Contextual Governance for Service Oriented Architecture Composition

Thesis by P. de Leusse

In partial fulfilment of the requirements for the degree of doctor of philosophy

Newcastle University

Newcastle upon Tyne, UK

2010

Acknowledgements

I am fortunate to have worked in the School of Computing Science at Newcastle University in close collaboration with the Security Architecture Group at BT Design and Innovate between 2006 and 2009. Many people in the School and at BT have helped me during this period and I have received valuable feedback on my research work. I do not have the space to acknowledge them all individually.

Several people deserve special mention with respect to the work in this thesis. First, I thank my initial supervisors, Dr Panos Perriorelis and Dr Theo Dimitrakos, for their supports, guidance and insights. It has been a privilege to work closely with Panos and Theo, and with Professor Paul Watson and Mr David Brossard. Paul has had a significant influence on the work I did at the school. David and I have worked together closely on this topic during this period, which has been a rewarding experience. I must also thank Dr Graham Morgan for useful discussions about this thesis. He is not responsible for any misconceptions that may persist. In addition, as reviewer, he has suggested a number of important improvements to this thesis. Dr Nektarios Georgalas of BT Design and Innovate was equally helpful.

Finally, and most importantly, I thank Anna. I owe her an incalculable debt. This has been a long journey for me. I know it must have seemed longer to her.

Abstract

Currently, business requirements for rapid operational efficiency, customer responsiveness as well as rapid adaptability are driving the need for ever increasing communication and integration capabilities of the software assets.

Functional decomposition into re-usable software entities, loose coupling, and distribution of resources are all perceived benefits of the investment on Service Oriented Architecture (SOA). This malleability can also bring about the risk of a more difficult oversight. The same service is ideally used in different applications and contexts. This situation forces a supporting infrastructure to allow and manage the adaptability to these different contexts of use.

In this thesis, the author proposes to govern such variations in a cost efficient way by composing the core business function offered by a service with other services implementing infrastructure capabilities that fulfil varying non-functional requirements.

However, as the number of services increases and their use in different contexts proliferates, it becomes necessary to automate policy enforcement and compliance monitoring. Furthermore, the composition of services into different business applications over a common infrastructure intensifies the need for end-to-end monitoring and analysis in order to assess the business performance impact. Managing the full life-cycle of service definition, deployment, exposure and operation requires management processes that take into account their composition with the infrastructure capabilities that take of non-functional requirements. In addition, policies may change during the life-time of a service. Policy updates may be the result of various reasons including business optimisation, of reaction to new business opportunities, of risk / threat mitigation, of operational emergencies, etc. It becomes therefore clear that a well-designed governance architecture is a prerequisite to implementing a SOA capable of dealing with a complex and dynamic environment.

Content

List of figures

List of tables

1. Introduction

The way enterprises conduct business today is changing greatly. The enterprise has become more pervasive with a mobile workforce, outsourced data centres, different engagements with customers and distributed sites [1]. In addition, companies seeking to optimise their processes across their supply chains are implementing integration strategies that include their customers and suppliers rather than looking inward. This increases the need for governing end-to-end transactions between business partners and the customer (B2B2C and B2B2G) [2].

As pervasive organisations connect their heterogeneous environments and systems, cross- and intra-enterprise compliance becomes more critical. The legal and regulatory frameworks become more complex and less forgiving. Companies have to comply with their own directives and regulations as well as comply with different legislations and regulations depending on the region of operation and the client or partner organisations' rules and legal constraints. IT use in the corporate environment, and in particular the governance of the IT infrastructure that enables business services, will need to provide means to measure and control compliance.

Globalisation and agility of integration require more systems along with more partners and more constraints and produce more complex environments where decision making processes are equally increasingly complex and crucial for this connected organisation. Change in a single process has the potential to impact more than one partner and disrupt a wider range of business processes.

It is important for any enterprise to understand how its business has performed at any given time in the past, present, and in the future. However, single partners no longer have a full visibility of all processes and their consequences. It becomes much harder for a single enterprise to therefore govern its collaboration with other enterprises in a safe and controlled way, to understand the use of its information and resources across the value chain, and to identify and assess the impact of violations of policies or agreements. There is a need for well-orchestrated, end-to-end operations management that provides controlled visibility, governance of network and IT state, timely assessment of the impact of security policy violations and the availability of resources. Hence, there is an increasing interest in Service Oriented Infrastructure (SOI) dashboards [3] showing real-time state of the corporate infrastructure including the B2B integration points.

Finally, another consequence of these changes in the organisational environment is the emergence of the notion of Virtual Organisations (VO). These are defined in [4] as "temporary or permanent coalitions of individuals, groups, organisational units or entire organisations that pool resources, capabilities and information to achieve common objectives". According to this definition, VOs can provide services and thus participate as a single entity in the formation of further VOs, hence creating recursive structures with multiple layers of "virtual" value-adding service providers. The required scalability, responsiveness, and adaptability, requires a cost effective resource distribution management solution for dynamic VO environments.

Effective solutions addressing these challenges require interdisciplinary approaches integrating tools from law, economics and business management in addition to distributed or "Cloud" computing. However, in this latter category, there are three

main aspects, as illustrated in Figure 1 to achieving effective governance in the context of high-value B2B interactions:

1. **Resource visibility** - brings the best fit for purpose visibility into the IT infrastructure used and its state. This aims at making sure that not only it is possible to find the resource but also that its purpose and constraints are well understood. With complex systems comprised of many resources (e.g. Web service, policy) there is a strong requirement to increase the visibility of each resource. Indeed a same functionality could be provided by different services and advertised in different places. In addition, one of the strengths of SOA being reuse and composition, there is an obligation to know how resources communicate and are wired together. The relevant management of dependencies amongst resources is indeed a crucial element of the visibility. Furthermore, with a single resource involved in several collaborations or discussions it is necessary to keep track of how this same functionality is proposed (i.e. its attached constraints). Increasing the visibility includes advertising its functionality as well as Non Functional Properties (NFP) correctly and its issuer or provider.

2. **Policy administration and management** - administers policies coming from different sources of authority and that may apply to different, potentially interrelated, contexts and business collaborations. In an organisation, the different levels of hierarchy manage their resources according to their responsibilities. As such managers set up rules on how certain requests from clients are going to be dealt with when the directors will set up the roles and limits of the manager's authority. As IT services attempt more and more to support business functions, the same types of policies should be applied to them, allowing for different levels of authority that apply in particular or more general cases. The same apply for the different areas of expertise where an account manager will dictate the pricing policy for a client and the lawyer will know how to write legal contracts. IT services are dependants of the IT specialists at different levels (e.g. deployment, security) as well as non-IT specialists. This aspect also deals with managing the selection and integration of the best policy decision and policy enforcement mechanisms to support the optimal use of IT resources and services in a given context and in compliance with corporate agreements. As introduced in the three points above, an SOA will suffer from having many services that may be available in different contexts and at different stages of their life-cycle. The management of the SOA is made through the use of policies and as such it is crucial to be able to manage how these are going to be used and enforced. In addition, such policy administration and management should allow detecting potential conflicts within the imbrications of services and their policies.

3. **Service provisioning** - deals with the processes that allow organisations to effectively manage the exposure conditions of their services. This aspect of governance aims at both allowing an efficient contextualisation of the service interface provided to potential partners as well as supplying this contextualisation with the processes necessary to render its governance flexible.

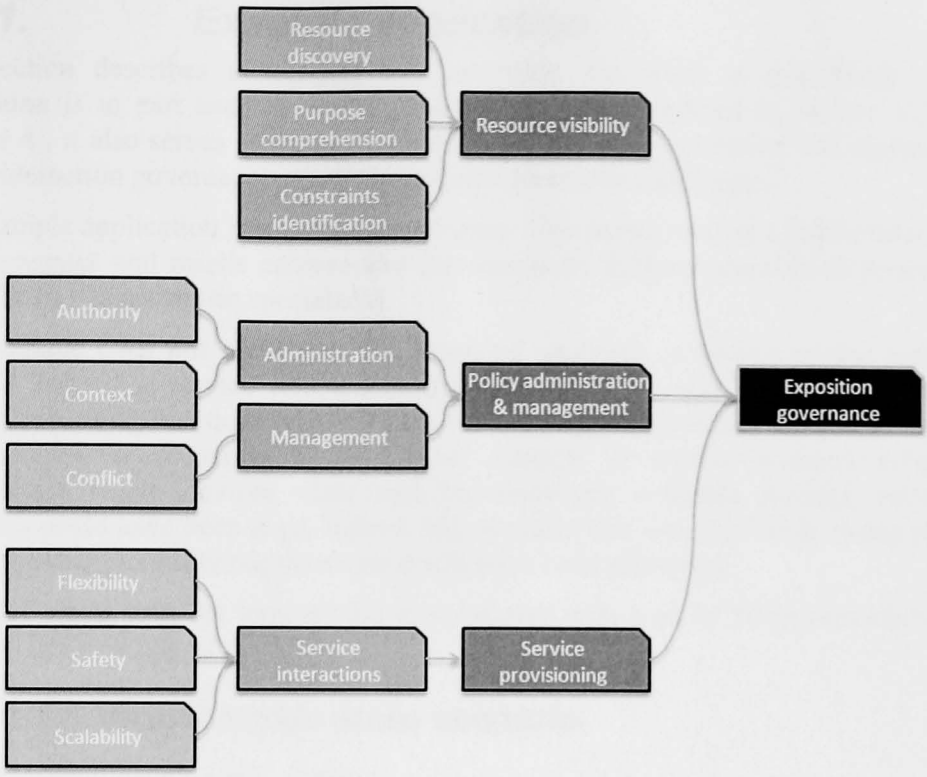


Figure 1. Interaction governance

In this thesis, the author aims to address the latter aspect of governance. The work presented in this thesis concerns the flexible management of services interactions where the properties of these exchanges are not a priori known. A related concern is that services should not suffer security and performance disadvantages as a result from exposing in this flexible manner. Thus governance can extend to supporting the safety of the interaction and scalability of the infrastructure supporting the exchanges.

The main contributions of this work are the design and evaluation of a flexible middleware architecture supporting the governance of B2B interactions. The author designed this architecture to support two domains of action – design and run times – that address requirements of the interactions management. The author shows how the middleware architecture can be implemented with a set of middleware services. The segregation of concerns proposed presents different advantages such as dynamic composition of the interaction contextualisation and systematic auditing. Additionally, this allows presenting an abstraction layer that is familiar from the enterprise context while providing regulated interactions in B2B context. For example, governed interactions can be used to secure exchanges in a more flexible manner than traditional approaches (c.f. section 2).

In Section 1.1 of this chapter the author describes an example application that serves to motivate the work presented in this thesis and provide requirements for that work. Section 1.2 summarises the scope of the work. Finally, section 1.3 concludes the chapter with the assertion of thesis contributions and an overview of the remainder of the thesis.

1.1. Example application

This section describes a scenario that motivates the work in this thesis. This application is in part used to derive research challenges defined in section 1.2. In Chapter 4 , it also serves as proof-of-concepts for the implementation and discussion of the interaction governance middleware that address the challenges.

The example application is a virtual music store. This music store is a public interface that aggregates and resells content and services from different specialised providers (e.g. Jazz or Classic music specialists).

This example does not mean that the proposed approach is limited to this type of scenario. Indeed, a service oriented enterprise architecture with an environment of services providing functions (e.g. payroll, fleet management) would have also been a good motivation scenario. In the same manner, a service oriented e-health environment where devices, data and functions are available through network interfaces could have been used. Indeed, any scenario that would provide strong reuse and exposition flexibility requirements could have been presented.

In the following text, the term profile is associated with a set of NFPs provided as a managed composition of services.

1.1.1. Virtual music store scenario

1.1.1.1 Description

This section describes the virtual music store as an example of a Virtual Organisation (VO). According to [5], VOs are frequently restructured, sustained to capture the value of a market opportunity and dissolved again to give way for the creation of the next VO from within the network of independent partners. This represents a need for adaptability that current systems, such as the GOLD middleware [6] or the current B2B gateway [7], attempt to address by providing one type of security profile.

The aggregated services are virtual music stores serving specialised markets or communities of interest. The basic service providers include copyright owners of musical recordings or their representatives who make these recordings available online and syndicated blogs or review sites. The music stores reach agreement with music providers enabling them to act as resellers of bundles of recordings from their catalogues. The virtual store is a VO consisting of the music store operator as well as content providers and it runs on top of an infrastructure provided by an infrastructure provider.

This scenario is based on the increasing trend of Long Tail retailing. The Long Tail concept describes the niche strategy of selling a large number of unique items in relatively small quantities, usually in addition to selling fewer popular items in large quantities. The concept was popularised by Chris Anderson in [8], in which he mentioned Amazon.com and Netflix as examples of businesses applying this strategy. Figure 2 illustrates the Long Tail concept.

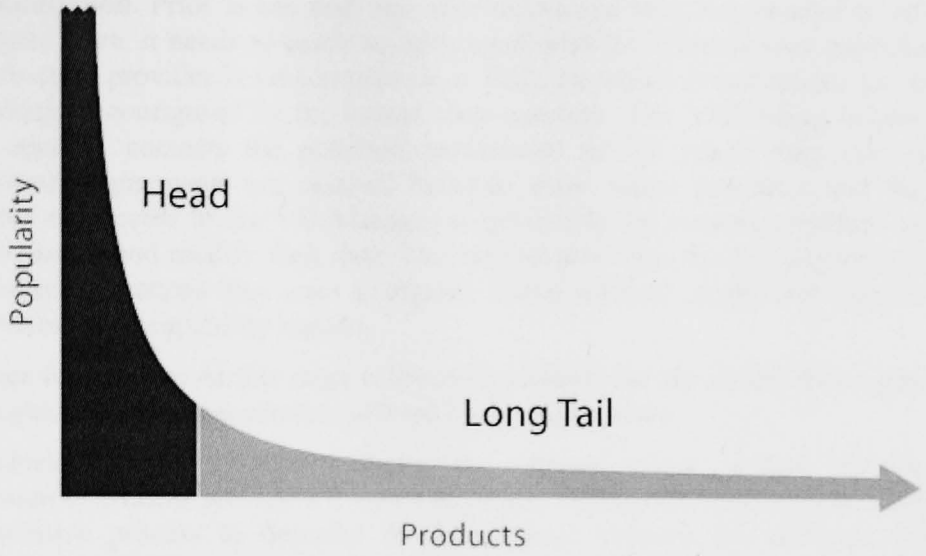


Figure 2. To the right is the long tail; to the left are the few that dominate [9].

The end customer of the virtual music store will be a member of the public. What they will see is a normal website where they will be able to search for and buy tracks and read reviews and blogs. This could be presented to them in much the same way as AbeBooks does, i.e. a search page and then each returned item is linked in from an independent seller; or stores could hide the aggregated nature of the service.

1.1.1.2 Stakeholders

In the music store, the main categories of partner are:

• **Infrastructure provider**: This role involves providing the VMS with a Virtual Hosting Environment (VHE), the B2B gateway. The purpose of the infrastructure is to hide the technical complexity of the middleware involved to the different participants in the virtual music store.

• **Music content provider**: This is a specialist content provider (e.g. record labels or other copyright owners).

• **Virtual music store operator**: The broker of music. It is assumed that the store operator will be the VO Initiator. As such the operator is responsible for instigating the opening federation process.

• **Value adding service provider**: This is a third party entrusted with providing Value Adding Services (VAS). These services provide non-functional proprieties (e.g. security, audit, translation) and allow the content providers and music store operator to leverage on the VHE to enhance their interoperability and quality of service.

1.1.1.3 VO Lifecycle

So let us look closer into the life-cycle of the virtual music store. The music store life-cycle starts with the initial agreements and discovery of the potential partners, the VO foundation. This is followed by the negotiation between these partners and the VO initiator to reach a firm collaboration agreement, this stage is called the partners' federation. Following this, the capabilities are virtualised and made available to the partners in the newly formed VO. Finally the adaptability faculty of the virtual store infrastructure is introduced.

VO foundation. Prior to any task and once the virtual shop has decided to establish the music store, it needs to reach an agreement with the infrastructure provider. The infrastructure provider is said to provide a VHE, on which it instantiates an 'empty' VO which is configured by the virtual shop operator. The VHE being in place, the shop operator contacts the potential participants of the music shop (i.e. content providers). Agreements are reached between these music providers and the shop operator and access to the VO Manager is granted to the content providers to setup their accounts and modify their data. The content providers can consequently publish the business functions they want to expose. These selected capabilities are published as services into a capability registry.

Partner federation. At this stage it becomes possible for the virtual shop operator to put in place the different services offered by the music store.

To achieve this, as introduced above, the operator creates a new VO for each federation of content providers it wants to create. Additionally, the operator defines a collaborative process to describe the interactions between the different business functions potentially present in the federation. Once this structure is in place, the operator can search the capability registry for the specific business functions it wants to aggregate and using the VO Manager sends a participation request to the relevant providers.

The providers contacted can inspect the process description and interaction description already provided by the store operator to take a decision upon participating in this federation. Having accepted the invitation, the content providers associate to the VO the VAS profile they want to apply to this federation. The VAS profiles are defined for each capability by its provider.

These profiles include infrastructure services used to secure and monitor the services. They are created and managed in much the same way as the federation between the operator and the music providers but include the VAS providers. The services are typically comprised of policy enforcement, authentication, authorisation and other added value services such as billing or auditing. In addition, the profiles are composed of policy templates that define the policies to be applied to each of the selected infrastructure services in the profile.

Following this, the operator can review and select the best matches in the positive answers it has received. With all the targeted business functions fulfilled, the virtual shop operator can continue the VO creation process and sends a creation order.

The VO management tool subsequently interacts with each partner's gateway. To allow the different identity providers to recognise each other's authority, it sends the relevant list of business cards associated with each business partner (role). In addition, the VO management tool sends the policies related to the implementation of the collaboration management for each business function. These policies come on the top of the security profiles setup and made available by the service providers.

Brokered services, such as the jazz music store aggregating the different content providers' services that offer jazz music, can be created along with their federation data following this method. The jazz music store is illustrated in Figure 3.

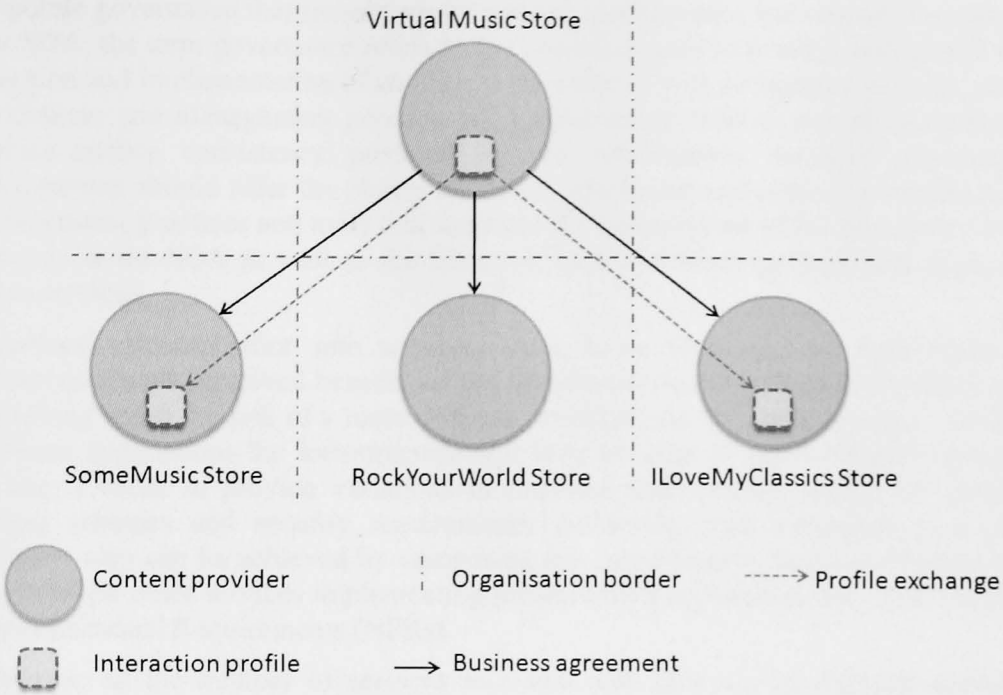


Figure 3. The Jazz Music Store VO

Capability virtualization. With the federation in place, it is possible for the participants to finalise the configuration of the instances of the services they selected and prepared for this specific VO. Before undertaking this, the gateway management interface allows the participant to inspect the configuration of its infrastructure. At this stage, the configuration of the infrastructure will have evolved as the services are exposed and activated. Additionally, the selected Federated Identity Provider (FIP) has built trust with the FIPs of the other partners in the VO. Furthermore, the baseline policies that restrict who can issue access policies about which resources have been activated.

Finally, the VAS profile that will be applied by the Partner for the business functions it performs is stored in a specific registry. To keep track of the configuration, the settings are associated with a unique collaboration ID. Upon configuration of the infrastructure, provisioning of policy templates and establishment of trust between the different VOs it becomes possible for the capability instances exposed to be invoked within the context of the virtual music store.

Adaptability. A music provider might want to participate in several such federations to increase its visibility. But different partners in various VOs will have distinct security needs and settings. By adjusting the VAS profile used in each federation to its specific needs, the content provider can more promptly offer its services.

1.2. Scope of work

The ever increasing amount of IT services along with all the potential states and types of configurations necessitate the development of adequate methods and tools for services governance. In [10], the concept of SOA governance is derived from corporate and IT governances. Corporate governance is referred to as the set of processes, customs, policies, laws and institutions affecting the way in which a corporation is directed, administered or controlled. IT governance is a subset of

corporate governance that focuses on the control, performance and risk of IT systems. For SOA, the term governance refers to the processes used to oversee and control the adoption and implementation of an SOA in accordance with recognised policies, audit procedures and management policies. SOA governance aims at providing optimum service quality, consistency, predictability and performance. An SOA governance environment should offer the ability to define, administer and enforce a combination of processes, practices and tools that facilitate the management of the life-cycle of the services in the SOA as well as the life-cycle of the different policies that apply on these services.

Functional decomposition into services, reuse, loose coupling, and distribution of resources are all perceived benefits of the investment on SOA. This malleability can also bring about the risk of a more difficult oversight. As the same service is used in different applications the infrastructure will have to adapt to these different contexts of use in order to provide variations in required functionality, quality of service, billing schemes and security requirements. Achieving such variations in a cost efficient way can be achieved by composing the core business function offered by a service with other services implementing infrastructure capabilities that fulfil varying Non-Functional Requirements (NFRs).

However, as the number of services increases and their use in different contexts proliferates, it becomes necessary to automate policy enforcement and compliance monitoring. Furthermore, the composition of services into different business applications over a common infrastructure intensifies the need for end-to-end monitoring and analysis to assess the business performance impact. Managing the full life-cycle of service definition, deployment, exposure and operation requires management processes that take into account their composition with the infrastructure capabilities that take charge of the NFRs. Finally, policies may change during the life-time of a service. Policy updates may be the result of various reasons including business optimisation, of reaction to new business opportunities, of risk / threat mitigation, of operational emergencies, etc. It becomes therefore clear that a well designed governance model is a prerequisite to successfully deploying services in a dynamic environment. More details on the objectives of such a SOA governance framework are given in the following paragraphs:

- **Resource contextualisation**: Permits resources to be efficiently configured for and managed at an end-to-end level is one of the main objectives of SOA governance.
- **Resource adaptation**: Enables diagnosis and remediation in an as automated as possible fashion. SOA systems can potentially become very complex, with many different policies and services. This not only allows to adapt resources to specific transactions in function of an organisation's rules but to manage this adaptation in a more configurable, reliable and secured way.
- **Contextualisation safety**: Manages the contextualisation so that contextual information cannot infringe into another context. Safety mechanisms should also ensure that the adaptation does not result in a loss or inadequate sharing of data.

1.3. Thesis contributions and overview of thesis

Section 1.2 identified three domains – resource contextualisation and adaptation as well as safety of the contextualisation process – that provide broad coverage of B2B interactions and a challenging set of requirements to support the governance of exchanges. In this context, the author's thesis contributes:

1. A set of middleware services that provide an efficient support for flexible resource contextualisation.
2. This enhanced flexible contextualisation can be governed in such a manner that it preserves the safety of the exchanges.

This statement will be justified by the design, implementation and deployment of novel middleware.

The remainder of the thesis substantiates the claim of novelty with respect to these systems. Before providing an overview of the thesis, the author elaborates on novel contributions in these three areas.

1.3.1. Overview of thesis

The work presented in this thesis addresses the research challenges identified in Section 1.2 by providing middleware to support the governance of B2B interactions. To this effect, the author proposes an SOA based architecture where the infrastructure capabilities providing support for NFPs that are not known a priori potentially come from third party providers. The approach is to provide a coherent definition of the interaction needs (i.e. in terms of NFPs) and compute these needs into an implementable and manageable aggregation of services. The architecture proposed also identifies the supporting – i.e. core – infrastructure necessary to provide the governance. The inherent flexibility of the architecture ensures that its implementation can be adapted to different application-specific requirements. Given this architecture, the author develops a prototype implementation. In addition, a proof-of-concepts application demonstrates the utility of the architecture implementation in the domain of SOA security. The thesis has the following structure.

Chapter 2 provides an overview of related works. The chapter concentrates on work on middleware that is fundamental to the architecture presented. The author also surveys work on contract-mediated interaction, policy driven middleware and on middleware support for flexible interactions management.

Chapter 3 describes the anatomy of the SOA based architecture for service exposure governance that, along with its implementation and proof of concepts utilisation, are the novel contributions. The services are based on an interceptor-mediated view of interactions that: (i) allows end users (e.g. web service manager, other governance system) to define their requirements in term of NFPs, (ii) supports the design of adaptable services, and (iii) provides a set of processes to checks the safety of the adaptation. This chapter develops work first reported in [11], [12] and [13].

Chapter 4 describes the implementation of the architecture described in Chapter 3. A proof-of-concept application based on the example in Section 1.1.1 demonstrates the utility of the implementation. In addition, this chapter describes a qualitative evaluation of the proof-of-concept application. This is based on work first reported in [14] and [15].

Chapter 5 concludes the thesis with a summary of contributions, including an evaluation with respect to the requirements set out in this chapter, and an overview of future work.

2. Background

In this chapter, the author introduces the reader to the domain of software integration in distributed computing and reviews the most pertinent related. These are: software architecture definition in an SOA, flexible resource contextualisation in an SOA, and management of this contextualisation's safety (c.f. section 1.3).

In section 2.1, the author introduces a short history of modern software integration through middleware. This part comprises content about how SOAs came to be and the development of the Enterprise Service Bus (ESB) concept which is a SOA based software infrastructure that acts as an intermediary layer of middleware through which distributed services and information can be made available. Section 2.2 identifies the principal reference models and architectures that could influence the author's definition of a concrete architecture for governance of safe and flexible resource contextualisation in an SOA. Section 2.3 reviews the most relevant works in the domain of flexible contextualisation while part 2.4 evaluates their counterparts in the domain of contextualisation safety. Finally, a conclusion on the related work and a summary of the different research challenges is proposed in section 2.2.

Throughout this chapter, the different elements described are assessed against potential research problematic that are then summarised in section 2.2.1.

2.1. The recent history of modern integration techniques

Distributed systems have become more and more complex. This is partly due to the amount of technologies developed and the frequency in which they appear. In particular the amount of platforms implemented (e.g. .NET, Java, Axis, WebSphere) as well as the various specifications related to the different issues associated with Enterprise Application Integration (EAI) or SOA (with several standardisation bodies such as W3C, OASIS or WS-I) have rendered the middleware environment more opaque. This situation, linked with the raise of inter-application communications [16] and the possible repeated changes of partnerships in these interactions [17], have caused a situation where Return On Investment (ROI) and ease of integration can be difficult to reach.

The traditional opportunistic integration is generally achieved using conventional application-to-application or point-to-point communication [18]. But these approaches have their limits. The complexity grows exponentially with the number of applications or points and the frequency they change. Furthermore, the maintenance and integration costs increase as the application becomes more complex.

Therefore, one objective of more recent integration approaches is to reduce the complexity of integration by replacing the point-to-point ad-hoc with systematic integration through a specialised integration platform. In the next sections, this trend will be analysed and its main outcome in term of integration practice for SOA.

2.1.1. Evolution of middleware

The first wave of integration practices aimed to provide APIs and interfaces between systems. As shown on Figure 4, this was mostly achieved either using custom Remote Procedure Calls (RPC) or messaging technologies like CORBA. This generation of middleware primarily aimed to achieve point to point integration and most of the

connectors were custom built. This generally allowed heterogeneous systems to communicate and in the case of messaging technologies, allowed to store and to forward messages. On the other hand, specific interfaces had to be developed for every system involved but at the same time the lack of widely spread communication semantics meant that many different formats prospered, thus generating a low level of reusability and rendering all coding and maintenance complex.

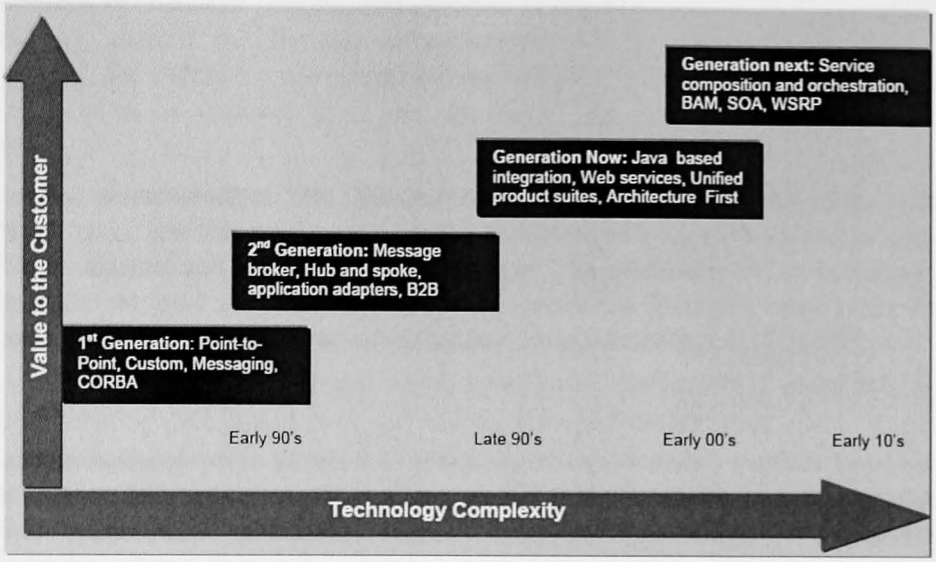


Figure 4. Figure 1 Integration history [18]

The next generation mainly aimed to improve the reusability as well as connectivity issues and introduced the spoke-hub distribution paradigm. Common integration infrastructures subsequently developed include, application servers and EAI brokers such as hub-and-spoke architectures that can potentially offer features such as message routing, transformation, business rules enforcement, transaction monitoring and auditing. This approach allowed reducing the connection complexity as with n nodes, a maximum of n - 1 routes are necessary to connect all nodes, compared to (n(n - 1))/2 nodes that would be required in a point-to-point network. In addition, this made possible to re-use and share the integration logic among multiple expositions. These advantages provided connectors that were potentially faster to put in place and easier to maintain. However, the broker or application server due to its hub-and-spoke nature can create a bottleneck effect, impacting on the performance as well as creating a single point of failure. Furthermore, although this type of infrastructure can support exchange format standards, this aspect was still being neglected during this age. Finally, this type of middleware links the connected systems together in a tightly coupled fashion, as it often intertwines the application and the integration logic.

With inter-application integration passing from a consideration to one of the main centres of interest, the current generation of middleware focuses on loosening the coupling and developing common exchange semantics. This brought up the emerging growth of the SOA paradigm and technologies such as Representational State Transfer (REST) [19] or Web Services (WS) and Message Oriented Middleware (MOM) in particular. With this concept, distributed systems can rely upon independent services in which the application specific logic is independent from the connection infrastructure. In addition, with the adoption of XML and the large effort put into defining common specifications and standards, more flexibility has been granted. Both software layers' segregation and semantic rationalisation allow the

creation of applications that are built by combining loosely coupled and interoperable end points. Services are widely used to abstract different application specific logics to reflect business activities [20]. These activities can be reused and combined to compose new services or processes [21] without impacting the underlying activities. This is generally acknowledged as being the main commercial interest of SOA [22] as it allows a faster and safer ROI. But one major factor that slows down the development of efficient process composition is the fair amount of hard coding still required by current middleware infrastructures to be connected to each other. Furthermore, the extensive use of connecting infrastructures, their complexity and the increasing need to connect them are rendering the middleware landscape more complex.

The author acknowledges that the acronym SOA for some people is simply a marketing term for the packaging of the communication infrastructure, which in actual fact should not matter. Interoperability and connectivity issues have been discussed for the past 20 years, however what SOA has brought about is the need to integrate at the middleware level using standardised technologies.

It is indeed interesting to note that while SOA eases the diversity and heterogeneity issues inherent to distributed systems; it does not completely solve them. In fact, the present standardised ways to abstract and simplify application specific logic used by middleware to address inter-software communication issues can also cause problems between middleware structures. These issues are intrinsic to heterogeneous environments where different interests and practices meet. The next generation of middleware should respond to this challenge and alleviate inter-middleware communications to allow more dynamic service and process compositions, thus enabling more effective business collaboration. This is suggested in the SOA Maturity model [23] which advocates that top level SOA infrastructures should be able to more dynamically adapt to changes.

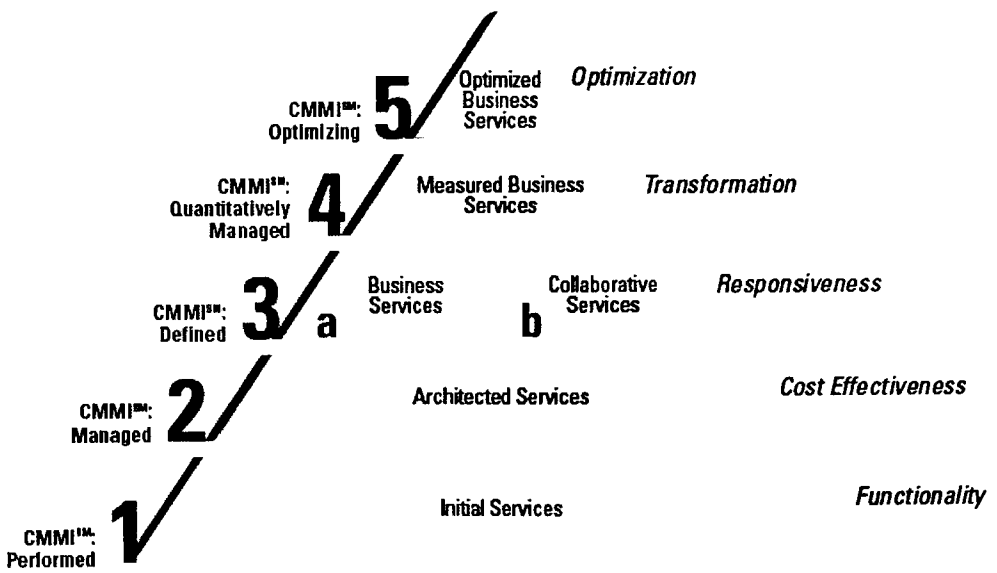


Figure 5. A Service-Oriented Architecture Maturity Model [23]

2.1.2. Enterprise Service Bus

A possible way of potentially enhancing middleware adaptability has been found in leveraging the infrastructures and practices produced during the evolution of integration software previously introduced.

The hub-and-spoke architecture has the benefit of being centralised, which provides a good structure for key features such as message routing or auditing as well as allows a higher level of reusability. However it does not scale well across heterogeneous and large distributed systems due to the centralised nature of the middleware itself. In addition, the historical lack of common exchange semantics hinders the creation of evolving collaborations. These issues were partially solved by SOA, but while allowing a more loosely coupled model, it requires time consuming low level coding. Moreover in a highly evolving environment, Message Oriented Middleware (MOM) necessitates a higher level of abstraction to allow the reusability and integration levels necessary for a fast ROI.

Figure 6 shows some of the higher level qualities of these infrastructures and introduces the concept of ESB. The ESB architecture applies knowledge learnt throughout the evolution of middleware and attempts to leverage the technologies subsequently developed [24]. Indeed the bus takes the centralised approached of the application server, the abstracted nature of the EAI broker with the distributed nature of MOM to provide a solution for rapidly adaptable middleware.

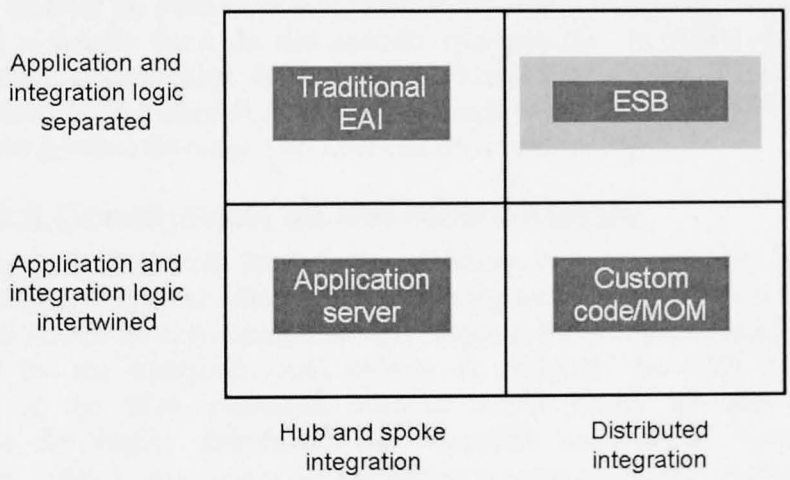


Figure 6. Integration approaches [25]

Although the exact definition of ESB varies according to author, company or features it includes, it is possible to draw a general picture.

An ESB is a SOA based software infrastructure that acts as an intermediary layer of middleware through which distributed end points are made available. It provides an abstraction layer that acts as entry point to a bus. Once the messages have been intercepted by the entry point, series of actions can take place in the bus. These actions take the form of services which are called according to various elements such as: message content, origin, destination and sets of rules predefined.

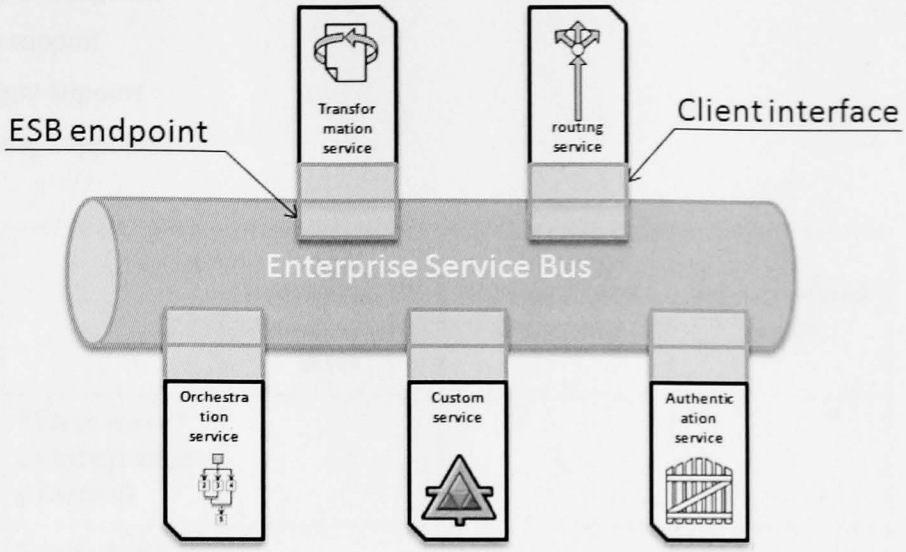


Figure 7. Generic ESB example

Figure 7 shows a basic ESB infrastructure in which services can communicate through an abstracted interface provided by the bus. Once a message is received by the bus end-point, series of actions can take place. The type of actions taking place is influenced by both the content of the messages received and the way the bus has been configured to handle them. In this specific example the bus offers authentication, transformation, content base routing and orchestration services. Finally, once the messages have been treated it is sent to the appropriate end point with the suitable data. A more detailed discussion on ESB can be found in [26].

2.1.3. Conclusions on the recent history

The concept of ESB tries to leverage different technologies and designs used throughout the middleware landscape thus moving away from the opportunistic point to point inter-software communication and adopting the centralised approach already introduced by the traditional EAI broker. In addition, the ESB concept takes advantage of the SOA paradigm benefits which jointly provides a potential architecture for highly distributed and adaptable as well as loosely coupled middleware. Table 1 summarises the key elements discussed in this section.

Tables 1-4 legend:

/ No support

L Light support

X Fully supported

	(1) Reusability of integration layer	(2) Shared communication semantics	(3) Decentralised model
"First wave" of integration practices	/	/	/
Spoke-hub distribution paradigm	L	/	/
Custom built SOA middleware	L	L	X
ESB	L+	L+	L

Table 1. Comparison of modern approaches to integration

However, if it is possible to increase the number of interaction enhancing infrastructures (.e.g. non-repudiation, message level security) offered by an ESB (c.f. [26] for list), it is clear that simply aggregating them does not suffice. Indeed, mechanisms to both define what infrastructures are to be used, how to use them and to provide a control over the information flow between these infrastructures are necessary.

2.2. Models and architectures

The main research challenge of this work is to investigate the two points listed in section 1.3 and provide a concrete architecture that describes a potentially implementable solution. There has been plenty of work from both academia and industry around the definition of architecture and indeed more specifically, architecture for SOA governance. Many of these works, such as [27] are not mentioned this section as they have not reached a certain level of visibility and or do not bring any addition to the models and architecture mentioned below. The following section reviews briefly the major efforts in this domain and describes their influential elements in the context of the current work.

2.2.1. Reference models and architecture for SOA

An abundance of specifications and standards have emerged from open standard organisations such as OASIS, OMG or The Open Group on the topic of SOA. In [28] the major instances are introduced and categorised.

These models and architectures, as expected from their status of "reference", do not present a readymade and expressive manner to implement a specific SOA. Indeed, they rather introduce the reader to the main artefacts forming a SOA along with, for some of them, guiding principles, processes, and technologies that could help organisations in the process of defining their own SOA.

In [29] however, an Execution Context illustrates the set of technical and business elements that form a path between those with needs and those with capabilities in the context of service providers and consumers interaction. All interactions are grounded in a particular execution context, which permits service providers and consumers to interact and provides a decision point for any policies and contracts that may be in force. On Figure 8, the Execution Context is represented with its links to the Interaction and Contract & Policy elements. The presence of the Service and Service description elements on this figure also give us a sign of their importance in this context.

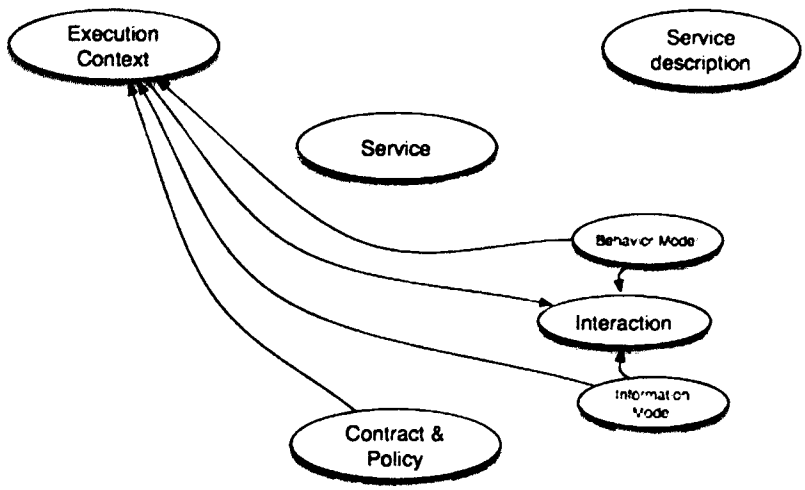


Figure 8. Execution Context [29]

Although this reference model provides valuable information on related concepts, it is insufficient to define a concrete architecture or implement a SOA governance infrastructure. Indeed, there is no description of the execution context itself or how it can be manipulated. These sets of information are necessary in order to design the contextualisation and adaptation processes which are key elements of the governance this thesis aims to describe.

In [30], SOA governance is defined as: "Governance in the context of SOA is that organisation of services: that promotes their visibility; that facilitates interaction among service participants; and that directs that the results of service interactions are those real world effects as described within the service description and constrained by policies and contracts as assembled in the execution context."

Figure 9 illustrates the vision of SOA governance in this report by lining the different types of governances within an organisation.

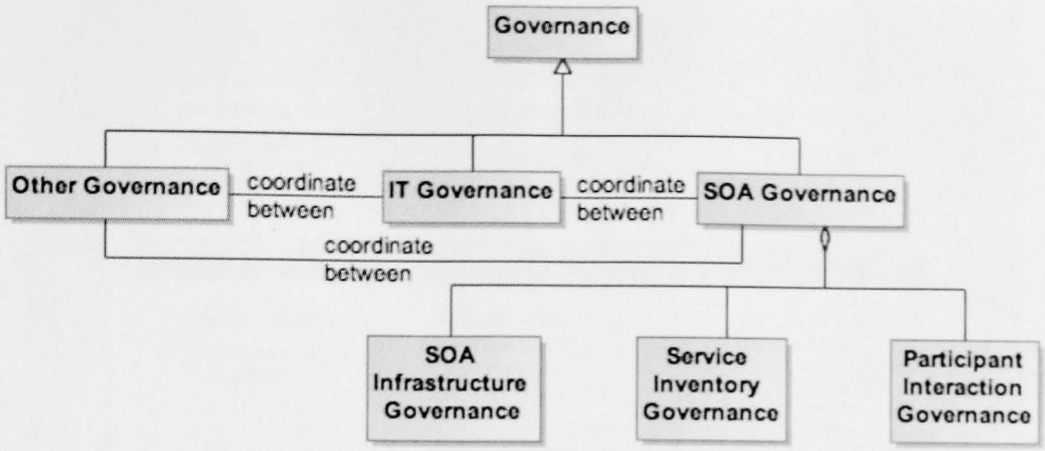


Figure 9. Relationship among types of governance [30]

Always according to this report, SOA governance applies to three aspects of service definition and use:

- SOA infrastructure – the "plumbing" that provides utility functions that enable and support the use of the service
- Service inventory – the requirements on a service to permit it to be accessed within the infrastructure
- Participant interaction – the consistent expectations with which all participants are expected to comply

As introduced in section 1, this thesis specifically looks into supporting the first item on this list, the SOA infrastructure.

With regards to this item, the Reference Architecture Foundation for SOA Version 1.0 defines ([30] page 95) a single requirement; a SOA governance architecture should take into account:

- "Governance requires that the participants understand the intent of governance, the structures created to define and implement governance, and the processes to be followed to make governance operational."

This point is focusing on the visibility and readability aspects of the governance processes as well as the availability of these processes as SOA services.

Although this reference architecture provides valuable information on related concepts and different requirements, it is insufficient to define a concrete architecture or design a SOA governance architecture.

In [31] a SOA reference architecture is described. This reference architecture defines a governance layer that mostly aims at managing different types of policies such as QoS or security. It is unclear from its short description if this layer is meant to only verify the compliance of services against these policies or if execution is also thought of.

However as illustrated on Figure 10, the authors of the SOA reference architecture take into account the challenge of contextualisation and interaction management.

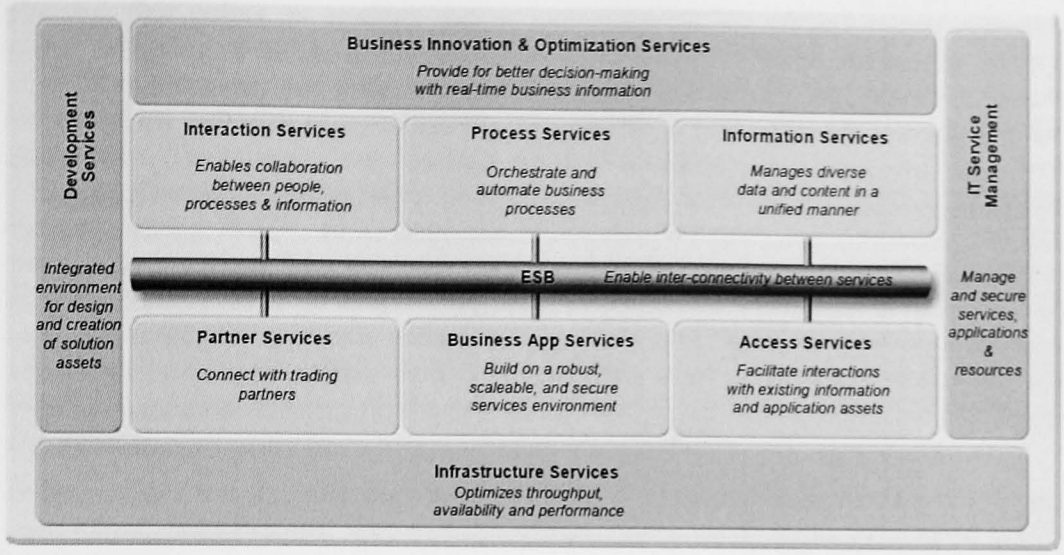


Figure 10. The Middleware View of the SOA Reference Architecture [31]

This reference architecture does provide valuable information about different requirements for SOA governance although they are not explicitly defined. Moreover, there lacks a discussion on designing issues which renders the creation of a concrete architecture and its implementation difficult to plan just from this reference architecture.

The most promising development with regard to an emerging SOA Governance architecture is the work conducted by the Open Group. Their proposal, presented in [32], for a SOA Governance framework describes the governance activities that are impacted by an SOA and puts forward some best-practice governance rules and procedures for those activities. However, as of September 2009 it has only the status of a Draft Technical Standard and still lacks essential elements such as a specification of detailed accountabilities along the service life cycle or a sound taxonomy that relates the core governance elements to each other.

Two of the main goals of this document are to provide a definition of a SOA Governance Reference Model (SGRM) and its constituent parts as well as a SOA governance vitality method to guide the customisation of the aforementioned model for specific contexts. A relevant approach for this thesis in relation to these two points would be to see how they are defined and can be elaborated upon in order to provide a concrete architecture that allows designing an efficient SOA governance infrastructure.

From the guiding principles listed in the SGRM and in the context of the current thesis the following points must be taken into account:

- *Contracts*: provider and consumer contracts shall exist between service providers and consumers in order to ensure the correct delivery of service.
- *Service metadata*: decisions and descriptions relating to services and their contracts shall be stored in a well known location, including relationships among services and their associated artefacts.
- *Automated processes*: SOA Governance processes need to be automated as it improves the reliability and traceability of the governance.

The two first points on contract and service metadata are indeed crucial elements when designing a SOA governance, as they impact on the ways the service exposition, discovery and assembly processes will be achieved. But without a clearer definition, all the work of investigating these points is left to the entity designing the governance. These topics are complex research issues on their own and have seen consequent research and industry efforts in the past, with the most widely accepted outcomes being WSDL [33], SAWSDL [34], WSMO [35] and OWL-S [36]. But many other solutions have been provided (e.g. WADL [37], METEOR-S [38]). It is therefore unrealistic to go as far as advising a concrete technology or implementation stack at the generic concrete architecture level as each approach will have its own advantages and inconvenient. However, providing a set of general, but precise and helpful, requirements falls into the role of an architecture and this point is lacking in this work.

Regarding this last point on processes, the SGRM defines three governing processes: compliance, dispensation, and communication which need to be performed on an ongoing basis.

- *Compliance*: the purpose of this type of process is to define a method to ensure that the SOA policies, Guidelines and Standards are adhered to. The compliance process provides the mechanism for review and approval or rejection against the criteria established in the governance framework (i.e. principles, standards, roles, and responsibilities etc.).
- *Dispensation*: this type of processes has the reverse function of the aforementioned type of compliance; it allows managing non compliance.
- *Communication*: these processes ensure access to and use of governance information.

These processes are during the three different stages of governance that are planning, design and operational. At the planning level, the processes are concerned with portfolio management. At design and operational time the governance processes aim at efficiently managing service and solution life-cycles.

In the context of this thesis, only design and operational times are relevant. For these stages, the SGRM lists the principal following requirements:

- Establishing and approving service
- Publishing services to enable reuse
- Managing multiple versions of a service
- Enabling service assembly for building composite services and applications
- Validating service contracts, functional and non-functional requirements
- Ensuring change management for SOA services which includes accurate impact analysis of deployed services

As for the contract and metadata points, this list of processes is a helpful hint of what needs to be taken into account. However, this work is lacking a discussion of the issues (e.g. virtualisation, management, safety) raised when designing these processes.

The TOG SOA Governance Framework mainly aims at describing SOA governance, its goals and how it impacts on the enterprise from a management point of view.

There are relevant principles that can be learnt from it, such as the points listed in this section. However, as of the time of writing, this work is still a draft. Additionally, it lacks the level of details about contextualisation description and execution that are vital for implementing contextualisation governance.

2.2.2. Autonomic computing based SOA governance model

The authors of [39] explore the relation between SOA Governance and Autonomic Computing, showing how principles and properties developed for the later can support SOA Governance. This leads the authors to create the term of Autonomic SOA Governance Infrastructures (ASGI) which is a governance infrastructure with autonomic capabilities. Following this, the authors describe a conceptual model for SOA governance based on the OASIS Reference Model for SOA, the work in [40], [41] and [42]. This conceptual model is represented on Figure 11.

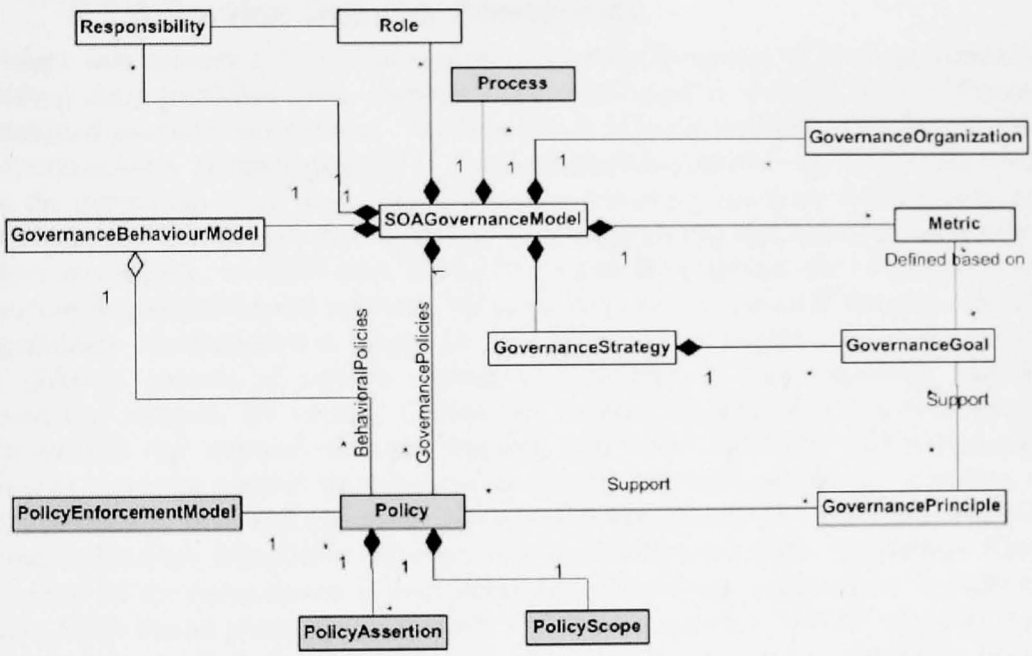


Figure 11. Conceptual SOA Governance Model - Global View [39]

On this conceptual model, policies and processes are divided in two stages:

- *Design time governance*: this group involves service identification, definition, creation and reuse.
- *Runtime governance*: this comprises management of the service life-cycle (i.e. deployment, consumption, versioning & change and retirement).

In this context, [39] defines the two key elements of the model as being policy and processes.

Governance policies: specify rules that SOA parties must adhere to. These can differ in function of the type, purpose and scope of the rule (e.g. business and corporate policies, behavioural policies, process policies, technical policies, QoS policies, testing policies).

Governance processes: identify actions that when executed contribute to achieve the goals of SOA governance.

With this in mind, [39] then approaches the autonomic computing domain and map the different types of governance policy to an autonomic property (Self-*). The attribute relevant to the current thesis is that of Self-configuration and its domain of application is defined according of the two stages aforementioned.

- Design time Self-Configuration concerns governance policies scope updating.
- Runtime Self-Configuration concerns enforcement of the governance policies.

This conceptual model does not specify any concrete pattern for the enforcement of the policies it introduces, nor identifies what the governance processes are. However, it gives information about the different stages the governance is active at and what are the main concerns for each of these phases.

2.2.3. Service Delivery Framework

Twenty first century (21C) Communication Service Providers (CSPs) are currently shifting their portfolios from traditional network-based to include more software-orientated products and services. The boundaries between networks, Information and Communication Technologies (ICT) and applications become very blurred allowing for the integration of all these capabilities into new wave services. Service Delivery Platforms (SDPs) [43] are the technology environments that facilitate this integration. More specifically, an SDP aims at enabling rapid development and deployment of new converged multimedia services. These services are composed of telecoms and IT capabilities. As illustrated in Figure 12, an SDP sits in the middle and bridges across to different sources of service capability. Examples of such capability include telephony, wireless, IP, content, Operational Support Systems (OSS) and 3rd party. Capabilities are exposed through standard functional interfaces. SDPs typically provide a service control environment, a service creation/assembly environment, a service orchestration and execution environment and abstractions for media control, presence/location, integration and other low-level communications capabilities. They are used for the composition of both consumer and business applications. In order to make SDPs carrier grade, i.e. sufficiently reliable and scalable for CSP adoption, it is current best practice to apply SOA principles to expose service capability, being network, OSS or 3rd party, through SOA adapters into a SOA based SDP [44]. The latter constitutes an IT platform, mostly an application server, using Web Service or other SOA technology standards to integrate services and compose applications.

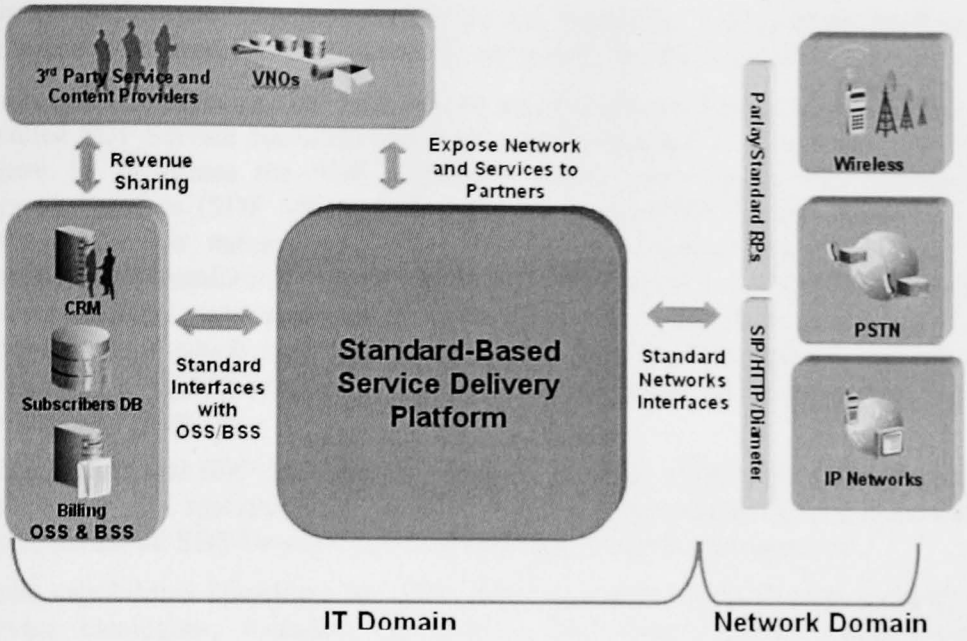


Figure 12. Service Delivery Platforms

SDPs available today are optimized for the delivery of a service in a given technological or network domain (examples include web, IMS, IPTV, Mobile TV, etc.).

There is lack of standardization work for SDPs. This gave rise to the TM Forum's Service Delivery Framework (SDF) programme [45]. In the context of SDF, services are defined as components that expose their functionality via one or more functional interfaces. A service becomes an SDF Service when it exposes one or more SDF Service Management Interfaces (SDF SMI) which manage the service lifecycle. The SDF programme focuses on the management of SDF Services, where management is referred to SDF Service Lifecycle management.

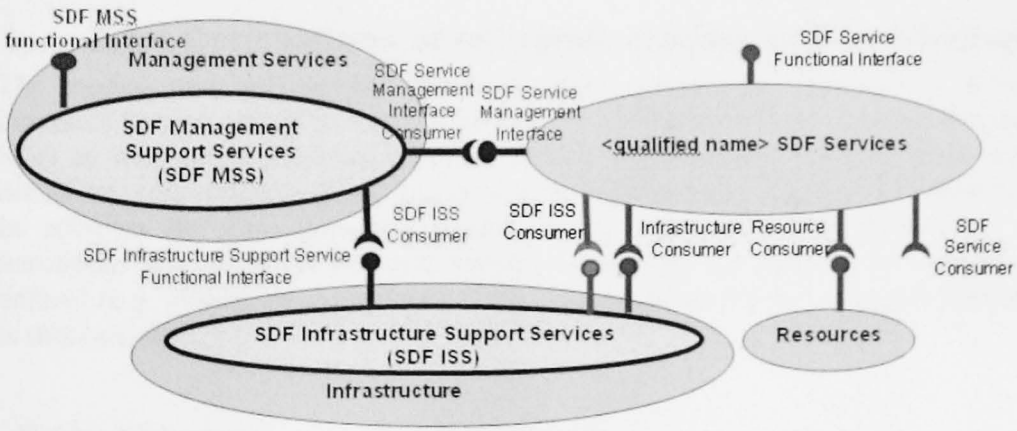


Figure 13. TMF Service Delivery Framework Reference Model

Figure 13, depicts a typical SDF service and the overall SDF Reference Model [46]. Functional capability of an SDF Service is exposed through service functional interfaces (SDF SFIs), which are graphically shown as "lollipops". Special case of SFIs are the SDF SMIs, which contain lifecycle management capabilities of the SDF

Service. Such capabilities include, but are not limited to: configuration, performance management, retirement, fault handling, versioning, monitoring and usage.

Finally, an SDF Service may itself rely on capabilities exposed by other services. This is called SDF Service consumer and is graphically denoted by a "socket". The rest of Figure 13 illustrates the SDF Reference Model comprising: SDF Management Support Services (SDF MSS): SDF MSS are responsible for the end-to-end SDF Service lifecycle management. This includes the support for operational (e.g. provisioning, installation, update/activation, monitoring capabilities) as well as business process automation. SDF MSS capabilities may either invoke SDF SMI capabilities, in which case they receive SDF Service management meta-data, or invoke other support services from the Infrastructure or the Management Services domains of Figure 13.

Infrastructure and SDF Infrastructure Support Services (SDF ISS): the Infrastructure domain provides specific capabilities to Management processes or SDF MSS that are usable across all SDF Services and facilitate their lifecycle management.

These capabilities constitute the SDF ISSs. Examples of SDF ISSs include: SDF Service catalogues, metadata repositories, user data (specific information for subscribers or other actors), resource management capabilities and charging capabilities.

Resources: Resources are capabilities that can be used by SDF Services and are exposed by network, IT infrastructure, OSS/BSS applications, or services on the Internet. They can exist anywhere, within or outside the CSP's domain, and offer their capability to SDF Services through their functional interfaces.

Its issuance from existing implementations into a reference model and its concrete approach make the SDF an interesting model to draw knowledge from. The main components of the framework as well as their relationships and the way they interact with their managed resources are described in the context of service delivery. However the SDF does not take the contextualisation governance domain into account and key elements such as policies or element visibility are note thought off.

2.2.4. Conclusions of reference models and architectures

The models and architectures for SOA, SOA governance and service delivery introduced above define, to a certain extent, the main architectural elements forming a SOA as well as the principal concepts related to governance. Indeed, these works define key concepts such as run and design times governance together with their roles. In addition the core concepts around service interaction (e.g. behaviour) are introduced and the main elements needed to manage the delivery of services are defined (e.g. management services). Table 2 summarises the key elements discussed in this section.

Tables 1-4 legend:

/ No support

L Light support

X Fully supported

	(4) Definition of SOA governance and related concepts	(5) High level description of key contextualisation elements and concepts	(6) Technical description of key contextualisation elements and their relationships
OASIS Reference Model	L	L	/
OASIS Reference Architecture	L	/	/
TOG SOA Reference Architecture	L	L	/
TOG SOA Governance Framework	X	L	/
Autonomic computing based SOA governance model	X	L	/
SDF	/	L	L

Table 2. Comparison of reference models and architecture for SOA

However, these references do not specify how to design the process of contextualisation and manage the visibility of the different elements required to enact it.

2.3. Flexible contextualisation

An area of interest that has focused on certain aspects of this investigation and that has received interest from the research community is the management of NFPs as a way to improve the adaptability of a resource exposed over the network.

Using previous work on NFP description (c.f. section 2.2 Conclusions of related works) and how to allow a concrete separation between a resource's functionality and its NFPs has been investigated.

In [47] a solution is proposed to manage a Web service NFPs using handlers. This work proposes a new model-driven development (MDD) framework, through the notion of feature modelling, to explicitly and graphically model a series of non-functional constraints in SOA. The framework consists of a:

- *feature model*: that defines non-functional properties in SOA.

- *NFP profile*: written using Unified Modeling Language (UML) to specify the NFPs, called UP-SNFR [48].
- *Model-Driven Development tool (MDD)*: called Ark, which generates application code (program code and deployment descriptors) according to a configuration (or instance) of the proposed feature model and a UML model defined with UP-SNFRs.

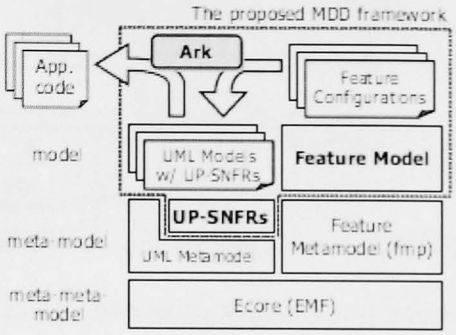


Figure 14. An Architectural Overview of the MDD framework proposed in [47]

The architecture proposed by the authors of the MDD framework is illustrated in Figure 14. By allowing developers to model NFP as features, the proposed framework allows logically constructing and validating NFPs in SOA. Ark automatically enforces NFPs in applications by transforming a feature configuration to application code with UP-SNFRs.

The separation of concerns between the different levels of abstractions in the domain model, the NFP as a feature descriptions and the generation of application code allows decoupling the descriptions of the different types of NFPs possible from the specific handlers implementing them and the way they will be aggregated to support the exposure of the service.

However by using a strategy that provides NFPs by the way of handlers the authors assume that the service providers take ownership of the NFPs implementations which limits their reuse and availability. In addition, this framework does not provide any support for the service life-cycle management which restricts the use of the service in one particular context. Finally, the use of NFP profiles could allow the use of different semantics and grammars, providing another level of flexibility, but current approaches have not investigated this possibility.

In [49], the authors present a Security Service Bus (SSB), an infrastructure that relies on a communication bus for providing flexible composition between application components and security components and mutually between security components. This SSB is evaluated against a scenario with a personal content management system platform that aims to offer its users a uniform interface for managing and sharing their personal content that is scattered over various devices.

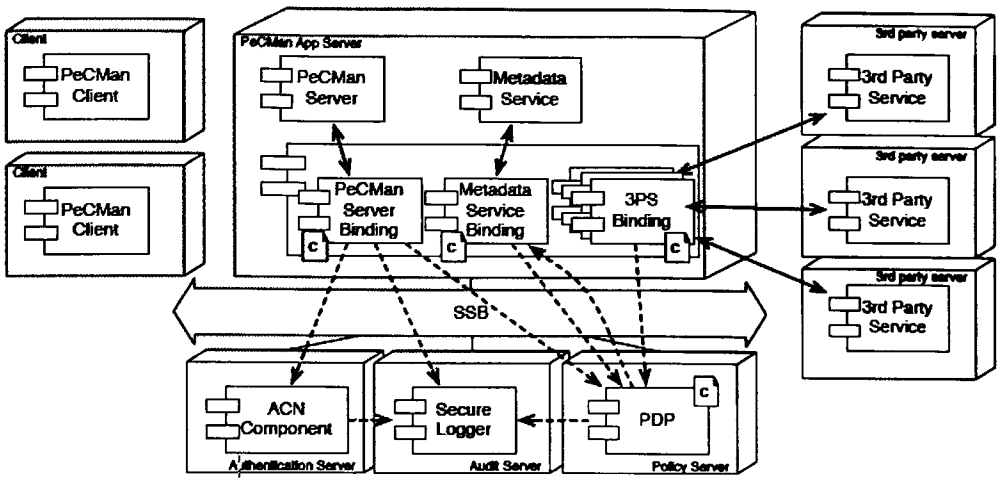


Figure 15. SSB-based Composition of the Application with Security Components [49].

In the SSB there are two types of component security and application bindings.

Security: this type provides the security functionalities. It is assumed in this architecture that the security components are reusable modules that can be invoked, managed and composed with applications and with each other. The functionality is expressed through a security interface and the component can be configured through a management interface.

Application binding: This type of component has two roles enforcing decisions and providing policy information.

In this architecture, the aggregation of components is made through an artefact named the security contract. This contract is expressed using the same data model as the one used to define component interfaces. It comprised two parts: the required and provided sections, defining what is expected from a contract and what is offered.

The SSB architecture describes a component based design that supports run time aggregation of security related functionalities. However, this work only takes into account security related functionalities, more complex exposure requirements that encompass more than security may require a different type of architecture. In addition, the SSB does not show how to support the life-cycle of the aggregation of added functionalities which is an important factor when dealing with independent, distributed components that could be, for instance, unavailable at times. Finally, the SSB does not describe how to support the management of components fulfilling the security contract. The author believes that such architecture should define at a more concrete level this type of mechanism rather than stating the existence of a management interface.

2.3.1. Conclusions of flexible contextualisation

The works discussed in the previous section present interesting characteristics as well as limitations for the domain of flexible contextualisation. In this section, both advantages and limitations are presented together with their impacts on the current thesis.

Both MDD and SSB demonstrate that the hard coded handlers approach, which is currently the most widely spread approach [50], has strong limits in terms of

flexibility and reusability. In an distributed environment where components could be reused many times, the impact changes on these components could have on their many clients could prove difficult to manage. Additionally, unless a strict control is put in place and a physical track of what handler is used on what service kept, keeping track of their usage in large scale systems seems effort consuming and further advocates against this practice. Moreover, managing the exposition of a same service in different contexts is difficult as the handlers are hard coded. Finally, hard coding the handlers into the service themselves can be costly time wise as a developer is required to understand how they work and how they can be assembled with the service they are expected to support the exposition of.

The MDD [47] approach deals with this limitation by dynamically aggregating handlers in a NFP profile through a model. This method presents the benefit of dividing the representation of the contextualisation strategy (i.e. NFP profile) from the complete domain definition (i.e. feature meta model) and from the run implementation of the strategy. This allows for more flexibility and reuse. However, the code enacting each NFP cannot be shared and allowed to evolve (e.g. reconfigured).

The SSB approach attempts to mitigate this limit by using distributed components that can be invoked and configured to fulfil the NFPs. For supporting the exposition of services in large scale distributed systems, this technique of using discoverable and configurable network enabled components looks more efficient in regards of early comments. Not only does it allow reuse and configuration for specific contexts but through the use of a common transport layer it permits to keep track of what components are used in what context. Furthermore, it authorizes the governance infrastructure to audit the message exchanges and evaluate how the components perform.

Yet these different techniques still present strong limitations. They necessitate an additional layer in order to allow using a resource in different contexts. In addition, they do not allow controlling the life-cycle of the NFP profile, which would bring further flexibility. Furthermore, they each use their own set of NFP description semantics which can be a limited approach when dealing with distributed resources that can be deployed in different organisations or countries.

Tables 1-4 legend:

/ No support

L Light support

X Fully supported

	(7) NFP description model	**(8) NFP profile description**	**(9) NFP profile instance**	**(10) Reusable & manageable NFP providers**

MDD Framework	X	X	X	
SSB	L	L	/	X

	(11) Run time aggregation of NFPs	(12) Domain agnostic	(13) Life-cycle management of the aggregation of NFPs	(14) Semantically described components
MDD Framework	L	X	/	X
SSB	X	/	/	L

	(15) Semantic agnostic
MDD Framework	/
SSB	/

Table 3. Comparison of approaches investigating adaptability of contextualisation

2.4. Safety of contextualisation

There is no work, as far as the author is aware, that targets the safety of contextualisation processes or middleware for distributed systems.

However, the domains of contextualisation can be, in this context, divided into two parts, the validity of the contextualisation strategy (c.f. NFP profile in section 2.3) and the security of the message exchanges between the different components enacting this strategy and the infrastructure supporting these exchanges.

In the context of this work, the optimal solution to support a flexible contextualisation strategy is an aggregation of distributed and reusable components that can be managed. Indeed, section 2.3 demonstrates that this approach is optimal to provide adaptive contextualisation for SOA. In order to control the safety of this strategy it is therefore critical to control what components are selected and the manner they are aggregated. Dynamic selection and composition of distributed components, such as web services, is a complex research topic on its own and will not be discussed here. The reader can instead refer to key works in this domain such as [51] or [52]. It is however noticeable that dynamic composition of services aims at bringing flexibility about the services selected, as opposed to allowing defining and enacting a contextualisation strategy.

SOA security will not be examined in this work as it is a complex research topic on its own.

Tables 1-4 legend:

/ No support

L Light support

X Fully supported

	(14) support for end to end message level security	(15) verify validity of the contextualisation strategy
SOA security	X	/
Dynamic selection and composition	/	L

Table 4. Comparison of approaches investigating safety of contextualisation

The present work does not aim at making significant contributions in these two fields. It aims however at leveraging on them in this new context in order to provide an enhanced investigation.

2.1. SOA Governance

SOA governance has been much talked about over the past few years. Industry middleware actors (e.g. SOA middleware vendors, consultants) have been the biggest sources of both hype and innovation [10].

The term "SOA governance" has also sometimes been treated as a marketing term for the packaging of the set of features that allow managing and improving the visibility of distributed resources. Such issues are well understood and solutions have been researched and developed for the past 10 years. In fact, SOA governance frameworks build on top of such work by addressing the need to make the supporting service management and monitoring layer interoperable and introduce processes that allow governing multiple interrelated services and policies in SOA deployments as one whole.

ESB vendors, services deployment platforms and Service registry providers (e.g. HP, IBM) include what they define as governance tools in their products. These products such as HP Systinet with its Governance Interoperability Framework (GIF) [55] or IBM WebSphere Service Registry and Repository (WSRR) [56] are mainly providing service registry and metadata (e.g. policy) repository services along with their supporting features. Some of these products also provide some support for service versioning and policy as well as service management. These are valuable contributions towards a common SOA governance specification. However, these products only address the challenges of visibility and for some of them policy

administration and management which, although are central issues, do not provide solutions for the problem of service provisioning. Indeed, preparing SOA assets to be used by consumers in a flexible and efficient manner is a key aspect of SOA governance as it allows reducing integration expenses.

2.2. Conclusions of background

The work surveyed in section 2, and the short history of integration techniques, can be viewed as the basis to provide resource exposition governance in a flexible and efficient manner. The surveyed work illustrates the need to design the architecture of the governance infrastructure as a distributed system where elements can be discovered, assembled and managed adequately. An aim of this work is to provide a concrete architecture that allows for safe and flexible governance of resource exposition in SOA as proposed in section 3.

Another area of interest that has focused on certain aspects of governance and that has received more interest from the research community is the description of Quality of Service (QoS) attributes, NFPs, services functionalities and architecture descriptions.

Several projects have looked into different ways of defining and expressing NFPs, using either Architecture Description Language [57], taxonomy [58] or ontology [59]. Some of these projects do not target any precise type of IT systems [57], while a few specifically investigate the domain of SOA [58]. As underlined at several points (c.f. 2.2.2, 2.2.3) in the previous section on related works, being able to describe the different components of a dynamic system is a critical element. However the author feels that this topic has already reached the stage where efficient descriptions of architectures, services and their properties can be used. Additionally, as illustrated several times in the aforementioned works, knowledge can be delegated to a specific layer of an architecture.

In the next chapter, the research challenges for flexible and safe resource contextualisation in a SOA are summarised.

2.2.1. Summary of research challenges

Large scale and dynamic service-oriented systems require a set of technologies and mechanisms to be deployed in order to be rendered more easily manageable and transparent through governance. A wide spectrum of complementary concerns needs to be taken into account when designing such a solution.

The following challenges have been defined using the scenario described in section 1.1 and the description of the related works listed in chapter 2.

1. Reusability of integration layer. The different functionalities forming the governance infrastructure and Value Adding Services (VAS) that may be invoked should be adequately segregated so as to allow reuse.

2. Shared communication semantics. Interoperability is a key element in middleware design.

3. Decentralised model. The governance infrastructure should enable a flexible resource location mechanism where the choice of the resource depends on contextual information in addition to the network endpoint of the governed services. Additionally, the infrastructure should support resource virtualisation, policy segregation and execution state in multi-tenancy usage scenarios.

4. Definition of SOA governance and related concepts. For the readers of the concrete architecture it is critical to explicitly make the goals understandable.

5. High level description of key contextualisation elements and concepts. For the concrete architecture to be implemented, its different elements and their relationships must be made understandable.

6. Technical description of key contextualisation elements and their relationships. For the concrete architecture to be implemented, the way its different elements can be designed and assembled must be defined.

7. NFP description model. A model or sets of models allowing the different elements potentially entering in the resource exposition governance should be provided. Alternatively or in addition, this type of model should be extendable.

8. NFP profile description. A resource provider should be allowed to express resource exposition requirements.

9. NFP profile instance. A same profile description can be enacted using different middleware. This includes allowing the selection of infrastructure capabilities and the corresponding policy schemes.

10. Reusable & manageable NFP providers. When relevant, the different elements composing a resource's exposition and its supporting infrastructure should be proposed in such a manner that they can be configured for the specific interactions they required for. In addition, the use of reusable elements should be supported.

11. Run time aggregation of NFPs. The composition of the different elements composing a resource's exposition and its supporting infrastructure should be supported at run time.

12. Domain agnostic. The exposition governance should not be specialised into a specific kind of exposition attribute. Additionally, it should not allow governing the exposition of only a specific type of resource.

13. Life-cycle management of the aggregation of NFPs. The composition of reusable and manageable elements supporting the exposition of a resource should be made manageable as a process. As such, the governance infrastructure should allow for the management of its life-cycle.

14. Semantically described components. The different elements entering in a exposition and its support should be adequately describe to allow for their automatic discovery and usage.

15. Semantic agnostic. The infrastructure should enable translating its own internal understanding of what are the objectives for specific governance instances into a way that is comprehended by partners.

16. Support for end to end message level security. The proposed architecture should ensure a safe exposition of the services. Potential clients or threats should not be able to bypass the governance capability put in place due to one of its capabilities failure (e.g. security, bad design).

17. Verify validity of the contextualisation strategy. A resource provider should be allowed to express its needs in a clear manner that doesn't leave any ambiguity as to how the governance should enact them. It should also support communicating

governance requirements to trusted partners and ensure consistency persists between the internal governance logic and what is advertised or agreed persists.

As shown in this section, there currently is no model, architecture or technology that fulfils the aforementioned sets of challenges. Instead, vendors have a tendency to aggregate the different products they have developed over the years that supports the management of distributed resources and academic works tend to investigate specific issues related to this topic. Finally the architectures and models proposed by open groups of experts are too abstract and general to allow implementing such middleware.

These challenges, coupled with leveraging the technical advantages discussed in the previous section on background, led to singling out a specific architecture which, according to the author experience and previous work in this domain (c.f. section on summary of contribution) sufficiently address the core need of exposition governance for SOA.

3. Anatomy of a SOA governance architecture

In Chapter 2, the author described relevant background and related works. Four main areas, evolution of middleware, models and architectures, flexible contextualisation and safety of contextualisation have allowed the identification of 17 research challenges.

3.1. Introduction of the anatomy

In the following section, the concrete architecture for governance of safe and flexible contextualisation of resources for SOA governance is presented. Initially, the author defines the main concepts necessary to the good understanding of the anatomy's description. Next, the data structures used in the concrete architecture are presented. Then, the components forming the architecture are described and categorised in either a core category or a management category. Finally, the processes that link these different data structures and components are shown.

This architecture is meant to be platform and language independent and no specific tools or frameworks will be discussed in this chapter.

The elements defined in this section are derived from the knowledge acquired through Chapter 2. Additionally, the implementation described in section 4.2 relies on the concepts, data structures, components and processes described in the current chapter.

3.2. Anatomy concepts

Prior to the descriptions of the concrete elements forming the architecture it is necessary to define several key concepts: capability, policies, processes, users, and governance times.

3.2.1. Capability

Capability: a capability is a functional unit in the governance context. Each capability is assumed to be capable of being deployed as a web service with its own service management, policy administration framework (control pane) and operational interfaces (data pane). Each capability is also policy driven as this permits configurability and flexibility. The different capabilities are meant to have their own distinct grammars and policy languages in order to keep their own advantages, capacities and evolution potential in their respective domains. For instance, the identity and access management can be separated and can use different grammars.

The interoperability issues generated by this situation are addressed at the messaging level, which is in itself a core capability, and through transformations that are possible thanks to transformation policies as shown in the transformation process step (section 3.5 point 7).

This also enables the interchange of core capabilities within their categories and according to the research challenges specified in section 2.2.4 when necessary. Non-core capabilities (e.g. auditing) can also be added through the same manner.

All capabilities are either a business or an infrastructure:

Business capability: This is an organisation's traditional function (e.g. accountancy, fleet management, credit check). It is exposed as a service and can be the result of an

aggregation of other business capabilities. In the case of the virtual music store content providers are business capabilities.

Infrastructure capability: This is a supporting capability fulfilling non-functional requirements such as identity management or access control. In an SOA, a set of infrastructures are typically aggregated to support the exposition of a business capability. Infrastructure capabilities can be segregated in two categories: core and non-core. Core infrastructures are functionalities that are vital to the governance architecture internal behaviour, these are described in section 3.4.1. The non-core infrastructures can include all type of non-functional property providers (e.g. billing, audit, transport protocol).

Some capabilities may be treated as business in a context and infrastructure in another. For instance, an identity management service in the virtual music store use case is treated as an infrastructure as it provides a non functional property. However, the identity management service can make use of its own exposure governance in which context it is treated as a business capability.

Section 3.4 on Components anatomy presents the components specific to exposure governance.

3.2.2. Policies

Policies are documents describing behaviours that capabilities or processes must comply with. They typically comply with different specific standards (e.g. WS-Policy [63], XACML). The main issues about policy in the governance framework are their enforcement and the potential necessity to translate same policies into different grammars (e.g. an access control infrastructure could be using either XACML or SecPAL). In the following paragraphs, the main policy types of SOA governance are introduced.

Profile policies: Profile policies identify and define policies or template that applies within their domains. The most important ones will regard the dependences and constraints related to the use of a profile.

Infrastructure capability policies: These policies are attached to particular infrastructures and consider potential I/O metadata, usage and management schemes.

Business capability policies: The business capabilities are similar to the infrastructure but for the possibility to assign exposure strategies to them.

Section 3.3 on Data structures anatomy shows the data specific to exposure management.

3.2.3. Processes

A process is a procedure that uses the above building blocks in order to meet exposure governance objectives. A distinction can be made between governance as well as policy and service management processes. The management processes target policies (e.g. authoring, association, enforcement, reporting) and services (e.g. publication, exposure) and are outside of the scope of this investigation. The governance processes aim at coordinating management and exposure governance processes.

Section 3.5 on Processes anatomy describes the processes specific to exposure governance.

3.2.4. Users

User: the entity, physical or logical that uses a service. More concretely, in the current context, there are three types of user:

Business or infrastructure capability administrator: these users should be allowed to define their requirements and/or to specify the context in which their resources should be exposed. They can also change the configuration of the profiles they want applied to their resources. An infrastructure administrator can also change the description of its resources or when relevant change the context (e.g. SLA, type of client) the infrastructure will be available for.

For instance, a music content provider, as a business capability is administered by a user. The same applies to an infrastructure capability such as an identity management service.

Governance administrator: much in the same way as the capability administrators, this user can modify the configuration or change the services that serve as core capabilities.

Another system: as specified in section 2.2.4 on research challenges, there is a need for automation of management tasks. Through the management interface, other systems such as another governance middleware or a management tool can extract, modify or deleted data (c.f. section 3.6 on Usage patterns). This type of access is managed in the same way as human user in regards to security.

3.2.5. Governance times

This governance architecture targets two phases, the design and run times of the capabilities which contextualisation it is meant to govern.

Design time governance: at design time, users are allowed to define their choices in terms of exposition behaviour. This phase is illustrated in Figure 16 where a user configures part of the governance to suits its needs.

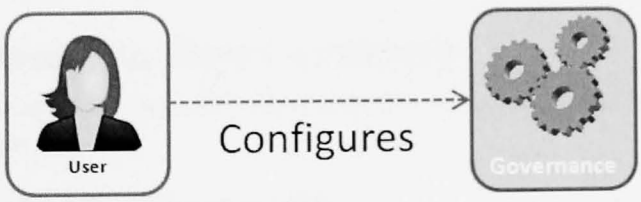


Figure 16. Design time governance

Run time governance: At run time, the governance enacts the requirements specified at design time and allows for the conversations between the business capabilities, and its clients to take place. This is achieved by brokering the messages forming these interactions and by aggregating infrastructure capabilities which provide the NFPs. This stage is shown on Figure 17 where the governance middleware links the business capability to a client and call infrastructure capabilities to supply the NFPs for the interaction. In this case, the infrastructures are sitting somewhere in the Cloud [66], but different options for their deployments are proposed in section 3.6.1.

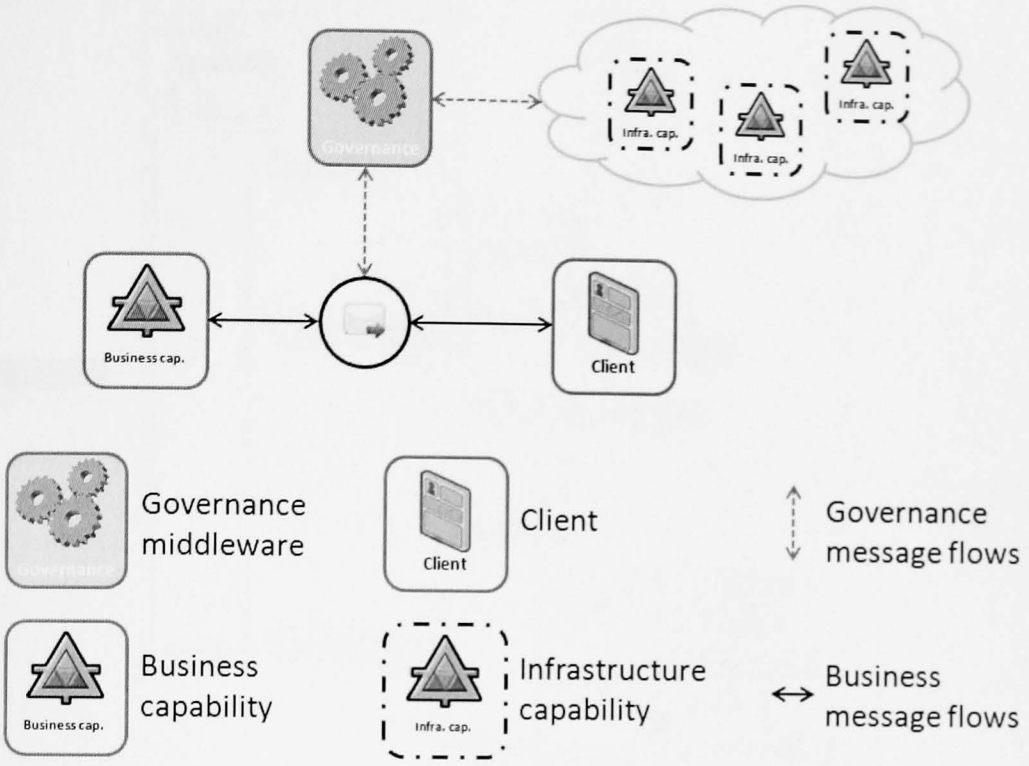


Figure 17. Run time governance

The separation between these two stages is not always as clear as it appears in Figure 16 and Figure 17. Indeed, the need to re-configure the properties or the quality of the enactment of an interaction can arise during run time and requires the governance to at least partially fall back to the design time phase. Additionally, the registration of new infrastructure capabilities can happen at any moment, so do the potential changes in their configurations or in the content of any policy they make use of.

3.3. Data structures anatomy

Following is the list of the core data elements that take part in the governance model and their properties.

3.3.1. Infrastructure profile

Infrastructure profiles are descriptors that define which aggregation of infrastructure capabilities (e.g. security services, audit) to use for the exposition of a business capability along with the different constraints associated with the use of these capabilities. Each profile associates infrastructures with their corresponding policy schemes, dependences (policy and service) and management processes.

In order to achieve this, the NFP requirements of the resource exposed are expressed in a normalised manner in a profile document which is used to define the way the resource is exposed. Figure 18 illustrates the schema that formally defines the data structure of the profile description. The XML schema can be found in appendix 1 and an instance of a profile can be found in section 4.2.4. In the following paragraphs the different elements forming the profile data structure are defined. Element names are

given in **bold** and concrete examples following the virtual music store use case in *italic*.

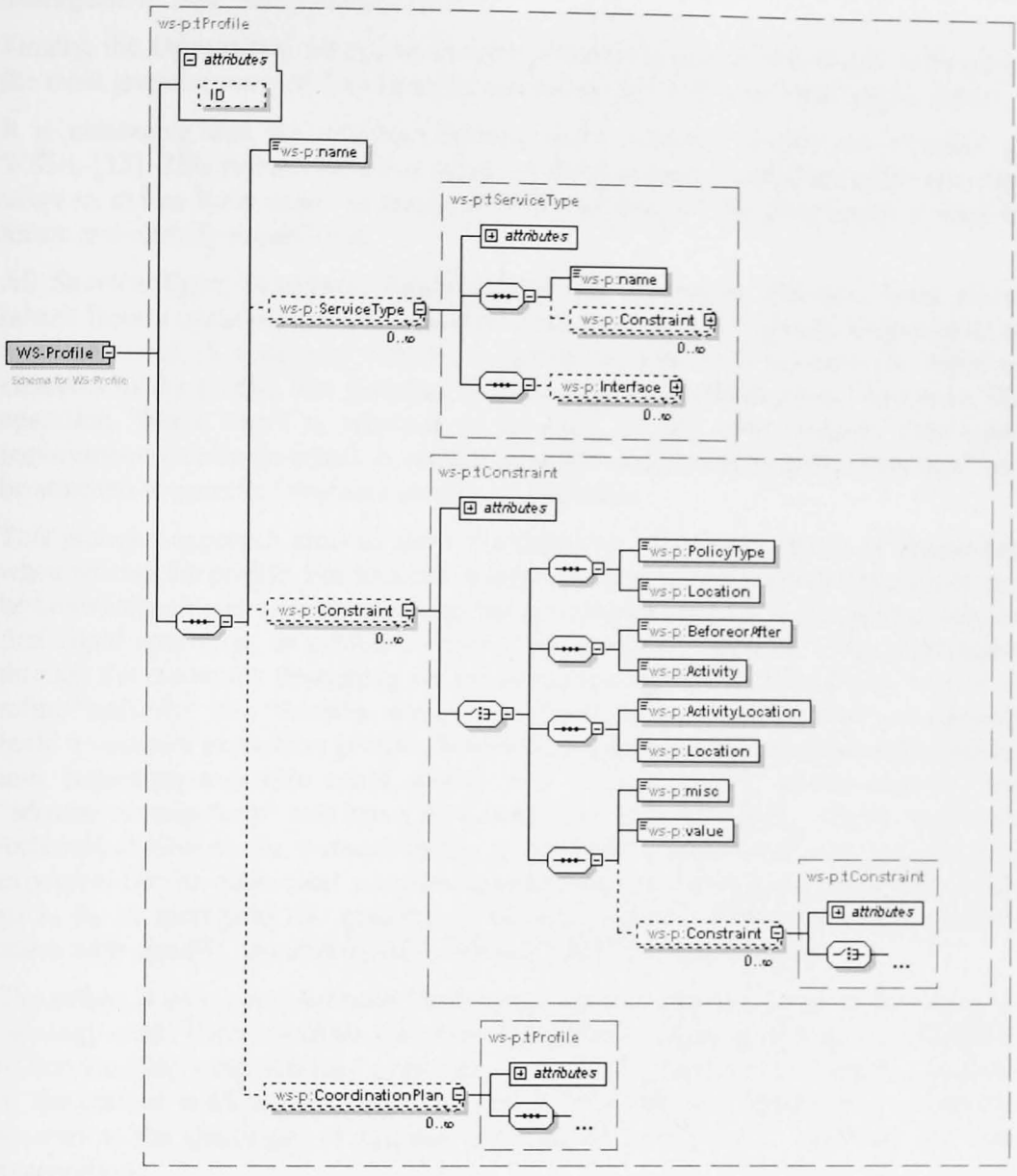


Figure 18. Illustration of the Profile definition XML schema

A **WS-Profile** is the top element of the description, it is uniquely identified by a number. Concretely, a profile could be use to "*expose a music content provider service*" in the context of the virtual music store.

The profile is composed of one-to-many **Service Types** that provides the top level of abstraction from a capability functionality and categorisation point of view. An examples of service types are "*identity management*" and "*access control management*".

Each service types can provide different **Interfaces** that allow to further decompose the logical structure of the profile. Examples of interfaces are "*management*" and "*operational*".

The next level of decomposition is the **Functionality** which allow to differentiate practical aspects of the same interface. Examples of functionalities are "*user management*" and "*token management*".

Finally, the **Operations** permit to describe concrete pieces of distributed software in the most granular manner. Examples of operations are "*add user*" and "*delete token*".

It is noticeable that the structure defined above matches closely the structure of WSDL [33]. This reflects the SOA nature of the architecture proposed while allowing users to define their needs in terms of NFPs as well as the governance system to locate and classify capabilities.

All **Service Type**, **Interface**, **Functionality** and **Operation** elements listed above inherit from a common element **Activity**. Activities allows to specify **Constraints** to these elements. A constraint permits to define the data flow between the different elements of the profile. For instance, before using an "*validate token*" operation, the operation "*build trust*" is required to be used. In the same manner, data types requirements can be specified. A certain "*XACML access control policy template*" can be attached to specific "*evaluate assertion*" operation.

This granular approach aims to allow the definition of different levels of abstraction when writing the profile. For instance, a music content provider administrator may not be knowledgeable in security but trusts the governance provider to be. In this case the first could specify in its profile a simple "*service type*" "*security*". The governance through the taxonomy describing the infrastructure capabilities would then be able to refine "*security*" into "*identity management*" and "*access control*" and subsequently build a concrete exposition profile. Similarly, a music content provider administrator user requesting a profile could specify two "*service types*" "*access control*" and "*identity management*" and leave the selection of the appropriate specific and more technical choices to the instance of the governance architecture that is governing its exposure. On the other hand a knowledgeable business capability administrator could go as far as specifying the anatomy of "*access control*" infrastructure capabilities it wants with specific "*security policy grammar*" and "*policy templates*".

The author is aware that methods for dynamic service selection based on taxonomy or ontology exist. These methods can allow functionality based selection as well Quality of Service (QoS) and Service Level Agreement (SLA). However it is not the objective of the current work to investigate this topic. The proposed approach is to provide answers to the challenges of dynamic selection of infrastructure capability and their aggregation.

Management plan: The management plan is a specific type of WS-Profile and is provided by infrastructure capability providers and compiles the necessary steps necessary to achieve in order to include provided activities. For instance, a "*XACML policy decision point access control*" infrastructure capability provider can specify that before using the "*evaluate access request*" "*operation*" the "*operation*" "*build trust*" needs to be successful.

Coordination plan: The coordination plan is another type of WS-Profile that is progressively assembled and completed with the business capability exposition profile. The coordination plan is an ordered list of management plans that need to be taken into account in order to activate an instance of the profile. Additional coordination plans can be provided for the different stages in a profile life-cycle: deactivate, reactivate and remove.

In Figure 19, the different entities forming the profile and their relationships are shown.

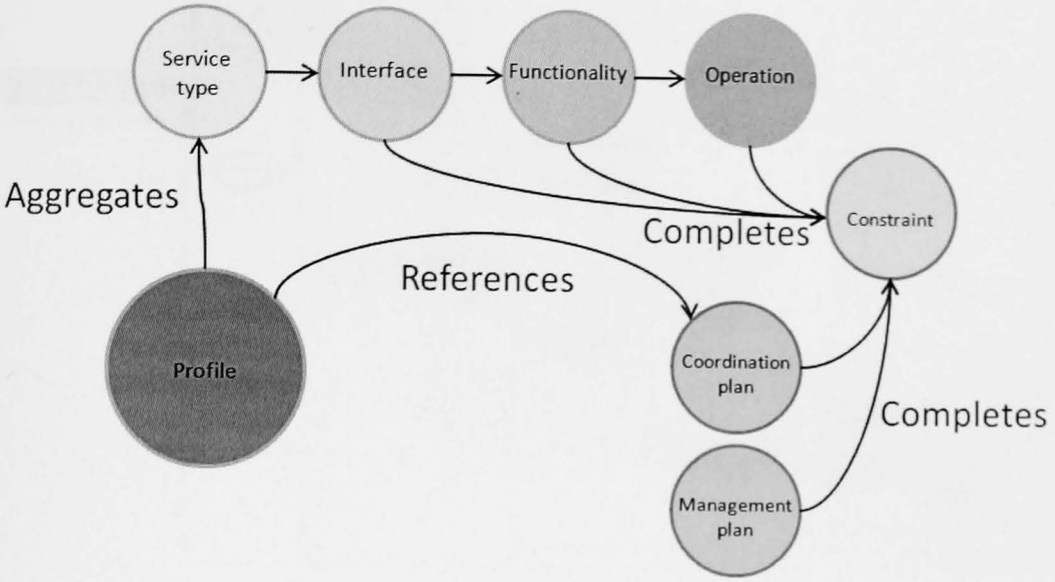


Figure 19. Profile description taxonomy

3.3.2. Context

Contexts are documents that allow the specification of a) a domain of contextualisation and b) what is the data specific to this domain. Another document, called a context selector, allows c) associating profiles and contexts.

Figure 20 illustrates the schema that formally defines the data structure of the context description. The XML schema can be found in appendix 2 and an instance of a context can be found in section 4.2.4.

Context specific data: The context specific data is a combination of data passed to the profile instantiation by the business capability and data created by infrastructure capabilities passed to others through the governance model in order to configure them. For instance, when requesting security in the exposition profile, a music content provider can pass its own "*credentials*" in order to complete the context. In the XML context definition, this part is defined by the **configData** element, the **target** being the activity in the profile the **data** applies to.

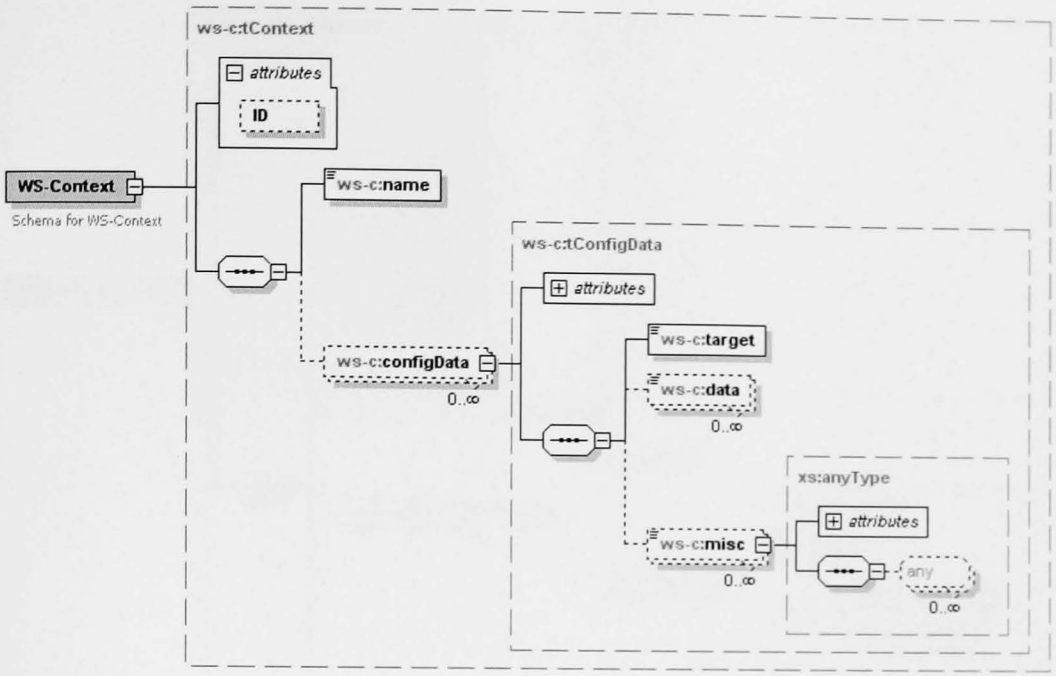


Figure 20. Illustration of the Context definition XML schema

3.3.1. Context selector

Association of profiles and contexts: This document allows to link different contexts to the profiles their associated with.

Domain of contextualisation: The domain of contextualisation allows the governance architecture to recognise what a context applies to. The interception domain is a combination of a potentially shared scope and state. This allows linking a profile to business capabilities or message exchanges. This part of the context is typically formed by a "*transaction ID*", a "*federation ID*" as introduced in [53], a WS-Addressing [62] "*message ID*" or even an operation type the profile instance is required for (e.g. "*request*" or "*response*"). In the XML context definition, this part is defined by the **configSelector**, the **target** being the identifier of the document containing the selection logic, the **data** being the potential representation of the selection logic (i.e. if it is a simple XPath in the implementation proposed) and the **operation** being request, response or both.

Up time: This element allows specifying the availability of the exposition. For better performance, the profile and exposition logic can be requested to be maintained "*all the time*". Alternatively, the exposition logic can be maintained "*only when relevant*" or at "*precise dates and times*". In the XML context definition, this part is defined by the **upTime**, the **type** allowing to define periods the profile will be maintained for.

Figure 21 illustrates the schema that formally defines the data structure of the context selector description. The XML schema can be found in appendix 3 and an instance of a context selector can be found in section 4.2.4.

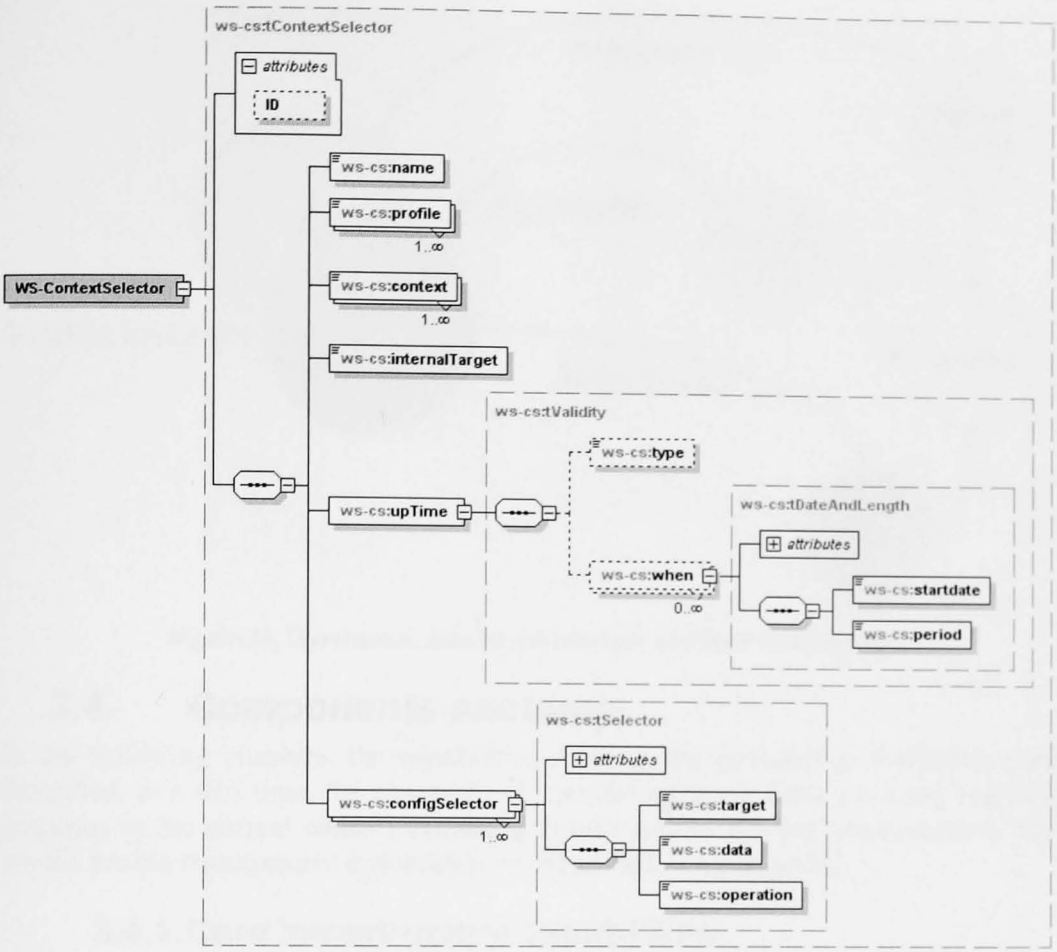


Figure 21. Illustration of the Context Selector definition XML schema

3.3.2. Summary of the data structure descriptions

Each exposure, with its data flows, dependencies, policies and management processes is defined through the three data structures defined above: profile, context and context selector. These elements, together with the capabilities they are attached to along with their relationships are recapitulated in Figure 22.

Figure 21 illustrates that a profile instance is formed by the combination of a profile and contextual data. This contextual data is divided into two categories: profile and capability oriented.

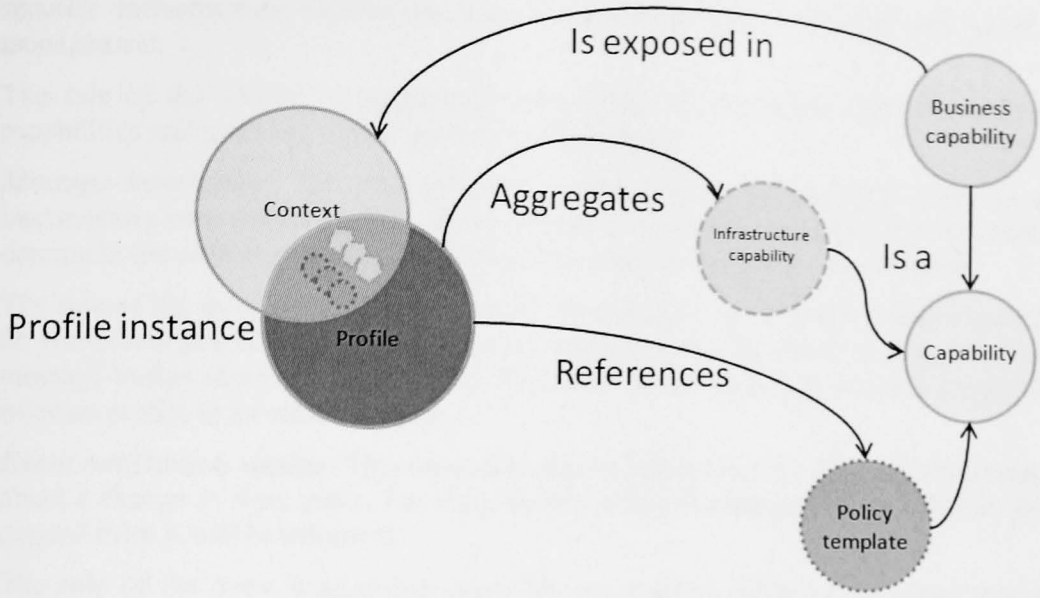


Figure 22. Governance data model concepts and their relationships

3.4. Components anatomy

In the following chapters, the capabilities forming the governance middleware are described. In a first time, the core and non specific infrastructures are listed and their purposes in the current context explained. Subsequently, the key infrastructures that are the profile management capabilities are described in more detail.

3.4.1. Core infrastructure capabilities

The core capabilities are the working base of the model as they each provide specific functionalities that are necessary to the execution of either all or part of the governance.

Following is the list of the governance model core infrastructures and their basic properties. It is noticeable that all these capabilities, are not specific to the governance anatomy. Therefore, describing them in details is out of the scope of this current investigation.

User interface: the user interface is a set of capabilities that allow users (e.g. administrator, other system) to access parts or all functionalities of the governance middleware.

Access control: An access control infrastructure is used in order to check authorization. It generally consists of a specialised service that checks security assertions against access control requests. This is typically achieved through the use of access control policies and security assertions written in specialised grammar such as XACML [60] or SecPAL [61].

The role of the access control capability in the governance anatomy is to ensure end to end control of the massages' content access.

Identity management: The role of the identity broker is to allow users to identify themselves. Authentication of the entity that acts as user is indeed a key aspect of SOA where different domains (e.g. companies, branches) will interoperate. This is generally accomplished by using security tokens. Depending on their anatomies,

security infrastructure capabilities can provide both access control and identity management.

The role of the identity management infrastructure is to ensure that the different capabilities and users are indeed who they claim to be.

Message interceptor: The message broker, often called the mediator, acts as an intermediary between two points. It can receive messages from multiple destinations, determine the correct destination and route the message to appropriate channels.

The role of the message broker is to allow the exposure governance infrastructure to be seamlessly placed between the business capability and its client. Additionally, the message broker allows the deployment of context based routing in order to assign the relevant profile to a specific context.

Event notification engine: This capability allows infrastructures to send notifications about a change in their states. For instance if a policy is changed, all capabilities that depend from it will be informed.

The role of the event notification capability is to allow different infrastructures to learn about a change of state and update their own behaviour accordingly.

Metadata repository: Often referred to as a policy store or simply a repository, the role of this infrastructure is to allow storing metadata such as policies, taxonomies or ontology. Together with the service registry, this is the most commonly found element in existing governance solution.

Policy management: The policy management is a set of capabilities that allow manipulating policies (e.g. apply metrics, detect and resolve conflicts). The author assumes that policy management capabilities are provided and their specific characteristics are left opened. Policy management is a complex topic in itself that is investigated in both academia and industry. Throughout this document, as stated in section 3.2.2, policies are meant as statements that define and constraint some aspect of a capability's behaviour.

Service registry: The service registry is a repository where Web services are listed. On production of their credential, users and systems can then discover the services which are potentially organised in different categories.

Service management: Service management comprises a set of capabilities that allow manipulating (e.g. configure, instantiate) services and when relevant their instances.

3.4.2. Profile management infrastructure capabilities

The management model supports the interactions between the different elements of the infrastructure. The main elements of this layer are the user interface, the profile management, policy management and the service management services. In this section, the author will describe the profile management and define its links with the user interface, policy and service management capabilities.

Profile management: Profile management represents the key element of this research. It aims at allowing the administration of profiles life-cycles and connects to the other services in order to so. This is done to guarantee that profiles are defined, enabled, monitored and disabled when relevant in agreement to user's needs and infrastructures requirements.

Figure 23 presents a top level view of the profile management infrastructures together with the user interface, management infrastructures, registry and repository along with their relationships. On this figure, it can observed that through the user interface a user can define a profile and specify a context. This data is then sent to the profile management that stores it in the meta data repository. When relevant, the policy management infrastructure performs tasks related to policies such as indicating and fixing inconsistencies. The profile management is also connected to the service management, which enables tasks related to capability selection and usage. The service management is connected to the service registry which stores capability specific data including the location of the infrastructure capabilities.

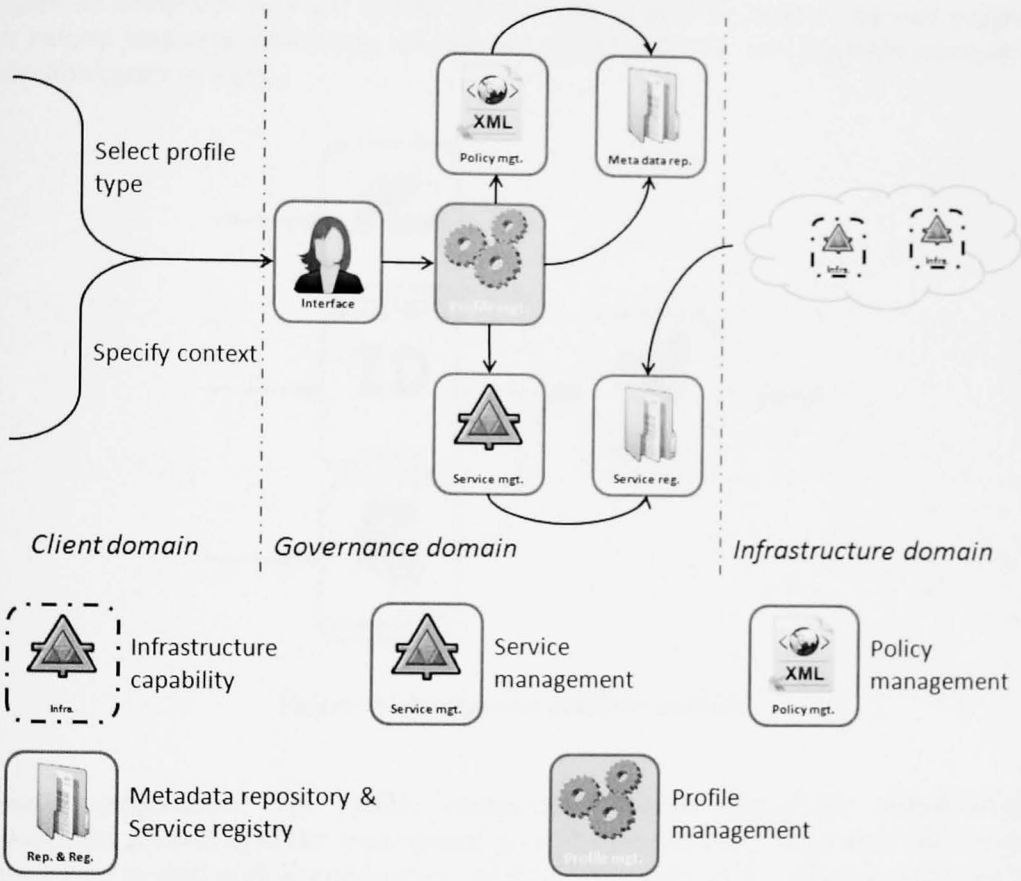


Figure 23. Architectural diagram of the governance framework – interactions at design time

In the following paragraphs the components related to profile management are described in more detail.

Profile initiator: The first step of the profile management is to translate the external request, received from the user interface, into the grammar used internally by the components described in section 3.3. This flexibility in terms of language is meant to bring an increase level of adaptability towards specific domains or types of users.

In order to allow for different grammars to be used, this component is based on the abstract factory pattern [67]. The factory proposes a set of different functions that are implemented by the instances and profile specific implementation for each case. Using services allows improving the flexibility of the system by permitting physical distribution as well as reuse of existing software.

The expected core functions of the profile initiator capability are illustrated in Figure 24.

«interface» **Profile initiator**
+CRUDProfile(in profile_representation, out acknowledgement) *+CRUDContext(in context_representation, out acknowledgement)* *+CRUDContextSelector(in context_selector_representation, out acknowledgement)*

Figure 24. Profile initiator interface description

All the functions on this component (Create, Read, Update, Delete profile, context and context selector) allow external users (e.g. human administrator, other governance system) to interact with the documents holding the governance data.

Figure 25 illustrates how the factory pattern can be used in order to provide support for natural language processing, an abstract profile grammar and a profile name or id selection query system.

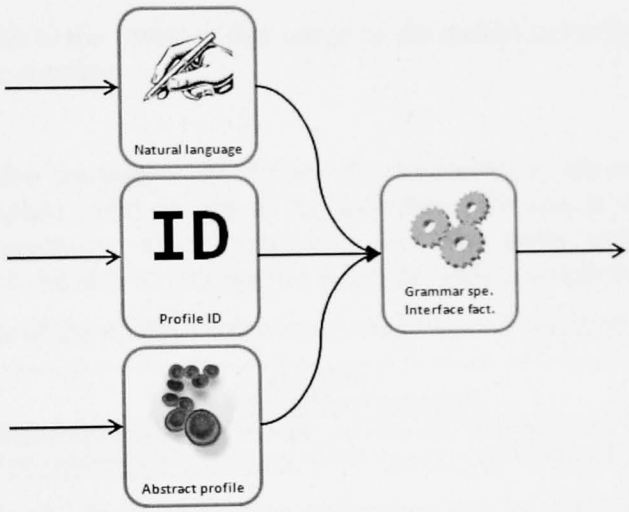


Figure 25. Management interface overview

Profile composition: The profile composition engine selects the infrastructure capabilities according to the requirement given by the initiator. The component works recursively to deal with dependencies. At different stages of the infrastructure profile life cycle, this engine will select infrastructures at different levels of abstraction, more abstract (e.g. category in the taxonomy) at the beginning and more concrete (e.g. Web service) towards the enactment stage.

The core functions of the profile composition capability are illustrated in Figure 26.

«interface» **Profile composition**
+composeAbstractProfile(in profile_reference, out abstract_profile_reference) *+composeConcreteProfile(in abstract_profile, out conctrete_profile_reference)* *+verifyProfile(in profile_reference, out acknowledgement)* *+bindProfile(in profile_reference, in context_reference, out acknowledgement)*

Figure 26. Profile composition interface description

The *compose abstract profile* function permits the definition of a complete and safe profile at the abstract level (e.g. without the concrete services). This process is further described in 3.5.1, 3.5.2 and 3.5.3.

The *compose concrete profile* function serves the same purpose as its abstract counterpart but with the selection of concrete services that enact the infrastructure capabilities required. This process is further described in 3.5.4 and 3.5.5. Given the sequential nature of the profile composition process it is possible that an abstract profile cannot be composed into a concrete one. This may occur when no service implementation is available to enact a service type described in the abstract process. In this case, mediation over the nature of the abstract process and whether it can be modified has to take place between the Business Capability owner and the Contextualisation Governance provider.

The *verify profile* function allows to go through the elements (e.g. dependencies, usage policy) forming the profile in order to insure the completeness and safety of execution.

Finally, *bind profile* is the function that connects the profile to the business capability whose exposure it manages.

Profile composition management: When the composition engine has deemed a profile to be complete, with no gap in the data flow, this one is sent to the profile composition management. Upon reception, this one starts creating the profile coordination plan to be able to manage the different infrastructures together.

The core functions of the profile composition capability are illustrated in Figure 27.

«interface» **Profile composition management**
+composeAbstractCoordinationPlan(in profile_reference, out coordination_plan_reference) *+composeConcreteCoordinationPlan(in profile_reference, out coordination_plan_reference)*

Figure 27. Profile composition management interface description

The *compose abstract coordination plan* builds the coordination plan at the abstract level.

Compose concrete coordination is its counterpart at the concrete level.

Profile life-cycle management: The life-cycle management supervises the profile life-cycle. It contacts the composition engine management when it's been informed that a profile is ready to be instantiated (by the management counterpart) and according to the up time agreement, requires the composition management to start the instantiation according to the coordination process built by the profile composition management. Additionally, the life-cycle management can request update(s) from the composition engine when necessary (e.g. service not adequate anymore, etc).

The core functions of the profile composition capability are illustrated in Figure 28.

«interface» **Profile life-cycle management**
+activateProfile(in profile_reference, out acknowledgement) *+deactivateProfile(in profile_reference, out acknowledgement)* *+enactProfile(in profile_reference, out deployed_profile_reference)*

Figure 28. Profile life-cycle management interface description

Activate profile allows to start and register the status of an exposition logic, *deactivate profile* enforces the opposite. Both of these functions should assume that the profile is ready to be used and expose the business capability when called.

The *enact profile* function is called in order to the finalise the instantiation and exposed the profile enhanced business capability.

Profile selector: When relevant, the profile selector chooses the relevant profile using a set of given parameters. Typically, the parameter is the context, however different cases can arise when the profile selector can request the life-cycle management to verify the profile and react accordingly. This separation of concern between the profile selector and the life cycle management and the fact that the profile selector advises its life-cycle counterpart, allows for different selection mechanisms that can specialised according to data and environment.

For instance, when an event specifying that an used policy has been changed in an infrastructure capability is triggered, the selector can then decide what profile it should ask the life-cycle management to review first. This selection could be based on how much resource (e.g. network, number of infrastructures involved) the different active profiles use, for performance reason, or what is the SLA with the different users involved.

The core functions of the profile composition capability are illustrated in Figure 29.

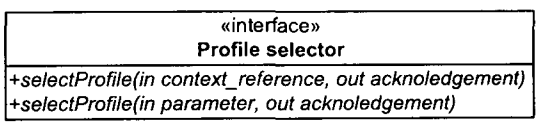


Figure 29. Profile selector interface description

Both *select profile* function of the profile selector infrastructure allow deciding on what particular profile is relevant. In the top example, the context is used to make the selection.

3.4.2.1. Service management requirements

Service management infrastructures are less commonly encountered pieces of infrastructure than their policy management counterparts. In this regard, the author felt it necessary to define the key requirements and expectations from the exposition governance point of view.

As underlined in section 3.2.1 in the definition of the concepts, capabilities are expected to be manageable when relevant (i.e. when it is sensible for configuration, performance or security reason).

The capability management component should be comprised of a) a capability instance factory, b) a capability selector factory and c) a capability management factory. These three elements respect the abstract software factory pattern in order to allow for a more flexible support of additional types of capabilities.

The management and instance factory elements are used to configure (e.g. setup with context aware policy) and/or replicate a capability (e.g. copy service onto another server). The latter is particularly useful for infrastructure capabilities that may have to deal with heavy workloads (e.g. included in many or demanding profile instances) or different requirements (e.g. a Service Level Agreement could necessitate high

availability). The management interface takes advantage of a management layer of a service, typically implemented using WS-DM [68], in order to configure the said service. This is also useful for infrastructures such as security service which require some sort of interaction and configuration before they can be used.

The selection factory allows selecting capabilities according to specific characteristics and grammars.

In Figure 30, the three functions previously introduced are illustrated along two examples of factories for each. The capability instance factory is implemented in two different manners, one making use of the Apache Muse [69] software, the other one indicating that capabilities provided through it cannot be instantiated. The capability selector instances drawn suggest that a QoS based and a user experience based selections are available. Finally, the manager factory illustrates the choices between a WS-DM and custom based remote capability management.

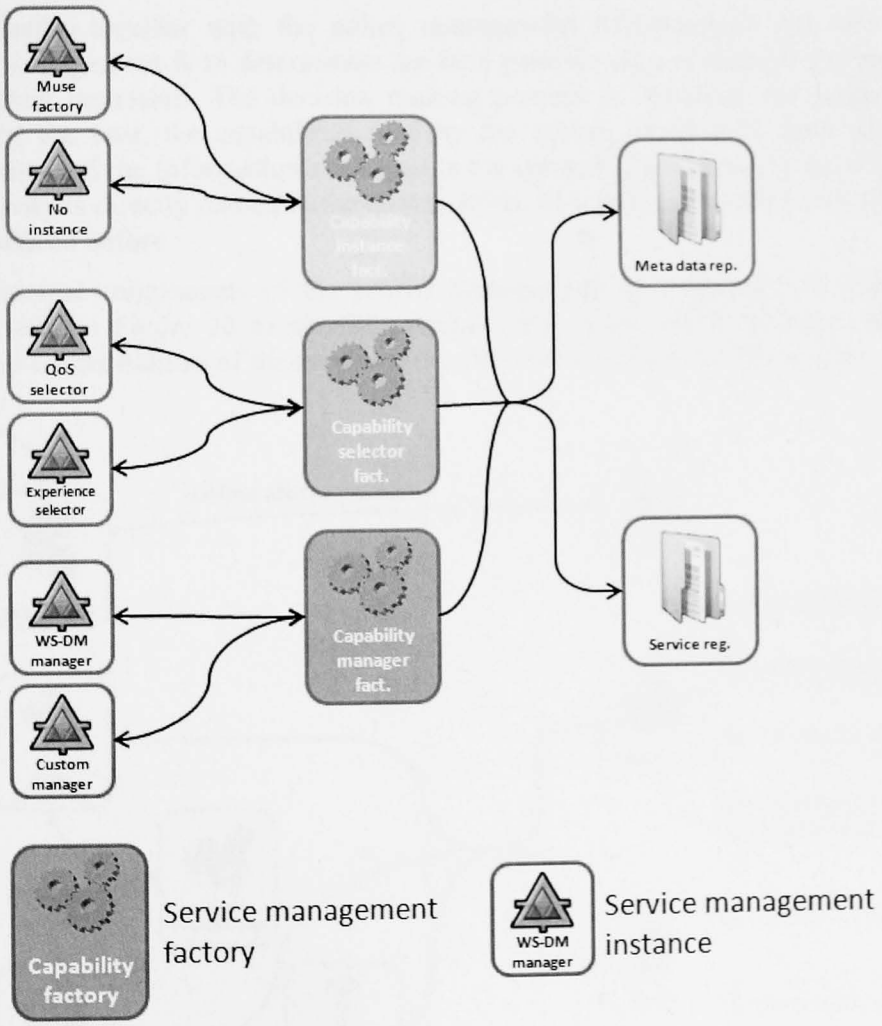


Figure 30. Service management overview

3.4.3. Summary of the components descriptions

Profile management is divided into two main logical domains, the profile consistency management and the profile life-cycle management.

These domains can be respectively split in several steps: defining the infrastructure capabilities, the policy templates, the service dependencies and the information flow for the first one and defining the profile management process as well as publishing the infrastructure profile for the second.

The first aim of the profile management is to manage the life-cycle of the profiles. This consists in allowing the profile to be defined, instantiated, maintained accessible, updated and deleted.

In addition to the profile instance's life-cycle management, an important task that is attributed to the profile manager is to handle the adaptability of the profile instance. This feature is needed to allow for an efficient interchange between the infrastructures used.

Additionally, together with the policy management infrastructure, the role of the profile management is to determinate the best possible way to achieve the profile in the context requested. The decision making process is based on the requirements given by the user, the capabilities held by the system along with their associated constraints and the information contained in the context. The degree of automation of this activity is directly related to the quality of the data held in the other core elements as introduced before.

The principal components of the profile management are illustrated in Figure 31. Please refer to Figure 23 to see the external connections. On this figure, the main expected functionalities of the service and policy management are illustrated.

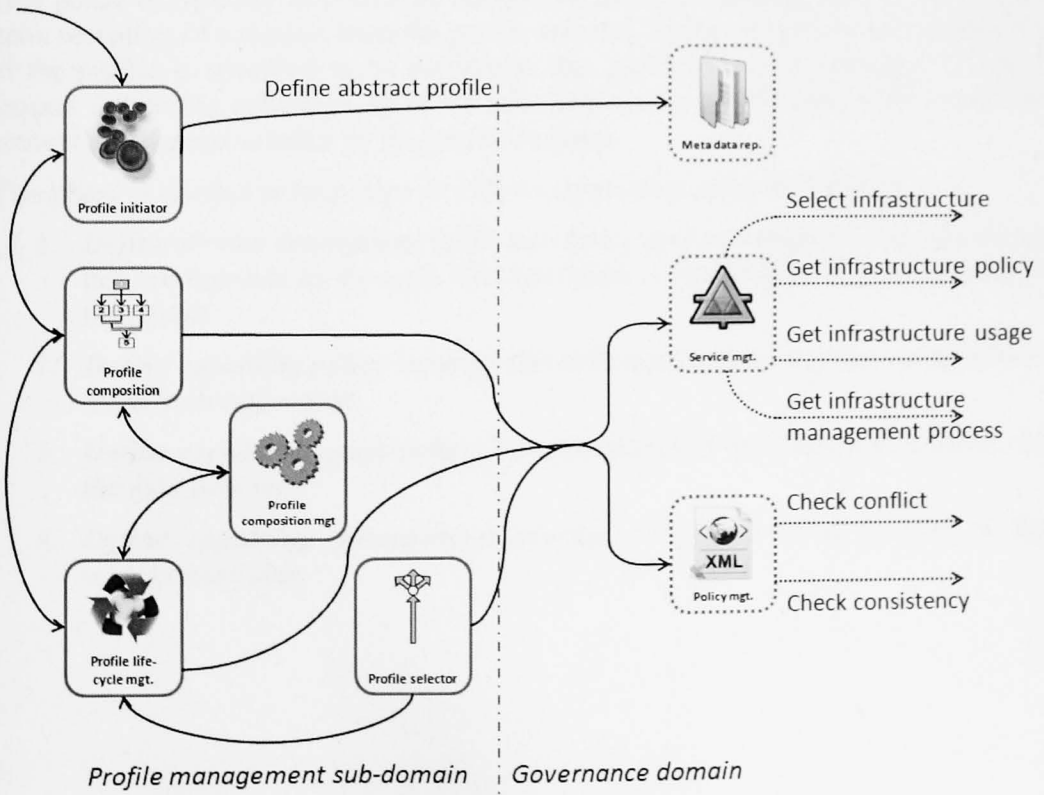


Figure 31. Profile management overview

This distribution of roles and the potential hierarchical anatomy of the model is meant to reflect the unreliable nature of distributed systems where components, here infrastructure capabilities, are provided by third parties and may not be always available or present in the same state.

In this section, the author described the infrastructure capabilities forming the exposure governance architecture. The roles of these infrastructures and their communication interfaces have been defined. In the next section, the governance processes that allow the infrastructures to work together are explained.

3.5. Processes anatomy

In the previous section, the author described the infrastructure capabilities forming the exposure governance architecture. In this section, the governance processes that allow the infrastructures to work together are explained.

The steps that lead to the creation of a safe and sound infrastructure profile are listed in the following paragraphs along with the components they involve. Concrete examples illustrating all the steps are shown in chapter 4 on implementation.

3.5.1. Define infrastructure capability

The first phase of the life-cycle is to gather all the information available about each infrastructure type listed in the profile formed by the profile initiator.

This task is enacted by the profile composition infrastructure capability and only targets infrastructures that are explicitly mentioned in the profile description document.

This phase is typically activated by the profile life-cycle management infrastructure upon reception of a request from the profile selector. In the case where the availability of the profile is specified to be permanent, the profile life-cycle manager is sent a request for profile activation upon the submission of a profile and of its associated context and context selector by the profile initiator.

This phase is divided in four steps as follows (steps illustrated in Figure 32):

1. ***Define service description***: in the taxonomy used to categorise infrastructures, this corresponds to the type. The operation is repeated for each capability in the profile.
2. ***Define capability policy scheme***: this information is held in the constraints of the infrastructure type.
3. ***Define capability usage policy***: this information is held in the constraints for the data flow part.
4. ***Define capability management process***: this information is held in the management plan.

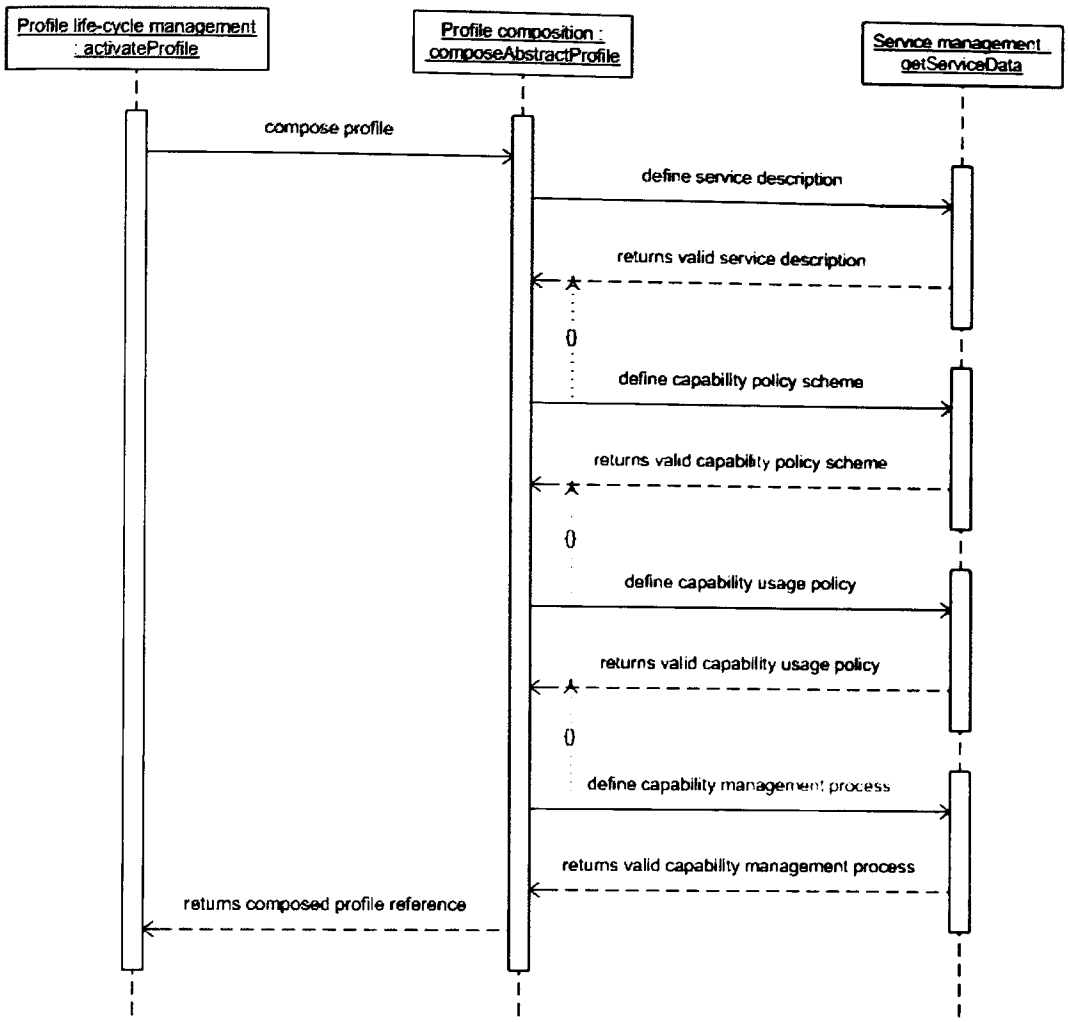


Figure 32. UML sequence diagram of the define infrastructure capability governance process

All this data is supplied within the profile description taxonomy depicted in section 3.3.1. This taxonomy is strictly based on the requirements of this research. The author takes into account that more advanced description techniques such as ontology based ones could be used. However complex the description is, these four basic steps remain and would be subdivided in more technology specific ones.

It is assumed that the infrastructure providers have registered their infrastructures correctly.

3.5.2. Define policy templates

The second phase aims at getting more information about each infrastructure's type policy templates and how to manage them. Amongst this information, the meta-data transformations specify how to translate data from different grammars and policy types into this particular type.

This step is also performed by the policy composition capability when composing a profile (stages illustrated in Figure 33).

5. ***Select capability***: the operation is repeated for each capability in the profile.

6. ***Define policy template***: the policy templates are held as constraints in the capability descriptions when relevant.
7. ***Define domain of meta-data transformations (i/o meta-data)***: this step looks into the data flow of this activity in the profile description and define if a transformation is required.
8. ***Define policy management processes***: when a transformation is required, the profile composition looks for such transformation in the infrastructure repository. This is achieved by looking at the data flows and usage of the infrastructures registered as translators. Upon selection of the appropriate translation infrastructure, this one is inserted accordingly in the profile.

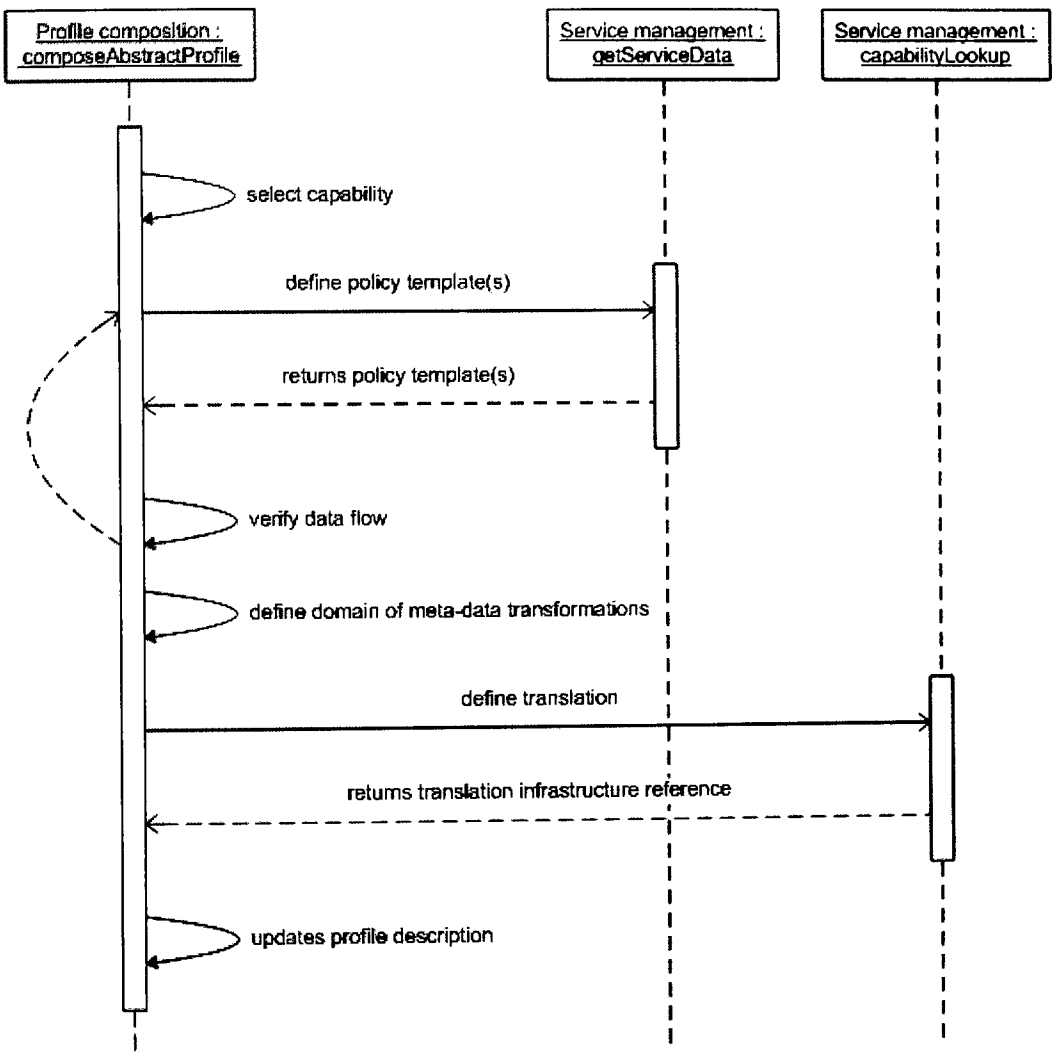


Figure 33. UML sequence diagram of the define policy templates governance process

In a similar manner as in the previous stage on infrastructure type definition, this data is provided by the profile taxonomy when infrastructures are registered. Based on the level of complexity of the semantics used in an implementation, transformation strategies could be generated by the policy management capability. However, this level of automation is both potentially complex and resource demanding and the

author assumes that it is more realistic for the translations to be provided as infrastructure capabilities.

3.5.3. Define infrastructure capability dependencies

Once all the infrastructure types provided in the profile request have been defined and their policies identified, the profile composition engine verifies if the profile is safe and complete. In order to achieve this, this capability works recursively and identifies potential gaps in the data flows and infrastructure usages. The capability management process and usage policy are parsed and check for consistency against the other infrastructure capabilities present in the profile at this stage. For instance, if an access control capability type is present at this stage, depending on its anatomy, it may require an external identity management infrastructure. In this case, this relationship would be present in the usage or management policies of the specific access control capability category. The stages forming this step are illustrated in Figure 34.

9. ***Select infrastructure capabilities***: the operation is repeated for each capability in the profile.
10. ***Define operation bindings***: at this stage, the data flow is checked, the data should pass from a capability's operation into another operation without any gap. If a gap is detected, the missing operation is looked after and the profile composition capability goes through steps 3.5.1 and 3.5.2.
11. ***Define capability invocation pattern***: for each operation registered in the profile, when relevant, an invocation pattern is available and defines what potential other operation would need to be invoked before it. This type of data is stored as a constraint when the infrastructure is registered and defines it behaviour. If a dependency is detected, it is looked after and the profile composition capability goes through steps 3.5.1 and 3.5.2.
12. ***Validate capability dependencies***: for each step in the profile this stage checks if both steps 10 and 11 defined above are fulfilled.

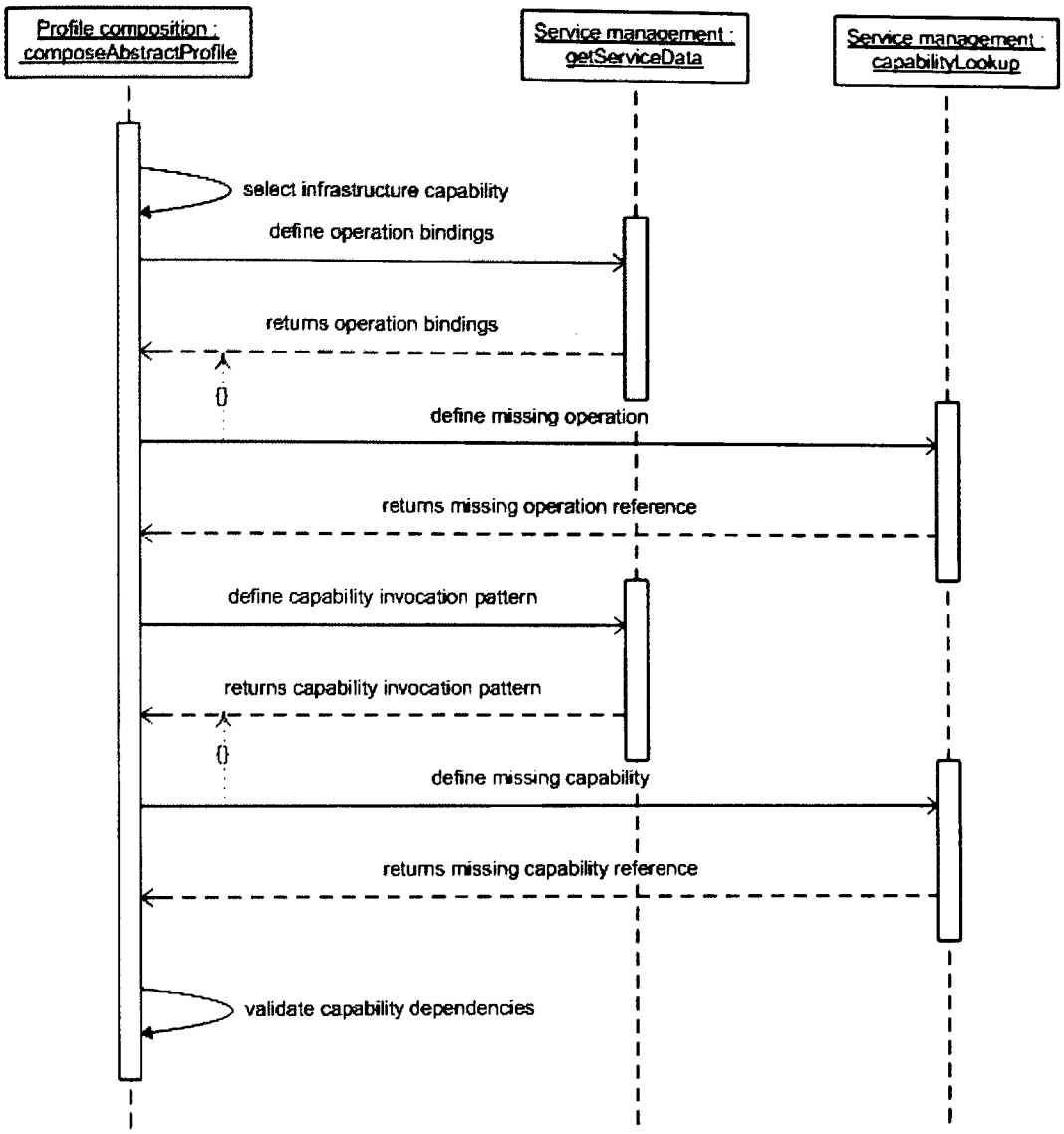


Figure 34. UML sequence diagram of the define service dependencies governance process

Following step twelve, the profile is stored and deemed safe and complete at an abstract level by the governance.

3.5.4. Define information flow

At this stage specific services enacting the infrastructure capability categories listed in the profile are selected. Additionally, the manner in which these services are aggregated and the transformations needed in order for the relevant data to pass from one to another are confirmed and validated.

13. ***Select infrastructure capabilities***: the concrete service implementing the infrastructure required in the profile is sought.
14. ***Define policy meta-data transformations***: the meta-data transformation specific to the service are resolved in the same manner as defined in section 3.5.2. It is noticeable here that according to how the semantics describing the

profile and the potential infrastructure forming it are defined, this stage could be useless.

15. ***Validate policy dependencies***: in the same manner as the previous stage for meta-data transformation, this stage looks into dependency validation specific to the service as 3.5.3 at the infrastructure category level.

By step fifteen, the profile is a set of defined and ordered infrastructure capabilities along with their specific management plans.

3.5.5. Define profile management processes

With a complete list of what services will allow enacting the profile, the profile composition management capability attempts to order the management processes and define an implementation strategy, named coordination process, which orders the operations needed in order to create the profile instance.

16. ***Select service management processes***: for each service, the profile composition management infrastructure gather the management plan.
17. ***Select policy management processes***: in a system where policies are managed through defined and enforceable management processes, the later are also obtained.
18. ***Define coordination process***: the different services management plans are correlated and duplicates when they exist are removed.
19. ***Bind management processes & coordination process***: the coordination plan document is assembled.
20. ***Validate dependencies***: finally, the coordination plan is checked for completeness.

Upon completion of the coordination process, the different management processes are attached to it at their relevant positions. Following this, the profile instance is sent back to the composition engine for validation.

Step twenty signifies that the profile is complete at the concrete level; it contains information related to what services will be used, how to configure them and how to pass data from one operation to the next. In this state, it only lacks contextual data to render it operational as required.

These steps are summarised in Figure 35, where the transformation of profile into profile instance with specific services fulfilling the infrastructure capabilities can be visualised.

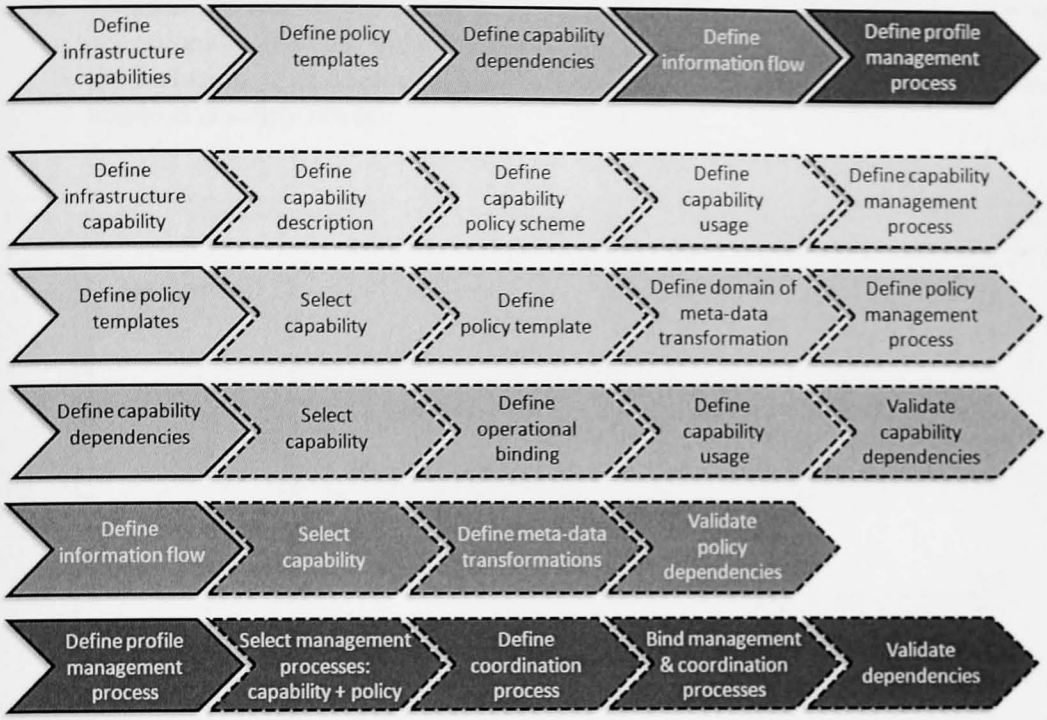


Figure 35. Infrastructure profile creation

Once a profile is complete at the concrete level, the next stage is the instantiation. Finally, the business capability can be enhanced with the instantiated profile and exposed. This separation of logic insures that a profile can be reused with different services. Additionally, the enhanced service is only used in the particular context that is relevant to its specific users.

Once a profile has been validated (c.f. steps from one to twenty), it can be used in order to allow the exposition of a business capability. The following steps define this process.

3.5.6. Select infrastructure profile instance

The first stage of the business capability exposure sees the profile selector component choosing the relevant profile instance. The selector then contacts the profile life-cycle management, which upon receiving the request of business capability exposure, requires the profile composition engine to define how the capability and profile instance work together. The stages forming this step are illustrated in Figure 36.

1. ***Discover infrastructure profiles***: this step makes use of the context and context selector to find the relevant profile(s) associated with the targeted exposure.
2. ***Select infrastructure profile(s)***: the relevant profile is selected.
3. ***Define bindings to business capability***: the goal of this step is to define how the business capability will be linked to the instantiated profile. Concretely, the data flow between the business capability and the profile needs to be completed.

4. ***Validate service dependencies***: The previous step that looked into how to operationally link the business capability to the profile may have necessitate the inclusion of an additional set of capabilities. This step insures that this new addition is safely made.

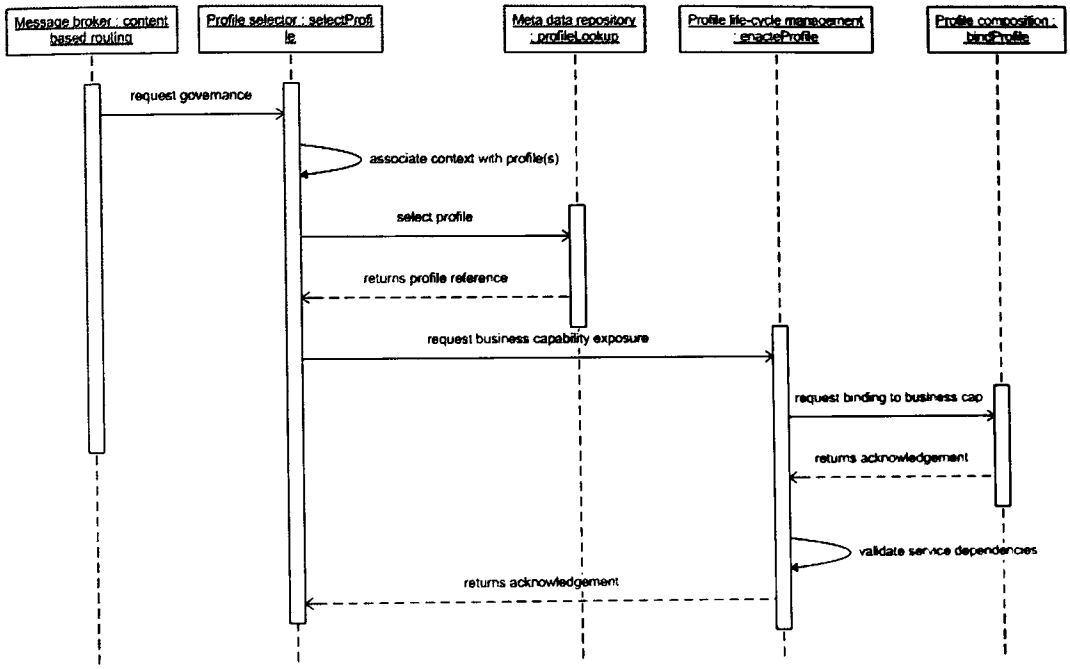


Figure 36. UML sequence diagram of the select infrastructure profile instance governance process

Due to the relatively unreliable nature of such a large scale distributed system where different entities provide components it is necessary to insure that the profile is still complete and can still be instantiated.

3.5.7. Define service-specific policies

Steps five to seven allow the profile composition engine to refine the different policies with the business capability data when relevant. This is the first step towards the contextualisation of the profile.

5. ***Select infrastructure capability***: This step is repeated for each infrastructure in the profile.
6. ***Refine policy template***: Existing templates are instantiated with contextual data found in the context document.
7. ***Update service policies***: The previously filled template is then pushed to the service through its management interface when relevant.

3.5.8. Define information flow

The same actions are then taken with the transformation strategies in the next three steps. This is made to insure that the transformations are adequate and match the filled templates.

8. ***Select infrastructure capabilities***: This step is repeated for each infrastructure in the profile. For performance reason, an implementation could here decide to only take capabilities where translations are required into account.
9. ***Refine policy meta-data transformations***: The existing transformation template is here refined with context specific transformation logic.
10. ***Validate policy dependencies***: Once again, the profile is checked for completeness and safety. For performance reason, an implementation can decide to either skip this step or target the specific infrastructures that the previous step 9 looked into.

3.5.9. Define service exposure management processes

This phase defines how the profile enhanced business capability is going to be exposed. The goal is to identify the chronological list of actions that need to take place in order for the service exposure to take place in a safe manner.

11. ***Select profile management processes***: All the management processes are here selected again.
12. ***Select business capability management processes***: the same is done for the management processes relevant to the business capability.
13. ***Define coordination process***: The coordination is validated and completed when a gap in the data flow in found.
14. ***Bind management processes & coordination process***: At this stage, the business capability management processes are included in the profile coordination.
15. ***Validate dependencies***: The latest stage may have introduced new infrastructures or translations, the dependencies are therefore validated along with the overall data flow of the exposition logic management process.

3.5.10. Publish service instance

Finally, the coordination plan is executed, the different infrastructure configured and their bindings is done. Additionally, policy stores are updated and the business capability, now enhanced with the appropriate NFPs, is published.

16. ***Update capability policy stores***: The goal of this stage is to update policy stores with contextualised versions of the templates when relevant.
17. ***Update infrastructure bindings***: During this phase, the specific bindings for each service enacting an infrastructure capability is defined and updated.
18. ***Expose profile enhanced business capability to service endpoint***: The instantiated profile and the business capability are exposed as a service.
19. ***Publish service***: Finally, the enhanced business capability exposed though the instantiated profile is advertised as relevant and awaiting traffic.

The process of refining the policies with the contextual data and the exposition of the governed business capability are illustrated end the profile management. These steps that allow contextualising the profile and publishing the enhanced business capability are summarised in Figure 37.

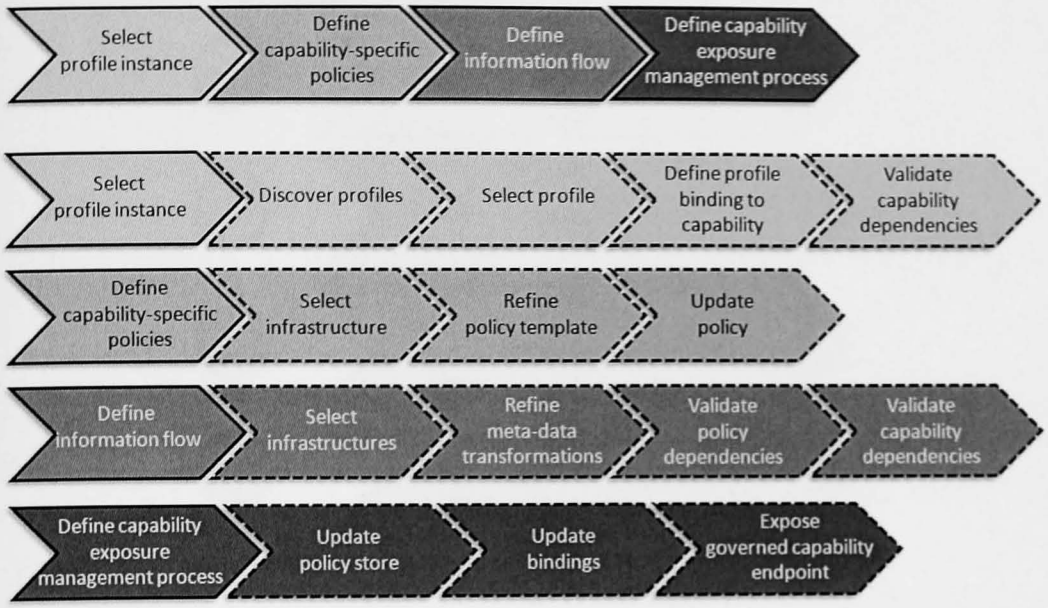


Figure 37. Business capability exposure

3.5.11. Summary of processes anatomy

The four main stages of the governance life-cycle are illustrated in Figure 38. In stage one, the profile is prepared and made ready to use in a specific context. At stage two, the business capability and exposition context requirements and conditions are inspected and prepared. Later on, at stage three the profile is refined and adapted to a specific exposition context. Finally, the instantiated profile and business capability are bound together and exposed to users.

The proposed approach divides the governance process in logical steps which are enacted by specialised components. This permits increasing the flexibility and reusability of the exposition logic as well as supporting the high level of distribution of the infrastructure as a service paradigm.

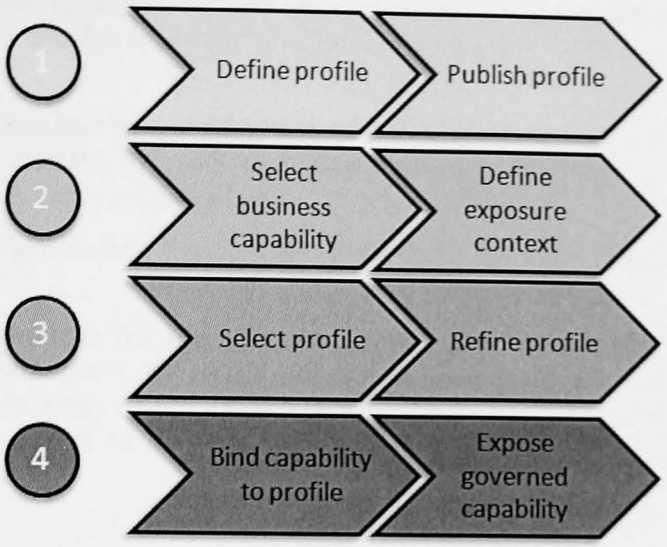


Figure 38. Governance life-cycle

3.6. *Usage patterns*

In this section, the different situations the architecture described in the previous sections can be used in are presented. Segregation is made between the deployment options for the deployment of the infrastructures forming the governance middleware and the usage of the governance itself.

3.6.1. Infrastructure capabilities deployment patterns

The core capabilities deployment patterns define how these capabilities are physically deployed. This impacts the governance middleware in terms of performance, ease of maintenance and accessibility for the capability providers.

Distributed pattern: In this pattern, the infrastructures are distributed over the network. This pattern is used in the Virtual music store scenario presented in section 4.2.1.

Hub and spoke pattern: In order to reduce potential network latency or insure that the physical location of the governance middleware is suitable for the workload it is meant to handle, it is possible to deploy the core infrastructures at the same location.

Standalone pattern: The stand alone pattern encompasses that all infrastructures potentially used in the governance are at the same location. This pattern can be required when the context requires that the traffic network be minimal.

These patterns can be applied in an assorted manner. For instance, an environment where the security related infrastructures are going to know of be very demanding could take advantage of having only these specific capabilities deployed in a standalone or hub and spoke pattern, with the rest of the infrastructures distributed over the network.

3.6.2. Governance middleware deployment patterns

Governance patterns identify who has ownership of the governance middleware configuration, the responsibility to monitor events and transaction characteristics and

therefore who defines and authorizes changes to the existing middleware. In addition, this defines who decides when and in what conditions new infrastructure services can be added.

The patterns described in the following paragraphs are derived from studying how other pieces of middleware such as brokers, message queues or Enterprise Service Buses (ESBs) are deployed and used.

Approaches to governance middleware deployment patterns range from internal to federated:

Internal governance: All the interactions requiring governance may lie wholly within the organisation's perimeter and not be accessible from outside of it. For example, a single entity may span over several sites or country where each area uses different software with different transport protocols or interface styles.

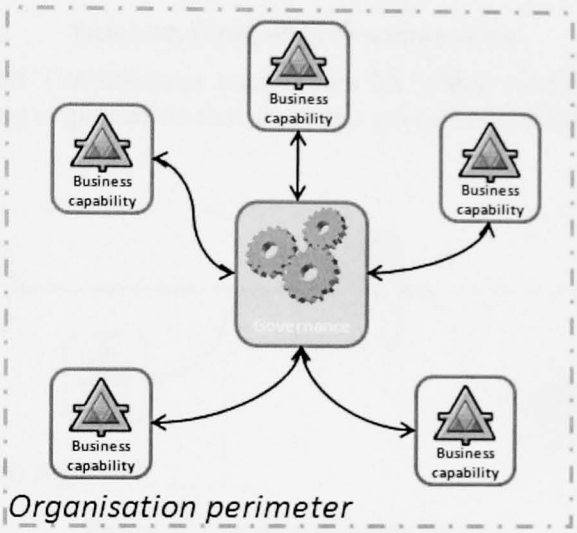


Figure 39. Internal governance pattern

Perimeter governance: All business capabilities lying within one organisation are made accessible to other organisations through the governance middleware. This situation may arise when an organisation wants to leverage on the governance middleware to expose its business capabilities to potential partners in a more flexible manner than internally. This pattern is the one used in the Virtual Music Store scenario in section 4.2.

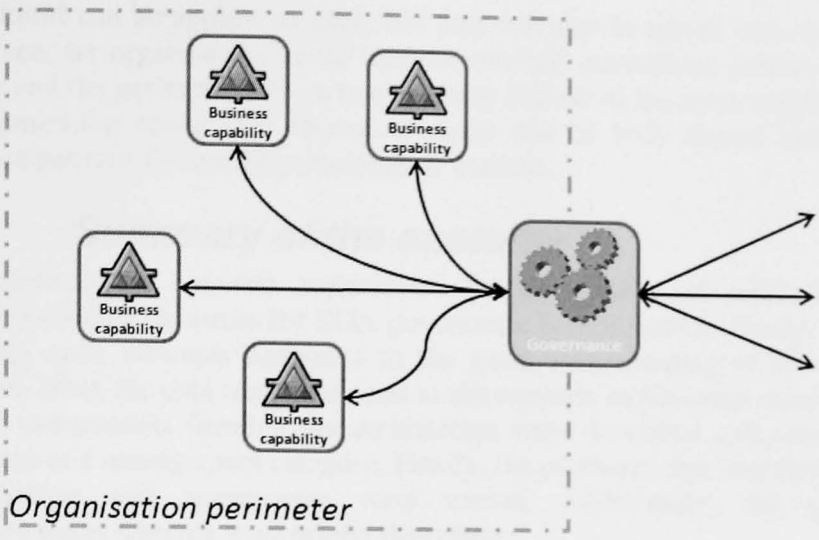


Figure 40. Perimeter governance pattern

Shared governance: The business capabilities for which interactions are governed belong to cooperating organisation that share the governance middleware.

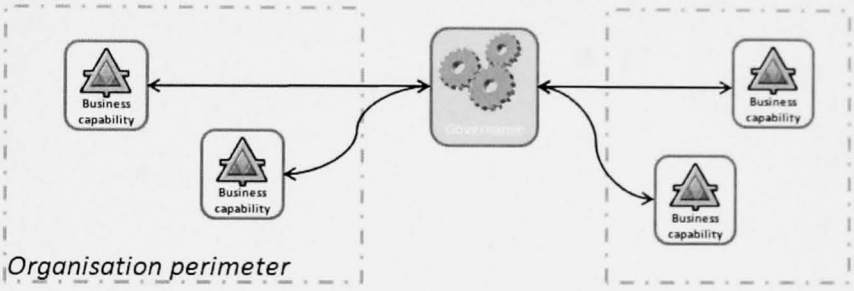


Figure 41. Shared governance pattern

Federated governance: In this pattern the organisations governance middleware are federated. The interactions between business capabilities inside of this federation can take advantage of the collaboration between the governance middleware.

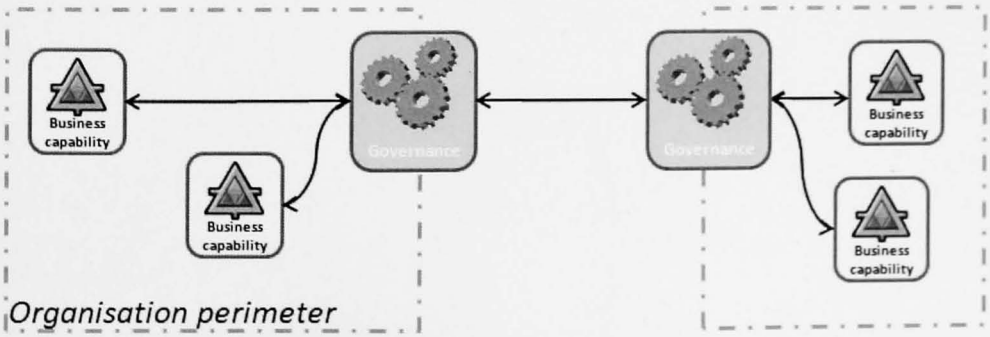


Figure 42. Federated governance pattern

These patterns can be applied as such, but they can also be mixed with one another. For instance, an organisation could use the internal governance pattern within its perimeter and the perimeter pattern to expose by default its business capabilities. The same organisation could still choose to make use of both shared and federated governance patterns for specific situations or partners.

3.7. Summary of the anatomy

In this section, the concrete architecture for governance of safe and flexible contextualisation of resources for SOA governance was presented. Firstly, the author defined the main concepts necessary to the good understanding of the anatomy's description. Next, the data structures used in the concrete architecture were presented. Then, the components forming the architecture were described and categorised in either a core or a management category. Finally, the processes that link these different data structures and components were shown. Additionally, the governance architecture usage patterns were briefly described.

In the chapter 4, the author will describe how the architecture presented in this chapter was implemented in a concrete example of virtual music store as presented in chapter 1.1.1.

4. Implementation of resource contextualisation governance

This chapter describes the implementation of the concrete architecture for resource contextualisation governance in SOA defined in Chapter 3. The implementation is used in the context of a Virtualised Music Store as presented in 1.1.1. Finally, a series of tests are performed on the functional validity of the implementation and the results are discussed.

The use of the resource exposure governance is essentially seamless. The resource owner or the governance middleware administrator, depending on how the responsibilities are shared, is responsible for inspecting when governance requests (e.g. profiles) are sensible or can be a threat to the governed resource's integrity (e.g. conflicting business goal, inadequate security). Additionally, the only significant supplementary effort, in comparison of not using any NFP support for the governed resource, is the provision of adequately described infrastructure capabilities. This is alleviated by the fact that infrastructure providers will want to expose their services in order, for instance, to generate revenue.

The objective of this implementation is to implement a Virtual Music Store connected to content providers. In this scenario, the role of the governance middleware is to allow seamlessly connect content providers that are using different security settings.

4.1. Governance middleware

This section provides a brief overview of relevant aspects of Web Service development with the Java language [70] and the Apache Axis2 [71] SOAP [72] and WSDL [33] engine that has been chosen as the demonstrator platform for implementation of resource contextualisation middleware services. Nevertheless, as is apparent from the following discussion, the general approach to development of the middleware is neither Axis2 nor Java specific.

The Apache Axis2 engine deployed on an Apache Tomcat [73] application server was chosen because it provides straightforward mechanisms for developing, deploying and using Web Services. It is also a widely used platform with which to experiment.

Each core infrastructure capability was developed, deployed and then registered in the registry using the same registration data as the none core infrastructure capabilities described further (c.f. Table 5 for example of infrastructure registration data). This approach was used to allow sharing the same code to discover and enact all the capabilities.

The registry itself was provided by an instance of the eXist-db Open Source Native XML Database [74] which natively exposes the data it holds through a SOAP interface provided by a servlet [75] mode based on Apache Axis. In addition, this web enabled XML database can be queried using XQuery 1.0 [76] or XPath 2.0 [77]. All these elements permit an efficient and straightforward manipulation of XML data over the network.

Finally, the message interceptor was provided by Apache Synapse [78]. This choice was driven by the relatively well documented usage of this project, its undemanding development and deployment requirements. In addition, the implementation required

a relatively efficient message broker as all the messages in the scenario were going to pass through it.

4.2. Virtual music store scenario

4.2.1. Description

This section presents the virtual music store (VMS) scenario introduced in section 1.1.1. This presentation is a practical illustration of the concrete architecture proposed in chapter 3. The security governance gateway presented intercepts messages addressed to a VMS, enforces security policies and integrates the defined security capabilities (e.g. identity management, authorization service) in order to secure the VMS communications with its content providers.

With this scenario, the author aims to demonstrate that the concrete architecture presented in section 3 can be applied to provide complex NFPs such as access control, identity management and policy enforcement. Indeed, these security-related NFPs necessitate configuration of the capabilities themselves, but also between them. For instance, in a certain federation anatomy, a Secure Token Service (STS) [54] providing identity management will not only need to be configured, but will also require a process of trust establishment with the different STSs it is meant to interact with. The situation can be different in different federation anatomies, thus necessitating a tight management of how to control the assembly of capabilities in a safe manner.

4.2.2. Partners and roles

In this scenario, the aggregated services are virtual music stores serving specialised markets or communities of interest. The basic content providers include copyright owners of musical recordings or their representatives who make these recordings available online. The virtual music store reaches agreement with content providers enabling it to act as re-seller of bundles of recordings from their catalogues. The VMS is a VO consisting of the music store operator that communicates to content providers through a security governance gateway whose security services are provided by external security providers through the SaaS (Security as a Service) paradigm.

The end customer of the VMS will be a member of the public. What they will see is a web site where they will be able to search for tracks. The user interface of the VMS setup for the implementation is presented in Figure 43.

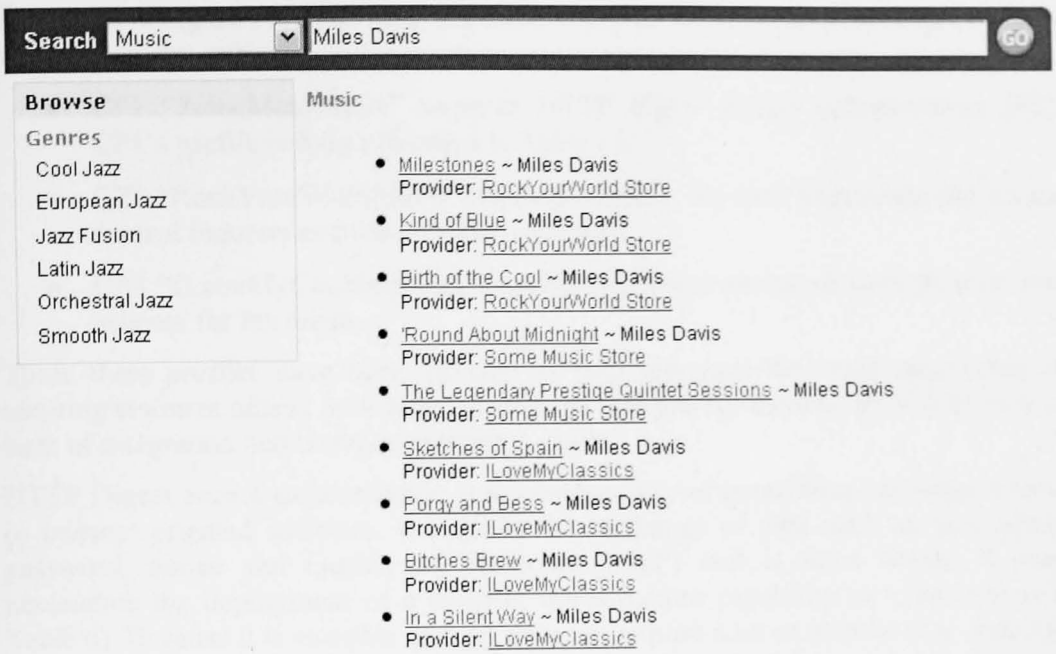


Figure 43. Virtual Music Store - user interface

As shown in Figure 44, in the music store scenario, the main categories of partners are:

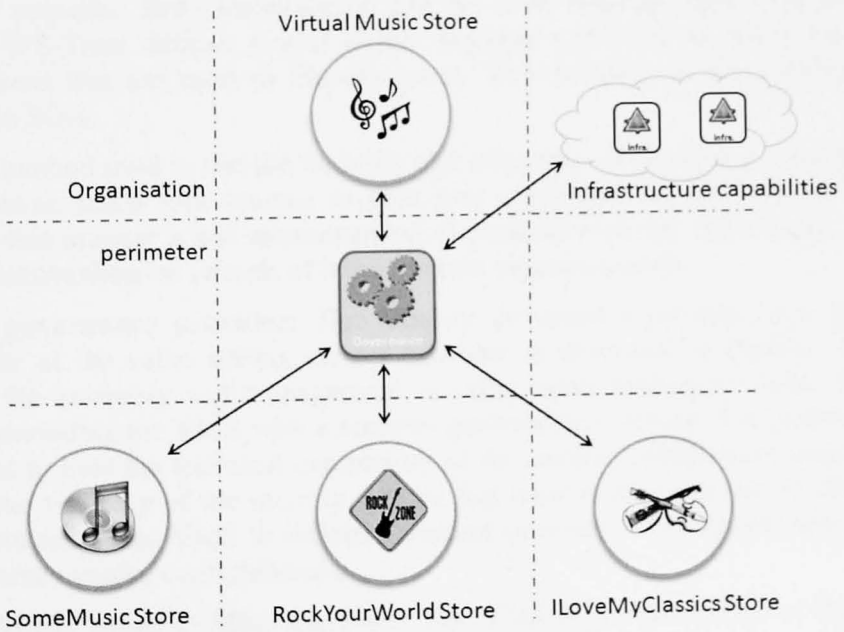


Figure 44. Virtual music store scenario with security governance gateway

Content Provider (CP): The CP is a stakeholder of the role music content provider as described in chapter 1.1.1. It is a specialist content provider (e.g. record labels or other copyright owners). Three different content providers take part in this particular scenario. Each one of them is using a different security profile that it shares with the VMS operator through the security governance gateway. The security profiles express their requirements in term of security in a set of XML files using the formalised way

illustrated in Figure 19. Following is the list of the content providers with an overview of their security settings:

- CP1 "SomeMusicStore" requires HTTP digest access authentication [80]. CP1's profile is fully presented in Table 11.
- CP2 "RockYourWorldStore" requires SecPAL for both expressing the access control requests and identify the requesters.
- CP3 "ILoveMyClassicsStore" requires login and password with its own xml schema for the token.

These three profiles have been selected as they represent the main approaches to securing resource access in distributed systems and present three level of difficulty in term of integration and contextualisation support.

HTTP Digest access authentication is a common way of controlling resources access in internet oriented software. It requires the exchange of data such as user name, password, nonce and Quality of Protection (QoP) and is none trivial. It does necessitate the deployment of a specific infrastructure capability to support it (c.f. Table 6). Because it is complex and yet does not require a lot of content (e.g. message payload) to demonstrate, this profile will be used in the rest of this section to illustrate the scenario.

SecPAL is used for expressing decentralised authorization policies. It is similar to XACML in that it uses a set of subject, resource and action to define access control rules and requests. Both technologies can be used together with WS-Trust [82] standard. WS-Trust defines a trust model together with how to issue, renew and cancel tokens that are used to identify users. This method for identifying user is common in SOA.

The third method used to test the usability and efficiency of the architecture proposed uses a custom made xml schema to pass user credentials in the message header. Although this manner is not recommended, it presents with the opportunity to test a simple contextualisation in term of infrastructure support needed.

Security governance provider: The security governance provider is a particular stakeholder of the value adding service provider as described in chapter 1.1.1 that provides the assembly and management of other value adding services. This role involves providing the VMS with a security governance gateway. The purpose of the gateway is to hide the technical complexity of the security middleware involved and enhance the visibility of the security actions that need to be enforced. In addition, it allows connecting the VMS to different content providers that expose their catalogs using diverse security configurations.

Virtual Music Store (VMS) operator: The VMS is a stakeholder of the virtual music store operator as described in chapter 1.1.1. Ultimately, the administrator of the VMS is responsible for selecting its different CPs partners.

Security as a Service (SaaS) provider: The SaaS provider is a stakeholder of the value adding service provider as described in chapter 1.1.1. This is a third party entrusted with providing a security service such as identity management, access control or audit. These services are managed by the security governance and allow the content providers and music store operator to leverage on the security governance gateway to enhance their interoperability while keeping the necessary level of security. In this experiment, these services, shown in Figure 44 are registered and

described using the same taxonomy as the one used in the security profile. Although these providers are not known from the VMS and CPs, for the sake of this experiment, it is assumed that these actors trust the security governance provider who in turn trusts the security service providers. The author believes that in a production environment this would be overcome by providing solutions such as QoS [83] or dynamic trust based [84] selection of services.

4.2.3. SOA security governance

In the music store scenario, an instance of the governance middleware named security governance is designed.

Figure 45 illustrates how security capabilities can be composed into a security management solution during service operation. In this figure, the contextual governance discussed throughout this document is part of the *SOA Security Governance Layer* element. This solution intercepts messages addressed to a set of resources, enforces security policies and integrates the defined core security capabilities, such as identity brokerage, authorisation services, or other managed security services in order to secure the resources' communications.

It is noticeable that defining an independent enforcement layer (c.f. SOA Security Governance Layer on Figure 45) connected through messaging middleware to the security capabilities that are governed by another dedicated layer implies an additional level of complexity. However this trade-off is compensated by the gains in flexibility, dynamicity and by allowing auditing of the results of each action separately if necessary.

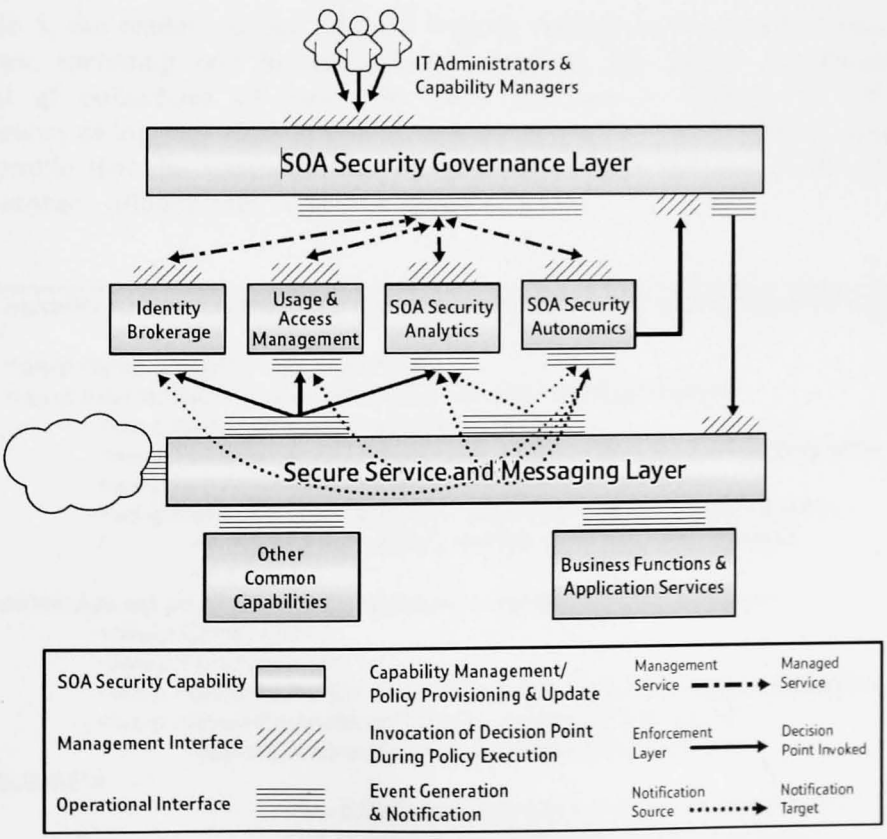


Figure 45. SOA security governance overview [81]

4.2.4. Security governance life-cycle

This section describes the life-cycle of the virtual music store's security. It starts with the governance gateway configuration then continues with the initial agreements and discovery of the potential partners which leads to the formation of a new VO. This is followed by the security profile management that allows for the verification and instantiation of the security profile given by content providers. Finally, the adaptability faculty of the virtual store infrastructure is introduced.

Security governance gateway configuration. Prior to the VO formation and at any time during the following steps, the security governance gateway is meant to reach agreements with security service providers to use their services within a particular context (e.g. VO) and/or with specific requirements (e.g. QoS) in order to be able to provide security for the different content providers used by the VMS. In a production environment, these agreements could be reached through Quality of Service (QoS) based selection, commercial agreement after human or electronic negotiation. In our case, different services are added, included relevant ones.

In the experiment presented, three infrastructure capabilities are registered as such in the service repository. The infrastructure are: two security related services including a HTTP digest authentication and a SecPAL Authorization as well as a logging service.

The logging service keeps a track of the text sent to it, in a preconfigured collection. This logging infrastructure's registration data is shown in Table 5. The XML schema defining how capabilities can be registered is shown in Table 15. In this prototype, the same schema is used for registering the capabilities and expressing the profiles.

In Table 5, the reader can see that the logging infrastructure capability posses two interfaces, including one for management purpose that allows the creation and removal of collections of logs. The other interface is defined for the actual functionality of logging. Both interfaces link the related WSDLs and their locations. It is noticeable that the functionalities are ordered through the use of the constraint "BeforeorAfter", allowing for automatic usage.

```
<ws-p:Capability ID="ac06e340-6763-11df-a08a-0800200c9a66" xmlns:ws-p="com.bt.ws-
profile">
        <ws-p:name>Logging</ws-p:name>
        <ws-p:Interface ID="ac06e341-6763-11df-a08a-0800200c9a66">
                <ws-p:name>Management</ws-p:name>
                <ws-p:Functionality ID="8db9cd54-676b-11df-a08a-0800200c9a66">
                <ws-p:name>CreateLogEntriesCollection</ws-p:name>
                <ws-p:Constraint ID="f8f0b7c1-6764-11df-a08a-0800200c9a66">
                        <ws-p:ActivityLocation>WSDL</ws-p:ActivityLocation>
                        <ws-
p:Location>muker.ncl.ac.uk:8080/LoggingInfrastructure/mgt</ws-p:Location>
                </ws-p:Constraint>
                </ws-p:Functionality>
                <ws-p:Functionality ID="9d940791-676b-11df-a08a-0800200c9a66">
                <ws-p:name>RemoveLogEntriesCollection</ws-p:name>
                        <ws-p:Constraint ID="ac06e343-6763-11df-a08a-
0800200c9a66">
                                <ws-p:BeforeorAfter>before</ws-p:BeforeorAfter>
                                <ws-p:Activity>self:8db9cd54-676b-11df-a08a-
```

```
0800200c9a66</ws-p:Activity>
                    </ws-p:Constraint>
                <ws-p:Constraint ID="a84e2761-676b-11df-a08a-0800200c9a66">
                    <ws-p:ActivityLocation>WSDL</ws-p:ActivityLocation>
                    <ws-
p:Location>muker.ncl.ac.uk:8080/LoggingInfrastructure/mgt</ws-p:Location>
                </ws-p:Constraint>
                </ws-p:Functionality>
        </ws-p:Interface>
        <ws-p:Interface ID="ac06e342-6763-11df-a08a-0800200c9a66">
                <ws-p:name>Functionality Logging</ws-p:name>
                <ws-p:Functionality ID="d93e3051-676a-11df-a08a-0800200c9a66">
                    <ws-p:name>CreateEntryLog</ws-p:name>
                    <ws-p:Constraint ID="ac06e343-6763-11df-a08a-
0800200c9a66">
                        <ws-p:BeforeorAfter>before</ws-p:BeforeorAfter>
                        <ws-p:Activity>self:8db9cd54-676b-11df-a08a-
0800200c9a66</ws-p:Activity>
                    </ws-p:Constraint>
                    <ws-p:Constraint ID="ac06e344-6763-11df-a08a-
0800200c9a66">
                        <ws-p:ActivityLocation>WSDL</ws-p:ActivityLocation>
                        <ws-
p:Location>muker.ncl.ac.uk:8080/LoggingInfrastructure/function</ws-p:Location>
                    </ws-p:Constraint>
                </ws-p:Functionality>
                <ws-p:Functionality ID="d93e3051-676a-11df-a08a-0800200c9a66">
                    <ws-p:name>RemoveEntryLog</ws-p:name>
                    <ws-p:Constraint ID="452ef0b2-676b-11df-a08a-0800200c9a66">
                        <ws-p:BeforeorAfter>before</ws-p:BeforeorAfter>
                        <ws-p:Activity>self:d93e3051-676a-11df-a08a-
0800200c9a66</ws-p:Activity>
                    </ws-p:Constraint>
                    <ws-p:Constraint ID="452ef0b9-676c-11df-a08a-0800200c9a66">
                        <ws-p:ActivityLocation>WSDL</ws-p:ActivityLocation>
                        <ws-
p:Location>muker.ncl.ac.uk:8080/LoggingInfrastructure/function</ws-p:Location>
                    </ws-p:Constraint>
                </ws-p:Functionality>
        </ws-p:Interface>
</ws-p:Capability>
```

Table 5. Registration data of logging infrastructure

In Table 6, the HTTP digest authentication and decoding infrastructure capability is registered in much the same way.

```
<ws-p:Capability ID="218ee5e1-6836-11df-a08a-0800200c9a66" xmlns:ws-p="com.bt.ws-
profile">
        <ws-p:name>HTTP Digest Authz</ws-p:name>
        <ws-p:Interface ID="218ee5e3-6836-11df-a08a-0800200c9a66">
                <ws-p:name>Management User</ws-p:name>
                <ws-p:Functionality ID="218ee5e5-6836-11df-a08a-0800200c9a66">
                    <ws-p:name>CreateUser</ws-p:name>
                    <ws-p:Constraint ID="218ee5e7-6836-11df-a08a-0800200c9a66">
                        <ws-p:ActivityLocation>WSDL</ws-p:ActivityLocation>
                        <ws-
p:Location>muker.ncl.ac.uk:8080/HTTPDigestAuthz/UserMgt</ws-p:Location>
```

```
                    </ws-p:Constraint>
                </ws-p:Functionality>
        </ws-p:Interface>
        <ws-p:Interface ID="5f913a01-6836-11df-a08a-0800200c9a66">
                <ws-p:name>Functionality</ws-p:name>
                <ws-p:Functionality ID="5f913a03-6836-11df-a08a-0800200c9a66">
                        <ws-p:name>ReadMessage</ws-p:name>
                        <ws-p:Constraint ID="5f913a07-6836-11df-a08a-0800200c9a66">
                                <ws-p:BeforeorAfter>before</ws-p:BeforeorAfter>
                                <ws-p:Activity>self:218ee5e5-6836-11df-a08a-
0800200c9a66</ws-p:Activity>
                        </ws-p:Constraint>
                        <ws-p:Constraint ID="5f913a05-6836-11df-a08a-0800200c9a66">
                                <ws-p:ActivityLocation>WSDL</ws-p:ActivityLocation>
                                <ws-
p:Location>muker.ncl.ac.uk:8080/HTTPDigestAuthz/Function</ws-p:Location>
                        </ws-p:Constraint>
                </ws-p:Functionality>
                <ws-p:Functionality ID="639220c5-897d-4f52-8db4-5d03456d4eab">
                        <ws-p:name>GenerateHeader</ws-p:name>
                        <ws-p:Constraint ID="5f913a05-6836-11df-a08a-0800200c9a66">
                                <ws-p:ActivityLocation>WSDL</ws-p:ActivityLocation>
                                <ws-
p:Location>muker.ncl.ac.uk:8080/HTTPDigestAuthz/Function</ws-p:Location>
                        </ws-p:Constraint>
                </ws-p:Functionality>
        </ws-p:Interface>
</ws-p:Capability>
```

Table 6. Registration data of HTTP digest authentication and decoding infrastructure

In Table 7, the SecPAL authorization infrastructure capability is registered in much the same way.

```
<ws-p:Capability ID="6449fda1-6766-11df-a08a-0800200c9a66" xmlns:ws-p="com.bt.ws-
profile">
        <ws-p:name>SecPAL Authz</ws-p:name>
        <ws-p:Interface ID="6449fda2-6766-11df-a08a-0800200c9a66">
                <ws-p:name>Management User</ws-p:name>
                <ws-p:Functionality ID="9b2829e5-682a-11df-a08a-0800200c9a66">
                        <ws-p:name>CreateUser</ws-p:name>
                        <ws-p:Constraint ID="ac06e343-6763-11df-a08a-0800200c9a66">
                                <ws-p:BeforeorAfter>before</ws-p:BeforeorAfter>
                                <ws-p:Activity>self:6449fda3-6766-11df-a08a-
0800200c9a66</ws-p:Activity>
                        </ws-p:Constraint>
                        <ws-p:Constraint ID="93450270-6767-11df-a08a-0800200c9a66">
                                <ws-p:ActivityLocation>WSDL</ws-p:ActivityLocation>
                                <ws-
p:Location>muker.ncl.ac.uk:8080/SecPAL/UserMgt</ws-p:Location>
                        </ws-p:Constraint>
                </ws-p:Functionality>
                <ws-p:Functionality ID="9b2829e7-682a-11df-a08a-0800200c9a66">
                        <ws-p:name>UpdateUser</ws-p:name>
                        <ws-p:Constraint ID="93450270-6767-11df-a08a-0800200c9a66"/>
                </ws-p:Functionality>
                <ws-p:Functionality ID="9b2829e9-682a-11df-a08a-0800200c9a66">
                        <ws-p:name>DeleteUser</ws-p:name>
```

```
                    <ws-p:Constraint ID="93450270-6767-11df-a08a-0800200c9a66"/>
                </ws-p:Functionality>
        </ws-p:Interface>
        <ws-p:Interface ID="6449fda3-6766-11df-a08a-0800200c9a66">
                <ws-p:name>Management Trust</ws-p:name>
                <ws-p:Functionality ID="029887b9-682a-11df-a08a-0800200c9a66">
                        <ws-p:name>CreateTrust</ws-p:name>
                        <ws-p:Constraint ID="95450273-6767-11df-a08a-0800200c9a66">
                                <ws-p:ActivityLocation>WSDL</ws-p:ActivityLocation>
                                <ws-
p:Location>muker.ncl.ac.uk:8080/SecPAL/TrustMgt</ws-p:Location>
                        </ws-p:Constraint>
                </ws-p:Functionality>
                <ws-p:Functionality ID="9b2829e1-682a-11df-a08a-0800200c9a66">
                        <ws-p:name>DeleteTrust</ws-p:name>
                        <ws-p:Constraint ID="95450273-6767-11df-a08a-0800200c9a66"/>
                </ws-p:Functionality>
        </ws-p:Interface>
        <ws-p:Interface ID="644a24b0-6766-11df-a08a-0800200c9a66">
                <ws-p:name>Functionality Token</ws-p:name>
                <ws-p:Functionality ID="029887b7-682a-11df-a08a-0800200c9a66">
                        <ws-p:name>CreateToken</ws-p:name>
                        <ws-p:Constraint ID="ac2088f1-6767-11df-a08a-0800200c9a66">
                                <ws-p:BeforeorAfter>before</ws-p:BeforeorAfter>
                                <ws-p:Activity>self:6449fda2-6766-11df-a08a-
0800200c9a66</ws-p:Activity>
                        </ws-p:Constraint>
                        <ws-p:Constraint ID="ac2088f4-6767-11df-a08a-0800200c9a66">
                                <ws-p:ActivityLocation>WSDL</ws-p:ActivityLocation>
                                <ws-
p:Location>muker.ncl.ac.uk:8080/SecPAL/TokenMgt</ws-p:Location>
                        </ws-p:Constraint>
                </ws-p:Functionality>
                <ws-p:Functionality ID="87d36ee1-682a-11df-a08a-0800200c9a66">
                        <ws-p:name>EvaluateToken</ws-p:name>
                        <ws-p:Constraint ID="ac2088f1-6767-11df-a08a-0800200c9a66">
                                <ws-p:BeforeorAfter>before</ws-p:BeforeorAfter>
                                <ws-p:Activity>self:029887b9-682a-11df-a08a-
0800200c9a66</ws-p:Activity>
                        </ws-p:Constraint>
                        <ws-p:Constraint ID="ac2088f4-6767-11df-a08a-0800200c9a66"/>
                </ws-p:Functionality>
        </ws-p:Interface>
</ws-p:Capability>
```

Table 7. Registration data of SecPAL Authorization infrastructure

VO formation. Prior to any collaborative task and once the virtual shop has decided to establish the music store, it needs to reach an agreement with the security governance provider.

VO formation and management as such are out of the scope of the current work. However, the author uses a form of virtual context that supports the level of functionality required in order to demonstrate the feasibility of the proposed concrete architecture. As suggested in section 5.4 on future works, collaborative governance spanning across multiple organisations and user domains of expertise should be investigated in the future.

Therefore, in the current context, all negotiations between the different partners are meant to take place outside of the scope of the implementation.

With a governance gateway in place, the VMS operator can register his business capabilities. In this scenario, there is one business capability that has two functionalities 1) consume music content (i.e. input point for content providers) 2) send request for content (i.e. search query to content providers). The data used for the registration of this business capability is shown in Table 8.

```
<ws-p:Capability ID="8b73c250-6af7-11df-a08a-0800200c9a66" xmlns:ws-p="com.bt.ws-
profile">
        <ws-p:name>VMS</ws-p:name>
        <ws-p:Interface ID="8b73c251-6af7-11df-a08a-0800200c9a66">
                <ws-p:name>Functionality Read Content</ws-p:name>
                <ws-p:Functionality ID="8b73c252-6af7-11df-a08a-0800200c9a66">
                        <ws-p:name>ReadMusicContent</ws-p:name>
                        <ws-p:Constraint ID="8b73c253-6af7-11df-a08a-0800200c9a66">
                                <ws-p:ActivityLocation>WSDL</ws-p:ActivityLocation>
                                <ws-
p:Location>muker.ncl.ac.uk:8080/VMS/ReadContent</ws-p:Location>
                        </ws-p:Constraint>
                </ws-p:Functionality>
        </ws-p:Interface>
        <ws-p:Interface ID="8b73c254-6af7-11df-a08a-0800200c9a66">
                <ws-p:name>Functionality Create Content Query</ws-p:name>
                <ws-p:Functionality ID="8b73c255-6af7-11df-a08a-0800200c9a66">
                        <ws-p:name>CreateContentQuery</ws-p:name>
                        <ws-p:Constraint ID="8b73c256-6af7-11df-a08a-0800200c9a66">
                                <ws-p:ActivityLocation>WSDL</ws-p:ActivityLocation>
                                <ws-
p:Location>muker.ncl.ac.uk:8080/VMS/CreateContentQuery</ws-p:Location>
                        </ws-p:Constraint>
                </ws-p:Functionality>
        </ws-p:Interface>
</ws-p:Capability>
```

Table 8. Registration data of VMS business capability

Additionally, the VMS operator can contact the potential participants of the music shop (i.e. content providers). Agreements are reached between these content providers and the VMS operator regarding the conditions in which the business will be run and the VO operated. Following this, the content providers deposit their security profiles in the gateway together with the relevant data about their business functions (e.g. music catalogue web service). These security profiles contain the data necessary for the governance gateway to understand how to securely connect to the CPs services. More precisely, this means that CP3's security profile will comport a reference to the appropriate login XML schema. CP2's security profile on the other hand will provide for instance a reference to the SecPAL assertion it expects and what service, interface and operation the security governance gateway needs to contact in order establish trust and therefore allow for its identity to be established.

Figure 46 shows the governance user interface that connects to the profile initiator component in order to manipulate the different elements of the exposure governance. In this figure, the "governance features" in the left menu allow accessing and modifying business capabilities, abstract and concrete profiles, contexts and context selectors. In the same illustration the profile used to govern the exposure in the

context of the content provider 3 is shown. This profile is remarkable as the interaction with the content provider 3 does not require any infrastructure, the logging infrastructure being added by the governance operator. The implemented governance, by convention, will add the given xml nodes in the SOAP header of the request for which the profile is used.

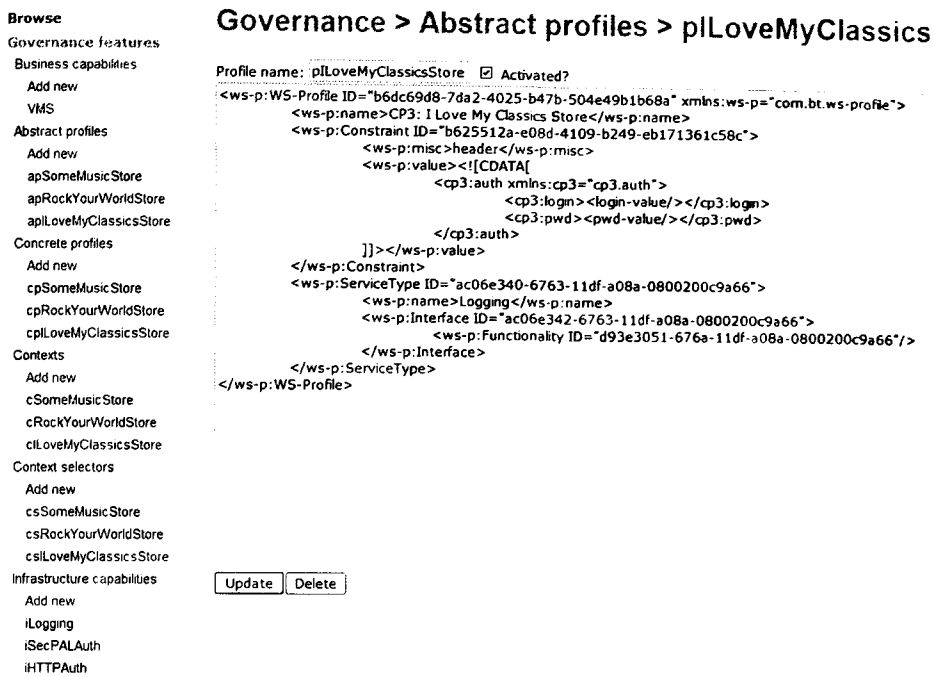


Figure 46. Governance user interface and "I Love My Classics Store" profile

The contextual data for the "I Love My Classics Store" profile is shown in Table 9. The configuration data specifies that for the given targeted element, particular values are to be changed accordingly. The targeted element in the case illustrated is the profile constraint shown in the "I Love My Classics Store" profile.

```
<ws-c:WS-Context ID="f80176e4-2dfa-45c9-a36f-d160f3185335" xmlns:ws-c="com.bt.ws-
context">
        <ws-c:name>clLoveMyClassicsStore</ws-c:name>
        <ws-c:configData>
                <ws-c:target>b625512a-e08d-4109-b249-eb171361c58c</ws-c:target>
                <ws-c:misc>
                replace:
                                <login-value/>=cp3
                                <pwd-value/>=pwd
                </ws-c:misc>
        </ws-c:configData>
</ws-c:WS-Context>
```

Table 9. Contextual data for "CP3: I Love My Classics Store" profile in the VMS

Finally, the context selector allows connecting a profile, a context, a business capability and a routing definition. Table 10 shows the context selector data for the CP3 profile and VMS business capability. This context selector specifies that when

the VMS business capability sends a request containing the "ILoveMyClassicsStore" expression, the CP3 profile and CP3 context should be used. Additionally, the up time property is set to always.

```
<ws-cs:WS-ContextSelector ID="d4b4bb5e-8e31-4140-a57c-7fb7d3e88ed0"
xsi:schemaLocation="com.bt.ws-context-selector context-selector.xsd" xmlns:ws-
cs="com.bt.ws-context-selector">
        <ws-cs:name>csILoveMyClassicsStore</ws-cs:name>
        <ws-cs:profile>b6dc69d8-7da2-4025-b47b-504e49b1b68a</ws-cs:profile>
        <ws-cs:context>f80176e4-2dfa-45c9-a36f-d160f3185335</ws-cs:context>
        <ws-cs:internalTarget>8b73c255-6af7-11df-a08a-0800200c9a66</ws-
cs:internalTarget>
        <ws-cs:configSelector>
                <ws-cs:data>ILoveMyClassicsStore</ws-cs:data>
                <ws-cs:operation>request</ws-cs:operation>
        </ws-cs:configSelector>
        <ws-c:upTime>
                <ws-c:type>always</ws-c:type>
        </ws-c:upTime>
</ws-cs:WS-ContextSelector>
```

Table 10. Context selector for "CP3: I Love My Classics Store" profile and the "CreateContentQuery" operation of the business capability

Table 11 holds the profile registered for CP1: Some Music Store. The content of this profile is shown in a more readable manner in Figure 47. The profile specifies the VMS business capability interaction with CP1 for both request and response. From this profile, the governance middleware knows that the interaction requires HTTP digest authentication mechanism. The credentials expected are provided in the context. Additionally, the governance middleware learns that the content of the response message is "encrypted" and the method used to encrypt it. The method of encryption is provided in the context.

The context and context selector for this interaction take the same form as the examples given in Table 9 and Table 10.

```
<ws-p:WS-Profile ID="edf66321-683f-11df-a08a-0800200c9a66" xmlns:ws-p="com.bt.ws-
profile">
        <ws-p:name>CP1: Some Music Store profile</ws-p:name>
        <ws-p:Capability ID="218ee5e1-6836-11df-a08a-0800200c9a66" xmlns:ws-
p="com.bt.ws-profile">
                <ws-p:name>HTTP Digest Authz</ws-p:name>
                <ws-p:Interface ID="5f913a01-6836-11df-a08a-0800200c9a66">
                        <ws-p:name>Functionality</ws-p:name>
                        <ws-p:Functionality ID="639220c5-897d-4f52-8db4-
5d03456d4eab"/>
                </ws-p:Interface>
                <ws-p:Constraint ID="0e91d933-f999-465b-8b80-2126b151c5cc">
                        <ws-p:misc>operation</ws-p:misc>
                        <ws-p:value>request</ws-p:value>
                </ws-p:Constraint>
        </ws-p:Capability>
        <ws-p:Capability ID="218ee5e1-6836-11df-a08a-0800200c9a66" xmlns:ws-
p="com.bt.ws-profile">
                <ws-p:name>HTTP Digest Authz</ws-p:name>
```

```
                <ws-p:Interface ID="5f913a01-6836-11df-a08a-0800200c9a66">
                        <ws-p:name>Functionality</ws-p:name>
                        <ws-p:Functionality ID="5f913a03-6836-11df-a08a-0800200c9a66"/>
                </ws-p:Interface>
                <ws-p:Constraint ID="d7acb749-eead-46dc-8ec8-d967dce88304">
                        <ws-p:misc>operation</ws-p:misc>
                        <ws-p:value>response</ws-p:value>
                </ws-p:Constraint>
        </ws-p:Capability>
        <ws-p:ServiceType ID="ac06e340-6763-11df-a08a-0800200c9a66">
                <ws-p:name>Logging</ws-p:name>
                <ws-p:Interface ID="ac06e342-6763-11df-a08a-0800200c9a66">
                        <ws-p:Functionality ID="d93e3051-676a-11df-a08a-
0800200c9a66"/>
                </ws-p:Interface>
        </ws-p:ServiceType>
</ws-p:WS-Profile>
```

Table 11. CP1 Exposition profile

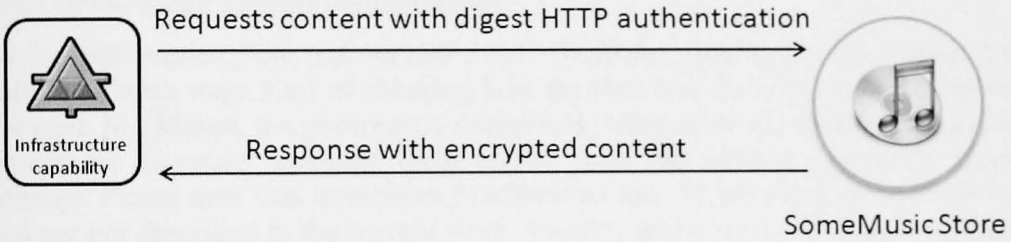


Figure 47. CP1 Exposition profile

Security profile management. At this stage it becomes possible for security gateway to validate and enact the different security profiles.

To achieve this, the profile management component of the gateway goes through a set of actions that starts with the verification of the security profile proposed to the definition of the processes necessary for the instantiation and management of the future implementation of the security profile.

These domains can be respectively as described in section 3.5 split in several steps: defining the infrastructure capabilities, the policy templates, the service dependencies and the information flow for the first one and defining the profile management process as well as publishing the infrastructure profile for the second.

More details on the specific stages of the profile management and how they execute are given in the following paragraphs. Tests assessing the relevance and quality of these stages are found in sections 4.2.5 and 4.2.6.

Define security infrastructure (c.f. section 3.5.1): The first stage of this process is to check for each security service if the description given in the security profile is adequate and complete according to the profile description taxonomy. If that test passes, the abstract definition of the service as well as its requirements are gathered. Please note that at this stage the governance has not selected a particular instance of any running service but has selected the appropriate categories in the taxonomy.

The security governance gateway does separate copies of the security profile at the beginning and end of this process. This allows keeping a track of what the domain of governance requested is as opposed to the profile that will be completed and proposed.

Define policy templates (c.f. section 3.5.2): This second stage verifies if the policy type that will be used require any translation (when using different grammars). In addition, this stage checks how they need to be managed. In the described implementation this stage is not necessary.

Define infrastructure dependencies (c.f. section 3.5.3): Following this individual check, the relationships between the services, both on the managerial and operational sides are verified.

The last stage goes through the security profile and check for missing components. For instance, a content provider could have specified an access control method without mentioning any identity management.

Following this, the VMS operator could review and select the best matches in the positive answers it has received, potentially eliminating content provider with too weak or too costly security configurations.

Define information flow (c.f. section 3.5.4): With the security profile complete and safe, the fourth stage aims at checking how the data will flow from one operation to the next. In addition, the governance determines, when relevant, whether a translation process is necessary amongst them and/or with the content provider's security settings. Please note that translation mechanisms may be provided by external tools and are not described in the current work. Finally, the concrete services enacting the infrastructure capabilities are selected.

The result of this step is saved as a security profile that can be instantiated.

Define profile management processes (c.f. section 3.5.5): With the profile instance in place and the services selected, the final stage is for the profile composition management component to define the different steps that will be necessary in order to call, configure and connect the security services. The result of this last stage is stored together with the now complete and safe security profile as a coordination plan.

When necessary the security governance gateway can then enact this complete profile through this management process.

Select infrastructure profile instance (c.f. section 3.5.6): When required to enact a security profile, the gateway search into its database and selects it. It then makes sure that the data can flow between the enacted security profile and the VMS interactions they manage.

Define service-specific policies(c.f. section 3.5.7): The profile management goes through all the concrete services selected in the security profile and instantiates the policy templates by inserting the context data.

Define information flow (c.f. section 3.5.8): If necessary, the transformations templates are also instantiated according to the results of the previous step on translation.

Define service exposure management processes (c.f. section 3.5.9): The profile manager orders and binds the different management processes of the security profile and the VMS interactions they manage.

Publish service instance (c.f. section 3.5.10): Finally, the policies created in step 26 are pushed to the different relevant stores (e.g. the SaaS capabilities driven by them) through the enactment of the management processes. To finish, the VMS clients of the CP services are bound to their instantiated profiles and exposed.

Adaptability. A VMS will want to be able to include as many content providers as it can, but different partners will have distinct security needs and settings. By reviewing and accepting the different security profiles used by each partner according to its specific needs, the VMS can more promptly make use of their contents. This necessitates and is limited by the capability of the security governance gateway to find security related services that provide for these different requirements.

4.2.5. Functional tests

In this section, the different tests done in order to evaluate if the Virtual Music Store functions according to expectations are presented together with their results. For each test, a parallel "witness test" is performed to ensure that the result is valid.

For the purpose of testing, the VMS does not perform any caching and any query is directly forwarded to the Content Providers. This is possibly not an adequate practice for this type of system in a commercial environment, but renders the testing easier.

Test 1: Content Provider 1 integration. For this first test, all the infrastructure capabilities are registered in the governance gateway. However, only the first exposition profile is made available. If the gateway is well implemented, the infrastructures and profile description well written, this test is expected to pass.

Validation test. In order to insure that test 1 result is valid, in this validation stage, the logging infrastructure capability is unregistered. As it has been marked as a mandatory infrastructure in all profiles, this test is expected to fail.

Test 2: Content Provider 2 integration. For this second test, all the infrastructure capabilities are registered in the governance gateway. However, only the second exposition profile is made available. If the gateway is well implemented, the infrastructures and profile description well written, this test is expected to pass.

Validation test: In order to verify test 2 result, the Functionality Token interface is erased from the SecPAL Authz infrastructure description. As this renders the SecPAL Authz infrastructure unsuitable for the contextualisation this test is expected to fail.

Test 3: Content Provider 3 integration. For this third test, all the infrastructure capabilities are registered in the governance gateway. However, only the third exposition profile is made available. If the gateway is well implemented, the infrastructures and profile description well written, this test is expected to pass.

Validation test: In this validation test, the schema defining user credentials supported by the custom authorization infrastructure used by CP3 is modified (username node is changed to user). This test is expected to fail.

Test 4: All Content Provider integration. For this fourth test, all the infrastructure capabilities are registered in the governance gateway. In addition, all the exposition profiles are made available. If the gateway is well implemented, the infrastructures and profile descriptions well written, this test is expected to pass.

Validation test: In this validation test, the HTTP digest infrastructure capability is unregistered. The expected outcome of this test is that only content coming from

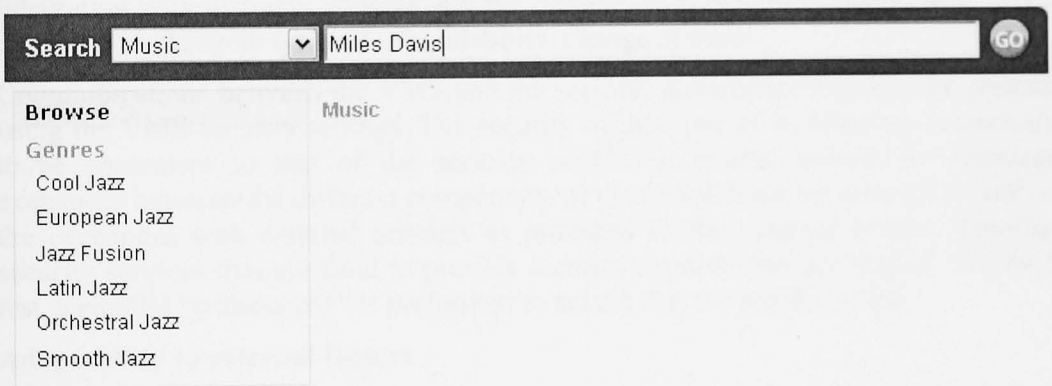
content providers 2 and 3 will appear on the VMS interface and therefore the test will fail.

Table 12 shows the results of the tests. The result is set as passed when the adequate music tracks per content provider are displayed on the web interface and failed otherwise. As all the tests produce the expected outcomes, the exposition governance implementation is therefore validated from a functional point of view.

Figure 48 and Figure 49 illustrate the outcomes of test 1 and its validation counterpart.

Figure 48. Test 1 result

Figure 49. Test 1 validation test result



Test	Result	Expected result
1	Passed	Passed
1 validation	Failed	Failed
2	Passed	Passed
2 validation	Failed	Failed
3	Passed	Passed
3 validation	Failed	Failed
4	Passed	Passed
4 validation	Failed	Failed

Table 12. Implementation tests results

4.2.6. Adaptability tests

The purpose of this evaluation is to demonstrate the adaptability of the security governance middleware. In order to do so, the author has assessed its capacity for integration with different systems and the capacity to manage this adaptability when dealing with different types of unavailability, change or event.

Communications between the VMS and the security governance solution are secured using the VMS security settings. The security of this type of middleware is assumed to be equivalent to that of the security profiles it enacts. Indeed, the message exchanges between the different components of the middleware are secured as well as the exchanges with external services as provided by the external service. External security services that are used to provide security requirements are trusted. For each test, a parallel "witness test" is performed to ensure that the result is valid.

Adaptability to external factors

Test 1: Infrastructure capability change. This test is similar to "test 3: Content Provider 3 integration validation test". In order to demonstrate that the governance middleware is capable to adapt to changes in the way the external infrastructure capabilities behave, this earlier test was repeated. Once again, this test is expected to fail.

Validation test. In order to insure that test 1 result is valid, in this validation stage, the custom authorization infrastructure used by CP3 is registered twice. The first registration makes use of the unmodified and valid node while the second one is registered with the modified node as in the earlier test. This test is expected to pass.

Test 2: Introduction of new infrastructure capability. To ensure that the introduction of new infrastructure capabilities is supported and does not hamper the good functioning of the governance middleware, the same type of infrastructure is deployed and registered several times. In this case, the custom authorization infrastructure used by CP3 is deployed and registered three times. This test is expected to pass.

Validation test. In order to insure that aforementioned test's result is valid, each registered custom authorization infrastructure was in turn unregistered until none was left. The first two tests are expected to pass while the third iteration leaves no authorization service for CP3 is expected to fail.

Test 3: Change a profile. In this test, the mandatory logging infrastructure capability is taken out of the abstract profile and is therefore not required anymore. This test is expected to pass.

Validation test. In this validation test, the logging infrastructure capability is put back in the abstract profile but unregistered from the service registry. This test is expected to fail.

Test 4: Change a security policy In this test, the values of the username and password are modified in the profile description and at CP3's side authorization code. This test is expected to pass.

Validation test. To verify the validity of the previous test, CP3's side authorization code is changed back to allow only the previous values to be accepted. This test is expected to fail.

Adaptability to internal factors

Test 5: Swap a security component for another, similar, component. This test is similar to "Test 2: Introduction of new infrastructure capability". However, the HTTP digest infrastructure capability was used instead of the custom authorization infrastructure. This test is expected to pass.

Validation test. As for "Test 2: Introduction of new infrastructure capability" validation test. The first two iterations are expected to pass while the third iteration leaves no authorization service for CP1 is expected to fail.

Reusability

Test 6: Use an infrastructure capability in different profiles. With the presence of the logging infrastructure capability in the abstract profile and the use of the abstract profile as the base for all security profiles, this test has been performed through previous tests. This test is expected to pass.

Validation test. In order to demonstrate that the governance middleware tolerates the presence of a registered but unused infrastructure, the logging infrastructure was taken out of the abstract profile, but still registered in the service registry. This test is expected to pass.

Test 7: Reuse stored profile in a different context. An abstract profile has been used, in order to define minimum requirements, as the base for all security profiles. This test has therefore been performed through previous tests. This test is expected to pass.

Validation test. In this scenario, the use of the abstract profile was made mandatory in order to guaranty a minimum security. However, in different scenarios, this might not be the case and it is trivial to deduce from the previous tests that this possibility, when implemented can be supported by a governance middleware.

Table 13 shows the results of the adaptability tests. The result is set as passed when the adequate music tracks per content provider are displayed on the web interface and

failed otherwise. As all the tests produce the expected outcomes, the exposition governance implementation is therefore validated from a functional point of view.

Test	Result	Expected result
1	Failed	Failed
1 validation	Passed	Passed
2	Passed	Passed
2 validation	Passed, Passed, Failed	Passed, Passed, Failed
3	Passed	Passed
3 validation	Failed	Failed
4	Passed	Passed
4 validation	Failed	Failed
5	Passed	Passed
5 validation	Passed, Passed, Failed	Passed, Passed, Failed
6	Passed	Passed
6 validation	Passed	Passed
7	Passed	Passed
7 validation	/	/

Table 13. Adaptability tests results

The potential scope of the contextualisation governance is wide and the concrete architecture has been thought of to take into account complex scenarios as well as simple ones. The tests presented in this section do not cover all aspects possible as this would necessitate more effort than it is possible in the timeframe of this thesis. Instead, the tests showed a practical illustration of the possibilities made available by the use of a well-defined contextualisation governance architecture.

4.1. Discussion

The authors of [42] (cf. related work section 2.2.2), have devised six types of adaptation that a flexible security middleware suite should support:

S1 Change a local parameter of a security component (e.g. the encryption method for an audit service).

S2 Introduce new security functionality (e.g. add a secure logging component).

S3 Compose/recompose a deployed security component with one or more application components. Application components depend on the security component but the security component can also depend on the application component(s) (e.g. for context-based access control).

S4 Swap a security component for another one (e.g. replace the authorization decision engine).

S5 Compose a security component using a (new) third-party component that is deployed elsewhere.

S6 Change a security policy. Since the security policy explicitly depends on application-level concepts, any change in a security policy can require further adaptations.

The system proposed in this paper, based on the security profile, is meant to introduce flexibility to the way in which the NFP aggregation lifecycle is managed and user requirements expressed. Therefore the following discussion points have been added:

S7 Enact the security profile at different stages in different situations (e.g. CP1 steps 1-39, CP2 steps 21-39).

S8 Express identical requirements using different semantics.

The objective of this discussion is to review the adaptability and to define its limits. The security profiles submitted by the content provider to the security governance middleware have been used in order to determine the scope of the adaptation.

S1 Changing a security service's configuration requires the user to change the security profile. With the change committed, the governance middleware will make use of Service Management's access to the security service's management interface to perform the change. However, the governance infrastructure will go through the process of profile management to ensure that the change is valid and can be realized. If the security service does not support this change a different one may be selected or the modification rejected.

S2 If the relevant security services are registered as accessible in this context and with these requirements, a security profile needs to be updated with the additional security functionality.

S3 Composition of security as well as other value-adding services can be realized. Of course, the quality of the segregation of a security service between different contexts (e.g. different interactions and conversations) depends on its implementation and may not be possible. For instance, a service may or may not support multi-domain instantiations and configurations.

S4 As presented in section 3.5, replacing a security service with another, similar one is achievable as long as a potential replacement is registered and accessible. If a security service is found missing at runtime, it is possible to start again from step 13 onwards, to regenerate an instantiated profile while storing the incoming and outgoing messages.

S5 The ability to compose external providers' security services is the very foundation of this work.

S6 In this model, modifying a security policy is equivalent to changing a security component local parameter (c.f. **S1**). This is due to the expected policy driven nature

of the infrastructure capabilities. A component local parameter will be modified using policy, in the same technical manner as an access control assertion (i.e. security policy).

S7 It is possible to store a profile's state at any stage of its lifecycle. However, validation stages are required in most cases to ensure the profile's validity at the time of use (e.g. whether the security service is still available).

S8 This point has not been verified in this set of experiments. However, it depends on the different semantic styles used as well as their interoperability.

5. Summary of contributions and future work

In the previous section, an implementation of the proposed concrete architecture was presented. Subsequently, series of tests both functional and targeting the adaptability were presented to demonstrate the validity of the implementation. Finally, a discussion based on the evaluation of a similar approach was proposed.

In this chapter the author summarises the contributions of the work and then concludes the thesis with suggestions for future work.

5.1. Summary of research challenges

In the following paragraphs, the author assesses the proposed approach with respect to each of the research challenge provided in Chapter 2.

1. Reusability of integration layer. The separation of concerns in the architecture proposed (c.f. Figure 17, Figure 23, Figure 31) aims at providing a coarse-grained distribution of the components. In this architecture the level of granularity proposed answers to the levels or reutilisation and potential performance optimisation techniques (e.g. replication) that can be used in distributed systems. In addition the message broker nature of the approach proposed offers a high potential for reusability.

2. Shared communication semantics. This requirement is principally dealt with by the brokering nature of the architecture proposed and the service orientation that brings ease of connectivity.

3. Decentralised model. The separation of concern and service orientation of the architecture allows avoiding a central core and relying on distributed components. The use of dedicated layers to administrate policies (c.f. policy management core infrastructure in section 3.4.1) and potential components (c.f. service management core infrastructure in section 3.4.1) permits the architecture to be flexible when dealing with resource location.

4. Definition of SOA governance and related concepts. The core objectives of the proposed architecture are presented in section 1 and further developed through a practical example in section 1.1.

5. High level description of key contextualisation elements and concepts. Sections 3.2 and 3.3 present the different elements forming the governance architecture and their relationships. Each element is described individually before a more global definition where the components are presented together is proposed.

6. Technical description of key contextualisation elements and their relationships. In addition to the descriptions as mentioned in point 5, sections 3.4 and 3.5 give a more detailed description of the key elements and a point by point description of all the steps happening prior and during a contextualisation. These steps are further divided into actions to give further details on the different processes taking place.

7. Non Functional Property (NFP) description model. With the use of the infrastructure profile definition proposed in section 3.3.1, potential partners (e.g. end users, other governance middleware, NFP providers) and this current architecture are able to express requirements in term of NFP and communicate them.

8. NFP profile description. The infrastructure profile along with the mechanisms that allow selecting capabilities and their policies are described in Chapter 3 on the anatomy of the infrastructure. The grammar used to describe the profile can be adapted to fit specific needs. The selection of capabilities and policies are also handled by dedicated layers enabling flexible and adaptable mechanisms.

9. NFP profile instance. The stages described in the series of steps necessary in order to create a safe and complete profile and then instantiate it as well as expose the business capabilities are described in section 3.5. The policy driven approach used in the architecture presented along with a dedicated layer for service management (c.f. section 3.4.2.1 on Capability management) allow specific management processes and policies to efficiently specify and supervise the characteristics of the infrastructures and profiles life-cycle. For the profile, these life cycles management processes are used by a dedicated component (i.e. Profile Life-Cycle Management in Figure 31). Finally the process of instantiating the profile and its different components (e.g. services, policies) allows multi-tenancy usage scenarios.

10. Reusable & manageable NFP providers. The choice of service based architecture was partly to be able to rely upon distributed pieces of software that can be made configurable and reusable. As presented in point 9 above, the architecture comprises a dedicated layer management to manage the different components and allow for their optimal reusability. This in turn, allows for the governance to interchange and reconfigure the different capabilities taking part in the governance process.

11. Run time aggregation of NFPs. The stages described in the series of steps necessary in order to create a safe and complete profile and then instantiate it as well as expose the business capabilities (c.f. section 3.5) present such policy schemes and how they can be contextualised. The transformation mechanism relies on three elements, the policy template, the transformation policy and the contextualised data. The first two elements are provided at design time while the later is given at run time. This system is flexible but becomes more and more complex as the number of transformation potentially necessary increase.

12. Domain agnostic. The limitations of the architecture proposed in terms of the types of resources it can govern and the categories of contextualisation it can provide are limited by the NFP description models it holds and the NFP providing components (i.e. infrastructure capabilities) it is aware of.

13. Life-cycle management of the aggregation of NFPs. The stages described in the series of steps necessary in order to create a safe and complete profile and then instantiate it as well as expose the business capabilities are described in section 3.5. These stages are progressing from an abstract profile with a high level view of what the domain of exposition will be into an instantiated and enacted profile that is complete and comprises concrete as well as configured instance of services.

14. Semantically described components. A normalised manner is used to define the way infrastructure capabilities are exposed. This taxonomy is described in Figure 19 of section 3.3.1.

15. Semantic agnostic. The proposed architecture can be made to understand different manners to express the domain of exposition thanks to the user interface abstract service pattern used (c.f. section 3.4.2) and the policy driven approach of the architecture. The limitation to this approach being that a translation between a new

semantic and an existing one, understood by the governance middleware, must be provided.

16. Support for end to end message level security. By maintaining no visible link between the final enhanced exposed resource and its pre-governance source (i.e. the business capability), the architecture, within the scope of its contextualisation, prevents threats from reaching the resource directly (c.f. steps 23 and 38 of the business capability exposure in section 3.5). In addition, the state of the different resources (e.g. NFP providers, policies) can be constantly monitored to prevent an unwanted execution of a profile. Finally, in order to prevent information leaks, security and auditing capabilities such as a logging service (c.f. section 4.2), when available, can be seamlessly included in the profile to evaluate data miss-management or prevent privacy issue. A tested scenario covering such requirement is described in the evaluation section (c.f. section 4).

17. Verify validity of the contextualisation strategy. During the profile life-cycle, dependencies are systematically sought and resolved. In addition, the profile usability can be enhanced for particular environments using different ways to express the exposition requirements through the user interface abstract service pattern (c.f. section 3.4.2).

Table 14. Summary of research challenges

Legend L – supported X – supported and specified in detail " " – not supported	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Proposed concrete architecture	**X**	**X**	**X**	**X**	**X**	**L**	**X**	**L**	**X**	**L**	**X**	**X**	**X**	**X**	**X**	**X**	**X**

Table 14 recapitulates the research challenges in the same manner as Table 1, Table 2, Table 3 and Table 4, but specifying what points are fulfilled by the proposed concrete architecture as opposed to those satisfied by the related works.

5.2. SOA governance model and Service Delivery Frameworks

This section focuses on the framework's conformance to the principles of the standard Service Delivery Framework developed by the TeleManagement Forum, introduced in section 2.2.3 and more thoroughly described in [44].

Figure 50 presents the main SDF concepts overlaid by those of governance model, showing the detailed mapping of the proposed governance model on top of the standard framework.

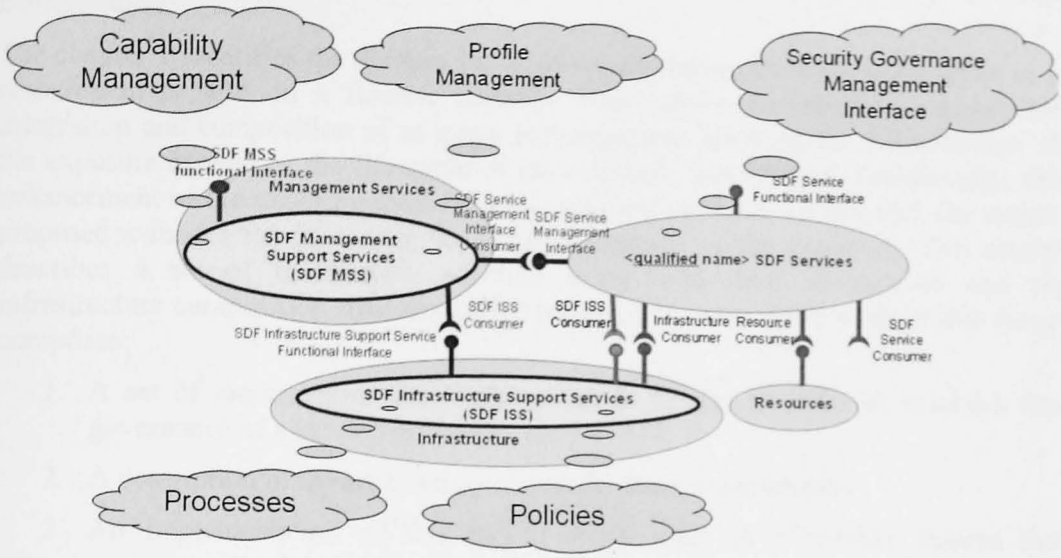


Figure 50. Mapping the Governance Model on the SDF standard

The governance model presented in the previous section has been developed in full compliance with the SDF standard of the TMF. This alignment extends the applicability of the governance model in service domains that conform to SDF and fulfils the respective research challenges 5 and 6 introduced in section 2.2.4.

The functional interface of the SDF in this mapping is the governance management interface as it is the governance model interface that allows for the management of capability.

The management service consists of the different components described in section 3.4. Specifically these are the Profile management, Capability management and the governance layer base which allows cementing and managing the governance enabling infrastructures.

The infrastructure support services are these governance enabling infrastructures that allow for the governance model to run along with the potential infrastructure capabilities that can be used to provide non functional support.

The SDF Infrastructure may host further supporting facilities for the remaining governance model aspects. It can provide Policy-based Management facilities in order to handle service policies and rules, as specified in the governance model description. It can also use the service catalogue to capture dependencies of a service onto other component artefacts. Finally, it can offer authentication and authorisation mechanisms for access control in order to define and assign privileges regarding the scope of operations entities can conduct within the SDF or capabilities they bear and can make available to SDF.

5.3. General summary

In this thesis, the author has provided a description of an architecture for contextual governance for SOA. This model is based on requirements that underline the need for policy and process management, resource life-cycle management, visibility and contextualisation. One of the roles and tested use case of the proposed model is to

handle the security of web services exposed through it by managing their security configuration.

The chapter 1 identifies the domain for action as managing the contextualisation of a resource in a SOA in a flexible manner. This creates the need to support the integration and composition of as many infrastructures allowing the enhancement of the exposure as well as the life cycle of the exposure thus created. Additionally, the enhancement of the exposure must be made safe and secured. To this end, the author proposed a model for governing the different aspects of the exposure. This model describes a set of middleware services along with their interactions and an infrastructure composition structure called profile. The core of the work in this thesis comprises:

1. A set of requirements that define what is needed in order to establish the governance of a service exposure.
2. A description of the architecture that fulfil these requirements.
3. An implementation of this model along with an evaluation against the requirements aforementioned.

This work addresses the requirements for contextualisation governance specified in Section 2.2.4.

A notable feature of the author's approach is a careful separation of concerns and modularity of the architecture. Figure 51 recalls the architecture presented in section 3 and presents a layered view of it.

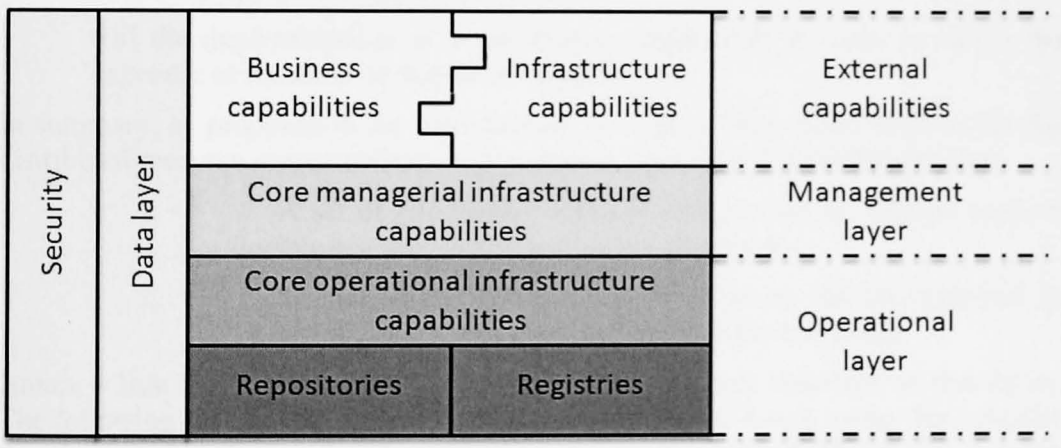


Figure 51. Layered view of the governance architecture

The contextualisation governance is realised through a set of middleware that allow enhancing the exposition of resources by composing external NFP infrastructures and managing the composition. This set of middleware is itself separated in four logical layers: the external capabilities that encompass both the business and infrastructure capabilities, the management layer that comprises core infrastructure capabilities with supervision roles, the operational layer which includes the functional core capabilities such as event processing, repositories and registries and finally the data and security layers allowing to describe the components and secure the architecture. This separation is fundamental to the flexibility of the middleware.

In these layers, each of these capabilities can be deployed as a components with its own management, policy administration framework (control pane) and operational interfaces (data pane). Each capability is also policy driven as this permits configurability and flexibility. The different capabilities are meant to have their own distinct grammars and policy languages in order to keep their own advantages, capacities and evolution potential in their respective domains.

The interoperability issues generated by this situation are addressed at the messaging level through transformation and governance as demonstrated by the transformation steps (section 3.5.2 point 7).

This also enables the interchange of core capabilities within their categories when necessary. Non-core capabilities (e.g. auditing) can also be added through the same manner.

It is noticeable that choosing to have these components to be independent and follow their own path implies an additional level of complexity. However this trade-off is compensated by the gains in flexibility, dynamicity and by allowing auditing of the results of each action separately if necessary.

The author has demonstrated the novelty of the design and implementation of the contextualisation governance middleware with respect to the related work discussed in chapter 2. Novel contributions include:

(i) the development of a flexible SOA based architecture that is based on fundamental work on distributed systems flexibility,

(ii) the provision of systematic support for exposure governance that is adaptable to different application contexts,

(iii) the implementation of a governance middleware in order to secure the exposure of resources in the shared domain.

In summary, as proposed in the introduction, through a SOA based architecture for distributed resource contextualisation governance, the author has established that:

1. A set of middleware services can provide an efficient support for flexible resource contextualisation governance.
2. This enhanced flexible contextualisation can be organised in such a manner that it preserves the safety of the exchanges.

Annex 4 lists the published work related to the architecture described in this thesis. The following section will present a discussion of potential future works that continue the theme of exploring fundamental concepts in the context of realistic systems implementation.

5.4. Future work

The current state of this work defines a concrete architecture for governance of safe and flexible contextualisation of resources in a SOA. Future investigation could evolve the material and expand on a variety of relevant topics. The following are some possible areas.

5.4.1. Different use cases

Potential future works include the use of governance middleware in different types of scenarios. Following is a list of potential use cases considered by the author:

- The Product and Service Assembly (PSA initiative) Catalyst [85], relies on a set of catalogues grouped in an infrastructure named the Active Catalog. One potential future research would be to investigate the place of such complex registry in the governance architecture or alternatively how the PSA initiative could take advantage of the architecture presented in this thesis.
- The Internet of Things and Service (IoTS) could be another environment where the governance architecture presented could play an interesting role. In [86] the co-authors and I identify the interest of security governance in the IoTS where Things could learn of potential threats and how to deal with them from other governed Things. Scenarios in Service Oriented Manufacturing chains are currently being discussed.

5.4.2. Flexible SOI governance

The investigation currently undertaken by the author and following this work comprises the study of this architecture in the context of flexible Service Oriented Infrastructure (SOI).

SOI provides a system for supplying information technology infrastructure as a service. Key aspects of SOI include virtualisation of all configurable infrastructure resources such as compute, storage, and networking hardware and software to support the running of applications. Consistent with the objectives for SOA, SOI facilitates the reuse and dynamic allocation of necessary infrastructure resources [53].

The main objective of this investigation is to determine how it would be possible to provide support to end users of SOI through a high level dashboard providing user centric semantics. Through this user centric layer, levels of NFPs and preferences (e.g. performance, security) could be chosen and selected. These preferences would then create sets of rules (i.e. profile in section 3.3.1) that would govern the SOI through monitoring, dynamic selection of infrastructures and virtualisation.

5.4.3. Improved profile expression

In section 2.2, the author has established that investigating the semantics necessary to describe IT system requirements is a legitimate and complex research topic on its own. One potential topic for future work is to utilise the requirements presented in paragraphs 2 on background in order to provide a more comprehensive and useable profile description model.

Alternatively, future work could explore how different semantics used to describe such requirements in different domains could be taken into account and used by governance middleware.

5.4.4. Federated governance

In section 3.6.2 on deployment patterns, the federation of governance middleware is introduced. The study of such use of governance middleware could attempt to determine a) what are the processes and techniques necessary in order to achieve efficient federation and b) what are the different types of federation achievable (e.g. 1 master with dependant governance middleware, equally ranked) and c) in what context it would interesting to use them. In such environments it would then be possible to investigate the potential of negotiating properties of the governance.

5.4.5. Trust brokering

In the context of governed interactions, service providers could find themselves in a situation where they do not know what services and by extension which providers are catering for different properties of the exchanges (e.g. security in the security governance scenario in section 4.2). This presents the governance architecture with the issue of allowing the different potential participants to express how they would define the level of trust required for entities in order to enter in collaboration with them. The following are some of the challenges of trust brokering in this type environment:

- Allowing a coarse grained definition of the level of trust required: a participant should be permitted to express what it expects from each property of the interaction (e.g. identity management, access control, virtualisation). Additionally, the participant should also be able to specify the level of trust required to access certain types of data in the interaction.
- Dynamic trust allocation: in this type of rapidly evolving environment, a participant's life-span can be very short. How to define how trustable this entity is when it has never been part of any collaboration and is potentially unknown to all participants should be taken into account.
- Properties and enforcement of trust: with potentially many different actors covering diverse aspects of a collaboration, defining what trust is and how it should be enforced could also become a challenge and should therefore be examined.

5.4.6. SOA governance middleware performance improvement

In the SOA security governance middleware presented in section **4,** policy templates, instances and processes are exchanged over the network. In complex scenarios or environments where the network is critical, this could create a stress on the physical infrastructure and the core infrastructure capabilities used as the more complex the profile, the more such data need to be exchanged.

One possible future research direction would be to investigate a) potential techniques that could be used to reduce the amount of data necessary to exchange or b) the number and types of the exchanges themselves between the core infrastructure capabilities of the architecture while keeping the level of distribution as high as possible.

5.4.7. Monitoring and Auditing

In chapters 3 and 5, the author introduces the relevance and possibility of introducing elements of monitoring and auditing into the architecture. These elements could further enhance the safety and reactivity of the architecture and allow further investigations towards autonomous behaviours. Due to the loosely coupled and distributed nature of the model, introducing such capabilities could be made on the top of the architecture proposed.

6. References

[1] The long nimbus, from The Economist, 23 October 2008, available at: http://www.economist.com/research/articlesbysubject/displaystory.cfm?subjectid=348909&story_id=E1_TNQTTRGQ

[2] Gresty C., Dimitrakos T., Thanos G. and Warren P, Meeting Customer Needs, BT Technology Journal, Vol 26, No 1, September 2008

[3] Deans P. and Wiseman R., Service-Oriented Infrastructure: Proof of Concept Demonstrator, BT Technology Journal, Vol 26, No 1, September 2008

[4] BR Katzy, G.S., The Virtual Enterprise. Information Age, 1995

[5] Katzy, B.R. Design and implementation of virtual organizations. in System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on. 1998. Pages?

[6] Periorellis, P., Cook, N., Hiden, H. et al. GOLD infrastructure for virtual organizations. Concurrency and Computation: Practice and Experience Vol. 20(11), pp - 1273-1288, 2008. Use this one as standard to fix your references.

[7] Dimitrakos, T., et al. Contract performance assessment for secure and dynamic virtual collaborations. in Enterprise Distributed Object Computing Conference, 2003. Proceedings. Seventh IEEE International. 2003.

[8] Anderson, Chris. "The Long Tail" Wired, October 2004.

[9] The Long Tail, in a nutshell, available at: http://www.longtail.com/about.html

[10] L. Frank Kenney, Daryl C. Plummer, Magic Quadrant for Integrated SOA Governance Technology Sets, 31 March 2009, Gartner RAS Core Research Note G00166481.

[11] de Leusse, P., Periorellis, P., Watson, P. and Maierhofer, A., Secure & Rapid Composition of Infrastructure Services in the Cloud, In The Second International Conference on Sensor Technologies and Applications, SENSORCOMM 2008, 25-31, August 2008, Cap Esterel, France, pp 770-775, IEEE Computer Society, 2008

[12] de Leusse, P., Periorellis, P., Watson, P. and Dimitrakos, T., A semi autonomic infrastructure to manage non functional properties of a service, In UK e-Science All Hands Meeting 2008, 8-11 September, Edinburgh, UK, National e-Science Centre, 2008.

[13] de Leusse, P., Dimitrakos, T., and Brossard, D., A governance model for SOA, IEEE 7th International Conference on Web Services (ICWS 2009), July 6-10, 2009, Los Angeles, CA, USA.

[14] de Leusse, P. and Brossard, D., Distributed systems security governance, a SOA based approach, Third IFIP International Conference on Trust Management, Springer, June 15-19, 2009, Purdue University, West Lafayette, USA.

[15] de Leusse, P.and Dimitrakos, T., SOA-based security governance middleware, The Fourth International Conference on Emerging Security Information, Systems and Technologies, IEEE Computer Society, July 18-25. 2010, Venice/Mestre, Italy.

[16] Natis, Y.V., et al., Predicts 2007: SOA Advances. 2006.

[17] T. Dimitriakos, G.L., et al, Towards a Grid Plateform Enabling Dynamic Virtual Organisations for Business Applications. 2005.

[18] Radhakrishnan, S., Integrating Entreprise Applications: Backgrounder. 2005.

[19] Fielding, Roy. Architectural Styles and the Design of Network-based Software Architectures (PhD Thesis). s.l. University of Irvine, California, 2000.

[20] Erl, T., Service-Oriented Architecture Concepts, Technology, and Design. 2005.

[21] Peltz, C., Web services orchestration and choreography. Computer, 2003. 36(10): p. 46 - 52.

[22] Erradi, A.M., P., wsBus: QoS-aware Middleware for Reliable Web Services Interactions. e-Technology, e-Commerce and e-Service, 2005. EEE '05. Proceedings. The 2005 IEEE International Conference on, 2005: p. 634 - 639.

[23] Bachman, J., S. Kline, and B. Soni, A New Service-Oriented Architecture Maturity Model. 2005.

[24] Corporation, S.S., SONIC ESB: AN ARCHITECTURE AND LIFECYCLE DEFINITION. 2005.

[25] Chappell, D., Enterprise Service Bus. 1st ed. Vol. 1. 2004: O'Reilly. 247.

[26] de Leusse, P., Periorellis, P., Watson, P., Enterprise Service Bus: An overview, CS-TR No 1037, School of Computing Science, Newcastle University, Jul 2007

[27] Arsanjani, A. Liang-Jie Zhang Ellis, M. Allam, A. Channabasavaiah, K., S3: A Service-Oriented Reference Architecture, in IT Professional, May-June 2007, volume 9, Issue 3, pages 10 - 17, ISSN 1520-9202, IEEE Computer Society.

[28] Kreger, H., Estefan J., Navigating the SOA Open Standards Landscape Around Architecture, June 2009.

[29] OASIS Reference Model for SOA, Version 1.0, OASIS Standard, October 2006: docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf

[30] OASIS Reference Architecture (Foundation) for SOA, Version 1.0, OASIS Public Review Draft 1, April 2008: docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-pr-01.pdf

[31] The Open Group SOA Reference Architecture (The Open Group SOA RA), Draft Technical Standard (to be published in 2009); refer to www.opengroup.org/projects/soa-ref-arch

[32] The Open Group SOA Governance Framework, Draft Technical Standard (to be published in 2009); refer to www.opengroup.org/projects/soa-governance

[33] W3C Note, Web Services Description Language (WSDL) 1.1, 15 March 2001, available at: http://www.w3.org/TR/wsdl

[34] W3C Recommendation, Semantic Annotations for WSDL and XML Schema, 28 August 2007, available at: http://www.w3.org/TR/sawsdl/

[35] W3C Member Submission, Web Service Modeling Ontology (WSMO), 3 June 2005, available at: http://www.w3.org/Submission/WSMO and http://www.wsmo.org/

[36] W3C Member Submission, OWL-S: Semantic Markup for Web Services, 22 November 2004, available at: http://www.w3.org/Submission/OWL-S

[37] W3C Member Submission, Web Application Description Language, 31 August 2009, available at http://www.w3.org/Submission/wadl

[38] Kunal Verma, Karthik Gomadam, Amit P. Sheth, John A. Miller, Zixin Wu"The METEOR-S Approach for Configuring and Executing Dynamic Web Processes", Technical Report . Date: 6-24-05

[39] José Antonio Parejo Maestre, Pablo Fernández Montes, and Antonio Ruiz Cortés, SOA Governance: Exploring Challenges & Benefits from an Autonomic Perspective, 2nd Workshop on Autonomic and SELF-adaptive Systems, 2009, San Sebastian, Spain

[40] Marks, E.A.: Service-Oriented Architecture Governance for the Services Driven Enterprise. John Wiley & Sons (2008)

[41] Brown, W.A., Laird, R.G., Gee, C., Mitra, T., SOA Governance: Achieving and Sustaining Business and IT Agility. IBM Press (December 2008)

[42] Bernhardt, J., Seese, D., A conceptual framework for the governance of service-oriented architectures. In: Service-Oriented Computing ICSOC 2008 Workshops. Springer (2009)

[43] Bruce G., Streamlining the Telco Production Line, BT Technology Journal, Vol 26, No 2, April 2009

[44] TeleManagement Forum Service Delivery Framework Programme, Service Delivery Framework Overview, TMF TR139, September 2008

[45] Georgalas N., Achilleos A., Freskos V.and Economou D., Agile Product Lifecycle Management for Service Delivery Frameworks: History, Architecture and Tools, BT Technology Journal, Vol 26, No 2, April 2009

[46] TMF061, Service Delivery Framework Reference Architecture, Release 1.0

[47] H. Wada, J. Suzuki and K. Oba, "A Feature Modeling Support for Non-Functional Constraints in Service Oriented Architecture," In Proc. of the 4th IEEE International Conference on Services Computing (SCC), Salt Lake City, UT, July 2007

[48] H. Wada, J. Suzuki and K. Oba, "Modeling Non-Functional Aspects in Service Oriented Architecture," In Proc. of the 3rd IEEE International Conference on Services Computing (SCC), Chicago, IL, September 2006.

[49] T. Goovaerts, B. De Win, and W. Joosen., Infrastructural support for enforcing and managing, distributed application-level policies. Electronic Notes, in Theoretical Computer Science, 197(1):31–43, Feb., 2008.

[50] Apache Axis, http://ws.apache.org/axis/

[51] Evren Sirin, , James Hendler, and Bijan Parsia. Semi-automatic composition of web services using semantic descriptions. In Web Services: Modeling, Architecture and Infrastructure workshop in ICEIS 2003, Angers, France, April 2003.

[52] Liu, Y., Ngu, A. H., and Zeng, L. Z. 2004. QoS computation and policing in dynamic web service selection. In Proceedings of the 13th international World Wide Web Conference on Alternate Track Papers &Amp; Posters (New York, NY, USA, May 19 - 21, 2004). WWW Alt. '04. ACM, New York, NY, 66-73.

[53] Dimitrakos, T., Brossard, D. and de Leusse, P., Securing business operations in an SOA, BT Technology Journal Vol. 26, Issue 2, BT, 2009.

[54] Dimitrakos, T., Brossard, D., de Leusse, P. and Nair, S. K., Security of Service Networks, In Handbook of Information and Communication Security, Stavroulakis, P. and Stamp, M. (eds.), pp 351-382, Springer-Verlag, 2010, ISBN 978-3-642-04116-7

[55] HP SOA Governance Interoperability Framework (GIF), Governance interoperability framework reference, February 2008

[56] Dudley, C., Rieu, L., Smithson, M., Verma, T., Braswell, B., WebSphere Service Registry and Repository Handbook, Redbooks, IBM, March 2007

[57] Zhang, S. Integrating Non-Functional Properties to Architecture Specification and Analysis, in Third International Conference on Information Technology: New Generations, 2006

[58] Galster, M., Bucherer, E., A Taxonomy for Identifying and Specifying Non-functional Requirements in Service-oriented Development, in IEEE Congress on Services 2008, 2008

[59] Dobson, G., Hall, S., Kotonya, G., A Domain-Independent Ontology for Non-Functional Requirements, in IEEE International Conference on e-Business Engineering, 2007

[60] OASIS, Extensible Access Control Mark Up Language (XACML) v2, 2004, available at: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

[61] Moritz Y. Becker, Cedric Fournet, Andrew D. Gordon, SecPAL: Design and Semantics of a Decentralized Authorization Language, Technical Report MSR-TR-2006-120, Microsoft Research, September 2006.

[62] W3C Member Submission, Web Services Addressing (WS-Addressing, WS-A), 10 August 2004, available at: http://www.w3.org/Submission/ws-addressing/

[63] W3C Member Submission, Web Services Policy 1.2 - Framework (WS-Policy), 25 April 2006, available at: http://www.w3.org/Submission/WS-Policy/

[64] WS-I, Basic Profile Version 1.0, 16 April 2004, available at: http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html

[65] Microformats, hCard 1.0, available at: http://microformats.org/wiki/hcard

[66] Armbrust, M., Fox, A., Griffith, R., Joseph, A., D., Katz, R., H., Konwinski. A., Lee, G., Patterson D., A., Rabkin, A., Stoica, I. and Matei ZahariaAbove the Clouds: A Berkeley View of Cloud Computing (EECS-2009-28)

[67] Gamma E., Helm R., Johnson R. and Vlissides J., (1993). Design patterns: abstraction and reuse of object-oriented design. In Proceedings of the ECOOP'93 Conference, Kaiserslautem, Germany; published by Springer Verlag

[68] OASIS Web Services Distributed Management (WSDM), www.oasis-open.org/committees/wsdm/

[69] Apache Muse - A Java-based implementation of WSRF 1.2, WSN 1.3, and WSDM 1.1. available at: http://ws.apache.org/muse/

[70] Java Technology Reference, Oracle, available at: http://java.sun.com/reference

[71] Apache Axis2/Java, The Apache Software Foundation, available at: http://ws.apache.org/axis2/

[72] SOAP Version 1.2, W3C Recommendation (Second Edition), 27 April 2007, available at: http://www.w3.org/TR/soap/

[73] Apache Tomcat, The Apache Software Foundation, available at: http://tomcat.apache.org/

[74] eXist-db Open Source Native XML Database, available at: http://exist.sourceforge.net/

[75] Java Servlet technology, Oracle, available at: http://java.sun.com/products/servlet/

[76] XQuery 1.0: An XML Query Language, W3C Recommendation, 23 January 2007, available at: http://www.w3.org/TR/xquery/

[77] XML Path Language (XPath) 2.0, W3C Recommendation, 23 January 2007, available at: http://www.w3.org/TR/xpath20/

[78] Apache Synapse Enterprise Service Bus (ESB), The Apache Software Foundation, available at: http://synapse.apache.org/

[79] The Role of XML Gateways in SOA, Optimizing Performance, Security and Policy Operations, Layer 7 Technologies, White Paper

[80] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., Sink, E., and Stewart, L. 1999. HTTP authentication: Basic and digest access authentication. Internet RFC 2617, June 1999.

[81] de Leusse, P., Brossard, D. and Georgalas, N., Securing business operations in an SOA, Wiley SCN Journal, Special Issue on Security and Trust Management for Dynamic Coalitions, 2010

[82] WS-Trust 1.4, OASIS Standard, 2 February 2009, Available at: http://docs.oasis-open.org/ws-sx/ws-trust/v1.4/os/ws-trust-1.4-spec-os.html

[83] Zeng, L., Benatallah, B., Ngu, A., H., H.,Dumas, M., Kalagnanam, J. and Chang H., QoS-Aware Middleware for Web Services Composition, IEEE Transactions on Software Engineering, p. 311-327, May 2004

[84] Galizia, S., Gugliotta, A. and Domingue, J., A Trust Based Methodology for Web Service Selection, Proceedings of the International Conference on Semantic Computing, p. 193-200, 2007, ISBN. 0-7695-2997-6, IEEE Computer Society

[85] Product and Service Assembly Initiative, available at: http://www.psainitiative.co.uk/

[86] de Leusse, P., Periorellis, P., Dimitrakos, T. et al., Self Managed Security Cell, a security model for the Internet of Things and Services, In Proceedings. The First International Conference on Advances in Future Internet, AFIN 2009, 18-23 June 2009, Athens/Glyfada, Greece, pp 47-52, IEEE Computer Society, 2009

List of acronyms

ADL: Architecture Description Language

AS: Application Server

ASGI: SOA Governance Infrastructures

B2B: Business To Business

B2B2C: B2B To Customer

B2B2G: B2B To Government

BC: Business Capability

BSS: Business Support Systems

CORBA: Common Object Request Broker Architecture

CP: Content Provider

CPU: Central Processing Unit

EAI: Enterprise Application Integration

ESB: Enterprise Service Bus

GIF: Governance Interoperability Framework

IoTS: Internet of Things and Service

IT: Information Technology

ICT: Information and Communication Technologies

JSON: JavaScript Object Notation

MDD: Model-Driven Development tool

MOM: Message Oriented Middleware

NFP: Non-Functional Property

NFR: Non-Functional Requirement

OASIS: Organization for the Advancement of Structured Information Standards

OSS: Operational Support Systems

PSA: Product and Service Assembly

QoS: Quality of Service

REST: REpresentational State Transfer

ROI: Return On Investment

RPC: Remote Procedure Call

SAML: Security Assertion Markup Language

SaaS: Security as a Service

SDF: Service Delivery Framework

SDF ISS: SDF Infrastructure Support Services

SDF SMI: SDF Service Management Interfaces

SDF MSS: SDF Management Support Services

SDP: Service Delivery Platform

SecPAL: Security Policy Assertion Language

SLA: Service Level Agreement

SMC: Self-Managed Cell

SOA: Service Oriented Architecture

SOI: Service Oriented Infrastructure

SSB: Security Service Bus

TMF: TeleManagement Forum

TOGAF: The Open Group Architecture Framework

UDDI: Universal Description Discovery and Integration

UDP: User Datagram Protocol

UML: Unified Modelling Language

VAS: Value Adding Service

VHE: Virtual Hosting Environment

VMS: Virtual Music Store

VO: Virtual Organisation

W3C: World Wide Web Consortium

WS: Web Service

WS-A: WS Addressing

WS-I: WS Interoperability

WS-DM: WS Distributed Management

WSDL: WS Description Language

WSRR: WebSphere Service Registry and Repository

XACML: eXtensible Access Control Markup Language

XML: eXtensible Markup Language

XSL: eXtensible Stylesheet Language

XSLT: XSL Transformation

YAML: YAML Ain't Markup Language

Appendix

1. Profile description XML schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ws-p="com.bt.ws-profile"
targetNamespace="com.bt.ws-profile" elementFormDefault="qualified"
attributeFormDefault="unqualified">
	<xs:element name="WS-Profile" type="ws-p:tProfile">
		<xs:annotation>
			<xs:documentation>Schema for WS-Profile</xs:documentation>
		</xs:annotation>
	</xs:element>
	<xs:element name="Constraint" type="ws-p:tConstraint">
		<xs:annotation>
			<xs:documentation>Schema for WS-Profile</xs:documentation>
		</xs:annotation>
	</xs:element>
	<xs:complexType name="tProfile">
		<xs:annotation>
			<xs:documentation>tProfile is the top level node of a WS-Profile. It
contains all the information expressed or required.</xs:documentation>
		</xs:annotation>
		<xs:sequence>
			<xs:element name="name" type="xs:string"/>
			<xs:element name="ServiceType" type="ws-p:tServiceType"
minOccurs="0" maxOccurs="unbounded"/>
			<xs:element name="Constraint" type="ws-p:tConstraint"
minOccurs="0" maxOccurs="unbounded"/>
			<xs:element name="CoordinationPlan" type="ws-p:tProfile"
minOccurs="0" maxOccurs="unbounded"/>
		</xs:sequence>
		<xs:attribute name="ID" type="ws-p:tUUID"/>
	</xs:complexType>
	<xs:element name="Capability" type="ws-p:tServiceType"/>
	<xs:complexType name="tServiceType">
		<xs:annotation>
			<xs:documentation>tServiceType is the top level of abstraction for an
activity in the WS-Profile.</xs:documentation>
		</xs:annotation>
		<xs:complexContent>
			<xs:extension base="ws-p:tActivity">
				<xs:sequence>
					<xs:element name="Interface" type="ws-p:tInterface"
minOccurs="0" maxOccurs="unbounded"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:complexType name="tInterface">
		<xs:annotation>
			<xs:documentation>tInterface may allow to differentiate different
parts of the same activity in the WS-Profile. For instance, one activity could have both
management and operational interfaces.</xs:documentation>
		</xs:annotation>
		<xs:complexContent>
			<xs:extension base="ws-p:tActivity">
				<xs:sequence>
					<xs:element name="Functionality" type="ws-
```

```
p:tFunctionality" minOccurs="0" maxOccurs="unbounded"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="tFunctionality">
                <xs:annotation>
                        <xs:documentation>tFunctionality may allow to differentiate different
functions of the same interface in the WS-Profile. For instance, one interface could allow to
manage users and items, these would be two different functionalities.</xs:documentation>
                </xs:annotation>
                <xs:complexContent>
                        <xs:extension base="ws-p:tActivity">
                                <xs:sequence>
                                        <xs:element name="Operation" type="ws-
p:tOperation" minOccurs="0" maxOccurs="unbounded"/>
                                </xs:sequence>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="tOperation">
                <xs:annotation>
                        <xs:documentation>tOperation links to the concrete operations of a
functionality in the WS-Profile. For instance, one user management functionality could have
operations about deleted, updating or creating users.</xs:documentation>
                </xs:annotation>
                <xs:complexContent>
                        <xs:extension base="ws-p:tActivity">
                                <xs:sequence/>
                        </xs:extension>
                </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="tConstraint">
                <xs:annotation>
                        <xs:documentation>tConstraint allows connecting constraints to the
different levels of abstraction in the WS-Profile. For instance, an operation could require
another one to called immediatly before in order to work.</xs:documentation>
                </xs:annotation>
                <xs:choice>
                        <xs:sequence>
                                <xs:element name="PolicyType" type="xs:string"/>
                                <xs:element name="Location" type="xs:string"/>
                        </xs:sequence>
                        <xs:sequence>
                                <xs:element name="BeforeorAfter">
                                        <xs:simpleType>
                                                <xs:restriction base="xs:string">
                                                        <xs:enumeration value="before"/>
                                                        <xs:enumeration value="after"/>
                                                </xs:restriction>
                                        </xs:simpleType>
                                </xs:element>
                                <xs:element name="Activity" type="xs:string"/>
                        </xs:sequence>
                        <xs:sequence>
                                <xs:element name="ActivityLocation" type="xs:string"/>
                                <xs:element name="Location" type="xs:string"/>
                        </xs:sequence>
                        <xs:sequence>
                                <xs:element name="misc" type="xs:string"/>
```

```
                                    <xs:element name="value" type="xs:string"/>
                                    <xs:element name="Constraint" type="ws-p:tConstraint"
minOccurs="0" maxOccurs="unbounded"/>
                                </xs:sequence>
                        </xs:choice>
                        <xs:attribute name="ID" type="ws-p:tUUID"/>
                </xs:complexType>
                <xs:complexType name="tActivity">
                        <xs:annotation>
                                <xs:documentation>TActivity defines an abstract component of the
Policy.</xs:documentation>
                        </xs:annotation>
                        <xs:sequence>
                                <xs:element name="name" type="xs:string"/>
                                <xs:element name="Constraint" type="ws-p:tConstraint"
minOccurs="0" maxOccurs="unbounded"/>
                        </xs:sequence>
                        <xs:attribute name="ID" type="ws-p:tUUID"/>
                </xs:complexType>
                <xs:simpleType name="tUUID">
                        <xs:restriction base="xs:string">
                                <xs:pattern value="[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-
9]{12}"/>
                        </xs:restriction>
                </xs:simpleType>
</xs:schema>
```

Table 15. WS-Profile XML Schema

2. Context description XML schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ws-c="com.bt.ws-
context" targetNamespace="com.bt.ws-context" elementFormDefault="qualified"
attributeFormDefault="unqualified">
	<xs:element name="WS-Context" type="ws-c:tContext">
		<xs:annotation>
			<xs:documentation>Schema for WS-Context</xs:documentation>
		</xs:annotation>
	</xs:element>
	<xs:complexType name="tContext">
		<xs:sequence>
			<xs:element name="name" type="xs:string"/>
			<xs:element name="configData" type="ws-c:tConfigData"
minOccurs="0" maxOccurs="unbounded"/>
		</xs:sequence>
		<xs:attribute name="ID" type="ws-c:tUUID"/>
	</xs:complexType>
	<xs:complexType name="tConfigData">
		<xs:sequence>
			<xs:element name="target" type="ws-c:tUUID"/>
			<xs:element name="data" type="ws-c:tUUID" minOccurs="0"
maxOccurs="unbounded"/>
			<xs:element name="misc" type="xs:anyType" minOccurs="0"
maxOccurs="unbounded"/>
		</xs:sequence>
		<xs:attribute name="ID" type="ws-c:tUUID"/>
	</xs:complexType>
	<xs:simpleType name="tUUID">
		<xs:restriction base="xs:string">
			<xs:pattern value="[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-
9]{12}"/>
		</xs:restriction>
	</xs:simpleType>
</xs:schema>
```

Table 16. WS-Context XML Schema

3. Context Selector description XML schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:ws-cs="com.bt.ws-
context-selector" targetNamespace="com.bt.ws-context-selector"
elementFormDefault="qualified" attributeFormDefault="unqualified">
	<xs:element name="WS-ContextSelector" type="ws-cs:tContextSelector"/>
	<xs:complexType name="tContextSelector">
		<xs:sequence>
			<xs:element name="name" type="xs:string"/>
			<xs:element name="profile" type="ws-cs:tUUID" minOccurs="1"
maxOccurs="unbounded"/>
			<xs:element name="context" type="ws-cs:tUUID" minOccurs="1"
maxOccurs="unbounded"/>
			<xs:element name="internalTarget" type="ws-cs:tUUID"/>
			<xs:element name="upTime" type="ws-cs:tValidity" minOccurs="1"
maxOccurs="1"/>
			<xs:element name="configSelector" type="ws-cs:tSelector"
minOccurs="1" maxOccurs="unbounded"/>
		</xs:sequence>
		<xs:attribute name="ID" type="ws-cs:tUUID"/>
	</xs:complexType>
	<xs:complexType name="tValidity">
		<xs:sequence>
			<xs:element name="type" type="ws-cs:tPeriod" minOccurs="0"
maxOccurs="1"/>
			<xs:element name="when" type="ws-cs:tDateAndLength"
minOccurs="0" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:simpleType name="tPeriod">
		<xs:restriction base="xs:string">
			<xs:enumeration value="always"/>
			<xs:enumeration value="on-request"/>
			<xs:enumeration value="time-frame"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:complexType name="tDateAndLength">
		<xs:sequence>
			<xs:element name="startdate" type="xs:dateTime" maxOccurs="1"/>
			<xs:element name="period" type="xs:duration" maxOccurs="1"/>
		</xs:sequence>
		<xs:attribute name="ID" type="ws-cs:tUUID"/>
	</xs:complexType>
	<xs:complexType name="tSelector">
		<xs:sequence>
			<xs:element name="target" type="ws-cs:tUUID"/>
			<xs:element name="data" type="xs:string"/>
			<xs:element name="operation" type="ws-cs:tOperation"/>
		</xs:sequence>
		<xs:attribute name="ID" type="ws-cs:tUUID"/>
	</xs:complexType>
	<xs:simpleType name="tOperation">
		<xs:restriction base="xs:string">
			<xs:enumeration value="request"/>
			<xs:enumeration value="response"/>
			<xs:enumeration value="both"/>
		</xs:restriction>
	</xs:simpleType>
	<xs:simpleType name="tUUID">
```

```
            <xs:restriction base="xs:string">
                <xs:pattern value="[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}"/>
            </xs:restriction>
        </xs:simpleType>
</xs:schema>
```

Table 17. WS-ContextSelector XML Schema

4. Publications

Book chapter

[1] Dimitrakos, T., Brossard, D. and de Leusse, P., Security of Service Networks, Stavroulakis, Peter; Stamp, Mark (Eds.), 1st Edition., 2010, XX, 800 p. 250 illus., Hardcover, ISBN: 978-3-642-04116-7, 2010, available from: http://www.springer.com/engineering/signals/book/978-3-642-04116-7

Refereed journal articles

[2] Dimitrakos, T., Brossard, D. and de Leusse, P., Securing Business Operations in SOA, BT Technology Journal, vol.27, no.2 (https://www.btplc.com/Innovation/Journal/BTTJ/current/HTMLArticles/Volume26/22Securing/Default.aspx)

[3] de Leusse, P., Brossard, D. and Georgalas, N., Securing business operations in an SOA, Wiley SCN Journal, Special Issue on Security and Trust Management for Dynamic Coalitions, 2010

Refereed conference publications

[4] de Leusse, P. and Dimitrakos, T., SOA-based security governance middleware, The Fourth International Conference on Emerging Security Information, Systems and Technologies, IEEE Computer Society, July 18-25, 2010, Venice/Mestre, Italy

[5] de Leusse, P., Dimitrakos, T., and Brossard, D., A governance model for SOA, IEEE 7th International Conference on Web Services (ICWS 2009), July 6-10, 2009, Los Angeles, CA, USA

[6] de Leusse, P. and Brossard, D., Distributed systems security governance, a SOA based approach, Third IFIP International Conference on Trust Management, Springer, June 15-19, 2009, Purdue University, West Lafayette, USA

[7] de Leusse, P., Periorellis, P., Dimitrakos, T., and Srijith K. Nair, Self Managed Security Cell, a security model for the Internet of Things and Services, The First International Conference on Advances in Future Internet, AFIN 2009, IEEE Computer Society, June 18-23, 2009, Athens/Vouliagmeni, Greece, Best paper award

[8] de Leusse, P., Periorellis, P., Watson, P. and Dimitrakos, T., A semi autonomic infrastructure to manage non functional properties of a service, In UK e-Science All Hands Meeting 2008, 8-11 September, Edinburgh, UK, National e-Science Centre, 2008

[9] de Leusse, P., Periorellis, P., Watson, P. and Maierhofer, A., Secure & Rapid Composition of Infrastructure Services in the Cloud, In The Second International Conference on Sensor Technologies and Applications, SENSORCOMM 2008, 25-31, August 2008, Cap Esterel, France, pp 770-775, IEEE Computer Society, 2008

[10] de Leusse, P., Periorellis, P., Dimitrakos, T. and Watson, P., An architecture for non functional properties management in distributed computing, Doctoral Consortium on Software and Data Technologies - Proceedings of the Doctoral Consortium on Software and Data Technologies, DCSOFT 2008, In Conjunction with ICSOFT 2008, 2008, Pages 26-37

Non-Refereed Publications

[11] de Leusse, P., Periorellis, P. and Watson, P., CS-TR No 1037 Enterprise Service Bus: An overview, School of Computing Science, Newcastle University, Jul 2007