies the processes, polices, controls and governance mechanisms that are required to monitor its service-oriented systems. It also provides the organizational structure and defines the roles and responsibilities that are needed to operate the governance model. Levels of indirection and feedback loops are highly effective mechanisms to facilitate and orchestrate maintenance and evolution. These mechanisms are present in the best practices of governance frameworks and service-oriented systems, but maybe not as explicit and readily recognized as they should be—given their excellent track record as well as wide applicability and versatility [MGN+09].

3.3 Policy Taxonomy—Characterization of Governance Policies

The previous section discusses the scope of a feedback loop as a governance mechanism. Out of the six governance components, governance policies serve as excellent feedback loops during the service functioning. This is because policies embody the decisions that need to be made for effective service management. Policies also include principles and standards that people, who are involved with a SOA governance program, create to guide the organization towards the desired behavior and evolution of its SOA solutions [Bis08]. Policies play a great role in determining level of conformance of stated metrics based on which service reformation occurs. This section introduces taxonomy to characterize governance policies. There are already a number of other policy taxonomies that cover diverse characteristics such as: run-time and design-time policies; business, process, architectural and technical policies; absolute and derived policies. Rather than striving for an all-inclusive taxonomy, this taxonomy's goal is to characterize the policies that are applicable to governance in distributed environments better. To characterize policies, the following criteria are used: scope, structure, granularity, origin,

state, awareness, execution phase. Figure 9 presents an overview of the complete taxonomy.

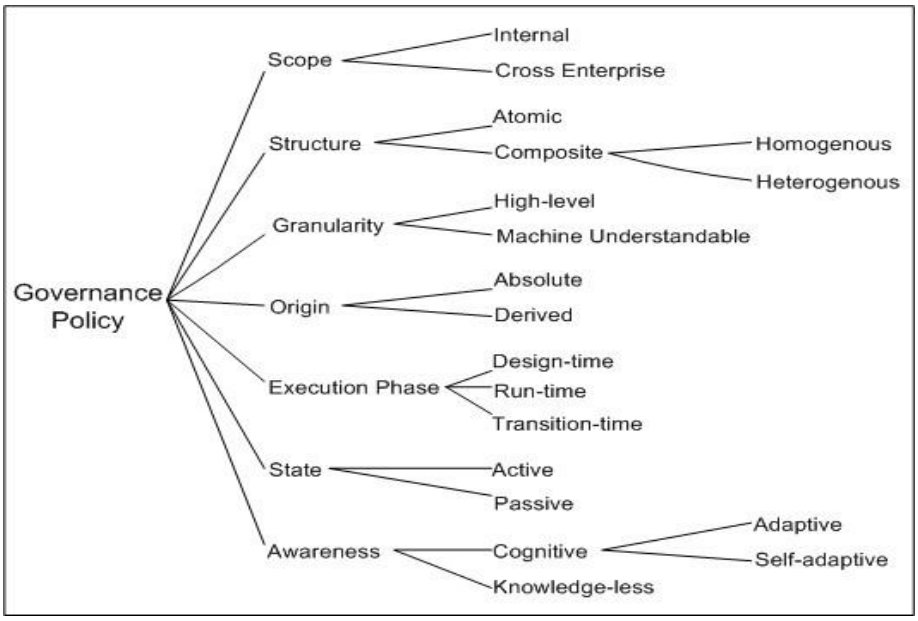

Figure 9: Taxonomy for Governance Policies

3.3.1 Scope

Internal: The policy is developed and consumed by developers within the same enterprise. Based on Luckham's enterprise layer [Luc01], the internal layer can be further refined to: Application, Collaboration, Middleware and Network.

Cross-Enterprise: The policy is provided by third party developers or by the enterprise whose service is being utilized.

3.3.2 Structure

Atomic: An atomic policy cannot be decomposed further to obtain more policies.

Composite: These policies are composites of other policies to achieve a goal or monitor a task. The composition can be static or dynamic. Also, policies that control the

same enterprise asset are called homogenous; policies combining to control different enterprise assets are termed heterogeneous.

3.3.3 Granularity

High-level: The policies are usually high level business goals and are implemented and monitored by people. These policies cannot be automated.

Machine Understandable: These policies are tied to the technical assets and are machine understandable rules that are usually monitored automatically without any assistance by the people component. These policies are fully automatic.

3.3.4 Origin

Absolute: These policies already exist in the policy store and can be applied to any service deployed.

Derived: Policies are derived from business goals and usually pertain only to the service defined to attain the goal.

3.3.5 Execution phase

Design-time: Policies are executed before the service deployment phase.

Run-time: Policies are executed dynamically at run-time once the service is published in the service registry. These policies, sometimes, are run-time instances of design-time policies. They monitor service activities when the service is under execution in the service consumer side.

Transition-time: These policies handle any change to the service which is under execution. These policies maintain SLA compliance and other contract rules from

breaking by measuring and monitoring any enterprise asset change that affect the service at run-time.

3.3.6 State

Active: The policy is associated to a service that is under consumption by the service consumer.

Passive: The policy along with the service has been published in the service registry but not consumed by the service consumer yet.

3.3.7 Awareness

Cognitive: The policy has knowledge to drive the addition, removal or selection of other policies.

Knowledge-less: The policy is merely associated with the service in the service description and executes as and when it is required to. The policy is not capable of adapting to changing environments.

This taxonomy has been created primarily for the purpose to characterize policies for governance execution in distributed environments. This taxonomy can also identify characteristics that would be helpful in classifying policies as the organization evolves to adopt the cloud computing infrastructure.

3.4 Summary

We described the use of policy as a governance component in this chapter in detail. We also defined policy taxonomy and identified the important components of a governance policy. In the next chapter, policies are classified and their characteristic based on the taxonomy is described. The taxonomy framed in this chapter will be used as an

instrument to precisely define the policy classification points in the service space, virtualization space and cloud space that this thesis aims to provide governance for.

Chapter 4 SOA Governance Lifecycle Based Classification of Policies with respect to Policy Characteristics

"Put simply, we need to ensure that rather than just building things, we are building the right things, the right way. Good governance that can make that happen. Poor governance results in teams doing whether they deem as important in their corner of the world, or more likely, whatever the easiest path is for them, by first focusing on things completely within their control."

—Todd Biske, Fortune 500 Enterprise Architect

4.1 Policies—A SOA Governance Lifecycle Based View

SOA is not sequential; it is iterative and fluid, with information management and governance needs spanning the traditional lines of design-time and run-time environments that as a result have blurred and overlapped. To be successful at SOA, enterprises need to be inclusive of all aspects of the SOA/services lifecycle. Advantageously, service lifecycle management ensures that each service is of the highest possible quality and used appropriately. Service lifecycle management using policies ensures the most apt choice of the service portfolio for delivering the highest possible business value over time. Also, introduction of SOA and SOA governance into an enterprise system introduces major architectural changes. Figure 10, based on IBM's SOA reference architecture [AZE+07], depicts the enterprise system after SOA adoption. The operation system layer includes legacy systems, current application systems, databases, existing software solutions like authentication solutions and logging and notification solutions. The enterprise's service component layer and the middleware layer

consist of all the services defined within the service oriented architecture. Services that are exposed can be invoked and choreographed to create composite services. The middleware layer enables service discovery and offers access to the service. This layer contains the service agreements that bind service consumers and providers. Service orchestration and choreography, to enable business process design and service flow creation supporting business requirements, are defined in the business process layer. The QoS layer, management and monitoring layer enables SOA capabilities to realize their non-functional requirements by monitoring compliance with SLAs, security rules and business KPIs. Finally, governance layer encompasses all aspects of the business-technical lifecycle management in SOA by providing guidance through policies.

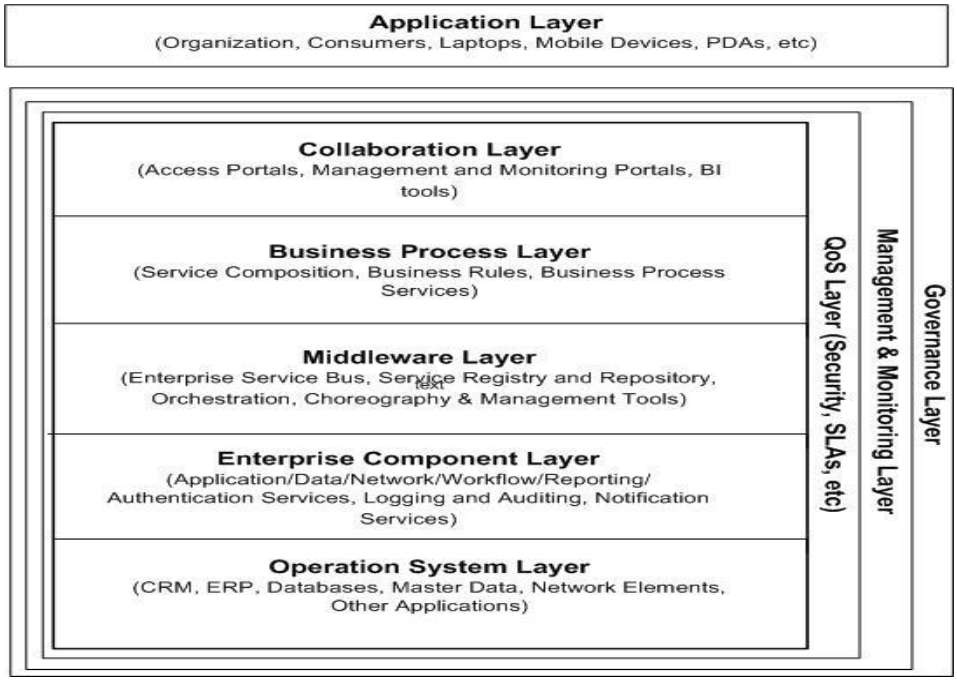

Figure 10: Enterprise Layer after Service Adoption

In order to control and govern service development lifecycle policies, it is useful to categorize policy types and management categories. *Policies are operating rules that are a means of maintaining order, security, or consistency and thereby, furthering a goal or*

mission. They are guidelines that a well-defined set of services ought to follow or system constraints or capabilities that determine the interaction between consumers and providers of services [All08a]. From this classification of service governance policies, this thesis provides an overview of the various categories of policies based on their empirically observable behaviour and the phase of the service lifecycle in which they are most applicable. Figure 10 shows a comprehensive view of the service development lifecycle comprising all the processes gathered from the above stated lifecycle approaches. To characterize different types of policies, we surveyed papers that deal with SOA governance in general and SOA governance policies in particular. This thesis summarizes policies featured in these papers and categorizes them according to six phases of the service lifecycle as defined by ITIL [CHR+08] and IBM SGMM [BLGM08] resulting in six policy types: *service planning, service design, service development, service transition, service operation,* and *multi-level policies.*

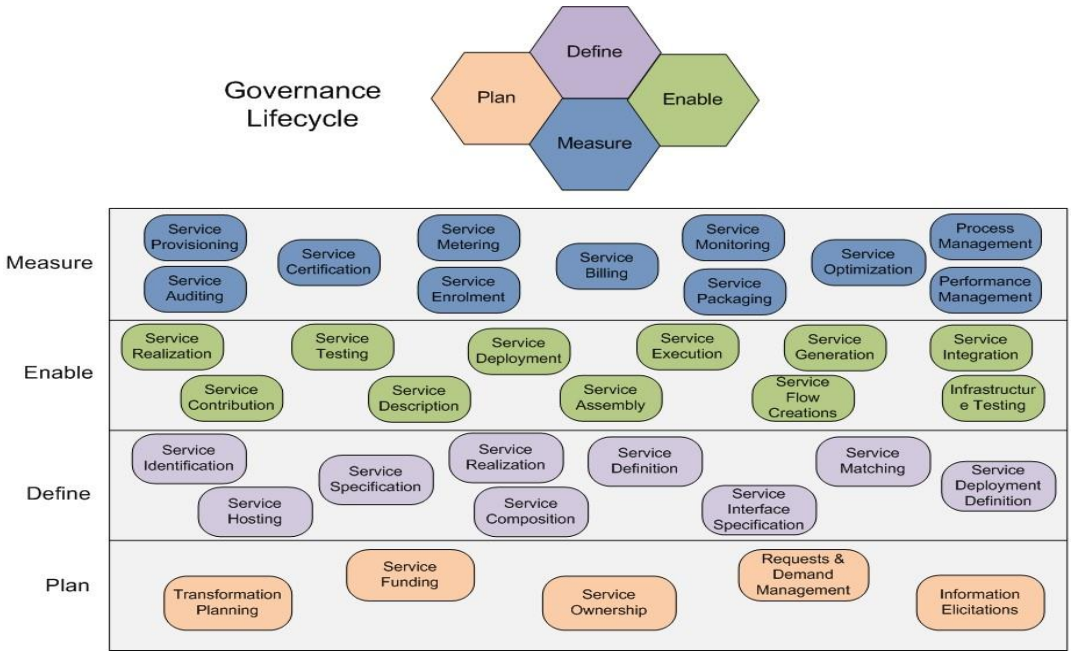

Figure 11: Service lifecycle and processes that can be governed

4.1.1 Service Planning Policies

Service Planning is usually the first stage of service oriented development wherein plans are made to engender reuse across lines of business and also creating services that facilitate agility and quick response to market opportunities [Mar08]. To ensure that the transformation to SOA is smooth and efficient, strategic decisions have to be made. This process involves budgetary control mechanisms, clearly identified roles and responsibilities, compliance with industrial and most appropriate standards, and adherence to reference architectures and design patterns [Mar08, ACH+07, Woo06]. Below we categorize the resulting policies. The diagram depicted in Figure 12 summarizes the service planning policies.

Organizational Policies: These policies define decision rights and responsibilities of people involved in service development [LL98] and form the basis for SOA adoption in an enterprise. They manage the interaction pattern between organizations across their line of business (LoB) [LL98, KKJ+08, RKT+07, AGSG08]. Where Organizational Policies handle, to a certain extent, the people aspect of service planning, *Architectural Policies* are established to address issues related to the framework of business processes and services and help drive down service development

Table 1: Organizational Policies

Scope	Task
Application	• Managing the content and usage of artifacts throughout their lifecycle • Defining roles and responsibilities of people involved in the development and governance of services • Managing organizational structure from the perspective of SOA • Ensures alignment of SOA activities with the business strategy and goals

cost by identifying specific services, or processes known to produce most favourable results [Mar08, ACH+07, Woo06, CMQ08, Geo05]. Under which, *Enterprise Architecture Policies* are related to business strategies such as EA goals, objectives and strategy. They also include the relationship of EA to capital planning, and program management. On the other hand, *Business Process Architecture policies* are those that relate to managing the logical and physical structure of infrastructure and component [WC07].

Table 2: Enterprise Architecture Policies

Scope	Task
Application	• Establishes compliance of Enterprise Architecture with the SOA goals defined • Creating informational view of the existing data assets before making them usable as consumable services • Managing compliance of EA standards with industry standards • Ensures alignment of business and technology functionality with IT

Table 3: Business Process Architecture Policies

Scope	Task
Application	Manages alignment of business process architecture and orchestration of business processes with the enterprise architecture

Ownership Policies that define the owners of the different artefacts in the development of a service. This is particularly useful while delivery/consumption of services across distributed services within an ownership domain and between ownership domains.

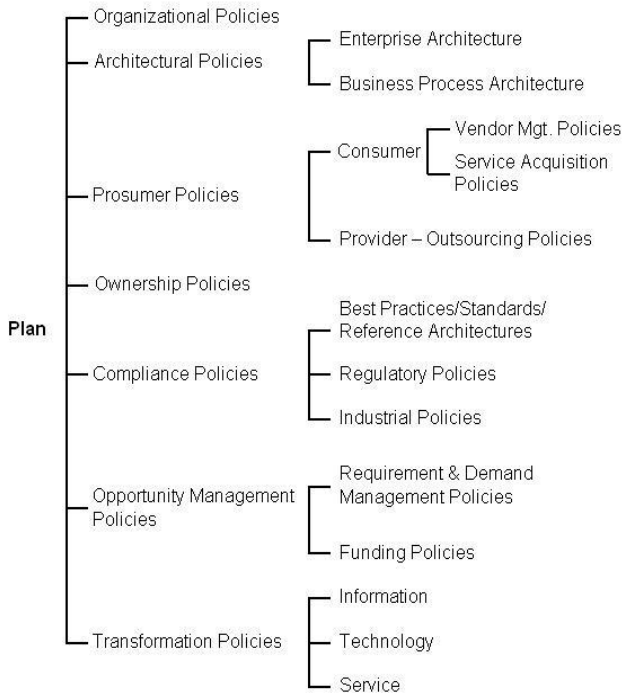

Figure 12: Service Planning Policies

Compliance Policies: These are Enterprise level compliance policies which handle conformance with industrial standards and regulatory policies based on the domain they work within. For example, industrial standards such as Sarbanes Oxley and domain specific policies include HIPAA (Healthcare), FIXX and IFX (Banking and Financial services), and ACCORD (Insurance). Moreover, compliance must be observed with the standards, best practices and architectures chosen for most efficient implementation of governance [MbC09].

Prosumer Policies: These policies relate to the high level relationship between consumers and providers from within or across the enterprise boundary. On the consumer side, *Vendor Management Policies* manage operational, reputational and strategic risks involved with the vendor relationship by stating contractual items such as performance standards, scope of service, security and confidentiality [Mar08]. There are also *Service*

Table 4: Vendor Management Policies

Scope	Task
Application, Cross- Enterprise	• Guides the management of vendor relationships • Ensures vendor performance capabilities are sufficient to meet IT requirements and the company's policies

Acquisition Policies that manage service purchase from providers based on clearly stated, performance based requirements, formulated costs and expected schedule.

Table 5: Service Acquisition Policies

Scope	Task
Application, Service, Cross-Enterprise	• Ensures service acquisition meets enterprise requirements maximising enterprise reuse and optimizing asset leverage for the organization • Conformance of service acquisition to SOA principles

Table 6: Ownership Policies

Scope	Task
Service	• Manages the ownership of business processes, assets, resources and services and their portfolios throughout their lifecycle • Managing the change of ownership as the service evolves incrementally to become an enterprise portfolio or backwards to become a business unit

Opportunity Management Policies comprise *Requirements and Demand Management Policies* that facilitate identifying, specifying and prioritizing requirements based on the high level business strategy. These policies facilitate evaluation of business and technology requirements and demand against existing resource portfolios to avoid duplication and achieve enterprise asset leverage and reuse. On the other hand, *Funding*

Policies manage the investment of funds within and across the enterprise maximizing profit. They also encourage companies to reduce investments in silos and maximize assets across the enterprise [MRVW05].

Table 7: Funding Policies

Scope	Task
Application	• Facilitates sharing of hardware and software infrastructure • Manages funding of shared services within and across the line of business • Facilitates funding of the SOA CoE group

Table 8: Requirements and Demand Management Policies

Scope	Task
Application	• Manages process that identify, aggregate, and prioritize requirements to achieve SOA goal • Measures conformance/difference between technical and business requirements, and existing portfolio to achieve asset leverage and reuse

Table 9: Transformation Policies

Policy	Scope	Tasks
Information	Middleware	• Manages mapping of data models to business logic • Manages data transformation between disparate schema structures
Technology	Collaboration, Network	• Ensures operational efficiency by optimizing IT portfolio during the alignment of business and IT • Ensures that IT systems are optimized and manageable while transforming to SOA needs

Transformation Policies [VGW+07, ETP09, TB92, RMW+10, FCWH06, LSBM08, LC07, TvLN08, GBG08] include the identification, classification and prioritization of domains and artifacts that satisfies the high level business vision and strategy. The sub-categories include *Information Transformation Planning Policies* identifying information

domains and services; *Technology Transformation Planning Policies* which are policies/standards for hardware, software, and development lifecycle and service reference architectures with the high level business goal taken into consideration; *Service Transformation Planning Policies* that are useful in identifying service domains, services and business processes.

Table 10: Compliance Policies

Policy	Scope	Tasks
Best Practices/Stds./Reference Architectures	Application	Adherence to and documentation of the standards & best practice
Regulatory Policy	Application	Implements regulatory compliance standards
Industrial Policy	Application	Implements industry specific standards such as HIPAA for healthcare, FIXX and IFX for banking and financial services, ACORD for insurance, etc

4.1.2 Service Design Policies

Service Design includes activities required to specify a service and smoothly transition into development or exposure of the service from an existing application. Services must be designed in a way such that they are able to meet the business requirements and goals set by the consumers and providers alike. With this perspective, *Service Design Policies* are those policies ensuring that the modeling of the service/s conforms to the desired business goal. Policies that fall under this category are described in Tables 11–18 and Figure 13.

Table 11: Architecture Policies

Policy	Scope	Tasks
Application Architecture	Application, Service	• Monitors the mapping of business services and service portfolio with the SOA solution • Controls the assembly and orchestration of services
Infrastructure Architecture	Application, Collaboration, Network	• Monitors that infrastructure leverage is most feasible for the SOA solution proposed • Manages and controls the governance of service components and artifacts based on the organizational policies
Information Architecture	Application, Middleware	• Monitors the mapping of a business object to logical, semantic and physical data model design • Controls the definition of data models and schemas
Service Architecture	Application, Service	• Deals with the components required to build a service, rules to combine components and services (for composite services) and the environment in which the service operates

Architectural Policies: The architectural policies at this level are more refined when compared to the planning phase. These policies control the identification and specification of design patterns that will be used for the implementation of services. The four primary architectural assets to be governed during SOA migration are: *application*, *information*, *technology/technical* and *service architecture*. The following paragraph details the policies governing these architectural assets and its objectives.

Application Architecture Policies comprise the way in which interfaces between application packages, databases, and middleware systems are defined from a functional point of view. SOA-based application architectures often place a premium on assembly/orchestration of services and in separating application-specific policy decisions from the underlying core business functions employed in support of those policies [Mar08]. *Information Architecture Policies* focuses on the process of modeling the

information, semantic models, logical and physical data models (*Data Architecture Policies*) that is needed to support the business processes and to address interoperability, integration, consolidation, and sharing of resources by correlating business processes to the business goals through the identification and definition of data/information relationships and dependencies (*Message Flow Policies*). *Technology Architecture Policies* enables identification of appropriate infrastructure, networks, computing and telecommunications infrastructure that supports and enables other architectures [Mar08]. They also involve identifying and building the right design patterns and domain specific languages for implementation. *Service Architecture Policies* refine the business process architecture into services. These policies define the way in which services must be reused, transformed or created. They also define in an abstract level the interaction between services and the functions that every service is expected to perform.

Table 12: Service Identification Policies

Policy	Scope	Tasks
Service Matching Policy	Service, Middleware	• Facilitates accurate identification of services based on description and other functional/non-functional requirements of SOA
Service Selection Policy	Service, Middleware	• Monitors static and dynamic service selection based on the architectural policies thus governing conformance between technical architecture and enterprise requirements
Service Sourcing Policy	Service, Within/Cross Enterprise	• Concerns the service supply process (whether outsourced or performed internally)

Acquisition Policies comprise Technology Acquisition Policies (the entire technical infrastructure needed to meet the requirements and demands of business) and *Resource Acquisition Policies* (e.g., resources such as people, funds, third party services, or contracts are identified here) [CWX10].

Table 13: Service Definition Policies

Scope	Task
Service	• Maintains mapping with the business architectural road map (thereby identifying the workload on every service)

Service Modeling Policies [QSPvS06] are the most important policies in this phase and include Service Identification Policies and Service Specification Policies [RR10].

Table 14: Service Usage Policies

Scope	Task
Application, Service, Within/Cross Enterprise	• Manages the constituency in which services are offered • Controls the business requirements that the offered service must satisfy

For a consumer, service identification policies provide guidance on how to identify high quality services that are of the most appropriate granularity. This involves having efficient algorithms to match the requirements from the pool of services (i.e., *Service Matching Policies*) [YHY+09, LZC09, TPT09, KNR09, LCHS05, YL08] and then selecting the best service based on requirements, performance, infrastructure and cost (i.e., *Service Selection Policies*) [DW09, ZZH06, ZJNZ09, BGG09, GCT+08, LW09, ZSCD08]. *Service Sourcing Policies* [AALN07, ZCC+02, Mar09] enable evaluation of potential suppliers and proactively narrowing down the list of suppliers for better achievement of business goals. *Service Specification Policies* confirm that the design and implementation meet the requirements and make sure that the service specification elicits the order in which the capabilities of a service can be executed and that the interfaces, constraints and Quality of Service (QoS) are specified appropriately.

Table 15: Service Dependency Policies

Scope	Task
Service	Manages the activities necessary to create binding between logical components during service execution and also services and their related artifacts

Service Definition Policies address the messaging structure, port types and the data types used in the message flow using schema definition languages such as XML Schema (*Information Model Policies*) (e.g., WS-Policy for the definition of XML Schema). *Service Definition Policies* [Jon05, SSOLH06, HKO10] must also address the effects and side effects of service operations and the kind of messages input or output by the service (*Behavioral Model Policies*). *Logical Model Policies* describe the type of service, the domains they belong to and their complete functionality. *Interface Definition Policies* describe the way the interface must be defined based on the extent of coupling or cohesion between services. For example, if two services are logically cohesive, then the service interface must only contain port types.

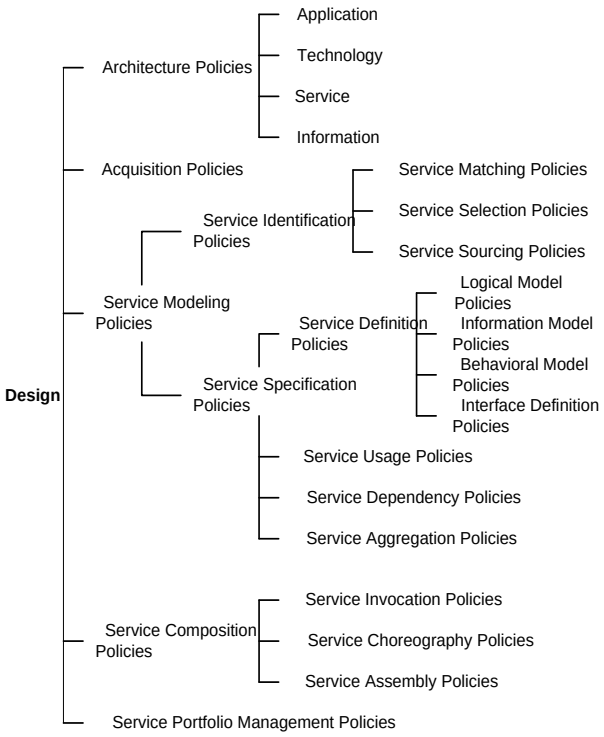

Figure 13: Service Design Policies

Service Usage Policy governs the usage of a service by the consumer. The costs of using the service, prohibited users, server abuse and spamming must be clearly specified within the service usage policy.

Service Dependency Policy enables the designers to clearly define the dependencies or relationships between services in a composite service. Moreover, the order in which the services will be executed must be specified [KDC+09, ZPPN07, BCD08, RBK05, EK01].

Service Aggregation Policy specifies the level of granularity that the service has or must have depending on the business goal. It also specifies the degree and the kind (i.e., Representational, Identity, Communication Protocol) of coupling between services

Table 16: Service Invocation Policies

Scope	Task
Service, Network	• Manages routing of requests during events such as version change, QoS support variation, etc

and the degree and type (Functional, Communicational or Logical) of cohesion amongst related services [Wu07, KR07, GN02b, QZC07, LDC09, WCH08, CVK10].

Table 17: Service Choreography Policies

Scope	Task
Service, Network	• Controls the multi behavioural service to be choreographed in a way such as to produce optimal and SOA solution aligned results • Manages the data flow and control flow through the process of service choreography

Service Composition Policies specify the necessary rules for the aggregation of multiple services into a single composite service [EMT06a, MWM07, HC09, ZSC06, CAA04, TX06, RB08, IYH07, YOLH05, WJY+08, BDNGD09, CL08, Pra07, SSE09, QSPvS06, ZHH+08]. One of the rules is the way in which each service must be invoked and the kind of messages that must be passed to and obtained from it (*Service Invocation Policies*) [YCMZ08, WWS08, LZLG07, LRD09, TYL05, Yu04]. *Service Choreography Policies* define the sequence of messages exchanges, the rules of interaction, specification of common behavior of services (WS-CDL) and agreements occurring between multiple services [XJ06, JFS08, MBLN09, AFC+, Pel03, BWR09].

Table 18: Service Flow Creation Policies

Scope	Task
Service, Network	• Controls the flow of services, aggregating to form a single unit of service fulfilling the enterprise's needs • Controls the order in which the consumed service operations need to be called at run-time

Service Portfolio Management Policy governs the management of the collection of assets and artifacts to optimize benefits. Policies for managing funds, resources, applications, hardware, software infrastructure and services fall under this category.

Service Flow Creation Policies specify the order of execution of the functionalities of a service or services. In a request/response communication pattern, the policy must specify the conditions in which the responses/requests will be invoked which is different from an event driven architecture wherein the basic events are specified.

4.1.3 Service Development Policies

Service Development is the phase where in development teams develop and test the composite application in an iterative manner and upload the artifacts to the enterprise service repository. Service development policies guide the development of services to optimize the profits obtained from developing service oriented systems. In a typical SOA environment, the standard WS-Policy along with the WS-Attachment specification allows a policy to be assigned to a service in an SOA. Some of the policies that fall in this category are described in Tables 19–26 and Figure 14.

Technology Standards Compliance Policies: Compliance with standards such as WS-I (Web Services Standards), appropriate versions of SOAP, REST, WSDL, and UDDI.

Table 19: Service Integration Policies

Scope	Task
Application, Network	• Ensures that the integration meets defined service level and security requirements and that services are readily consumable across technology, administrative, and organizational boundaries

Table 20: Service Orchestration Policies

Scope	Task
Application, Network	• Controls the selection of services based on business needs • Controls the interaction between services at data level and also their execution order

Service Realization Policies: Service Realization enables the development of services from the ground up or identification/wrapping of the legacy system module will be used to realize a given service. This includes *Service Integration Policies* that govern the process by which individual services or collaborating services or legacy systems organize themselves to implement the entire application/service. Legacy systems integration and wrapping policies, most of the application server policies fall under the category of integration policies [CI04, EKP+10, JCY10, YCD+09, I+07, BLK+09, SPM06, FOIY05, He09, UP08, BP09, JKP08].

Table 21: Service Provisioning Policies

Scope	Task
Application, Network, Collaboration	• Grants access to all the managed SOA artifacts • Manages deployment of services into production • Defining accountability for service provision and usage

(e.g., Java Application Server Policies such as Oracle OC4J, or Apache) and Integration server policies (e.g., WebSphere Application Server policies, or webMethods B2B Integration Server policies) fall under this category.

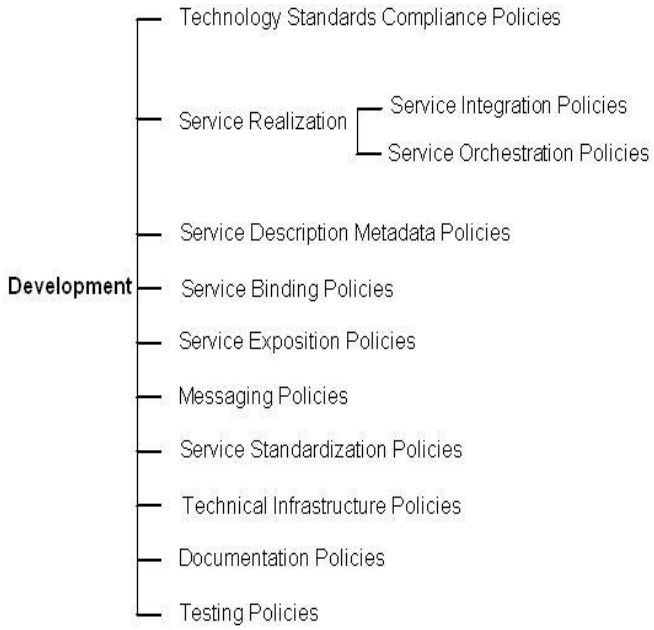

Figure 14: Service Development Policies

These policies vary on the basis of the systems that are integrated and the type of integration (e.g., UI Integration, Point-to-Point Integration, CGI Integration, or Data Integration). For example, if the intention is to integrate UNIX environments with Microsoft.

Table 22: Service Description Policies

Scope	Task
Middleware	• Manages service information provided in the metadata so as to achieve loose coupling • Provides an abstract definition of what information is necessary to deploy and interact with a service • Ensures that services can be easily reused by different types of clients

Technologies, and if the solution provided is SFU (Windows Services for UNIX), then SFU policies and standards such as FTP, HTTP, IP, IPSec, LDAP, PKI, POSIX standards.

Table 23: Service Binding Policies

Scope	Task
Middleware, Network	• Configures bindings to allow service access on one or several available interfaces based on SOA requirements • Monitors the process of binding to a service

Binding a Service to an interface means that that service is only accessible if the connection comes in on that interface. With WinGate, bindings can be configured to allow access on one, several or all of the available interfaces depending on the binding policy or network requirements [PKH+08, CDM+03].

Service Assembly Policies are those policies that enable the creation of new services which accurately transform the business requirements and also follow predefined rules and processes based on architectural standards and design patterns [Mar08, OLH+]. *Service Orchestration Policies* [PASB07, ST05, He09, BSX+08, YBG08, MMD10, MSK08, ZLK+07, MCBFR08] enable developers to specify the interaction between services at the message level, the execution order of interaction thus assuring controlled

execution flow of services. These include XML-based process standard definition languages policies such as BPEL (Business Process Execution Language) for Web Services. OASIS defines Service Provisioning as "*Provisioning is the automation of all the steps required to manage (setup, amend and revoke) user or system access entitlements or data relative to electronically published services.*" Examples of Service Provisioning Policies include SPML (Service Provisioning Markup Language) [R+03] policies and policies that are created by administrators based on the way services are customized (e.g., Provisioning Policies created using IBM Tivoli Identity Managers) [YLT+06, VdM96, GH05b, GH05a, BH06].

Service Description Metadata Policies: These policies enable the specification of user-defined metadata types that are used to enhance the service metadata to explain their semantics. For example, WSRR policy (Web Services Registry and Repository) that defines Service Description Entities (Physical, Logical and Conceptual) and the way in which these entities are related, classified or defined [FMA+06, AFPO08]. Service description defines *metadata* that fully describe the characteristics of services that are deployed on a network. Such metadata are important and fundamental to achieving the loose coupling that is associated with an SOA. It provides an abstract definition of the information that is necessary to deploy and interact with a service. *Messaging policies* provide Reliability mediation (e.g., map WS-* standards for reliable messaging over http to message-oriented-middleware reliability models [ZC05, AZHG06, ETP09]).

Table 24: Messaging Policies

Scope	Task
Network	• Ensures that messaging meets the non-functional and functional requirements of SOA • Reliable mediation

Technical Infrastructure Policies specify the usage of a particular tool or technology. These policies vary with the technology that is used [ZMCW08, BJS96].

Table 25: Technical Infrastructure Policies

Infrastructure Policies	Network Infrastructure Policies, Application Infrastructure Policies, System Infrastructure Policies
Registry and Repository	WSRR Policies, enXML policies, UDDI policies, XML Regsitry Policies,
Web services Management	.NET, SilverLight, Ajax Client Libraries
Service Development Tools	WSDL Editory Policies, Apache Axis Policies, Web service Testing tools, etc (all the tools that help in attaining a service oriented system).
Coding Policies	All policies set based on best practices.
IDE Policies	Eclipse, NetBeans, VisualStudio, etc
Application Server Policies	JBoss Policies, WebSphere Application Server, Zend Policies, etc
Database Policies	Database storage policies (this includes the time period at which information must be updated), access and retrieval polices, etc

Service Binding Policies specify the ways in which services must bind with other services. Examples include BasicHttpBinding, MessageBinding, SOAP Binding, and SOAP Addressing). This also includes Message Encoding Policies such as SOAP Encoding, and MTOM Encoding.

Table 26: Service Standardization Policies

Scope	Task
Service, Middleware, Network	Ensures interoperability and manages services from disparate sources

Service Exposition Policies include the different ways in which services can be exposed in a registry/repository (e.g., JSF policies for exposing services using rich user interfaces, and ESB Policies).

Messaging Policies enable the definition of messages that facilitate interaction between services (e.g., JMS standards [MHC01], OASIS—ebXML standards, ActionScript Messaging Format (AMF), and message exchange patterns such as request-response) [RWW]. Messaging Architecture Policies include routing protocols, patterns such as request-response, push-subscribe, message channel patterns, and transformation patterns Messaging Server Policies. Static or dynamic binding between services relies on messaging architectures.

Service Standardization Policies include appropriate service interface definitions [WT98, Wal99, Han98, MSSR08]. (e.g., WSDL definitions that specify the granularity, messaging structure, data types, data structures and naming conventions, WS-Interoperability other WS-* standards [Nol92, VB97, Asa94]).

Documentation Policies define time frames when the service artifacts must be uploaded in the repository for future reference, confirm that the documents are readily available and correctly, define styles in which documents must be presented [VC02] (e.g., Java document specification).

Testing Policies include WSDL (Service Interface Design) validation, Message and Schema validation, performance and load testing, verifying end to end service integration, query verification and validation, application integration, BPEL orchestration, testing for protocols such as SOAP over HTTP, HTTPs, JMS data driven testing, conformance with SLAs and QoS metrics [BGP00].

4.1.4 Service Transition Policies

Service Transition is the next stage after development and testing which focuses on delivering the business requirements in the form of services. In this section, we focus only on the release and deployment of services. Service Transition Policies enable appropriate and accurate deployment of services onto the repository/registry or at the provider side. The policies that fit in this category are,

Table 27: Service Configuration Policies

Scope	Task
Network	• Manages the setup of services and other resources (to adhere to the service contract and business goals) offered to the consumer • Controls dynamic clustering of services and also managing the load and synchronization of service configuration updates

Service Deployment Policies are involved in releasing the services to other participants, enterprises, application or services [GDA04, LLL10, HDS03, TWP+05, CHK+04][205]. These include *Service Configuration Policies* [XFZ08, MHD+05, GN02a, BMEWE05, CHK+04, CMT06], which define the arrangement of services, initial setting of services, information about hardware, software and other artifacts that affect the performance and functionalities of the service. They enhance the process's ability to support multiple types

of bindings, including transport protocols, security/network settings and message-encoding standards, between service clients and service hosts, simultaneously and efficiently.

Table 28: Service Publishing Policies

Scope	Task
Network, Middleware	• Ensures pre-publishing service validation activities such as testing, documentation, service certification, authorization rights, conformance to enterprise governance model • Ensures complete exposing and development of service • Ensures conformance to minimal service standards during publishing

Some of the policies are the ones framed by Nixes, Radia, SmartFrog (based on the style of deployment—manual, language based, model based or script based). Yet another example is SAP NetWeaver Process Integration. *Service Publishing Policies* focus on appropriate definitions of both service operations and service interface so that

Table 29: Service Versioning Policies

Scope	Task
Middleware, Application, Network	• Manages versioning/deprecation/retirement of services and different service versions within the registry with appropriate notification processes

identification of services is enhanced during service discovery [YCSG06, CZFD07, DSFR07, DCCD08, WRK05]. The communication protocols, data structures, messages that can be exchanged must be accurately mentioned while publishing to the registry. *Versioning Policies* concentrate on maintaining the correct version of services being executed in a highly distributed environment and informing the consumers about any updates on services [LMRD08, WZD07, BLM+08, Las08].

Table 30: Service Metering Policies

Scope	Task
Application, Network, Within/Cross Enterprise	• Monitors if the service attributes to be metered are explicitly mentioned in the service contract

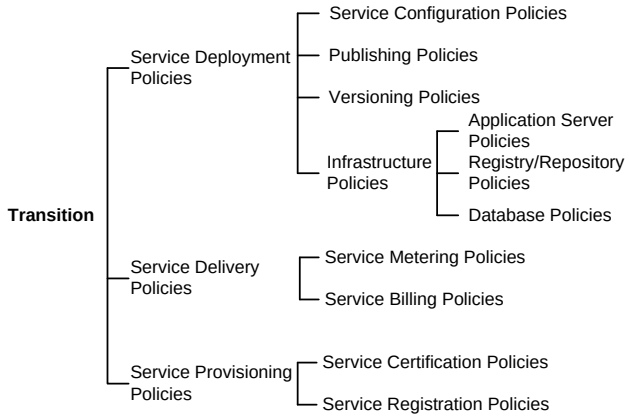

Figure 15: Service Transition Policies

Versioning can also be handled using XML extensions, middleware, service adapters, UDDI subscriptions, or WS-Addressing [GHR06]. Providers must specify the different ways in which versioning is handled. *Infrastructure Policies* specify the system necessary

Table 31: Service Billing Policies

Scope	Task
Application, Network, Within/Cross Enterprise	Defines rules that control service billing and obtain maximum ROI

for the identification and execution of services. These include the messaging protocols, Application server policies, Registry/Repository policies, middleware policies and database policies.

Table 32: Service Certification Policies

Scope	Task
Application, Network, Within/Cross Enterprise	• Specifies the service standard and service quality that the consumers can expect

Service Delivery Policies make sure that there is no ambiguity in the conditions set between the service provider and consumers. *Service Metering Policies* enable proper metering of the usage of a service by clients and specifying the parameters [AKK03, CLZ10] (e.g., subscription rates, usage events, quality, precision, leasing, lifetime, and hidden costs) which will be used for billing the clients. Usually, customers and providers agree on a billing scheme with their choice of parameters depending on the kind of service being offered (*Service Billing Policies*) [KTR07, MOV10, LBNH08, WS06]. These policies also include retrieval, payment and disbursement of payment to the service providers.

Service Provisioning Policies in this phase include *Service Certification Policies* [FF03, Mas07] which enables the specification of properties to facilitate prediction of the behavior of the service. Service Registration Policies specify the most appropriate way in which services must be published to the UDDI. It generally involves the design of an XML file that specifies the path to the service DLL, the interface that defines the service contract and the class that implements the contract. The file also specifies any environment variables that the service needs [PHS+10, LGZ+05, SB08].

4.1.5 Service Operations Policies

During this phase, services are managed, supported, controlled and rectified based on changing business contexts. The policies that fit this category are depicted in Figure 16.

Service Management and Support Policies include a wide range of policies [EMT06b, KLMB96, LBMS00, Dam02, JfBZj05, HDW02, SPL+03, SL02, Slo94, SRC05, YLT+06, GB07, UBJ+04, KW06],

- Change Management Policies
- Variability Management Policies
- Data Management Policies
- Configuration Management Policies
- Release Management Policies
- Knowledge Management Policies
- Asset Management Policies
- Incident Management Policies
- Capacity Management Policies
- Problem Management Policies (includes fault management policies)
- Availability Management Policies

These policies help in specifying adaptability requirements by determining the boundaries and key articulation points (technical SLA policy) [BZ08, PHS+08, GHH+01, GZH96]. Some of their functionalities include: life cycle management (specification)—

ensuring integrity in versioning, configuration and change management [YLT+06, LSS09, GRM06, FLO09, SSLTN05, IPK+06, SR], coordinating distributed service specification to meet a common goal; determining appropriate an level of generalization to meet strategic goals and organizational design, maintenance of business perspective in resource allocation, alternative service assignment and rule application, providing the right services to internal and external customers, partners—ensuring sharing where appropriate, identification of standard business and infrastructure services and situation applicability.

Run-time Policies describe desired or correct behavior post deployment or at run-time. These include [KK08, LGP+10, JZZ07],

Service Discovery Policies based on interface and behavioral models of the services and operations that need to be discovered at run-time [PHS+10, RG08, ZCG05, BGK04, SZK05]. This also involves patterns for matching services based on interface, structure, and behavior. UDDI policies and WS-Policy play major roles here.

Service Composition Policies allow the realization of flexible and adaptable applications by properly selecting and combining components based on the user request and context. This also allows run-time integration of service endpoints. (e.g., WS-Composition, CORBA Policies [RR10, EMT06a, SPS96]).

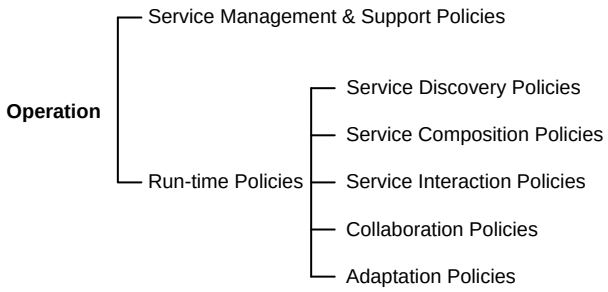

Figure 16: Service Operations Policies

Service Interaction Policies specify the selection of services, the way to access services and the exchange of messages between them. (e.g., WSDL policies which describe the operations of a service, run-time interaction protocols, WCF (Windows Communication Foundation) policies, and Apache Felix policies).

Collaboration Policies: Dynamic collaboration allows two parties not knowing each other to establish their collaboration protocol at run-time. In dynamic collaboration, as collaboration protocols will be negotiated and determined at run-time, collaboration policies need to be generated at run-time to manage the interaction [ZTW+06, LSZJ06, HY09, NZC08].

Adaptation Policies facilitate the evolution of services based on change in the underlying infrastructure, business requirement, environment, and context [SA07, ZSX10, BB10, NUH09]. This also includes load balancing policies [EMT06a, KW04, PKKJ04, KC03, ZCG05, SL05, HK04, CBBJ08, SA07, JJH+08, ZSC06]. They help in selecting and consuming the right service to meet business functional requirements, return on investment (ROI), and to fill contractual obligations such as SLAs for onward service consumers. They aid in the determination and negotiation of life cycle objects—metadata covering semantic and technical protocols that need to be aligned.

Monitoring policies [RK08, FFKS08, EMT07, R+08, SS08, KB09, Bet02, NUH09, SLZ09, BTPT06] are generally IT policies that define the monitoring of resources, applications and services in the organization for issues such as security, performance, and compliance [FFKS08, Mar08, NUH09]. They also assist in monitoring usage of business rules and effectiveness of agility. Example includes SLA Monitoring Policy—monitoring the contractual commitment of software services, including Security.

4.1.6 Multilevel Policies

Multilevel policies [Mar08] are the ones that are applied to all the phases of the service development lifecycle. Multilevel policies take one of three forms: Business (Plan-Time), Process (Design-Time) and Technical (Run-Time). They take one of these forms in every phase of the lifecycle. A Security Policy can be "Customer Trust" in its highest level and can be enforced at run-time as WS-Security Standards, SAML, XML Signature, or XML Encryption. Similarly, SLA can be "Meet subscriber requests and demands quickly" at its highest level and at a lower level, it could be "Server Response time >= 10ms" [Yao03, LMKB01, SL02, SRS+01, SS09, MWM07, YCA+09, AEYC, WHBLC06, PB05].

4.2 Survey Results

This section summarizes our survey by presenting data to the selected references and the service lifecycle phases as depicted in Figures 17–23. We collected 314 policy papers for this survey. Of these papers, 296 were categorized and used to produce Figures 17–23. The main focus of this survey was to identify current research trend in the field of policies applicable to SOA governance. This section elucidates the policies that might require more exploration which is beneficial for the SOA community.

Our survey followed the *systematic literature review method* stated by Kitchenham et al. [KC07] and Easterbrook et al. [ESSD08]. We searched for conference proceedings, journals and articles related to policies. We then tabulated all the policies and their corresponding functionalities followed by sampling of the search results and policy classification. Due to the tabulation of policies and its functionalities, identifying redundant policies was much easier leading to removal of duplicate policies from the master list of policies. Finally, we classified policies on the basis of their functionality and the components (people, development artifacts, planning/development/management processes, and services/software components) they affected.

4.2.1 Overall Distribution of Contributions by Policy Subtype

Figure 17 provides an overview of the overall distribution of papers by major service lifecycle policy type. The classification identifies that research effort is laid more towards policies that are machine understandable and this could be because service-oriented systems are extensively instrumented to keep track of useful figures such as execution time, availability, throughput, latency, or resource consumption which are readily available. In this thesis, monitoring policies are considered as those whose results are essential in deciding whether there is a need to adapt policies and/or processes in order to optimize outcomes [MGN+09]. Based on this definition, it is quite evident that more exploration is required towards monitoring policies that form the basis for managing evolution of service oriented systems.

Figures 18–23 exhibit the distribution of papers for each major service lifecycle policy type by policy subtype. The reader must note that the search results are not exhaustive and that we are trying our best to accommodate policies into the data representations as

and when we find them. Nevertheless, it is quite evident that research is focused primarily towards the technical policies (cf. Figure 20) and policies that deal with service behavior during deployment-time (cf. Figure 21) and run-time (cf. Figure 22). Figure 23 exhibits the involvement of the networking research community towards implementing service oriented policies and their high degree of participation. If the policies are broadly divided into design-time and run-time, we conclude that, currently there are more contributions towards managing the run-time properties of a service.

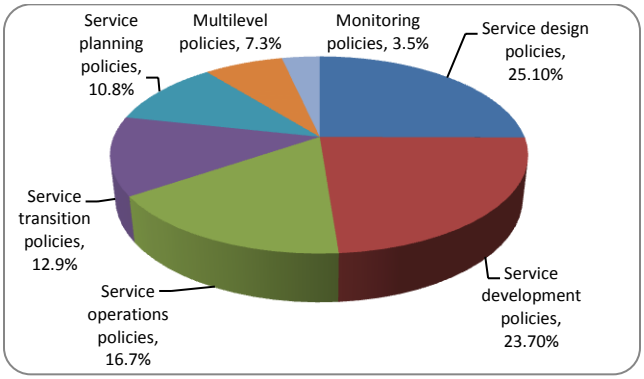

Figure 17: Overall distribution of contributions

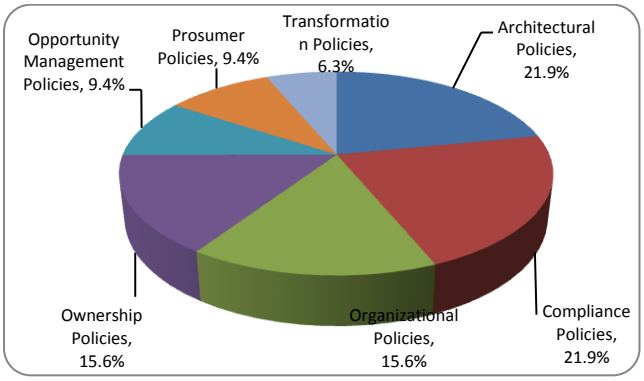

Figure 18: Distribution of contributions of service planning policies by planning subtype

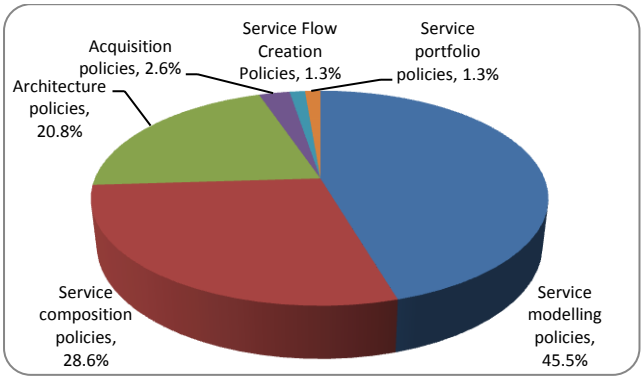

Figure 19: Distribution of contributions of service design policies by service design subtype

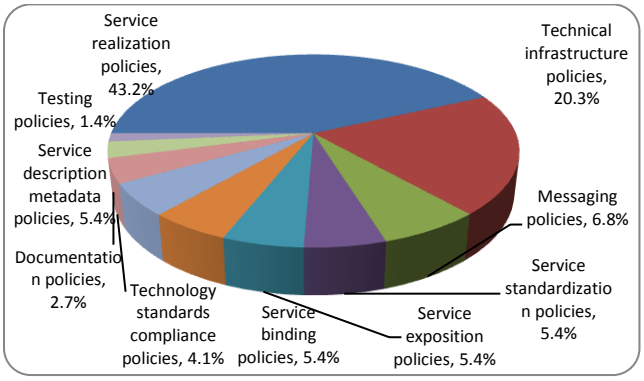

Figure 20: Distribution of contributions of service development policies by development subtype

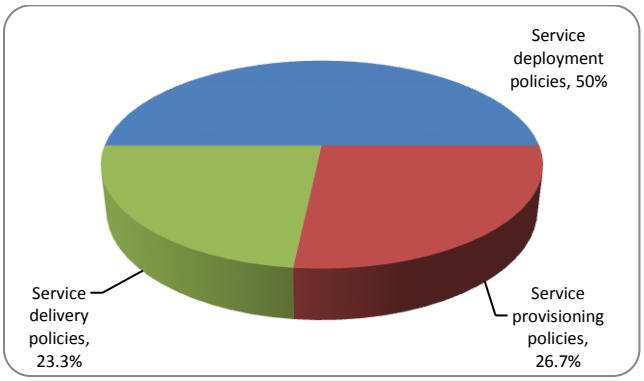

Figure 21: Distribution of contributions of service transition policies by transition subtype

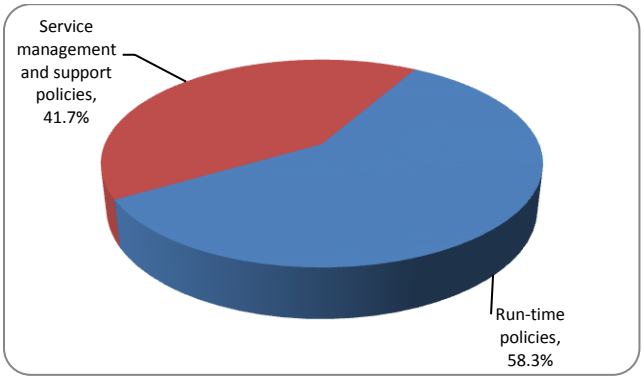

Figure 22: Distribution of contributions of service operations policies by operations subtype

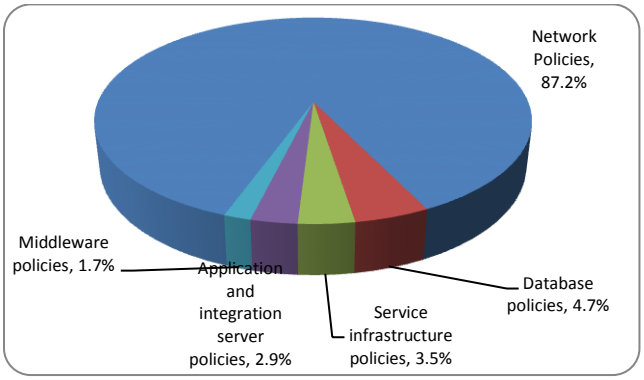

Figure 23: Distribution of contributions of domain specific technical policies by of domain specific technical policies sub-type

4.2.2 Chronological Distribution of Contributions by Policy Subtype

Figures 24–28 depict the research trend over the years in different policy areas. We conclude that the participation of research communities has been more in the area of service development, operation and transition policies than the service planning and design policies.

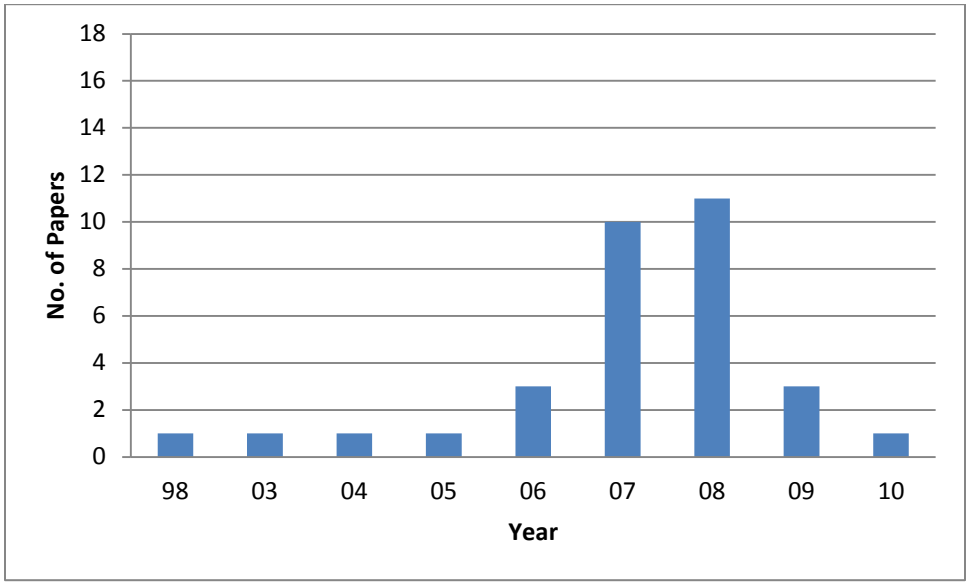

Figure 24: Chronological distribution of contributions of service planning policies by planning subtype

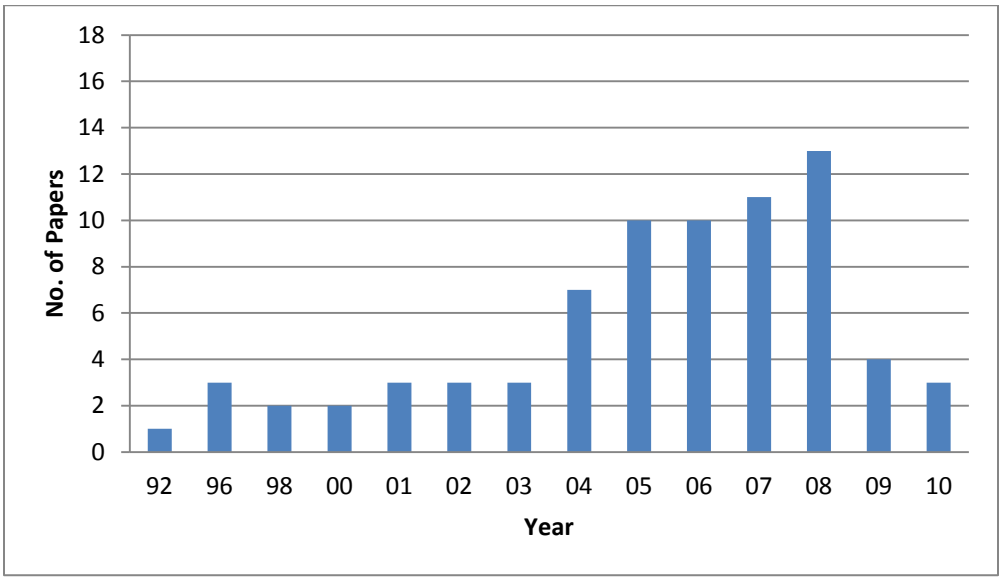

Figure 25: Chronological distribution of contributions of service design policies by design subtype

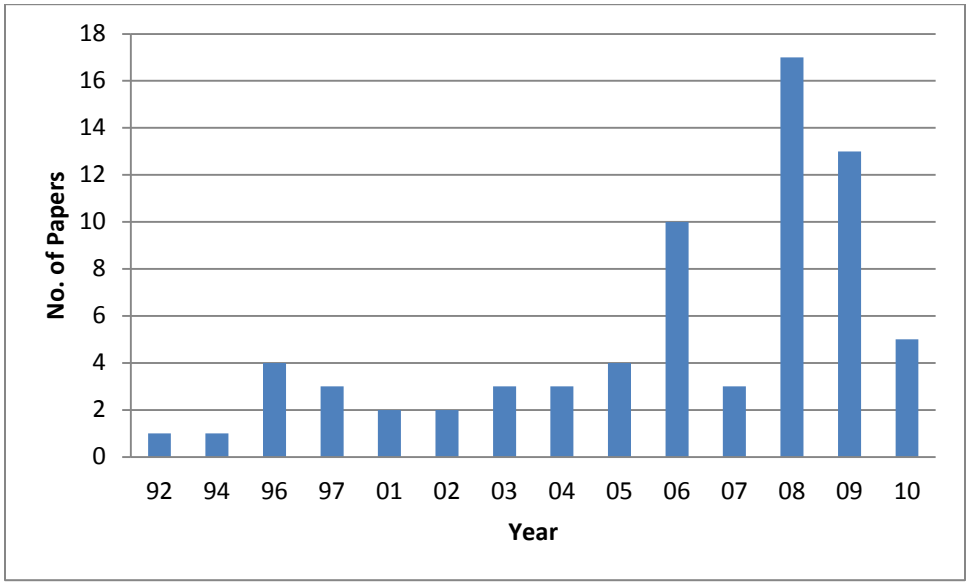

Figure 26: Chronological distribution of contributions of service development policies by development subtype

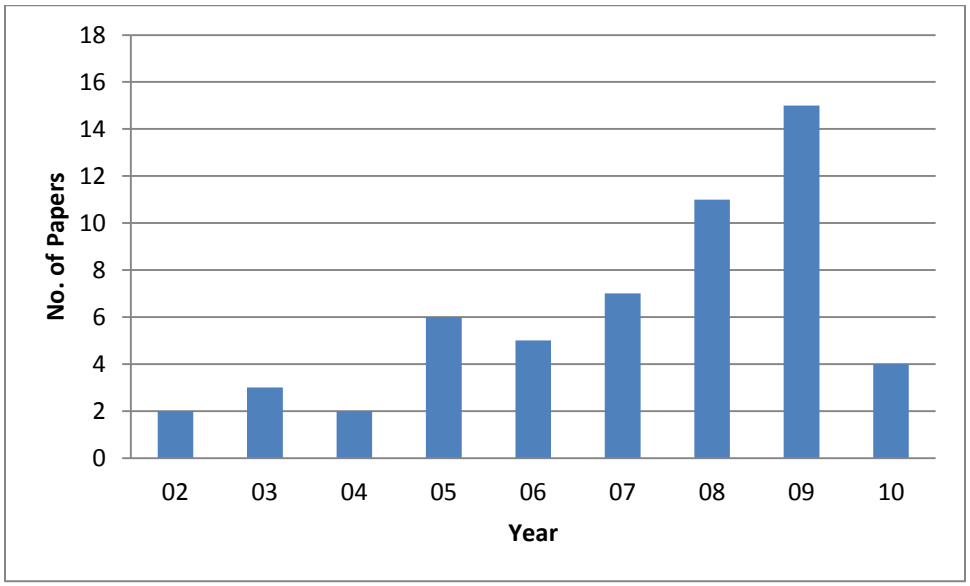

Figure 27: Chronological distribution of contributions of service transition policies by transition subtype

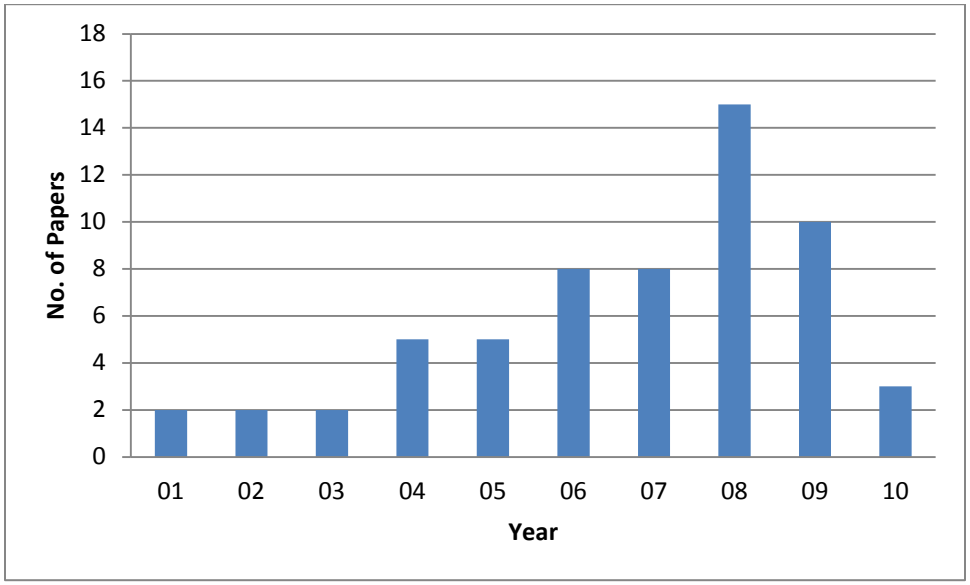

Figure 28: Chronological distribution of contributions of service operations policies by operations subtype

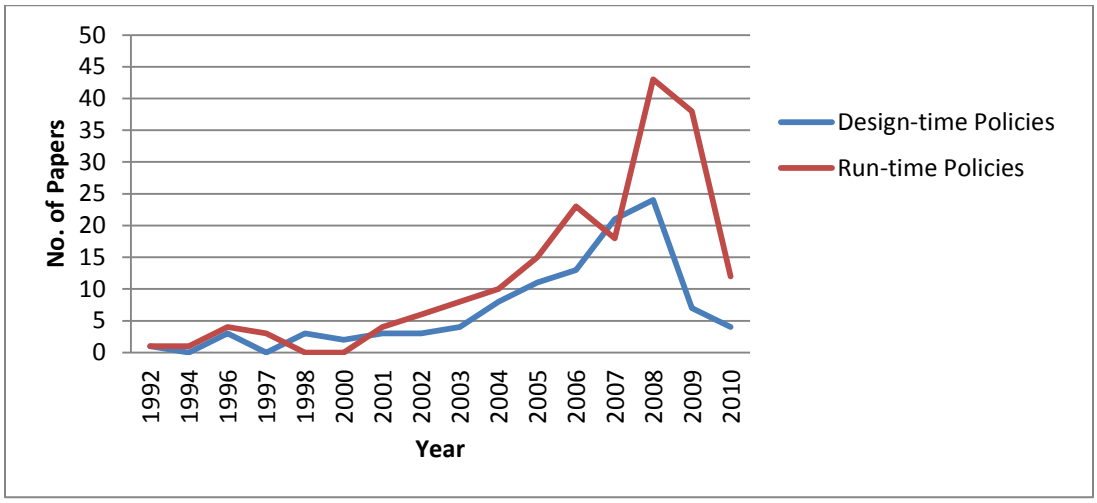

Figure 29: Chronological distribution of design-time policies v/s run-time policies

Viewing Figure 29 with an industrial perspective, it is observed that more focus is laid towards low-level policies and less focus towards high-level policies. The primary focus of SOA is business and technical alignment which implies that the gap between these business level policies and technical policies must be less. However, our results have shown that there is an increase in gap between both the set of policies, which then leads

to a state where one might never know how the machine-level policies change when the high-level business requirements change. With an academia perspective, the classification identifies that research effort is laid more towards policies that are machine understandable and this could be because service-oriented systems are extensively instrumented to keep track of useful figures such as execution time, availability, throughput, latency, or resource consumption which are readily available.

Based on the results obtained, it would be more appropriate that more focus be laid towards monitoring policies that identify how the business rules impact the machine understandable rules. Therefore, researchers must spend more effort in identifying dependencies between both the levels of policies and possibly close this gap that is prominent between the business and technical side of SOA.

4.3 Is SOA Governance Sufficient to Handle "Next Generation" Service Development?

Service oriented architecture undoubtedly offers numerous advantages such as effective integration with other organizations, dynamic yet simplified integration and collaboration, flexibility to adapt to evolving requirements, reduced development, maintenance and infrastructure costs by deploying same functionality (by provisioning reusable services) through standardized interfaces. When an organization decides to adopt the SOA strategy, they align their business processes with existing or new technical services. When this degree of maturity is reached (where SOA adoption is considered), the organization can better handle the construction of a business process from an existing set of interacting services.

Thus, SOA governance, employed across the enterprise, ensures that the benefits of SOA are achieved by governing, planning, and executing aspects of SOA. At this level of maturity, SOA governance's scope is service centric. However, as the maturity of an organization increases; integration, operational, security and other lifecycle issues of real SOA deployments arise that disrupt much of the anticipated ROI. Some examples of these issues are as follows,

- Managing multiple versions of service interfaces and their corresponding service implementations
- Dynamic routing of consumer requests between different service versions
- Handling discrepancies in capabilities and standards between different services/versions
- Handling changing service policies, location restrictions, changing protocols and integration with supporting infrastructures
- Managing changing consumer requirements, for example, two consumers using the same service might have conflicting requirements at a later period of time

The challenges stated above can be solved by providing the existing services and business processes through a facade—a level of indirection. Thus, the service consumer does not invoke the service directly, but through the invocation of a virtual service [Ope09]. Governance eventually matures and requires a new set of policies that govern the processes in this abstraction layer. To achieve this level of agility, the scope of governance policies evolve from being well-defined, enterprise wide and service centric to dynamic, adaptive, business centric and cross-enterprise wide. As the organization matures to implementing virtualization layers, it gains a considerable degree of

flexibility in the design of integrated systems, in that different services that would otherwise not be interoperable can be more easily integrated. Moreover, in SOA, the service interface that hides the service description layer is directly accessible to the consumers, thereby creating additional overhead of managing change in the consumer's environment. With the introduction of a virtualization layer, any service can be reconfigured without the consumers having to transform their code, thereby achieving higher credibility. Virtualized services will also facilitate organizations to better align business requirements with IT capabilities by providing robust, highly flexible, manageable and scalable services.

Despite the numerous advantages, the virtualization layer is primarily about selection and location of specific instances of services, whose characteristics may have been, defined in advance [Ope09]. As the organization matures, it demands even more automation, flexibility, mobility, and storage but with enormous reduction in cost, maintenance and management. The organization strives for an ability to handle evolving business strategy dynamically by orchestrating enterprise assets and modelling services on the fly to transform to self-adaptive business processes. This notion of a self adaptive enterprise is provided by the cloud layer that offers dynamically reconfigurable services. The only effective way of implementing governance in the cloud is by applying automated policies. Governance in the cloud deals with defining policies around managing the above factors and tracking/enforcing the policies at run-time when the applications are running. The policies, provided within a cloud computing system, are even more sophisticated, in which they manage processes such as security, usage, compliance, and availability. Where SOA governance manages the lifecycle of a service,

cloud governance not only governs services (Platform, Infrastructure or Software), but also the cloud environment in which these services are executed. Policies are required for access control, logging, auditing, monitoring, SLA for both services used and their underlying infrastructure, QoS, and change management. Therefore, in addition to SOA governance policies, cloud computing requires another set of policies that manage static, dynamic and transition period of services provided in the cloud.

The powerful capabilities of solutions enabled by SOA when combined with emerging paradigms including virtualization and cloud computing define the evolving architectural model for next generation IT solutions.

4.4 Summary

In this chapter, we presented an overview on the various categories and sub-categories of policies which are classified on the basis of service lifecycle. In particular, we described six categories identified—Service Planning, Service Design, Service Development, Service Transition, and Service Operations Policies—along with the category of Multi Level Policies. Finally, the limitations of SOA governance as the organization evolves are discussed in Section 4.3.

Chapter 5 Governance in the Cloud

"Clouds come floating into my life, no longer to carry rain or usher storm, but to add color to my sunset sky."

—Rabindranath Tagore

5.1 Introduction

This chapter begins by addressing the limitations of SOA governance as the maturity level of an organization increases from being service oriented, to virtualized, to adopting the cloud [Ope09]. We then present information on the virtualization layer and cloud layer and their governance in this chapter. The processes in the virtualization layer that need be governed are stated in detail and the policies that control them are discussed. The three ways of implementing governance policies in the virtualization layer is explained in detail. Finally, governance policies in the cloud layer are discussed.

5.2 Virtualization Layer

As described in the previous chapter, a virtualization layer is introduced in an enterprise system to address the operational issues during SOA deployment. The addition of this level of indirection minimizes development and maintenance cost by taking advantage of the diminishing dependency between the service implementation and its consumer. As mentioned by [Cis07], service virtualization manages change in SOA. Adding a virtual interface means hiding the implementation of a service and instead of publishing the service interface to the consumer; a virtual interface, which closely resembles the service interface, is presented to the consumer. This abstraction layer (in case of virtual services)

that exists between the virtual service interface and the actual service interface provides the following.

- Harmonizes differences between the interfaces
- Performs transformations as required
- Integrates with other infrastructures
- Allows policy-based changes to be made
- Allows changes in the implemented service; without affecting its dependant consumers
- Handles changing customer requirements more efficiently, by providing support for different security mechanisms, credential types, messaging models and protocols for each interface [Cis07]

The following figure depicts a virtualized layer in an enterprise system. Each and every layer of an enterprise system can be abstracted through an interface which enables standardized communication to the functionality, hiding the actual implementation, its complexity and platform types. Virtualization can be provided to services, infrastructure (platforms, network, storage mechanisms, servers), applications and business processes [Gri07]. Virtualization can be provided across enterprise layers in different ways. At the operating systems layer and enterprise component layer, virtualization can be provided using virtual machines, clustering techniques and grid computing techniques. Clustering techniques can be useful in providing abstraction in the middleware layer. For service virtualization and business process layer virtualization, SOA, to a certain extent, provides abstraction above the applications and network. At the business process layer, virtualization is provided using rules engines that orchestrates the services. BPEL is a

great example for virtualization as it enables the design of business processes without the need to understand the communication between services that are developed to achieve an end-to-end functionality. At the collaboration layer, Mashups can be utilized to present information that is an aggregation of contents from disparate sources [All08b].

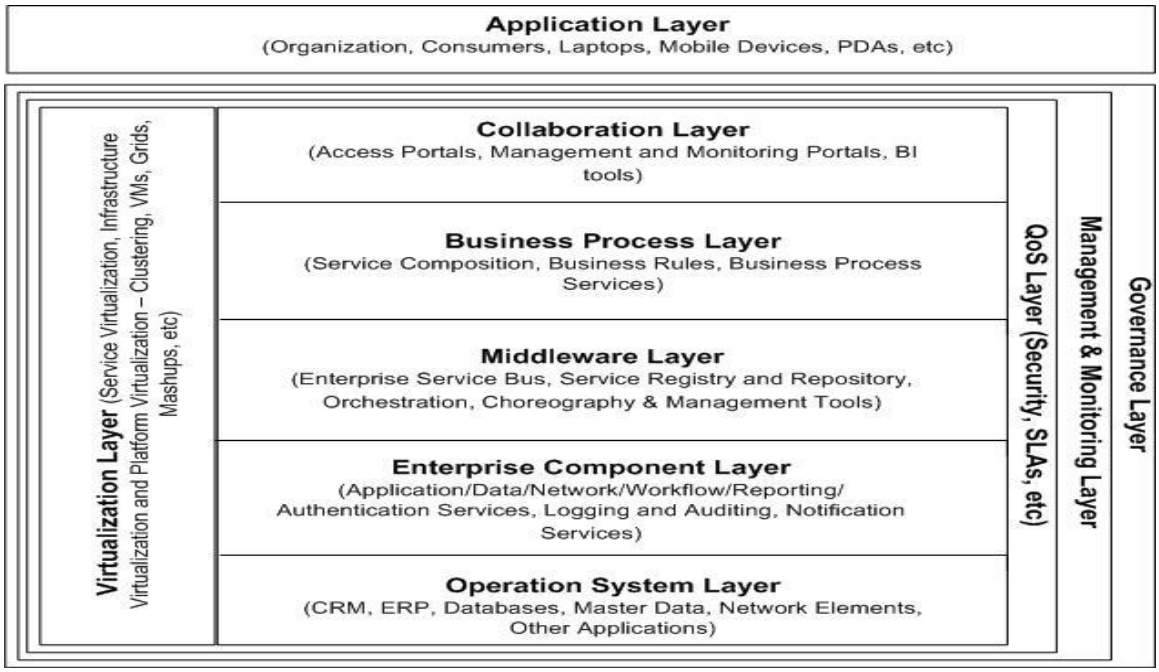

Figure 30: Introduction of Virtualized Layer in an Enterprise System

The virtualization layer, as mentioned in [Cis07], exhibits the following set of capabilities that enhance the loosely coupled nature of SOA.

Managing Service Versions

The service interface often requires enhancements or changes. These changes are usually performed at development-time rather than run-time. This in turn requires changes to be handled in the service interface which requires temporary suspension of consumer activities due to the tight coupling between the consumer and the interface established at run-time. One of the ways of offering the consumer, uninterrupted service,

is by supporting multiple virtual instances of a service with mapping back to its deployed service implementation and interface. The virtualization layer, thus, handles mediation, mapping and other transformation activities to accommodate the discrepancies between versions. It is also capable of handling message/address filtering and routing messages to the corresponding implemented service.

Apart from managing different service versions, the virtualization layer is advantageous during service creation and deletion. The logging, auditing and monitoring capabilities embedded with the abstraction layer provides the necessary support for tracking and controlling the consumers using different service versions. During service deletion or creation, the virtualization layer has the ability to route consumer requests (or return a suitable error response) to newer versions by applying appropriate syntactic and semantic transformation.

Managing Interoperability with Changing Service Standards and Capabilities

Even the field of standardization experiences the release of newer versions and this leads to incompatibilities and derails the interoperability between services, development environments and platforms. Often, capabilities and mechanisms provided by such standards are not expressed in a controllable way within the service contract established between the service and the consumer [Cis07]. A virtualization layer supports uncoupled deployment of new standards on each side of a service interaction without requiring that all participants upgrade in a synchronized manner.

Managing Changing Consumer Needs

One classic advantage of having a level of indirection is the ability to perform changes at one end without any discontinuity of operations at the other end. Different consumer

demands are addressed differently through the same service. The virtualization layer handles all the changes and differences that are demanded by the consumer. In addition, the layer also manages the evolving service contract by providing SLA monitoring and debugging facilities and arbitration among requirements without affecting the underlying service implementation.

Managing Location Changes

The abstraction layer is location aware and provides the ability to adhere to compliance policies across geographical boundaries. This safeguards the service against accessing any sensitive information across geographic boundaries and routes or transforms it appropriately. This is particularly useful when the consumer base is distributed.

Managing Infrastructure Changes

The virtualization layer handles integration of services with infrastructures such as registries and repositories, enterprise service buses, messaging systems and access management systems. This enables the consumers to worry less about the message exchange patterns, which version of transfer protocols are running, or which platform the infrastructure is running.

5.2.1 Governable Capabilities of a Virtualization Layer

Virtualization hides the complexity and nitty-gritty of the underlying infrastructure, and alternatively furnishes the user with a uniform, simple abstraction as the interface [OSP+09]. The introduction of governance in this abstraction layer will establish more stable and efficient infrastructure, aid in the performance of the virtualized infrastructure, and enhance collaboration and team work. In addition to the numerous advantages and functionalities stated in the previous section, the virtualization layer supports a variety of

capabilities and mechanisms [Cis07], which when governed can produce a stable offering of the following enterprise services.

- Identity Management
- Access Control
- Security
- Privacy
- Information transformation
- Filtering mechanisms
- Dynamic Routing & Routing mechanisms
- Logging, Monitoring and Auditing
- Managing failures
- Load Balancing
- Mediating transportation
- Mapping and message translation
- Optimization of Physical Resource Utilization
- Syntactic and semantic mediation

As the organization advances to incorporate a virtualized layer in their system; an increased sophistication in the governance requirements is necessary. The processes implemented in the abstraction layer require governance to ensure resources exist to support the proposed business goals.

5.2.1.1 Governing the Virtualization Layer based on IBM's governance lifecycle

Comparing the same with the *Plan, Define, Enable, Measure* stages of IBM's governance lifecycle, the following activities would be most appropriate in each of these stages during virtualization. In the Plan stage, the virtualization layer must serve itself as a business deliverable satisfying high level enterprise requirements. This would include activities such as physical resource consolidation, availability and recoverability operations, optimized and time-efficient service provisioning and recovery considerations. After the virtualization plan is set up, the enterprise must compute the ROI and provisioning times to ensure cost savings is more transparent.

During the define phase, a clear framework for planning and building the virtualization solution is established. Virtualization, to a certain degree, modifies the existing infrastructure and changes the service delivery process of the enterprise. Hence, the governance monitors, in addition to ascertaining conformance, must constantly evaluate the cost and performance to not exceed what has been already incurred. Governance enables proper documentation of the virtualization infrastructure, procedural controls for operations, and a clearly outlined test process.

During the enable phase, governance ensures consistent and stable operation of the virtualization layer and measuring improvements to existing virtualization capabilities. Governance guarantees that risks are identified and current performances are documented on a regular basis. Controls for introducing new technologies are systematically followed. Questions such as follows arise: Does the system performance suffice the consumer? Is the virtualization layer over utilized? Does the virtualization layer pass all the security tests? Do the failover tests function as anticipated?

In the measure phase, improving the stability and availability of IT services is the primary focus. Governance ensures that solid and consistent change management is in place. Moreover, all the current configurations of systems within the virtualization infrastructure must be maintained and preferably automated [Las10].

5.2.1.2 From SOA governance to Virtualization Layer Governance

For efficiently delivering SOA benefits, the enterprise requires a scalable technique for provisioning the virtualization layer for interoperability, visibility across services, and security. As the deployment of SOA and the virtualization layer evolves, complexity management requirements enable augmentation of this abstraction layer. This leads to an increase in the sophistication of repository capabilities including service discovery, service publication, SLA management, and security. The service discovery process will now include identifying virtualized enterprise assets; SLA management will reference both the virtualized assets and the physical assets. The management of this virtualized interface requires WS-Policy to be published in the middleware or the repository accessible to the consumer. Moreover, with this abstraction, the providers offer a wide range of options for selecting authentication tokens; information/data exchange protocols among dozens of other preferences. The selection of these options is supported in the WS-Policy framework but managing these preferences and supporting them is an uphill battle. During execution, the selection of a subset of preferences based on the criteria that change dynamically is, again, a policy decision. One way of managing dynamically changing preferences and change in policies is by defining a well-defined location where policies can be established and modified.

The concept of virtualization is designed to protect the consumer from any change and speed up the deployment process. For enterprise architects designing using SOA, virtual services and the abstraction layer provide known, well-defined locations for policy control and monitoring [Cis07]. As mentioned by [Cis07], virtualized layer and infrastructure provide control points where changes are defined and monitored with no modification of the underlying code and its performance. In the virtualized layer, service contracts and descriptions for implemented services in the form of WSDL can be imported from a UDDI directory or other source, enhanced to apply specific policy requirements, and then republished as a virtual service interface. For instance, the Cisco ACE XML Gateway acts as a policy enforcement and monitoring point to allow consistent policy application across all services and to monitor and audit application system behaviour [Cis07].

Current virtualization techniques suffer from being based on managed configurations with static properties [OSP+09]. Also, an increase in the heterogeneity and complexity of the underlying infrastructure leads to an escalation in the number of tuneable parameters and selecting a subset of these parameters based on the dynamics of resources and environmental context requires a cognitive way of policies and their enforcement techniques. In the virtualization layer, establishing smart controls would mean adapting the parameters through short term reasoning and experience. These smart controls must be able to govern any conflicting operation based on the environmental context. The following section classifies governance policies based on their degree of cognition.

5.2.2 Policy Characterization Based on Enforcement Style

The virtualization layer is a great location for supporting and providing policy enforcement and instrumentation points for control and monitoring purposes [Cis07]. Incorporating intelligence within policies is essential as cognitive rules are cost efficient, less-redundant and more reactive; which primarily is the purpose of adopting virtualization. Based on the policy enforcement style, policies can be converted into three broad types.

5.2.2.1 Static Policies

These policies do not bear any self-knowledge. The static policies exist in the registry, either as an inclusion in the service description or as a mapping to its corresponding service. This is useful when there in a one to one mapping between the service and its consumer. The policies have a well-defined description about the conditions in which they should execute and the processes they are required to monitor.

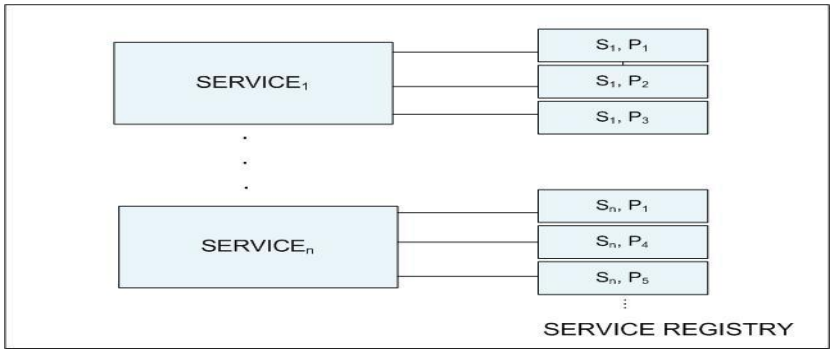

Figure 31: Static Policy in a Virtualization Layer

5.2.2.2 Adaptive Policies

These policies are capable of performing any context-based selection. These policies are the rules that enable the selection of policies appropriate for the service based on the service state and environment. The adaptive policies convey the selection information to

the middleware which in turn selects the policies that exist in the registry associated to the service. Whereas in one to one mapping between the service and policies, there was a defined set of policies statically associated with the service description, in this case of an adaptive policy, the list of policies associated with the service in the registry can change dynamically at run-time. There is a master-slave relationship between the policy associated with the service and the actual policies. Therefore, in a change event, the master policy needs to be modified without affecting the associated policies and their mapping. Control is offered through multiple service-policy entities and most often these entities execute control with/without communicating with each other [OSP+09].

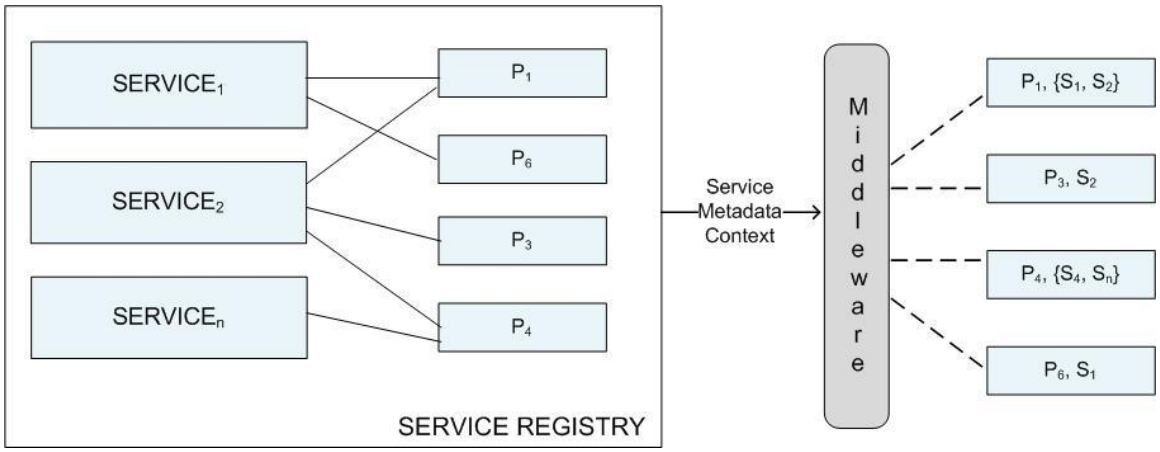

Figure 32: Adaptive Policy in a Virtualization Layer

5.2.2.3 Self-Adaptive Policies

Self-adaptive policies can perform any context-based addition or deletion of other rules/policies. They closely resemble adaptive policies but do not require any mapping to exist between the service description and the corresponding policies. Instead, the policies are stored in a separate location such as the policy store. These policies provide information to the middleware about the policies that must be added to or deleted from the service. Based on the gathered state and context information about the execution

environment, user requirements, and usage patterns the middleware offers a plan of action. Thus, in event of change, the master policy will be able to handle routing of requests to the appropriate policy. The policy store executes cognitive control. Any change in the policy store would thereby not affect the services under execution.

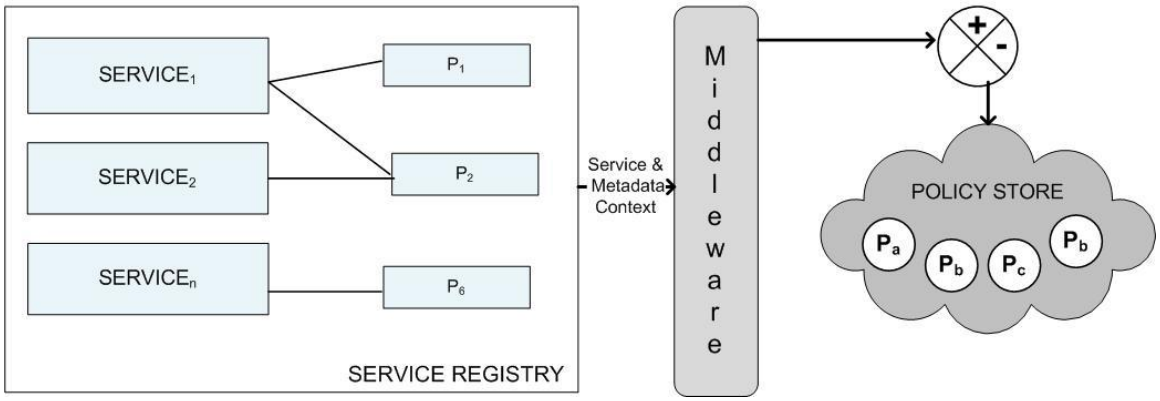

Figure 33: Self-Adaptive Policy in a Virtualization Layer

5.3 Adoption of the Cloud Layer

SOA has emerged as a practice of developing and delivering enterprise architecture by enabling a high degree of reuse and collaboration across enterprise and technological boundaries and also, by promoting high degree of collaboration. With the existence of SOA in an organization, its IT is highly responsive to evolving business requirements and not inhibited by the existing implementations. As mentioned in the previous section, an abstraction layer allows better sharing of resources and enables companies to build improved solutions using different delivery models and sourcing techniques. Until now, solutions have been focussed towards entire development lifecycle; be it of the services or the virtualization layer. At this level of maturity (SOA and virtualization), the organization's IT assets form an essential part of a company's capabilities, that were required to be kept under tight control to ascertain that the company's requirements were

fully met. As the organization matures (towards adopting cloud), IT assets are not viewed as company's capabilities but rather as being a facilitator of business processes.

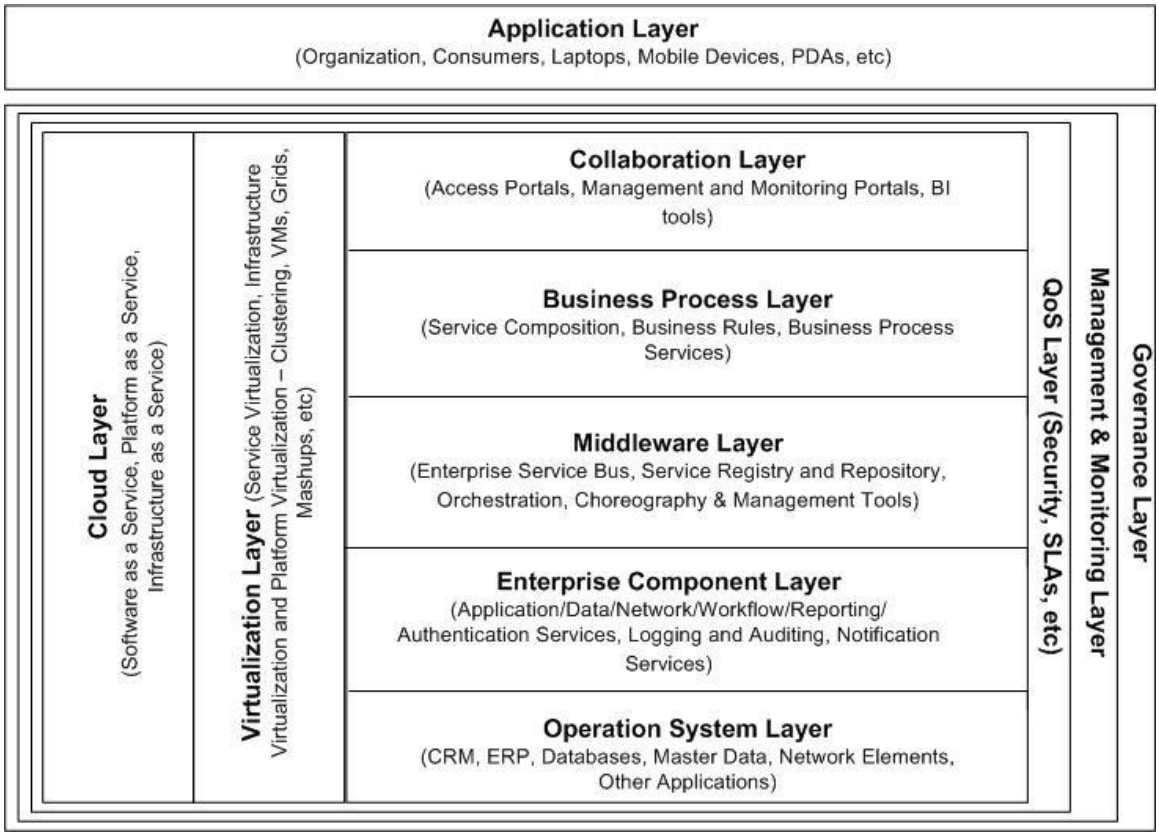

Figure 34: Introduction of Cloud Layer in the Enterprise Layer

The tight control of IT assets and their development lifecycles are replaced with partnerships with other organizations and standardization of applications [MG09]. This kind of a flexible control and enablement of real-time delivery of products, services and solutions over the Internet is offered by a paradigm called cloud computing. Cloud computing focuses more towards service delivery, hence it is more often referred to as a deployment architecture. Figure 34 depicts the introduction of a cloud layer into an enterprise system. Cloud computing primarily deals with the delivery of software solutions and infrastructure. As shown in the figure, adoption of a cloud would mean

providing applications, software solutions, storage mechanism, operating systems, hardware and portals as a service on an on-demand, pay-per-use outsourcing model [All08b]. A cloud layer within the enterprise enables better reuse, agility to the IT implementations and allows greater return on investment by injecting the pay-per-use concept.

Having a well-defined SOA within an organization eases the introduction of cloud computing into an enterprise system. Services, which are the basic units of SOA, can be deployed anywhere and accessed through ESB or over the Internet using communication standards such as WS-* standards. Furthermore, with SOA it is easier to share common services and work with third-party service providers, as most of the concerns surrounding SLAs, security, and service integration are handled by SOA directly. Both cloud and SOA are all about delivering services to the business with increased agility, speed and cost effectiveness, and that can lead to greater innovation and improved return on investment.

5.4 Introduction to Cloud Computing

Cloud computing broadly describes off-premise, on-demand computing where the end-user is provided applications, computing resources, and services (including operating systems and infrastructure) by clouds services provider via the Internet [Par09].

Cloud Computing focuses on:

a) Virtualization of infrastructure and services;

b) Automated provisioning of services;

c) Increased availability and connectivity with end users [IMS09].

Based on the delivery model, services in the cloud can take the following forms:

a) **Software as a Service**—business applications, development and test tools are offered via a virtual interface.

b) **Platform as a Service**—Web Services, Web 2.0 applications, middleware are offered as a platform on top of which applications can be built, assembled and run

c) **Infrastructure as a Service**—servers, storage, networking and other infrastructure can be offered as a service through a virtualized interface.

In addition to the above, clouds can be classified based on their access types—Private Clouds and Public Clouds. *Public Clouds* are managed and owned by service providers and accessed through subscription rules set by the provider and deliver select set of standardized business process, application and/or infrastructure services on a flexible pay per use basis. While *Private Clouds* are owned and managed by the organization itself and access is limited to assets within the organization and drive efficiency, standardization and best practices in the services provided while *retaining greater customization and control* than public clouds would permit [IMS09].

5.4.1 Challenges Involved in Cloud Computing

Some of the significant concerns associated with cloud computing and third-party provisioning are [Sch09]:

- Security and Access Control
- Managing Failure
- Managing Information Loss
- Cost incurred in transitioning existing functionality
- Hidden, ongoing costs of managing third-party relationships (internal or external)

- Reusability support
- Service Management
- Service Pricing
- Service Availability

In addition to the above challenges, there are a lot of other concerns that blur the line of trust between a cloud provider and a consumer. Perhaps the reason why usage of the cloud is still nascent in the enterprise is because of an increasing chorus of concerns being voiced about the usage of cloud resources. The concerns are [MG09],

- Cloud availability
- Cloud security
- Erosion of data integrity
- Data replication and consistency issues
- Privacy issues
- Lack of auditing and logging visibility
- Potential for regulatory violations
- Application sprawl & dependencies
- Inappropriate usage of Services
- Difficulty in managing intra-Cloud, inter-Cloud, and Cloud and non-Cloud interactions and resources.

The above challenges and issues are largely, if not absolutely, governance concerns. Fortunately, whatever has been learnt or implemented in SOA governance can be, in many ways, applied directly to the issues of governance in the cloud.

5.4.2 Governing the Cloud

As highlighted in the previous sub-sections, the focus of cloud is in the provisioning and delivery of shared services by virtualizing the applications, platform and infrastructure such that the user of the service gets a seamless experience without being aware of the underlying provisioning methodology. Governance in the cloud is about defining policies around managing the above factors and tracking/enforcing the policies at run-time when the applications are running. Different cloud vendors have varying degrees of flexibility when it comes to giving their clients access to the underlying infrastructure. It becomes imperative for businesses to understand these capabilities and define policies that mirror the needs of the business.

Cloud computing facilitates a remarkable amount of flexibility and scalability for deploying and managing enterprise applications on the cloud. Along with the flexibility comes a list of concerns that have to managed and monitored more closely than the traditional systems. Availability, security, privacy, location of cloud services and compliance are just some of the aspects of the cloud that have to be monitored and managed closely [Par09]. Organizations that carry out cloud deployment initiatives without an extensible and auditable policy-based governance system risk exposure to security compromises, service failures, inadequate performance, violated service level agreements, and costly legal and regulatory penalties [Ser11].

As mentioned previously, concepts of SOA and the cloud go hand in hand while providing service visibility and governance. The process of governance comes into prominence when organizations have already initiated the adoption of SOA. Governance in the world of SOA is divided into design-time governance (defining policies to

services) and run-time governance. This will be equally applicable to the services used from a cloud. Service visibility and governance will provide service discovery within a cloud and the SOA governance for managing the lifecycle of services that are available in the cloud. However, because cloud computing deals primarily with service deployment; governing the changes at run-time takes prime importance. The policies pertaining to change behavior of a cloud can be termed as *"transition-time governance"*.

Oracle defines a run-time governance system that allows organizations to define and deploy policies for governing the quality of the deployed SOA system at run-time [Ora10]. These policies cover all the aspects of management and monitoring a service at run-time. The established policies also exhibit the ability to handle any changes in the IT assets or services. Cloud computing focuses prominently on deploying and delivering enterprise assets as services, concepts of SOA governance and change management can therefore be applicable to cloud governance. Oracle's concept of architecturally dividing run-time governance into three phases can be personalized to change-time governance. However, unlike services or the virtualization layer where the enforcement points are the service lifecycle processes, registry or the virtualization layer itself; cloud governance policies scale across the entire cloud infrastructure. Moreover, in SOA or the virtualization layer, the infrastructure and registry falls under the control and visibility of the organization unlike the cloud computing architecture. Because of the broader scale and scope of cloud governance policies, it would be more useful to provide information about the policy during policy definition. This information can simplify policy administration tasks for the organization and improve visibility of the policy scope and characteristics to the consumers. Müller et al. have specified the importance of increasing

visibility in large-scale, adaptive systems [MPS08]. Employing Müller's propositions and extending Oracle's governance approach towards cloud governance, there are four main aspects attached to a cloud governance policy:

- **Policy Ownership (Who)—**specifies its owner thereby resolving the increasing concern over who owns the service and cultivating trust between third party providers, stakeholders and consumers
- **Policy Definition (What)—**defines the change-time constraints enterprises enforced on their cloud infrastructure. It also describes the primary purpose of the policy.
- **Policy Enforcement Point (Where)—**specifies if the scope of the policy is the infrastructure/platform/software or the service itself and if it spans across the local network or the entire cloud.
- **Policy Execution Stage (When)—**specifies if the policy execution would take place at run-time, design-time or change-time.

Figure 35: The 4 W's of a Cloud Policy

Based on the phase of execution, policies can be characterized into three kinds,

1. Design-time cloud governance policies
2. Run-time cloud governance policies
3. Transition-time cloud governance policies

The following subsections will discuss the three types of governance policies essential for the cloud.

5.4.2.1 Design-Time Cloud Governance Policies

Designing and developing services in the cloud is different because the consumers have little visibility to the service assets built by unknown, third-party developers when compared to the service development where the organization owned the SOA infrastructure and registry. Because services in the cloud (due to the simplicity in their deployment and consumption) can be accessed by anyone, usage of these cloud capabilities can go undetected by the IT management.

Furthermore, there has to be a central point for a cloud consumer to view the services and associated policies. Also, current trends show that the design-time policies can be effortlessly enforced when the development and QA process are in control, lack notoriously in the cloud environment. Clearly, SOA governance's best practices must be adopted to know about SOA registries/repositories and governance processes and control must be given to the appropriate assets/processes to avoid chaos and failure. In addition to governing the aforementioned processes, a list of design-time policies with the tasks they govern is stated in the Table 33.

Table 33: Design-Time Cloud Governance Policies

Design-Time Cloud Governance Policy	Task
QoS (Quality of Service) Policies	Policies around the cloud infrastructure across platform and application to be monitored
Service Level Agreements	SLAs span across both application and the underlying infrastructure
Service Policies	These policies establish rules around monitoring and managing disparate service in the cloud
Access Policies	Controls the organizational roles who bear access to the cloud infrastructure

5.4.2.2 Run-Time Cloud Governance Policies

At run-time, governance in the cloud means controlling data and information that does not reside on systems that is under the organization's control. Also, one cannot assume that the security standards in the cloud will be similar to the standards followed within the organization. This requires for security policies to be much more fine-grained and much more stringent. Scrutiny has to be laid on every service and every data exchange happening between services (thereby requiring a data policy definition).

Some of the run-time governance activities in the cloud are stated as follows,

- Defining processes to be executed when a service in the cloud is unavailable
- Defining processes to be executed if the entire cloud is unavailable
- Establishing event change mechanisms when there is a service or cloud failure
- Addressing data and compliance issues by defining policies for third-party cloud providers to perform logging and auditing operations
- Setting up policies to deliberately compose all cloud services with internal auditing and logging processes deployed on the cloud or at least on local network

- Defining policies to negotiate better access to logging data from the cloud provider, and implementing policies surrounding the control of leakage of private information in the cloud
- Implementing usage policies to control the excessive, and potentially expensive, use of services in the cloud in unauthorized ways

An effective run-time governance approach must provide the means to control, monitor, and adapt services, both with on-premises and cloud-based implementations, and needs to provide consistency across internal SOA and cloud SOA. To realize the location independence feature of a cloud, management and governance must span across SOA infrastructure boundaries.

Schmelzer specifies a solution to the above problem by using network intermediaries and gateways that keep a close eye on traffic between the corporate network and the cloud [Sch09]. These intermediaries can assess data from the cloud to identify leakage of any sensitive information, or even filter information sent to the cloud; apply service access policies, and service usage policies. Table 34 specifies the cloud run-time policies and the processes they control [Sch09].

Table 34: Run-Time Cloud Governance Policies

Run-Time Cloud Governance Policy	Task
Compliance Policies	Monitors compliance to all the policies and metrics defined in the cloud
SLA Monitoring Policies	Monitors compliance to SLA across both, the application and the underlying infrastructure
Cloud Monitoring Policies	Includes cloud application monitoring and cloud performance monitoring
Cloud Auditing Policies	Handles auditing, logging and reporting policies
Security Policies	Handles cloud asset security
Messaging and Routing Policies	Routes requests to other cloud assets during unavailability and performs error-handling
Pricing Policy	Manages metering and billing of cloud services
Usage Policy	Monitors usage of cloud assets
Cloud Interaction Policy	Manages interaction between cloud assets, interaction between the assets within the local network/infrastructure or across clouds

5.4.2.3 Change Management and Transition-Time Cloud Governance Policies

The primary concern when dealing with services in the cloud is to govern cloud resources that have the capability to change over time. During this transition phase, cloud service can be modified or replaced with other services, changes in service performance must be measured, service testing must be executed at run-time, verification and validation of cloud resources must be performed, conformance to SLAs must be managed. As cloud services are progressively evolving to be autonomic, governance processes must also steadily transform to achieve autonomicity. Other factors to be

managed at transition-time are: version control management, configuration management, handling changes occurring via customization,

Governance policies must be employed across every asset, associated with the cloud, capable of experiencing version changes. Some of the assets/processes include: service implementation, service contract, policy, development and deployment process, service/cloud infrastructure, data, and schema [Che11]. Some of transition-time policies along with the processes they control are specified in Table 35.

Table 35: Transition-Time Cloud Governance Policies

Transition-Time Cloud Governance Policy	Task
Change Management Policies	Monitors and manages changes occurring in the cloud assets
Versioning Policies	Manages versioning, upgrades and capabilities of service assets across the underlying cloud infrastructure
Compliance Policies	Manages compliance with the design-time and run-time policies
Security Policies	Manages security issues on any event of change
Privacy Policies	Handles any erosion of data integrity and executes role-based access policies

Where SOA is an architectural approach and philosophy guiding the development and management of applications, cloud is a deployment and operational model suited to host certain types of services within an existing SOA initiative. The cloud concept within the SOA context is one of service infrastructure, implementation, composition, and consumption. The SOA concept within the cloud context is one of application-level abstraction of cloud resources. Therefore, it is more intuitive to consider governance in the cloud as evolved SOA governance.

5.5 Summary

This chapter discussed the evolution of governance requirements as the organization matures into adopting virtualization and the cloud layer. Based on the taxonomy described in Chapter 3, we focus on policy awareness whereas in the cloud, the focus shifts towards controlling run-time processes and the changes affecting them during their execution. The governance policies in the both the layers, i.e., virtualization and cloud, are discussed in detail.

Chapter 6 Case Study

"Doing lots of little Web services projects all over the place with no governance isn't SOA, it's just playing."

—Anne Thomas Manes, Burton Group

6.1 Introduction

In the previous chapters, a large number of policies were collected and classified on the basis of their usage, execution phase, or enforcement type for services, virtualization layer, and the cloud. In this chapter, we introduce a case study that exhibits the process of employing governance policies within an enterprise that implemented SOA from scratch. The case study that we extend is an IBM Redpaper publication involving a company named JHKL Enterprises (JHKLE) whose focus was to expand its travel agency business by developing a website consisting of reusable services based on its underlying business processes [Kee09]. IBM solved JHKLE's business and technical challenges by transforming their architecture from siloed (cf. Figure 36) to service oriented (cf. Figure 37) [Kee09].

We extend IBM's case study by incorporating governance policies, on JHKLE's SOA adoption process, based on Oracle's governance policy leverage points and our policy classification described in Chapter 4 and Chapter 5. This case study identifies some potential risks that occur during IBM's SOA adoption and how the implementation of governance policies, at the most appropriate lifecycle phase, can be helpful in mitigating those risks. Sections 6.3 and 6.4 describe the changes in governance policy functionalities when JHKLE evolves to integrate the virtualization and the cloud layer. With the

implementation of governance policies, the company can savor the benefits of establishing better control and visibility over the processes.

With the expertise of IBM, JHKL Enterprises (a fictitious company), is looking to expand its travel sector [Kee09]. This case study sheds light on the usefulness of employing governance policies across service development lifecycle phases during SOA adoption and provides an extension by rendering information on migration to the virtualization layer and eventually the cloud. How governance is implemented in this progressive environment is the main focus of this case study.

6.2 The Case

The online website set up by JHKL Enterprises is a one-stop shopping for all travel services. The prime focus points for this agency are: flight reservations, hotel reservations, traffic conditions, cruise reservations, weather conditions and other map features. This Web site should offer easy-to-use services for customers and simplify the company's underlying business processes.

The current architecture of the agency's travel website is as shown below in Figure 36.

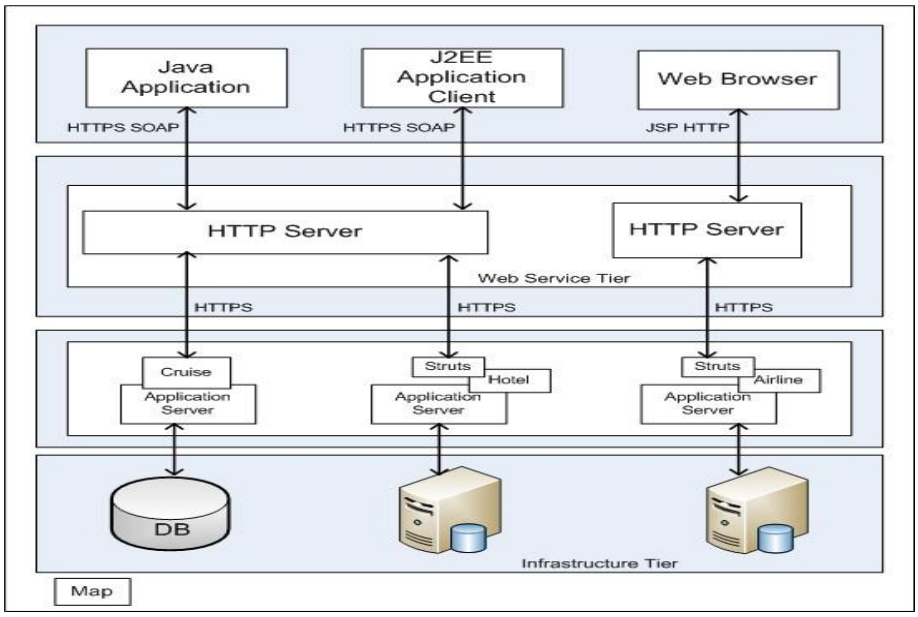

Figure 36: Current architecture of JHKL Enterprises [Kee09]

Before JHKLE's SOA adoption, they are presented with the following business and technical challenges.

Business Goals

- Expedite the process of information exchange with business partners by transforming the current architecture
- Remove business silos by leveraging SOA infrastructure and consumable service end points
- Provide non-disruptive additions of new business data and services

Technical Challenges

- The business system currently operates in a silo and concerns are raised when a multiple channel strategy is realized for customers

- Integrating third-party services into the overall solution is difficult. The inclusion of these third-party services is important, and helps to differentiate the company's travel offerings from the company's competitors
- Similarly, third-party companies are having difficulty integrating with company's services, limiting the use of agency's travel services by external parties.

The SOA strategy is adopted by the company to solve their business and technical challenges. The new architecture that was designed as a SOA solution is as depicted in Figure 37.

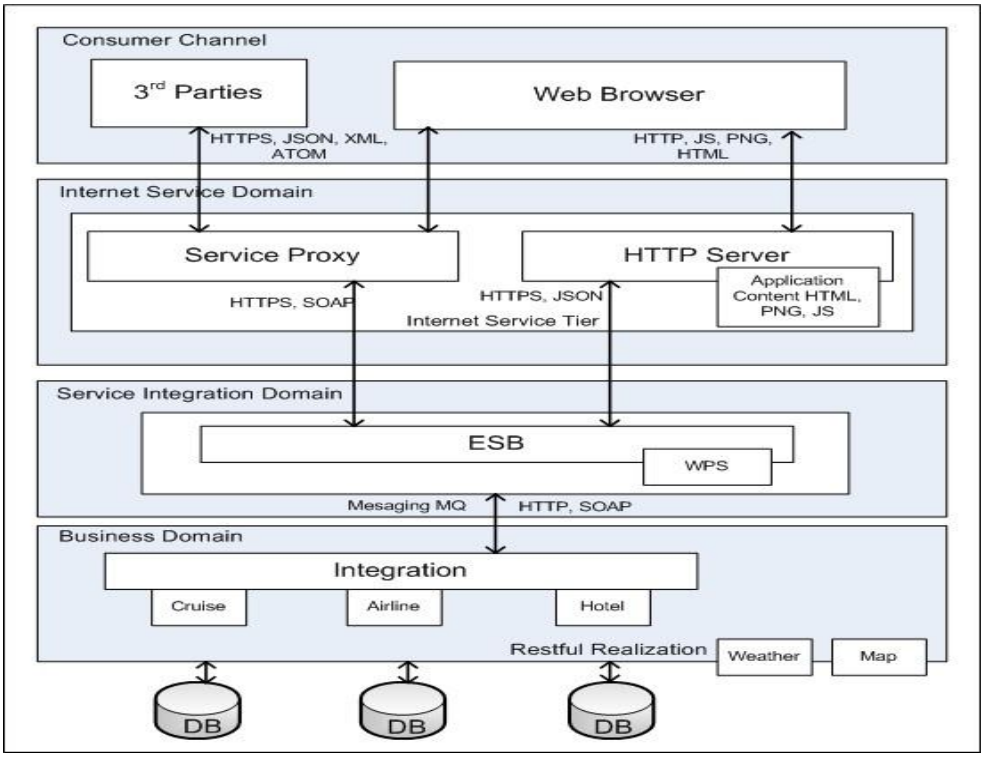

Figure 37: Proposed SOA based Architecture for JHKL Enterprises [Kee09]

To attain this architectural goal, JHKL enterprises chose to tread the path of a SOA development lifecycle. In this chapter, some potential SOA risks are stated first and the governance policies mitigating those risks are also mentioned.

Some of the risks that can occur during the initial phases of SOA adoption are: Once the service is under execution, who should be contacted for further changes? Who is responsible for fixing a potential service failure? How should the revenue generated from service execution be distributed? Therefore, during the phase of service creation, capabilities to identify which service is funded by whom and who owns what services must be identified thereby allowing governance to handle any unscrupulous access of service information. This establishment of roles and responsibilities and appropriate funding models is ensured via *service ownership policies* and *service funding models* respectively.

The business domain is primarily *travel* and to a certain extent, involves *banking*; *compliance policies* ensure that banking specific standards are implemented. In this case, a typical compliance policy will ensure that FIXX standard (standard for banking) and other regulatory compliance standards are understood and implemented appropriately which can otherwise lead to non-compliance to governmental and legal regulations.

Another risk during SOA adoption is a possible duplication of effort due to the failure of creating non-reusable services, potential overlapping of service portfolios and a failure to ensure interoperability across service domains. These issues can be combated by employing governance through *architectural policies.* These policies confirm that the architecture formulated support the stated business goals and principles. They also ensure that the architectural documents including use cases, service contract and interface design, development patterns are properly utilized and reference architectures and best practices and standards are adhered to.

Once the business process and service planning phase is completed, services must be identified, specified and realized. The service architecture must be defined and the service functionalities must be tested.

6.2.1 Services Candidate Identification

The fundamental benefits of well-defined services are to facilitate interoperability of services and to enable flexibility and reuse of business functionality within an enterprise and with partners [IMS09]. Based on the goals stated above, and the coarsest granularity of services as scope, the most apt service candidates for JHKL Enterprises would be airline, cruise, hotel, and credit card processing service as depicted in Figure 38.

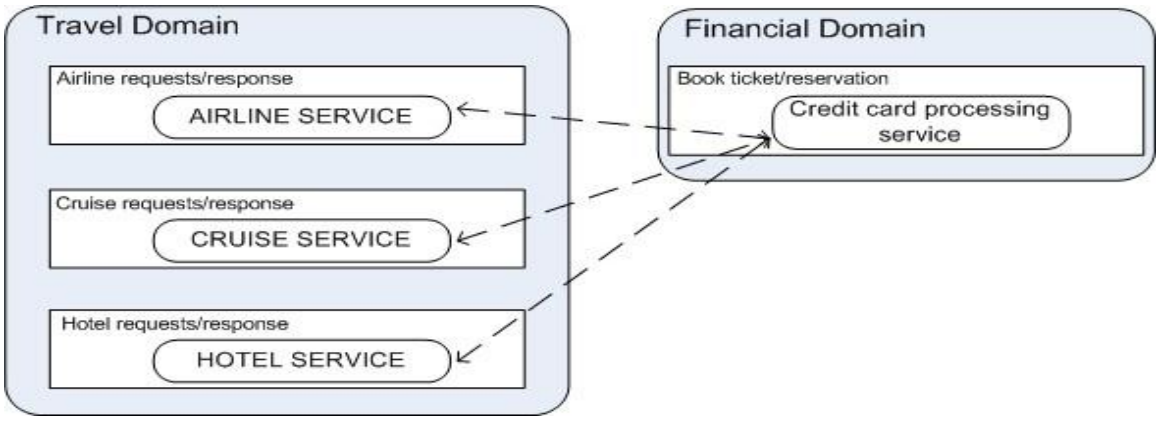

Figure 38: Service Candidate Identification

The above list of service candidates based on the business goals is neither a full list; nor with the finest granularity. There are several risks that JHKL Enterprises can face during this phase of service identification: What if the service identified is not compatible with the other service contracts? Is the service flexible to accommodate changes? Will the service artefacts be able to handle the load? These risks can be mitigated by the implementation of *service identification policies* that assure that each service activity

maps to the business service and any new business services created map directly to the business goals. These policies also ensure that the selection of services is communicated to dependent projects and that the services are tested thoroughly (for standards compliance and load) before their selection.

6.2.2 Service Prioritization

For JHKLE, realizing all the service candidates at the same time is not always necessary. Instead, some characteristics of service candidates must be helpful in determining which service demand most and immediate focus. *Service prioritization policies* help achieve tasks such as identifying service priorities based on stated metrics, confirming that service prioritization metrics comply with business requirements and that services are created on the priorities stated. An example of the priorities for JHKL Enterprises is stated in Figure 39 based on which service development activities can be prioritized.

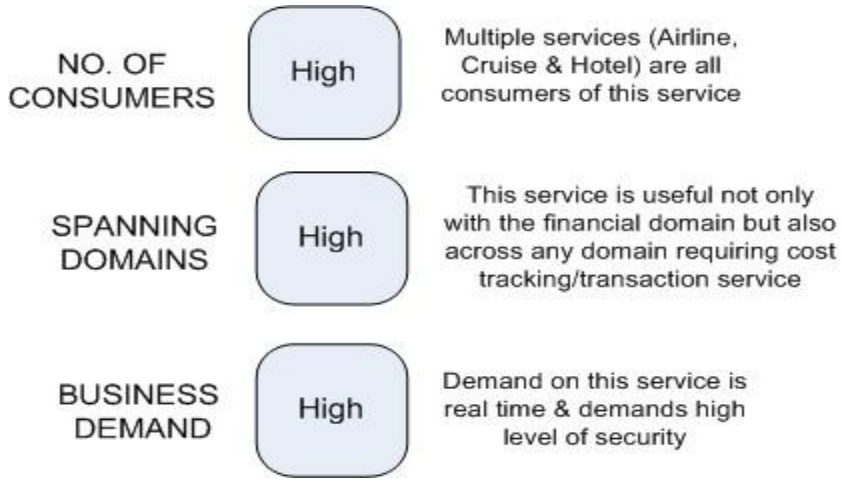

Figure 39: Service Prioritization Factors

6.2.3 Service Specification

Service specification policies manage compliance between service requirements and service specification. In addition, during the process of service specification, non-functional requirements must be stated along with the service. In this case, some of the non-functional requirements are security, logging (if the customer decides to purchase), availability of services, etc. During this design phase of a service, other processes include:

- Interface description—Service Interface must be based on the most acceptable standard. In this case, JHKL Enterprises can impose the usage of WS-I Basic Profile for generating compliant service interfaces. Governance through *Interface Definition Policies* ensures the usage of WS-I Basic Profile for the travel, airline and weather service interfaces.
- Context description—JHKL Enterprises requires a flexible architecture that can accommodate new services in the future. For example, the airline service might expand in the future accommodating services from other airline services from different geographical locations. In this case, the services developed must be designed with a thorough understanding of all the current service consumers so current service models and their usage is not affected in the event of change *(Behavioral Model Policies).*
    - Service dependencies—Services specified must be designed to satisfy the needs of other services depending on it (*Service Dependency Policy*). For example, JHKL Enterprises should confirm their service usage criteria to the

consumer only after calculating the performance levels of other services in mind.

6.2.4 Service Development Policies

In this phase, *service realization policies* involve managing activities such as formulating service level agreements as well as identifying and injecting service availability and security policies within the service. In the case where service are realized from existing systems; service performance results must be constantly fed back to the developers until it can meet the SLA requirements for the existing and newer customers of the service. One of the primary goals of JHKL Enterprises is to move away from the business silos and allow multiple connectivity to services across the organization. During the development phase, JHKL enterprises adopt the internal connectivity pattern through an ESB which allows access to appropriate service based on varying requirements irrespective of their location. However, some questions that can rise in this phase are: Assuming all the services are Java based, what would happen when a .Net service is invoked at a later period of time? Is there discrepancy in the way the internal and external consumers are treated? Are the current processes running in a batch, if yes, have all the processes replaced with real-time, event driven services?

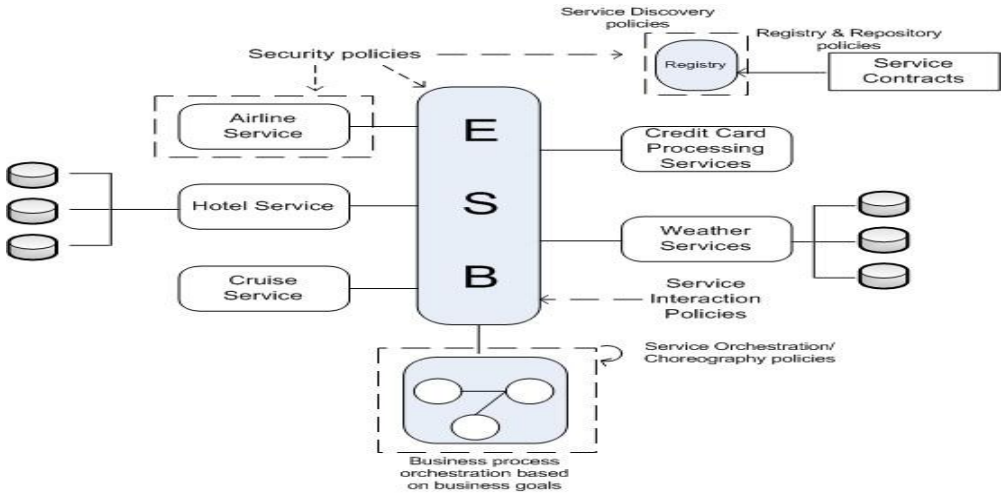

Figure 40: Governance Policies during Service Development

Based on the technical solution of JHKL enterprises' business goal and their concerns, as stated above, the following *technical policies* help govern and mitigate these risks.

- Payload policies that ensure that JHKL Enterprises have a consistent canonical data model which helps the organization achieve standardized definitions, tags and structure of data/services. (JHKLE can implement payload policies using XML schema policies, ATOM and JSON)
- JHKL Enterprises, based on the technology used (Web 2.0), must use WS-Security standards, SAML and XML Encryption to specify and implement their security constraints and requirements. *Security Policies* ensure that all the services implement the specified security constraints based on the accepted standard.
- WebSphere sMash: provides a development and run-time environment for creating the web applications. Governing application development using WebSphere sMash involves production of appropriate REST interfaces, mapping of REST requests to appropriate events. JHKL Enterprises uses

WebSphere sMash for production activities and it would be convenient to extend sMash to provide governance support by acquiring run-time information on REST interfaces.

6.2.5 Service Operations Policies

Once the service hits production, JHKL Enterprises must address the following concerns: When the service levels are increased, which service consumer is required to pay for the additional service artifacts provided? How is policy violations handled? How are operational factors like security, access, logging and billing addresses? These challenges can be address by the inclusion of governance policies that would manage run-time processes.

- *Messaging Policies*: Policies that will handle messaging transformation during service execution by transforming request-response messages from SOAP to REST and vice versa, thereby, enabling communication between provider's interface and the service consumer
- *Versioning policies* includes necessary heading and entity transformation, enrichment and filtering when requests are cast to multiple resource versions
- Quality of service mediation (*QoS policies, mediation policies*) is maintained by enforcing quality of service limits, throttling and shaping requests as necessary.
- JHKL Enterprises uses WebSphere DataPower (WDP) to integrate, secure and accelerate SOA. WDP maintains service operations according to the contract signed between consumer and provider for the resource usage. Based on this

agreement, the usage contract is monitored and enforced based on request rates and volume.

- At client side, user interface aggregation must be performed through standards. JHKL Enterprises uses Web 2.0 for their SOA adoption and JSR standards can be used to standardize all their web services interaction. If the company chooses to follow this standard, then governance will make sure the standards are followed appropriately across all the service and their interfaces.

Stated above are a few governance policies that can help govern certain activities in the SOA adoption process. The next section will describe about the processes that need to be governed, if JHKLE decides to implement the service virtualization layer.

6.3 Services to Virtualization Layer

From the user experience point of view, JHKLE provides the user uninterrupted service access even when there are changes being made to the database. Usually, travel details change every second and this requires intensive movement of data from and to the database. During this period, the service request handling slows down simply because the service interacts directly with the database for accessing any information. Once a change is made to the database, appropriate testing is required which further increases the latency. Hence the adoption of database virtualization is the most suitable solution for JHKL Enterprise. With this level of indirection, request loads can be balanced by routing requests appropriately. To the user it would appear as if they are accessing a physical database and this enhances productivity because the location of the user or the database has no effect in getting the job done.

Governance in the virtualization layer is straightforward because the virtualization layer provides significant policy enforcement and instrumentation points for control and monitoring. When the resources are virtualized, the policy enforcement is most important as it provides controlled access to resources based on authorization decisions and governance policies. Furthermore, centralizing the policy enforcement points reduces the agency's expenses of maintaining disparate services' security polices (or other database access policies) within the enterprise. The primary goal of JHKLE is to allow new services and channels to integrate quickly into their existing SOA architecture. When policies are added, deleted or updated; changed local policies must be appropriately dispatched to the corresponding policy engine or the service itself (if the policy is included in the service description). This nullifies the purpose for which the virtualization layer is adopted. Decoupling the policies from the services and decentralizing the policies would be the most ideal and efficient way of providing governance. Decoupling the policies completely from the services is not possible; hence the policies should evolve to being more context-aware.

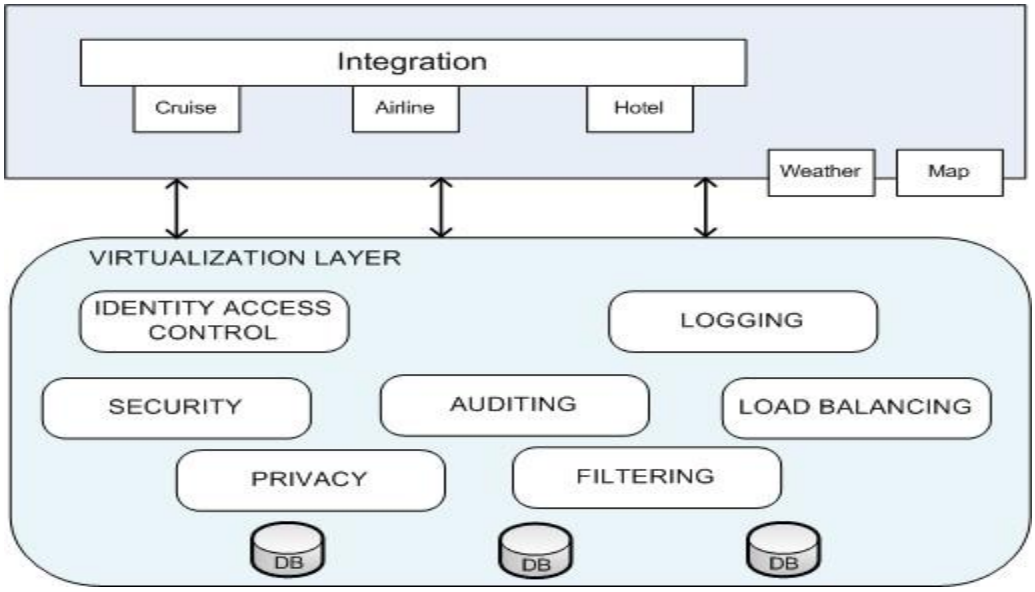

Figure 41: Database Virtualization in JHKL Enterprises

In Section 5.2.2, three kinds of governance policies are elucidated based on enforcement style. Because JHKLE services a lot of consumers, having policies embedded within the service would introduce latency and redundancy. Also, the agency does not deal with cross-enterprise establishments on a greater scale, thereby, policies change at a lesser time frequency but because all the three services (hotel, airline and cruise) have a fair amount of common characteristics, thereby common policies to govern most of the common processes. The most apt choice in this case is *adaptive policies*. With adaptive policies, the policies consist of service context and metadata and are able to instruct the middleware to choose the most appropriate policies from the registry. The processes that need to be controlled are specified in the Figure 41 describing the virtualized layer functionalities.

6.4 From SOA to the Cloud

The current architecture is built around the common tiers that separate the current business architecture into presentation, application, integration and virtualization within

the enterprise. Also, the servers must be provisioned to handle peak responses and will be wasted the response frequency is less. Moreover, web applications are known to go down when the traffic spikes up unexpectedly. Moreover, the hardware costs of building traditional web applications continue through the pre-production, beta and testing phases. Once tested, the hardware stays unused until a suitable candidate arises.

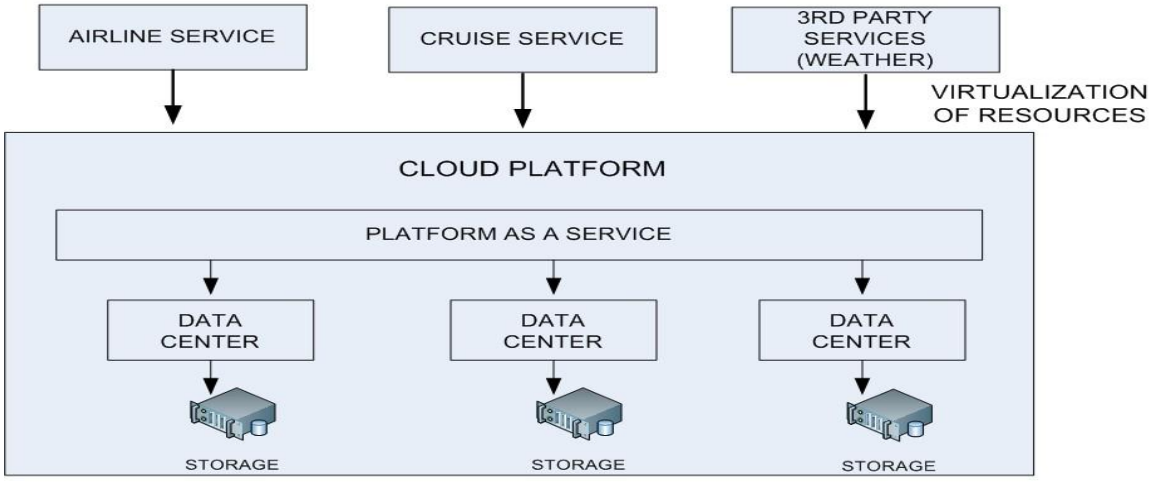

Figure 42: Providing Database as a Service

To combat the stated concerns, the cloud service provided by AWS (Amazon Web Service) is adopted by the agency. AWS can leverage on-demand provisioning of additional servers that allows capacity and costs to follow the traffic. The on-demand capability helps web applications scale-out and scale-in to match regular traffic spikes and thereby handle unexpected traffic spikes. Also, an AWS cloud can be utilized to simulate user traffic before a candidate release and the test bed is available only when required. A high level view of the adoption of AWS cloud computing infrastructure is described below.

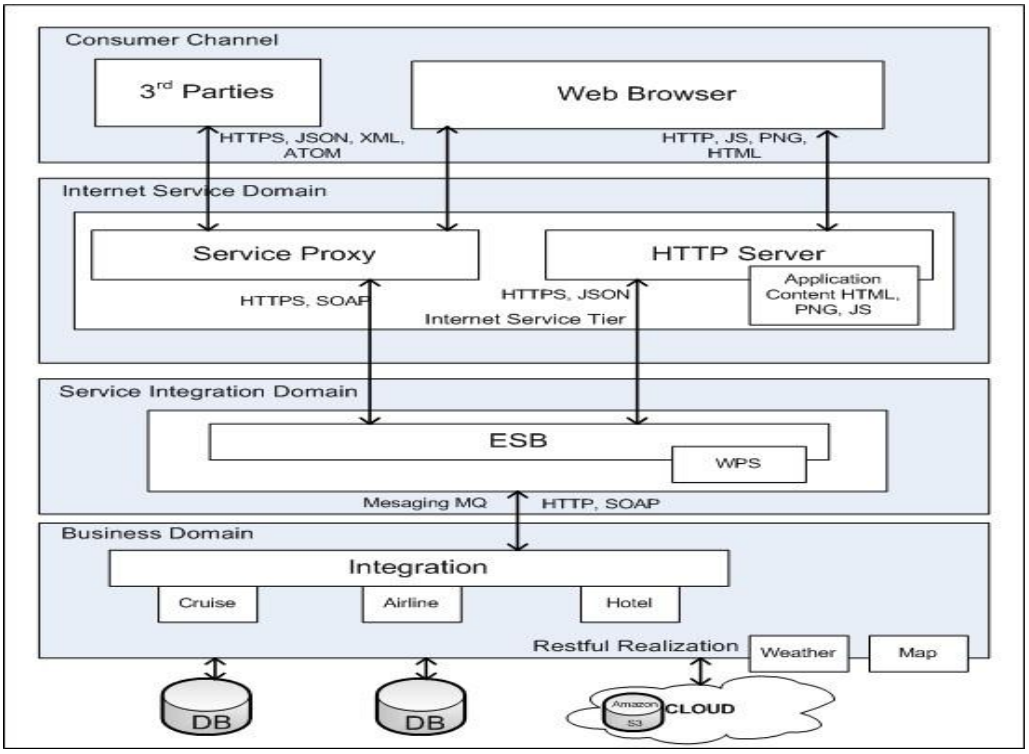

Figure 43: Amazon S3 in the SOA architecture of JHKL Enterprise

During the agency's migration to the cloud, it is important to identify processes that need to be governed at design-time and run-time. But cloud is typically a deployment model that complements the SOA infrastructure, change-time governance takes prime importance here. Change-time or transition-time (as it is called by the author) governance involves applying policies managing cloud service usage.

The big difference between governing services/virtualization layer and the cloud is the presence of a registry which is under the control of the enterprise. This registry manages the relationship between the services and the policies. However, cloud governance policies can range between local networks and third party providers. Simple issues such as difference in cloud security handling mechanisms between the enterprise and third party cloud providers can occur and these issues must be managed using governance

policies. Based on the three broad kinds of policies, the following policies are applicable in this case study.

During the adoption of AWS, the following policies must be defined. QoS policies, SLA, developer access policies and ownership policies. Yet another important factor associated with the clouds is time, so temporality is an important factor in framing cloud governance design-time policies. Whatever is framed at designed comes to fruition at run-time, thereby the policies framed here would be, compliance policies, monitoring the service contract, logging and security. Amazon S3 provides excellent logging facilities. At run-time, pricing policies are monitored, security policies are employed. During an event of change, compliance to design-time and run-time policies is monitored and managed. At change-time, security, SLA conditions, versioning are the primary processes that are managed here.

6.5 Summary

The primary aim of this case study is to explore the possibility of application of the categorized policies during a real-time implementation of SOA, virtualization layer and the cloud. However, this case study is coarse grained. Due to time and space constraints, only few aspects of governance were taken into account. Through this case study, it was observed that the previous chapters do not mention service migration governance which is stated as a drawback of our policy classification. As part of future work, it would be interesting to work through the entire process of SOA development and its migration to the cloud and observe the behavior of governance and its effect.

Chapter 7 Conclusions

"The very essence of leadership is its purpose. And the purpose of leadership is to accomplish a task. That is what leadership does–and what it does is more important than what it is or how it works."

—Colonel Dandridge M. Malone

7.1 Thesis in a Nutshell

Service-oriented architecture policy adds important business and technical flexibility and control to an SOA-based solution. Policy oriented governance offers full control and management of services from creation to retirement, including managing and maintaining the associations to business services [ACH+07]. At run-time, SOA policy provides ready access to change key operating characteristics of a service, including business parameters such as approval limits and transaction routing. During development, SOA policy controls key aspects of how services are built. The nature of SOA—highly distributed, heterogeneous, and very dynamic—means that it is critical for SOA artifacts to be governed by specific business, technical, and regulatory policies. Once the policies are defined, they are used throughout the service lifecycle. However, in the current industrial setting, policies are often created in ad hoc ways and communicated through mechanisms that are incompatible with services architecture and lifecycle model. To address this issue, it is of prime importance to group policies that services have to adhere to into classes that have similar management requirements. In this manner, policies can be identified and established as control points in every phase of service lifecycle. Policy taxonomy serves multiple purposes. Effectively implemented, the taxonomy provides the

cornerstone of an SOA strategy, supports linkage to other IT infrastructure issues (such as security) and can impart broader enterprise value. Yet another advantage of classifying policies in accordance with lifecycle is its apparent benefit in the management and evolution of service oriented systems. As the organization evolves, it moves to the virtualization layer and the cloud, governance evolves as well. Chapter 3 depicts taxonomy for governance policies. Chapter 4 discusses the classification of governance policies for SOA based on service development lifecycle in detail. This chapter summarizes our survey by presenting data to the selected references and the service lifecycle phases. The figures depicted provide an overview of the overall distribution of papers by major service lifecycle policy type, distribution of papers for each major service lifecycle policy type by policy subtype. For this survey 314 policy papers were collected.

Chapter 5 discusses the evolution of governance as organization maturity increases. The transformation in the enterprise layers is clearly depicted. Governance requirements and the governance policies in the virtualization layer and the cloud layer are described in detail. Chapter 6 evaluates, at a coarse level, the classification of policies through a case study.

7.2 Contributions

The key contributions of this thesis are stated as follows.

- A comprehensive catalog of governance policies, based on our extensive survey, that will be helpful for SOA personnel to select policies based on their business requirements

- An elucidation of major governance policy types as an organization implements the virtualization layer and the cloud layer
- An introduction of a new taxonomy to characterize governance policies which is grounded in the work of other researchers
- A case study is provided as an exercise that walks through the process of applying governance using policies, as the organization advances to the cloud level
- We diligently exercised the employment of our policy classification to an existing case study by IBM and also discussed the applicability of governance policies when the virtualization layer and the cloud layer are included into an enterprise's SOA model.

7.3 Future Work

While much progress has been made over the past few years in the area of virtualization and cloud, many open research problems and challenges remain in this enthralling field of governance for virtualization layer and the cloud.

Governance model construction for the cloud: The process of governing the cloud requires the construction of models that can work around the performance needs of clouds. These models must be able to continually improve governance practices; thereby establishing long-term and reliable relationships with the stakeholders and the consumers. While in the cloud, quantifying trust is a crucial issue. The governance framework must be capable of rewarding the cloud consumers with trust. Müller et al. have done extensive study in the identification and application of feedback loops in complex systems and the benefits of incorporating feedback loops within any management model is glaringly

obvious [MKS06]. However, developing feedback based governance models using quality criteria for a wide range of application domains that are beyond the enterprise boundary poses significant challenges and determine the success of cloud development.

Governing and leveraging uncertainty: Uncertainty can take the form of changing environmental contexts, evolving requirements, or changing regulatory policies. Cloud developers must be able to govern uncertainty which may not be noticeable at design-time and cloud administers must be able to determine the static and dynamic constituents of a cloud environment. This would require continuous validation of performance criteria at run-time.

Determination of quality attributes: Representing and modeling quality features and deriving metadata for representing these quality attributes where the context is continuously changing can be quite a bit of challenge. Managing and modeling dynamic quality attributes to support dynamic generation of instances within the cloud is an interesting arena of focus for software engineers.

Predicting Maintainability Changes: The state of the cloud is never constant, in which case, determining a maintainability quantification model for the current and any future additions to the cloud is not easy. Concepts from the software evolution area can be adopted to support cloud dynamics. It would be interesting to employ Müller's classic solution of "feedback loops" to the cloud and identify its effect in predicting and controlling change and evolution. Also, metrics form an important part of governance and designing standards and metrics in these multiple domains of applications are an interesting area of research.

Orchestrating the cloud: Services in the cloud are added and removed dynamically and in this scenario, choreographing the services to work coherently with each other, to achieve the stated business goals, can be quite enticing. How would one test services in the cloud? How does the concept of user acceptance testing and white box testing change with the cloud? Along the same lines, verifying and validating the cloud models at run-time, is an important topic that can facilitate trust between cloud sellers and buyers.

Bibliography

[AALN07] H. Agndal, B. Axelsson, N. Lindberg, and F. Nordin. Trends in service sourcing practices. *Journal of Business Market Management*, 1(3):187–208, 2007.

[ACH+07] M. Afshar, M. Cincinatus, D. Hynes, K. Clugage, and V. Patwardhan. SOA governance: framework and best practices. Oracle Whitepaper, 2007.

[AEYC] D.S. Allison, H.F. El Yamany, and M.A.M. Capretz. Privacy and trust policies within SOA. In *Proceedings International Conference for Internet Technology and Secured Transactions (ICITST 2009)*, pages 1–6. IEEE 2009.

[AFC+] F. Akkawi, D.P. Fletcher, T. Cottenier, D.P. Duncavage, R.L. Alena, and T. Elrad. An executable choreography framework for dynamic service-oriented architectures. In *Aerospace Conference (AERO 2006)*, pages 13–30. IEEE 2006.

[AFPO08] M.S. Aktas, G.C. Fox, M. Pierce, and S. Oh. XML metadata services. *Concurrency and Computation: Practice and Experience*, 20(7):801–823, 2008.

[AGA+08] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley. SOMA: A method for developing service-oriented solutions. *IBM systems Journal*, 47(3):377–396, 2008.

[AGSG08] M. Anzures-Garcá and L.A. Sánchez-Gálvez. Policy-based group organizational structure management using an ontological approach. In *Proceedings 3rd International Conference on Availability, Reliability and Security (ARES 08)*, pages 807–812. IEEE, 2008.

[AKK03] V. Agarwal, N. Karnik, and A. Kumar. Metering and accounting for composite e-services. 2003.

[All08a] P. Allen. Agile SOA governance: Illusion or reality? *Enterprise Architecture Advisory Service Executive Update, Vol. 11, No. 22*, 2008.

[All08b] R. Alluri. Cloud oriented architecture. 2008.

[Asa94] K. Asatani. Standardization of network technologies and services. *Communications Magazine, IEEE*, 32(7):86–91, 1994.

[AZE+07] A. Arsanjani, L.J. Zhang, M. Ellis, A. Allam, and K. Channabasavaiah. Design an SOA solution using a reference architecture. *IBM DeveloperWorks*, 2007.

[AZHG06] R. Afandi, J. Zhang, M. Hafiz, and C.A. Gunter. AMPol: Adaptive messaging policy. In *Proceedings 4th European Conference on Web Services (ECOWS 2006)*, pages 53–64. IEEE, 2006.

[BB10] R.M. Bahati and M.A. Bauer. Towards adaptive policy-based management. In *Proceedings Network Operations and Management Symposium (NOMS 2010)*, pages 511–518. IEEE, 2010.

[BC06] W.A. Brown and M. Cantor. SOA governance: how to oversee successful implementation through proven best practices and methods. *IBM, Somers, NY*, 2006.

[BCD08] S. Basu, F. Casati, and F. Daniel. Toward web service dependency discovery for SOA management. In *Proceedings IEEE International Conference on Services Computing (SCC 2008)*, pages 422–429. IEEE, 2008.

[BDNGD09] L. Baresi, E. Di Nitto, S. Guinea, and S. Dustdar. Multi-dimensional service compositions. In *Proceedings 31st International Conference on Software Engineering-Companion (ICSE-Companion 2009)*, pages 323–326. IEEE, 2009.

[Bet02] C. Bettini. Obligation monitoring in policy management. In *Proceedings 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY 2002)*, pages 2–12. IEEE, 2002.

[BGG09] E. Billionniere, D. Greiman, and K. Gosha. A comparison of social service selection techniques. In *Proceedings 8th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2009)*, pages 260–265. IEEE, 2009.

[BGK04] M. Bakhouya, J. Gaber, and A. Koukam. Service discovery and composition in ubiquitous computing. In *Proceedings International Conference on Information and Communication Technologies: From Theory to Applications (ICTTA 2004)*, pages 489–490. IEEE, 2004.

[BGP00] F. Bernabei, L. Gratta, and R. Pietroiusti. A policy based architecture for guaranteed QoS multimedia services. In *Proceedings IEEE Conference on High Performance Switching and Routing (ATM 2000)*, pages 401–409. IEEE, 2000.

[BH06] L. Boursas and W. Hommel. Policy-based service provisioning and dynamic trust management in identity federations. In *Proceedings International Conference on Communications (ICC 2006)*, volume 5, pages 2370–2375. IEEE, 2006.

[Bis08] T. Biske. *SOA Governance*. Packt Publishing, 2008.

[BJS96] E. Bertino, S. Jajodia, and P. Samarati. Supporting multiple access control policies in database systems. In *Proceedings IEEE Symposium on Security and Privacy*, pages 94–107. IEEE, 1996.

[BLGM08] W.A. Brown, R.G. Laird, C. Gee, and T. Mitra. *SOA governance: achieving and sustaining business and IT agility*. IBM Press, 2008.

[BLK+09] R. Brennan, D. Lewis, J. Keeney, Z. Etzioni, K. Feeney, D. O'Sullivan, J. Lozano, and B. Jennings. Policy-based integration of multiprovider digital home services. *Network, IEEE*, 23(6):50–56, 2009.

[BLM+08] K. Becker, A. Lopes, D.S. Milojicic, J. Pruyne, and S. Singhal. Automatically determining compatibility of evolving services. In *Proceedings IEEE International Conference on Web Services (ICWS 2008)*, pages 161–168. IEEE, 2008.

[BMEWE05] S. Babu Musunoori, F. Eliassen, and VS Wold Eide. QoS-driven service configuration in computational grids. In *Proceedings 6th International Workshop on Grid Computing (GC 2005)*, pages 304–307. IEEE Computer Society, 2005.

[BMT06] W.A. Brown, G. Moore, and W. Tegan. SOA governance-IBM's approach. *Somers, NY*, 2006.

[BP09] A. Bertolino and A. Polini. SOA test governance: enabling service integration testing across organization and technology borders. In *Proceedings International Conference on Software Testing, Verification and Validation Workshops (ICSTW 2009)*, pages 277–286. IEEE, 2009.

[BSX+08] H. Bai, M. Song, H. Xu, Q. Wang, and L. Fu. An optimized design of service orchestration. In *Proceedings 3rd International Conference on Pervasive Computing and Applications (ICPCA 2008)*, volume 2, pages 980–984. IEEE, 2008.

[BTPT06] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti. Run-time monitoring of instances and classes of web service compositions. 2006.

[BWR09] A. Barker, C.D. Walton, and D. Robertson. Choreographing web services. *IEEE Transactions on Services Computing*, pages 152–166, 2009.

[BZ08] V. Borovskiy and A. Zeier. Evolution management of enterprise web services. In *Proceedings IEEE Symposium on Advanced Management of Information for Globalized Enterprises (AMIGE 2008)*, pages 1–5. IEEE, 2008.

[CAA04] S.A. Chun, V. Atluri, and N.R. Adam. Policy-based web service composition. 2004.

[CBBJ08] F. Chauvel, O. Barais, I. Borne, and J.M. Jezequel. Composition of qualitative adaptation policies. In *Proceedings 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008)*, pages 455–458. IEEE Computer Society, 2008.

[CDM+03] A. Cole, S. Duri, J. Munson, J. Murdock, and D. Wood. Adaptive service binding middleware to support mobility. 2003.

[Che11] P. Cheliah. SOA governance in the cloud. *SOA Magazine Issue XLVII, February*, 2011.

[CHK+04] H. Chen, S. Hariri, B. Kim, Y. Zhang, and M. Yousif. Self-deployment and self-configuration of pervasive network services. In *Proceedings International Conference on Pervasive Services (ICPS 2004)*, page 242. IEEE, 2004.

[CHR+08] A. Cartlidge, A. Hanna, C. Rudd, I. Macfarlane, J. Windebank, and S. Rance. An introductory overview of ITIL V3. In *The IT Service Management Forum. http://www. itsmfi. org/files/itSMF_ITILV3_Intro_Overview. pdf*, volume 24, 2008.

[CI04] M.W. Chowdhury and M.Z. Iqbal. Integration of legacy systems in software architecture. *Specification and Verification of Component-Based Systems (SAVCBS 2004)*, page 110, 2004.

[Cis07] Service virtualization: Managing change in a service-oriented architecture. *Cisco, http://www.whitepapersdb.com/white-paper/2597/service-virtualization-managing-change-in-a-service-oriented-architecture, June*, 2007.

[CK07] S. Chang and S. Kim. A systematic approach to service-oriented analysis and design. *Product-Focused Software Process Improvement*, pages 374–388, 2007.

[CL08] P.P.W. Chan and M.R. Lyu. Dynamic web service composition: a new approach in building reliable web service. In *Proceedings 22nd International Conference on Advanced Information Networking and Applications (AINA 2008)*, pages 20–25. IEEE, 2008.

[CLZ10] S. Chen, J. Lukkien, and L. Zhang. Service-oriented advanced metering infrastructure for smart grids. In *Proceedings Asia-Pacific Power and Energy Engineering Conference (APPEEC 2010)*, pages 1–4. IEEE, 2010.

[CMQ08] Z. Chang, X. Mao, and Z. Qi. Formal analysis of architectural policies of self-adaptive software by bigraph. In *Proceedings 9th International Conference for Young Computer Scientists (ICYCS 2008)*, pages 118–123. IEEE, 2008.

[CMT06] A. Corradi, R. Montanari, and A. Toninelli. Dynamic configuration of semantic-based service provisioning to portable devices. 2006.

[Com04] A. Computing. An architectural blueprint for autonomic computing. *white paper*, 2004.

[CVK10] M.B. Chhetri, B.Q. Vo, and R. Kowalczyk. Policy-based management of QoS in service aggregations. In *Proceedings 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid 2010)*, pages 593–595. IEEE, 2010.

[CWX10] X. Cao, H. Wen, and H. Xia. Research on acquisition policy under some special cases. In *Proceedings International Conference on E-Health Networking, Digital Ecosystems and Technologies (EDT 2010)*, volume 1, pages 19–20. IEEE, 2010.

[CZFD07] T. Cheng, Z. Zhang, P. Feng, and J. Du. A CORBA-based integrated platform for service publishing, searching and sharing in virtual enterprise. In *Proceedings International Conference on Networking, Sensing and Control (ICNSC 2007)*, pages 530–535. IEEE, 2007.

[Dam02] N.C. Damianou. *A policy framework for management of distributed systems*. PhD thesis, Citeseer, 2002.

[DCCD08] Q. Du, C.H. Chi, S. Chen, and J. Deng. Modeling service quality for dynamic QoS publishing. In *Proceedings International Conference on Services Computing (SCC 2008)*, pages 307–314. IEEE, 2008.

[DSFR07] D. Di Sorte, M. Femminella, and G. Reali. On the performance of service publishing in IEEE 802.11 multi-access environment. *Communications Letters, IEEE*, 11(4):322–324, 2007.

[DW09] G. Dai and Y. Wang. Trust-aware component service selection algorithm in service composition. In *Proceedings International Conference on Frontier of Computer Science and Technology (FCST 2009)*, pages 613–618. IEEE, 2009.

[EAK06] A. Erradi, S. Anand, and N. Kulkarni. SOAF: An architectural framework for service definition and realization. 2006.

[EK01] C. Ensel and A. Keller. XML-based monitoring of services and dependencies. In *Proceedings IEEE Global Telecommunications Conference (GLOBECOM 2001)*, volume 3, pages 1646–1650. IEEE, 2001.

[EKP+10] B. Eze, C. Kuziemsky, L. Peyton, G. Middleton, and A. Mouttham. Policy-based data integration for e-health monitoring processes in a B2B environment: experiences from Canada. *Journal of Theoretical and Applied Electronic Commerce Research*, 5(1):56–70, 2010.

[EMT06a] A. Erradi, P. Maheshwari, and V. Tosic. Policy-driven middleware for self-adaptation of web services compositions. *Middleware 2006*, pages 62–80, 2006.

[EMT06b] A. Erradi, P. Maheshwari, and V. Tosic. Recovery policies for enhancing web services reliability. In *International Conference on Web Services (ICWS 2006)*, pages 189–196. IEEE, 2006.

[EMT07] A. Erradi, P. Maheshwari, and V. Tosic. WS-Policy based monitoring of composite web services. In *Proceedings 5th European Conference on Web Services (ECOWS 2007)*, pages 99–108. IEEE, 2007.

[ESSD08] S. Easterbrook, J. Singer, M.A. Storey, and D. Damian. Selecting empirical methods for software engineering research. *Guide to advanced empirical software engineering*, pages 285–311, 2008.

[ETP09] M. Eggenberger, D. Thurmond, and N. Parkash. Policy controlled message transformations. In *Proceedings World Conference on Services-I (SERVICES-I 2009)*, pages 578–584. IEEE, 2009.

[FCWH06] J. Feng, L. Cui, G. Wasson, and M. Humphrey. Policy-directed data movement in grids. 2006.

[FF03] T.K. Ferrell and U.D. Ferrell. Use of service history for certification credit for COTS [avionic software]. *Aerospace and Electronic Systems Magazine, IEEE*, 18(1):24–28, 2003.

[FFKS08] L. Fei, Y. Fangchun, S. Kai, and S. Sen. A policy-driven distributed framework for monitoring quality of web services. In *Proceedings IEEE International Conference on Web Services (ICWS 2008)*, pages 708–715. IEEE, 2008.

[FLO09] K. Feeney, D. Lewis, and D. O'Sullivan. Service-oriented policy management for web-application frameworks. *Internet Computing*, pages 39–47, 2009.

[FMA+06] W. Fang, L. Moreau, R. Ananthakrishnan, M. Wilde, and I. Foster. Exposing UDDI service descriptions and their metadata annotations as WS-resources. In *Proceedings 7th International Conference on Grid Computing (Grid 2006)*, pages 128–135. IEEE Computer Society, 2006.

[FOIY05] N. Fukuta, T. Osawa, T. Iijima, and T. Yamaguchi. Semantic service integration support for web portal. In *Proceedings International Conference on Web Intelligence (WIC 2005)*, pages 161–164. IEEE, 2005.

[GB07] J. Ganci and Inc Books24x7. *Best practices for SOA management*. IBM, International Technical Support Organization, 2007.

[GBG08] N. Guermouche, B. Benatallah, and C. Godart. Data messaging based approach for web service composition. In *Proceedings IEEE International Conference on e-Business Engineering (ICEBE 2008)*, pages 449–454. IEEE, 2008.

[GCT+08] G.R. Gangadharan, M. Comerio, H.L. Truong, V. D'Andrea, F. De Paoli, and S. Dustdar. License-aware service selection. In *Proceedings 10th IEEE Conference on E-Commerce Technology (CEC 2008) and the 5th IEEE Conference on Enterprise Computing, E-Commerce and E-Services (EEE 2008)*, pages 305–310. IEEE, 2008.

[GDA04] S. Ghamri-Doudane and N. Agoulmine. Hierarchical policy based management architecture to support the deployment and the discovery of services in ubiquitous networks. In *Proceedings 29th Annual International Conference on Local Computer Networks (LCN 2004)*, pages 126–133. IEEE, 2004.

[Geo05] J.C. Georgas. Knowledge-based architectural adaptation management for self-adaptive systems. In *Proceedings International Conference on Software Engineering (ICSE 2005)*, volume 27, page 658. Citeseer, 2005.

[GH05a] M. Ganna and E. Horlait. Agent-based framework for policy-driven service provisioning to nomadic users. In *Proceedings International Conference on Wireless And Mobile Computing, Networking And Communications (WiMob 2005)*, volume 4, pages 82–89. IEEE, 2005.

[GH05b] M. Ganna and E. Horlait. On using policies for managing service provisioning in agent-based heterogenous environments for mobile users. In *Proceedings 6th International Workshop on Policies for Distributed Systems and Networks (POLICY 2005)*, pages 149–158. IEEE, 2005.

[GHH+01] M. Garschhammer, R. Hauck, H.G. Hegering, B. Kempter, I. Radisic, H. Rolle, H. Schmidt, M. Langer, and M. Nerb. Towards generic service management concepts a service model based approach. In *Proceedings IEEE/IFIP International Symposium on Integrated Network Management (IM 2001)*, pages 719–732. IEEE, 2001.

[GHR06] M. Gudgin, M. Hadley, and T. Rogers. Web services addressing 1.0-SOAP binding. *W3C Recommendation, May*, 2006.

[GMP05] T. Gleason, K. Minder, and G. Pavlik. Policy management and web services. *Policy Management for the Web*, 2005.

[GN02a] X. Gu and K. Nahrstedt. Dynamic QoS-aware multimedia service configuration in ubiquitous computing environments. 2002.

[GN02b] X. Gu and K. Nahrstedt. A scalable QoS-aware service aggregation model for peer-to-peer computing grids. 2002.

[Gri07] A. Grigoriu. SOA and the virtualization of the enterprise. 2007.

[GRM06] S. Gorton and S. Reiff-Marganiec. Policy support for business-oriented web service management. In *Proceedings 4th Latin American Web Congress (LA-Web'06)*, pages 199–202. IEEE, 2006.

[GZH96] C. Gbaguidi, S. Znaty, and J.P. Hubaux. Service management: From definition to information modeling. In *Proceedings Global Telecommunications Conference (GLOBECOM 1996)*, volume 1, pages 457–462. IEEE, 1996.

[Han98] J. Han. A comprehensive interface definition framework for software components. In *apsec*, page 110. IEEE Computer Society, 1998.

[HC09] C. Huang and N. Crespi. Extended policies for automatic service composition in IMS. In *Proceedings IEEE Asia-Pacific Services Computing Conference (APSCC 2009)*, pages 254–259. IEEE, 2009.

[HDS03] R. Haas, P. Droz, and B. Stiller. Autonomic service deployment in networks. *IBM Systems Journal*, 42(1):150–164, 2003.

[HDW02] L. Hong, B. Dong, and D. Wei. A policy-based solution for management of enhanced network services. In *Proceedings of IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering (TENCON 2002)*, volume 3, pages 1684–1687. IEEE, 2002.

[He09] K. He. Integration and orchestration of heterogeneous services. In *Proceedings Joint Conferences on Pervasive Computing (JCPC 2009)*, pages 467–470. IEEE, 2009.

[HK04] R. Hirschfeld and K. Kawamura. Dynamic service adaptation. In *Proceedings 24th International Conference on Distributed Computing Systems Workshops (ICDCS 2004)*, pages 290–297. IEEE, 2004.

[HKG] R. High, S. Kinder, and S. Graham. IBM's SOA foundation: An architectural introduction and overview. *IBM developerWorks. http://www-128.IBM.com/developerworks/webservices/library/wssoa-whitepaper, Nov*.

[HKO10] A. Hock-Koon and M. Oussalah. Toward the definition of the loose coupling notion in a composite service. In *Proceedings 2nd International Conference on Computer Engineering and Applications (ICCEA 2010)*, pages 339–343. IEEE, 2010.

[HY09] D.D. He and J. Yang. Understand collaborative authorization policies: Models and specifications. In *Proceedings International Conference on Cloud Computing (CLOUD 2009)*, pages 206–213. IEEE, 2009.

[Hyn06] Kline S. Hynes, D. SOA web services cover story: SOA governance - gaining flexibility and retaining control. *http://soa.sys-con.com/node/204403*, 2006.

[I+07] N. Ibrahim et al. Automatic service-integration framework for ubiquitous environments. 2007.

[IBM07] IBM: Smart SOA: Best practices for agile innovation and optimization. *IBM White Paper, Nov*, 2007.

[IMS09] Adoption of service oriented architecture for enterprise systems in education: Recommended practices. *IMS Global Learning Consortium, Inc., http://www.imsglobal.org/soa/soawpv1p0/imsSOAWhitePaper_v1p0.pdf*, 2009.

[IPK+06] S. Illner, A. Pohl, H. Krumm, I. Luck, D. Manka, and F.J. Stewing. Policy-based self-management of industrial service systems. In *Proceedings IEEE International Conference on Industrial Informatics (INDIN 2006)*, pages 492–497. IEEE, 2006.

[IYH07] F. Ishikawa, N. Yoshioka, and S. Honiden. Developing consistent contractual policies in service composition. In *Proceedings The 2nd IEEE Asia-Pacific Service Computing Conference (APSCC 2007)*, pages 527–534. IEEE, 2007.

[JCY10] Z. Jiang, Y. Chen, and M. Yang. A research on multi-layer structure for dynamic service integration an innovative model based on SOA. In *Proceedings The 2nd IEEE International Conference on Information Management and Engineering (ICIME 2010)*, pages 590–594. IEEE, 2010.

[JfBZj05] T. Jing-fan, Z. Bo, and H.E. Zhi-jun. Policy driven and multi-agent based fault tolerance for web services. *Journal of Zhejiang University-Science A*, 6(7):676–682, 2005.

[JFS08] S. Jiang, J. Floch, and R. Sanders. Modeling and validating service choreography with semantic interfaces and goals. In *Proceedings IEEE International Symposium on Service-Oriented System Engineering (SOSE 2008)*, pages 73–78. IEEE, 2008.

[JJH+08] G. Jung, K.R. Joshi, M.A. Hiltunen, R.D. Schlichting, and C. Pu. Generating adaptation policies for multi-tier applications in consolidated server environments. In *Proceedings International Conference on Autonomic Computing (ICAC 2008)*, pages 23–32. IEEE, 2008.

[JKP08] J.Y. Jung, J. Kong, and J. Park. Service integration toward ubiquitous business process management. In *International Conference on Industrial Engineering and Engineering Management (IEEM 2008)*, pages 1500–1504. IEEE, 2008.

[Jon05] S. Jones. Toward an acceptable definition of service. *IEEE software*, pages 87–93, 2005.

[JZZ07] Y. Jin, J. Zhang, and X. Zheng. Specification and runtime enforcement of security policies. 2007.

[KB09] I. Kotenko and V. Bogdanov. Proactive monitoring of security policy accomplishment in computer networks. In *Proceedings IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS 2009)*, pages 364–369. IEEE, 2009.

[KC03] J. Keeney and V. Cahill. Chisel: A policy-driven, context-aware, dynamic adaptation framework. 2003.

[KC07] B. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. *Engineering*, 2(EBSE 2007-001), 2007.

[KDC+09] N. Kheir, H. Debar, F. Cuppens, N. Cuppens-Boulahia, and J. Viinikka. A service dependency modeling framework for policy-based response enforcement. *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 176–195, 2009.

[Kee09] Barcia R. Bishop K. Delio E. Kaushik R. Perrins M. Phillips D. Vay G. Keen, M. Case study: Web 2.0 SOA scenario. *IBM Redpaper*, 2009.

[KK08] S. Kuppuraju and A. Kumar. Enabling SOA governance for production deployed services. In *Proceedings IEEE Congress on Services-Part I*, pages 109–110. IEEE, 2008.

[KKJ+08] N. Kumar, P. Kumar, X. Jiang, S. Kowtha, E. Chow, H.P. Chang, M. James, D. Freides, G. MacArthur, and A. Mayer. Applying policy based enterprise management towards realization of real-time services. In *Proceedings Military Communications Conference (MILCOM 2008)*, pages 1–7. IEEE, 2008.

[KLMB96] M.J. Katchabaw, H.L. Lutfiyya, A.D. Marshall, and M.A. Bauer. Policy-driven fault management in distributed systems. In *ISSRE*, page 236. IEEE Computer Society, 1996.

[KNR09] K. Kannan, N.C. Narendra, and L. Ramaswamy. Extracting environmental information for improved web service matching and identification. In *Proceedings World Conference on Services-II (SERVICES-II 2009)*, pages 79–86. IEEE, 2009.

[KR07] Y. Kai and S. Robert. An ontology mediated web service aggregation hub. 2007.

[KTR07] V. Kaldanis, S.E. Tan, and R. Roswall. Charging and billing schemes for service provisioning in federated P2P environments. In *Proceedings 18th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2007)*, pages 1–5. IEEE, 2007.

[KW04] J.O. Kephart and W.E. Walsh. An artificial intelligence perspective on autonomic computing policies. In *Proceedings of Fifth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004)*, pages 3–12. IEEE, 2004.

[KW06] T. Klie and L. Wolf. Autonomic policy-based management using web services. In *Proceedings International Conference On Emerging Networking Experiments And Technologies (CoNEXT 2006)*, page 28. ACM, 2006.

[Las08] K. Laskey. Considerations for SOA versioning. In *Proceedings 12th Enterprise Distributed Object Computing Conference Workshops (EDOC 2008)*, pages 333–337. IEEE, 2008.

[Las10] N. Lasnoski. Virtualization and IT governance. *http://blog.concurrency.com/infrastructure/virtualization-and-it-governance/, April*, 2010.

[LBMS00] H.L. Lutfiyya, M.A. Bauer, A.D. Marshall, and D.K. Stokes. Fault management in distributed systems: A policy-driven approach. *Journal of Network and Systems Management*, 8(4):499–525, 2000.

[LBNH08] M. Le, BJF Beijnum, LJM Nieuwenhuis, and GB Huitema. An enterprise model for real-time inter-domain billing of services. In *Proceedings 12th Enterprise*

Distributed Object Computing Conference Workshops (EDOCW 2008), pages 405–412. IEEE, 2008.

[LC07] L. Li and W. Chou. Automatic message flow analyses for web services based on WSDL. 2007.

[LCHS05] K.M. Lee, K.H. Choi, S.P. Her, and D.R. Shin. Matchmaking algorithms to improve dynamic service matching in ubiquitous environments. In *Proceedings 4th Annual ACIS International Conference on Computer and Information Science (ACIS-ICIS 2005)*, pages 239–244. IEEE, 2005.

[LDC09] Z. Laliwala, A. Desai, and S. Chaudhary. Policy-based services aggregation in grid business process. In *Proceedings Annual IEEE India Conference (INDICON 2009)*, pages 1–4. IEEE, 2009.

[LGP+10] J.P. Loyall, M. Gillen, A. Paulos, J. Edmondson, P. Varshneya, D.C. Schmidt, L. Bunch, M. Carvalho, and A. Martignoni III. Dynamic policy-driven quality of service in service-oriented systems. In *Proceedings13th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC 2010)*, pages 1–9. IEEE, 2010.

[LGZ+05] J. Liu, N. Gu, Y. Zong, Z. Ding, and Q. Zhang. Service registration and discovery in a domain-oriented UDDI registry. 2005.

[LL98] JCS Leite and M.C. Leonardi. Business rules as organizational policies. In *Proceedings of the 9th international workshop on Software specification and design*, page 68. IEEE Computer Society, 1998.

[LLL10] T. Liu, T. Lu, and Z. Liu. An optimization approach for service deployment in service oriented clouds. In *Proceedings 14th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2010)*, pages 390–395. IEEE, 2010.

[LMKB01] H. Lutfiyya, G. Molenkamp, M. Katchabaw, and M. Bauer. Issues in managing soft QoS requirements in distributed systems using a policy-based framework. *Policies for Distributed Systems and Networks*, pages 185–201, 2001.

[LMRD08] P. Leitner, A. Michlmayr, F. Rosenberg, and S. Dustdar. End-to-end versioning support for web services. In *Proceedings IEEE International Conference on Services Computing (SCC 2008)*, pages 59–66. IEEE, 2008.

[LRD09] P. Leitner, F. Rosenberg, and S. Dustdar. Daios: Efficient dynamic web service invocation. *Internet Computing, IEEE*, 13(3):72–80, 2009.

[LS08] G.A. Lewis and D.B. Smith. Service-oriented architecture and its implications for software maintenance and evolution. In *Frontiers of Software Maintenance (FoSM 2008)*, pages 1–10. IEEE, 2008.

[LSBM08] F. Lécué, S. Salibi, P. Bron, and A. Moreau. Semantic and syntactic data flow in web service composition. In *Proceedings International Conference on Web Services (ICWS 2008)*, pages 211–218. IEEE, 2008.

[LSS09] J. Li, B. Stephenson, and S. Singhal. A policy framework for data management in services marketplaces. In *Proceedings International Conference on Availability, Reliability and Security (ARES 2009)*, pages 560–565. IEEE, 2009.

[LSZJ06] H. Liang, W. Sun, X. Zhang, and Z. Jiang. A policy framework for collaborative web service customization. In *Proceedings 2nd International Workshop Service-Oriented System Engineering (SOSE 2006)*, pages 197–204. IEEE, 2006.

[Luc01] D.C. Luckham. *The power of events: an introduction to complex event processing in distributed enterprise systems*. Addison-Wesley Longman Publishing Co., Inc., 2001.

[LW09] Y. Li and T. Wen. Quality and relation driven service selection for web services composition. In *Proceedings International Conference on New Trends in Information and Service Science (NISS 2009)*, pages 152–155. IEEE, 2009.

[LZC09] Q. Lv, J. Zhou, and Q. Cao. Service matching mechanisms in pervasive computing environments. In *Proceedings International Workshop on Intelligent Systems and Applications (ISA 2009)*, pages 1–4. IEEE, 2009.

[LZLG07] G. Li, W. Zhang, H. Li, and J. Guo. An efficient way to accelerate service discovery and invocation. In *IEEE International Conference on Systems, Man and Cybernetics (SMC 2007)*, pages 1304–1309. IEEE, 2007.

[Mar08] E.A. Marks. *Service-oriented architecture governance for the services driven enterprise*. John Wiley & Sons Inc, 2008.

[Mar09] R. Martignoni. Global sourcing of software development-a review of tools and services. In *Proceedings 4th IEEE International Conference on Global Software Engineering (ICGSE 2009)*, pages 303–308. IEEE, 2009.

[Mas07] M. Masone. Environmental certification for service networks and data centers. In *Proceedings 29th International Telecommunications Energy Conference (INTELEC 2007)*, pages 756–759. IEEE, 2007.

[MbC09] Y.H. Meng and H. Cai. Regulation policy of service industry and international trade in service. In *Proceedings International Conference on Management and Service Science (MASS 2009)*, pages 1–4. IEEE, 2009.

[MBLN09] A. Mahfouz, L. Barroca, R. Laney, and B. Nuseibeh. From organizational requirements to service choreography. In *World Conference on Services-I (SERVICES-I 2009)*, pages 546–553. IEEE, 2009.

[MCBFR08] S. Mosser, F. Chauvel, M. Blay-Fornarino, and M. Riveill. Web services composition: Mashups driven orchestration definition. *CIMCA 2008, IAWTIC 2008, and ISE 2008*, pages 284–289, 2008.

[MG09] M. Mertz and M. Gryning. Cloud computing and SOA. *www.mortengryning.dk/Work/coa.pdf*, 2009.

[MGN+09] H.A. Müller, P. Gupta, L. Nigul, R. Desmarais, A. Rudkovskiy, N.M. Villegas, and Q. Zhu. SOA governance optimizes the business and evolution of service-oriented systems. In *Proceedings of 3rd International Workshop on a Research Agenda for Maintenance and Evolution of Service-Oriented Systems (MESOA 2009)*, 2009.

[MHC01] R. Monson-Haefel and D.A. Chappell. *Java message service*. O'Reilly Media, 2001.

[MHD+05] J. McCabe, T.J. Harmer, P. Donachy, R.H. Perrott, C. Chambers, S. Craig, R. Lewis, B. Mallon, and L. Sluman. Grid service configuration and lifecycle management. In *Proceedings 1st International Conference on e-Science and Grid Computing (E-SCIENCE 2005)*, pages 7–pp. IEEE, 2005.

[Mil07] S. Mills. The future of business–aligning business and IT to create an enduring impact on industry. *IBM, Thought leadership paper. ftp://ftp.software.IBM.com/software/soa/pdf/future_of_business. pdf (2008-03-28)*, 2007.

[Mit05] K. Mittal. Service oriented unified process (SOUP). *IBM Journal*, 2005.

[MKS06] H. Müller, H. Kienle, and U. Stege. Autonomic computing: Now you see it, now you don't—design and evolution of autonomic software systems. *A. De Lucia & F. Ferrucci, International Summer School on Software Engineering (ISSE 2006)*, 2008:32–54, 2006.

[MMD10] L. Mostarda, S. Marinovic, and N. Dulay. Distributed orchestration of pervasive services. In *Proceedings 24th International Conference on Advanced Information Networking and Applications (AINA 2010)*, pages 166–173. IEEE, 2010.

[MOV10] J.B. Mwangama, V.G. Ozianyi, and N. Ventura. Charging and billing for composite services in a multi-service provider environment: The IMS case. In *Proceedings Wireless Communications and Networking Conference (WCNC 2010)*, pages 1–6. IEEE, 2010.

[MPS08] H. Müller, M. Pezzè, and M. Shaw. Visibility of control in adaptive systems. In *International Conference on Software Engineering (ICSE 2008)*, number 4, pages 0–11. Association for Computing Machinery, 2008.

[MR07] G. McBride and SOA Rational. The role of SOA quality management in SOA service lifecycle management. *IBM Developer Works, March*, 2007.

[MRVW05] J.B. Michael, S.E. Roberts, J.M. Voas, and T.C. Wingfield. The role of policy in balancing outsourcing and homeland security. *IT professional*, pages 19–23, 2005.

[MSK08] P. Mayer, A. Schroeder, and N. Koch. MDD4SOA: Model-driven service orchestration. In *Proceedings 12th International Enterprise Distributed Object Computing Conference (EDOC 2008)*, pages 203–212. IEEE, 2008.

[MSSR08] S. Mani, V.S. Sinha, N. Sukaviriya, and T. Ramachandra. Using user interface design to enhance service identification. In *Proceedings IEEE International Conference on Web Services (ICWS 2008)*, pages 78–87. IEEE, 2008.

[MWM07] M. Menzel, C. Wolter, and C. Meinel. Access control for cross-organisational web service composition. *Journal of Information Assurance and Security*, 2(3):155–160, 2007.

[NER+08] M. Niemann, J. Eckert, N. Repp, and R. Steinmetz. Towards a Generic Governance Model for Service Oriented Architectures. In *Proceedings Americas Conference on Information Systems (AMCIS 2008).* page 361.

[NFG+06] L. Northrop, P. Feiler, R.P. Gabriel, J. Goodenough, R. Linger, R. Kazman, D. Schmidt, K. Sullivan, and K. Wallnau. Ultra-large-scale systems-the software challenge of the future. *Technical report Software Engineering Institute Carnegie Mellon University ISBN*, 2006.

[Nol92] M. Nolan. Can industry standards satisfy DoD and service ATE standardization goals? Can a standardization policy be derived to satisfy everyone? In *Conference Record Systems Readiness Technology Conference (AUTOTESTCON 1992)*, pages 437–445. IEEE, 1992.

[NUH09] S. Nakajima, N. Ubayashi, and K. Hokamura. Runtime monitoring of cross-cutting policy. 2009.

[NZC08] S. Nepal, J. Zic, and S. Chen. WS-CCDL: A framework for web service collaborative context definition language for dynamic collaborations. In *Proceedings International Conference on Web Services (ICWS 2008)*, pages 781–784. IEEE, 2008.

[OLH+] A.K. Oracle, M. Little, R. Hat, S. Patil, and M.R. Bea. SCA service component architecture.

[Ope09] OSIMM maturity model for SOA. *Open Group*, 2009.

[Ora10] An agentless approach to runtime governance: Leveraging the platform to enforce policies. *An Oracle White Paper, http://www.imsglobal.org/soa/soawpv1p0/imsSOAWhitePaper_v1p0.pdf, April*, 2010.

[OSP+09] E. Onur, E. Sfakianakis, C. Papagianni, G. Karagiannis, T. Kontos, I. Niemegeers, IP Chochliouros, S.H. de Groot, P. Sjodin, M. Hidell, et al. Intelligent end-to-end resource virtualization using service oriented architecture. In *Proceedings GLOBECOM Workshops (2009)*, pages 1–6. IEEE, 2009.

[Par09] Understanding cloud computing. *Parallel Holdings Ltd., http://www.parallels.com/ca/spp/understandingclouds/*, 2009.

[PASB07] A.V. Paliwal, N. Adam, B. Shafiq, and C. Bornhoevd. Policy based web service orchestration and goal reachability analysis using MSC and CP nets. 2007.

[PB05] A. Paschke and M. Bichler. SLA representation, management and enforcement. In *Proceedings IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE 2005)*, pages 158–163. IEEE, 2005.

[Pel03] C. Peltz. Web services orchestration and choreography. *Computer*, pages 46–52, 2003.

[PHS+08] T. Phan, J. Han, J.G. Schneider, T. Ebringer, and T. Rogers. A survey of policy-based management approaches for service oriented systems. In *Proceedings 19th Australian Conference on Software Engineering (ASWEC 2008)*, pages 392–401. IEEE, 2008.

[PHS+10] T. Phan, J. Han, J.G. Schneider, T. Ebringer, and T. Rogers. Policy-based service registration and discovery. *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, pages 417–426, 2010.

[PKH+08] A. Pohl, H. Krumm, F. Holland, F.J. Stewing, and I. Lueck. Service-orientation and flexible service binding in distributed automation and control systems. In *Proceedings 22nd International Conference on Advanced Information Networking and Applications-Workshops (WAINA 2008)*, pages 1393–1398. IEEE, 2008.

[PKKJ04] A. Patwardhan, V. Korolev, L. Kagal, and A. Joshi. Enforcing policies in pervasive environments. 2004.

[Pra07] S. Prabhu. Towards distributed dynamic web service composition. 2007.

[PVDH06] M.P. Papazoglou and W.J. Van Den Heuvel. Service-oriented design and development methodology. *International Journal of Web Engineering and Technology*, 2(4):412–442, 2006.

[QSPvS06] D. Quartela, M.W.A. Steen, S. Pokraev, and M. van Sinderena. A conceptual framework for service modelling. 2006.

[QZC07] R. Quitadamo, F. Zambonelli, and G. Cabri. The service ecosystem: Dynamic self-aggregation of pervasive communication services. 2007.

[R+03] D. Rolls et al. Service provisioning markup language (SPML) version 1.0. *OASIS Committee Specification*, 2003.

[R+08] N. Repp et al. WS-Re2Policy: A policy language for distributed SLA monitoring and enforcement. In *Proceedings The 3rd International Conference on Systems and Networks Communications (ICSNC 2008)*, pages 256–261. IEEE, 2008.

[RB08] J. Rossebo and R. Bræk. Using composition policies to manage authentication and authorization patterns and services. In *Proceedings 3rd International Conference on Availability, Reliability and Security (ARES 08)*, pages 597–603. IEEE, 2008.

[RBK05] M. Randic, B. Blaskovic, and P. Knezevic. Modeling service dependencies in ad hoc collaborative systems. In *Proceedings The International Conference on Computer as a Tool (EUROCON 2005)*, volume 2, pages 1842–1845. IEEE, 2005.

[RG08] M. Rouache and C. Godart. A run-time service discovery process for web services compositions. In *Proceedings of the 10th international conference on Electronic commerce*, page 2. ACM, 2008.

[RK08] A. Razavi and K. Kontogiannis. Pattern and policy driven log analysis for software monitoring. In *Proceedings 32nd Annual IEEE International Computer Software and Applications (COMPSAC 2008)*, pages 108–111. IEEE, 2008.

[RKT+07] D. Roman, J. Kopeck, I. Toma, D. Fensel, and B. Sapkota. The place of policy in semantically enabled service-oriented architectures. 2007.

[RMW+10] A. Rajasekar, R. Moore, M. Wan, W. Schroeder, and A. Hasan. Applying rules as policies for large-scale data sharing. In *Proceedings International Conference on Intelligent Systems, Modelling and Simulation (ISMS 2010)*, pages 322–327. IEEE, 2010.

[Rog08] S. Rogers. Evolving SOA with IBM websphere. *IBM IDC White Paper, May*, 2008.

[RR10] J. Rossebø and R. Runde. Specifying service composition using UML 2. x and composition policies. *Model Driven Engineering Languages and Systems*, pages 520–536, 2010.

[RWW] L. Reitman, J. Ward, and J. Wilber. Service oriented architecture (SOA) and specialized messaging patterns.

[SA07] P. Supadulchai and F.A. Aagesen. Policy-based adaptable service systems architecture. 2007.

[SB08] H. Samset and R. Bræk. Dynamic service discovery using active lookup and registration. In *Services-Part I, 2008. IEEE Congress on*, pages 545–552. IEEE, 2008.

[Sch09] R. Schmelzer. Cloud governance: Something old, something new, something borrowed. *ZapThink Gate, http://www.imsglobal.org/soa/soawpv1p0/imsSOAWhitePaper_v1p0.pdf, July*, 2009.

[SEI07] SEI: Integration of software-intensive systems (ISIS) initiative: Addressing system-of-systems interoperability. *http://www.sei.cmu.edu/interoperability/index.cfm*, 2007.

[Ser11] Servicemesh's agility platform brings enterprise-proven cloud governance and lifecycle management to the next level. *ServiceMesh, http://www.marketwire.com/press-release/servicemeshs-agility-platform-brings-enterprise-proven-cloud-governance-lifecycle-management-1512271.htm, May*, 2011.

[SL02] M. Sloman and E. Lupu. Security and management policy specification. *Network, IEEE*, 16(2):10–19, 2002.

[SL05] B. Simmons and H. Lutfiyya. Policies, grids and autonomic computing. In *Proceedings Workshop on Design and Evolution of Autonomic Application Software (DEAS 2005)*, pages 1–5. ACM, 2005.

[Slo94] M. Sloman. Policy driven management for distributed systems. *Journal of network and Systems Management*, 2(4):333–360, 1994.

[SLZ09] M. Sun, B. Li, and P. Zhang. Monitoring BPEL-based web service composition using AOP. In *Proceedings 8th IEEE/ACIS International Conference on Computer and Information Science (ACIS-ICIS 2009)*, pages 1172–1177. IEEE, 2009.

[SPL+03] L. Sacks, O. Prnjat, I. Liabotis, T. Olukemi, A. Ching, M. Fisher, P. Mckee, N. Georgalas, and H. Yoshii. Active robust resource management in cluster computing using policies. *Journal of Network and Systems Management*, 11(3):329–350, 2003.

[SPM06] M. Szomszor, T.R. Payne, and L. Moreau. Automated syntactic medation forweb service integration. 2006.

[SPS96] A.R. Silva, J. Pereira, and P. Sousa. A framework for heterogeneous concurrency control policies in distributed applications. In *Proceedings 8th International Workshop on Software Specification and Design (IWSSD 1996)*, page 105. IEEE Computer Society, 1996.

[SR] P. Steenekamp and J. Roos. Implementation of service management policies: applying intelligent agent technology. In *Proceedings IEEE Network Operations and Management Symposium (NOMS 1996)*, volume 2, pages 402–412. IEEE.

[SRC05] C.S. Shankar, A. Ranganathan, and R. Campbell. An ECA-P policy-based framework for managing ubiquitous computing environments. 2005.

[SRS+01] Y. Snir, Y. Ramberg, J. Strassner, R. Cohen, and B. Moore. Policy QoS information model. *draft-ietf-policy-qos-infomodel-04.txt*, 2001.

[SS08] F. Seehusen and K. Stolen. A transformational approach to facilitate monitoring of high-level policies. In *Proceedings IEEE Workshop on Policies for Distributed Systems and Networks (POLICY 2008)*, pages 70–73. IEEE, 2008.

[SS09] V. Stantchev and C. Schröpfer. Negotiating and enforcing QoS and SLAs in grid and cloud computing. *Advances in Grid and Pervasive Computing*, pages 25–35, 2009.

[SSE09] C. Schneider, F. Stumpf, and C. Eckert. Enhancing control of service compositions in service-oriented architectures. In *Proceedings International Conference on Availability, Reliability and Security (ARES 2009)*, pages 953–959. IEEE, 2009.

[SSLTN05] N. Sheridan-Smith, J. Leaney, O. Tim, and M.H. Neill. A policy-driven autonomous system for evolutive and adaptive management of complex services and networks. 2005.

[SSOLH06] N. Sheridan-Smith, T. O'Neill, J. Leaney, and M. Hunter. A policy-based service definition language for service management. In *Proceedings 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006)*, pages 282–293. IEEE, 2006.

[ST05] M. Salehie and L. Tahvildari. A policy-based decision making approach for orchestrating autonomic elements. 2005.

[SZK05] G. Spanoudakis, A. Zisman, and A. Kozlenkov. A service discovery framework for service centric systems. 2005.

[TB92] V. Tiruveedhula and JS Bedi. On the evaluation of a hybrid hypercube architecture employing data movement policies. In *International Conference Technology Enabling Tomorrow: Computers, Communications and Automation towards the 21st Century. (TENCON 1992)*, pages 933–937. IEEE, 1992.

[TPT09] V.X. Tran, S. Puntheeranurak, and H. Tsuji. A new service matching definition and algorithm with SAWSDL. In *Proceedings 3rd IEEE International Conference on Digital Ecosystems and Technologies (DEST 2009)*, pages 371–376. IEEE, 2009.

[TvLN08] J. Tammo van Lessen and F.L. Nitzsche. Formalising message exchange patterns using BPEL light. In *Proceedings International Conference on Services Computing (SCC 2008)*, pages 353–360. IEEE, 2008.

[TWP+05] V. Talwar, Q. Wu, C. Pu, W. Yan, G. Jung, and D. Milojicic. Comparison of approaches to service deployment. 2005.

[TX06] J.F. Tang and X.L. Xu. An adaptive model of service composition based on policy driven and multi-agent negotiation. In *Proceedings International Conference on Machine Learning and Cybernetics (ICMLC 2006)*, pages 113–118. IEEE, 2006.

[TYL05] P.S. Tan, Z.L. Yin, and SG Lee. Realising service-oriented architecture through context-based dynamic service invocation. In *Proceedings 3rd IEEE International Conference on Industrial Informatics (INDIN 2005)*, pages 300–305. IEEE, 2005.

[UBJ+04] A. Uszok, J.M. Bradshaw, M. Johnson, R. Jeffers, A. Tate, J. Dalton, and S. Aitken. KAoS policy management for semantic web services. *IEEE Intelligent Systems*, pages 32–41, 2004.

[UP08] K. Umapathy and S. Purao. Representing and accessing design knowledge for service integration. In *Proceedings International Conference on Services Computing (SCC 2008)*, pages 67–74. IEEE, 2008.

[VB97] C. Vermeulen and B. Bauwens. Intelligent agents for on-line commerce: a crying need for service standardization. In *Proceedings 4th International Workshop on Community Networking (CN 1997)*, pages 87–92. IEEE, 1997.

[VC02] M. Visconti and C.R. Cook. An overview of industrial software documentation practice. 2002.

[VdM96] J. Van der Meer. Service management, provision and design. In *Proceedings IEEE Intelligent Network Workshop (IN 1996)*. IEEE, 1996.

[VGW+07] J.A. Vayghan, S.M. Garfinkle, C. Walenta, D.C. Healy, and Z. Valentin. The internal information transformation of IBM. *IBM Systems Journal*, 46(4):669–683, 2007.

[Wal99] M.G. Wales. WIDL: interface definition for the web. *Internet Computing, IEEE*, 3(1):55–59, 1999.

[WC07] S. Wang and M.A.M. Capretz. A policy driven approach for service-oriented business rule management. In *Proceedings 5th International Conference on Industrial Informatics (INDIN 2007)*, volume 2, pages 713–718. IEEE, 2007.

[WCH08] Y. Wei, X. Chen, and Y. Han. Service-oriented aggregation of distributed and heterogeneous information resources. In *Proceedings 7th International Conference on Grid and Cooperative Computing (GCC 2008)*, pages 214–220. IEEE, 2008.

[WHBLC06] D.J. Weitzner, J. Hendler, T. Burners-Lee, and D. Connolly. Creating a policy-aware web: Discretionary, rule-based access. *Web and Information Security*, pages 1–31, 2006.

[WJY+08] M. Wu, C. Jin, C. Yu, H. Yan, and J. Ying. QoS and situation aware ontology framework for dynamic web services composition. In *Proceedings 12th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2008)*, pages 488–493. IEEE, 2008.

[Woo06] B. Woolf. Introduction to SOA governance. *IBM Developer Works*, 2006.

[WRK05] S. Willmott, H. Ronsdorf, and K.H. Krempels. Publish and search versus registries for semantic web service discovery. In *Proceedings International Conference on Web Intelligence (WIC 2005)*, pages 491–494. IEEE, 2005.

[WS06] X. Wang and H. Schulzrinne. Pricing network resources for adaptive applications. *IEEE/ACM Transactions on Networking*, 14(3):506–519, 2006.

[WT98] D. Watkins and D. Thompson. Adding semantics to interface definition languages. In *Proceedings Australian Software Engineering Conference (ASWEC 1998)*, pages 66–78. IEEE, 1998.

[Wu07] R. Wu. Concept of component aggregation in information, service and business. In *Proceedings International Conference on Service Systems and Service Management (ICSSSM 2007)*, pages 1–4. IEEE, 2007.

[WWS08] M. Wiley, A. Wu, and J. Su. WSDL-D: A flexible web service invocation mechanism for large datasets. In *Proceedings 10th IEEE Conference on E-Commerce Technology (CEC 2008) and the 5th IEEE Conference on Enterprise Computing, E-Commerce and E-Services (EEE 2008)*, pages 157–164. IEEE, 2008.

[WZD07] R. Weinreich, T. Ziebermayr, and D. Draheim. A versioning model for enterprise services. 2007.

[XFZ08] P.C. Xiong, Y.S. Fan, and M.C. Zhou. QoS-aware web service configuration. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 38(4):888–895, 2008.

[XJ06] Q. Xiaoqiang and W. Jun. A decentralized services choreography approach for business collaboration. 2006.

[Yao03] W. Yao. Fidelis: A policy-driven trust management framework. *Trust Management*, pages 1071–1071, 2003.

[YBG08] U. Yildiz, R. Badonnel, and C. Godart. On service orchestration in mobile computing environments. In *Proceedings International Conference on Services Computing (SCC 2008)*, volume 2, pages 545–548. IEEE, 2008.

[YCA+09] H.F.E. Yamany, M.A.M. Capretz, D.S. Allison, D. Garcia, and M.B.F. Toledo. QoSS policies within SOA. In *Proceedings International Joint Conference on Web Intelligence and Intelligent Agent Technology (IEEE/WIC/ACM 2009)*, volume 01, pages 426–429. IEEE Computer Society, 2009.

[YCD+09] J. Yin, H. Chen, S. Deng, Z. Wu, and C. Pu. A dependable ESB framework for service integration. *Internet Computing, IEEE*, 13(2):26–34, 2009.

[YCMZ08] H. Yang, J. Chen, X. Meng, and Y. Zhang. A dynamic agent-based web service invocation infrastructure. In *Proceedings 1st International Conference on Advances in Computer-Human Interaction (ACHI 2008)*, pages 206–211. IEEE, 2008.

[YCSG06] J.J. Yu, R. Chen, H. Su, and S. Guo. Web services publishing and discovery on peer-to-peer overlay. In *Proceedings Asia-Pacific Conference on Services Computing (APSCC 2006)*, pages 329–334. IEEE, 2006.

[YHY+09] F. You, Q. Hu, Y. Yao, G. Xu, and M. Fang. Study on web service matching and composition based on ontology. In *Proceedings WRI World Congress on Computer Science and Information Engineering (CSIE 2009)*, volume 4, pages 542–546. IEEE, 2009.

[YL08] S.S. Yau and J. Liu. Service functionality indexing and matching for service-based systems. In *Proceedings International Conference on Services Computing (SCC 2008)*, pages 461–468. IEEE, 2008.

[YLT+06] X. Yang, T. Lehman, C. Tracy, J. Sobieski, S. Gong, P. Torab, and B. Jabbari. Policy-based resource management and service provisioning in GMPLS networks. In *Proceedings 25th IEEE International Conference on Computer Communications (INFOCOM 2006)*, pages 1–12. IEEE, 2006.

[YOLH05] K. Yang, S. Ou, A. Liotta, and I. Henning. Composition of context-aware services using policies and models. In *Proceedings IEEE Global Telecommunications Conference (GLOBECOM 2005)*, volume 2. IEEE, 2005.

[Yu04] J.J. Yu. Dynamic web service invocation based on UDDI. In *Proceedings IEEE International Conference on E-Commerce Technology for Dynamic E-Business (CEC 2004)*, pages 154–157. IEEE, 2004.

[ZC05] G. Zhao and D. Chadwick. Trust infrastructure for policy based messaging in open environments. 2005.

[ZCC+02] L.J. Zhang, H. Chang, T. Chao, J.Y. Chung, Z. Tian, J.M. Xu, Y.N. Zuo, S. Yang, and Q.Y. Ao. A manageable web services hub framework and enabling technologies for e-sourcing. In *Proceedings IEEE International Conference on Systems, Man and Cybernetics (SMC 2002)*, volume 6, pages 6–14. IEEE, 2002.

[ZCG05] D. Zhang, C.Y. Chin, and M. Gurusamy. Supporting context-aware mobile service adaptation with scalable context discovery platform. In *Proceedings 61st Vehicular Technology Conference (VTC 2005-Spring)*, volume 5, pages 2859–2863. IEEE, 2005.

[ZHH+08] Q. Zhao, G. Huang, J. Huang, X. Liu, H. Mei, Y. Li, and Y. Chen. An on-the-fly approach to web-based service composition. In *Proceedings IEEE Congress on Services Part II (SERVICES-2 2008)*, pages 208–209. IEEE, 2008.

[ZJNZ09] W. Zhen, L. Jianmin, Z. Nanrun, and L. Zhenrong. Semantic web service selection based on context and QoS. In *Proceedings International Conference on Web Information Systems and Mining (WISM 2009)*, pages 332–335. IEEE, 2009.

[ZLK+07] Y.C. Zhou, X.P. Liu, E. Kahan, X.N. Wang, L. Xue, and K.X. Zhou. Context aware service policy orchestration. In *Proceedings International Conference on Web Services (ICWS 2007)*, pages 936–943. IEEE, 2007.

[ZMCW08] F. Zulkernine, P. Martin, C. Craddock, and K. Wilson. A policy-based middleware for web services sla negotiation. In *Proceedings International Conference on Web Services (ICWS 2009)*, pages 1043–1050. IEEE, 2008.

[ZPPN07] J. Zhou, D. Pakkala, J. Perala, and E. Niemela. Dependency-aware service oriented architecture and service composition. |, pages 1146–1149, 2007.

[ZSC06] B. Zhang, Y. Shi, and Y. Chen. A policy-based adaptation method for service composition. In *Proceedings 1st International Symposium on Pervasive Computing and Applications (ICPCA 2006)*, pages 619–624. IEEE, 2006.

[ZSCD08] Z. Zhang, W. Sun, W. Chen, and J. Dong. A lifetime aware approach to service selection in mobile ad hoc networks. In *Proceedings 9th International Conference for Young Computer Scientists (ICYCS 2008)*, pages 539–544. IEEE, 2008.

[ZSX10] B. Zhang, Y. Shi, and X. Xiao. A policy-driven service composition method for adaptation in pervasive computing environment. *The Computer Journal*, 53(2):152, 2010.

[ZTW+06] X. Zhou, W.T. Tsai, X. Wei, Y. Chen, and B. Xiao. PI4SOA: A policy infrastructure for verification and control of service collaboration. 2006.

[ZZH06] Y. Zhang, S.S. Zhang, and S.Q. Han. Context-aware service selection engine for ubiquitous computing application. In *Proceedings The 6th World Congress on Intelligent Control and Automation (WCICA 2006)*, volume 1, pages 4269–4273. IEEE, 2006.

Appendix A – Abbreviations

AWS – Amazon Web Services

BPEL – Business Process Execution Language

EA – Enterprise Architecture

EAI – Enterprise Application Integration

ERP – Enterprise Resource Planning

ESB – Enterprise Service Bus

IAAS – Infrastructure As A Service

IDE – Integrated Development Environment

IT – Information Technology

JSON – JavaScript Object Notation

KPI – Key Performance Indicator

MTOM – Message transmission Optimization Mechanism

OASIS – Organization for the Advancement of Structured Information Standards

OSIMM – Open Group Service Integration Maturity Model

PAAS – Platform As A Service

QoS – Quality of Service

REST – Representational State Transfer

ROI – Return on Investment

SAAS – Software As A Service

SAML – Security Assertion Mark-up Language

SCA – Service Component Architecture

SLA – Service Level Agreement

SOA – Service Oriented Architecture

TOGAF – The Open Group Architecture Framework

UDDI – Universal Description, Discovery, and Integration

UML – Unified Modeling Language

URL – Uniform Resource Locator

WS-BPEL – Web Service Business Process Execution Language

WS-CDL – Web Services Choreography Description Language

WSDL – Web Services Description Language

WS-I – Web Services Interoperability Organization

XML – Extensible Mark-up Language