

REST and WS-\* Architectures

# SOA Governance IN ACTION

Jos Dirksen

MEAP

 MANNING





**MEAP Edition  
Manning Early Access Program  
SOA Governance in Action version 4**

Copyright 2012 Manning Publications

For more information on this and other Manning titles go to  
[www.manning.com](http://www.manning.com)

# *Table of Contents*

## **Part 1: Introduction**

- 1 Introducing SOA governance
- 2 Setting up the SOA governance environment
- 3 Using a case study to understand SOA governance

## **Part 2: Design time policies**

- 4 Service design and documentation policies
- 5 Security policies
- 6 Testing, performance, and the cloud

## **Part 3: Using tools for supporting SOA governance at runtime**

- 7 Using tools for runtime governance
- 8 Registration, discovery and lookup of services and policies
- 9 Using business activity monitoring

Appendixes

# 1

## *Introducing SOA Governance*

This chapter covers:

- What SOA Governance is
- Why SOA Governance is needed
- What the advantages are of applying SOA Governance
- What role tooling and open source play in SOA Governance

When working in a large company you often have many different departments building services and applications. For instance services that offer access to the customers database, create new accounts or that can be used to register possible prospects. The goal of offering this functionality, in a service-oriented manner, is to promote reuse and a standard way of working. However, if you don't have a process in place that can provide a set of guidelines on how these services must be created and must behave, these services will be created using different standards, different best practices and it will be very difficult to determine the quality of these services. With SOA Governance you define a set of guidelines your services need to comply with. Such a set of guidelines will make it easier to create new services, upgrade existing ones and monitor how your services are used.

When people first hear about SOA Governance they often think of large organizations, heavy processes, lots and lots of paperwork that pretty much prevents you, as a developer, of getting any work done. If you've read one of the books that have SOA Governance in the title, this view will be somewhat confirmed. SOA Governance, especially the governance part, sounds very heavy and restrictive, and this is something that can quickly scare people. But don't worry as we'll show you in this book, applying SOA Governance principles isn't that hard and not so different from the normal way you design or monitor the services you've created.

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=778>

Governance isn't something exclusive to IT, as we'll see in this chapter. It's applied throughout the industry. Let me give you an example of what happens in the aviation industry. In this industry governance is the most important way to make sure that airplanes are safe and don't drop out of the sky on a regular basis. In the aviation industry everything from construction, to maintenance, to flight monitoring happens under the strictest regulations. Every screw and bolt needs to be accounted for and every part of the plane is validated and exhaustively tested before it may be used. For this they use a strict set of governance guidelines to control and validate that the aircraft is constructed in a safe and controllable manner using materials they know the exact properties for. The services and applications you're developing most likely won't cause airplane crashes or nuclear explosions, but having a good set of guidelines, or policies as we'll call them, is important to make sure your services comply with the guidelines defined by your organization, and behave as you expect.

When you look at the organizational part of SOA Governance you'll indeed have to deal with various administrative processes, follow regulations and all this won't have much to do with actual software development. This, however, is only one part, and a very important one, of SOA Governance. During this phase the policies will be defined that you as a software developer will have to follow.

Many people think that you only need SOA Governance when you've got heavy WS-\* based architectures. This isn't the case, regardless of the technology you use for creating your SOA, be it REST based or WS-\* based, you'll need some sort of governance to assure that all your services follow the same security, quality, performance requirements required by your organization.

In this first chapter we'll dive directly into the details of SOA Governance. We'll explain why SOA Governance is important, what the benefits are when you've got SOA Governance in place, and of course we'll give an overview of how you can deal with SOA Governance in a practical and pragmatic manner. In the following chapters we'll show you how to start using it.

## **1.1 What is SOA Governance**

To understand what SOA Governance is we of course first have to look a bit deeper at what SOA is and what governance is. In this section we'll look at what SOA is and what Governance is. We'll start with the definition of SOA.

### **1.1.1 Definition of Service Oriented Architecture**

Let's start by looking what Wikipedia has to say about this. Although not an authority on the subject of SOA it gives a good idea what a lot of people think about when talking about SOA:

## DEFINITION

“Service-oriented architecture (SOA) is a flexible set of design principles used during the phases of systems development and integration in computing. A system based on a SOA architecture will provide a loosely-coupled suite of services that can be used within multiple separate systems from several business domains.” Wikipedia: [http://en.wikipedia.org/wiki/Service-oriented\\_architecture](http://en.wikipedia.org/wiki/Service-oriented_architecture)

What you see in this quote is that it only focuses on the technical aspect of SOA; we're talking about service design principals, systems development and integration. Although this is an important part of SOA, there is more when it comes to SOA. When you talk about SOA you should look at it from a couple of different angles. To really define SOA we should look further than just the technical aspect.

### **SOA != WS-\* + SOAP + UDDI + WSDLs**

As you've probably read from the table of contents or the introduction to this chapter, when we talk about SOA or services in general in this book, we don't necessarily mean the traditional web services stack. Even though often SOA is equated with using the standards based WS-\* stack, this is only one possible solution. When you look at is currently deployed in the 'enterprise', you'll mostly see the traditional WS-\* approach. In the Web 2.0 space, to give it a name, we see the opposite. When web services first became popular you saw a rise of public APIs based on SOAP, WSDLs and XML. The last couple of years though, especially in the public space, these types of services have pretty much all disappeared or have been replaced with REST based services. A similar trend is going on in the enterprise space. Not so drastic as on the Internet, but we see that, also in the enterprise, the value of a REST architecture has been accepted. We're now slowly going to a situation where the best solution is used for a problem. This doesn't mean the WS-\* stack is going anywhere soon. What you'll see is these two architectural types running side by side. In this book we'll look at both WS-\* and REST and see how governance can be applied to these kind of services.

Basically when we talk about SOA we talk about the following different views:

- Business view: This view focuses on the value and the advantages SOA offers for the business. This is an important view because ultimately we adopt a SOA architecture to improve the way we do business. From this perspective SOA can help to quickly create new products, adapt to changes in the market, reduce costs and improve the return on investment (ROI) etc.;
- IT view: From this point of view we want to know how SOA can help the IT in quickly adapting to changes. Using SOA the IT department can save costs by reusing services and can better determine who needs to be billed for the usage of services.;
- Technical view: The services provided to the business should be designed following a

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=778>

set of SOA best practices. This view focuses on how services should be created by defining the application architecture, technical guidelines and best practices.

In figure 1.1, you see a very simple use case where an organization wants to make it easy for its customers to request access to a specific service for which they need an API key.

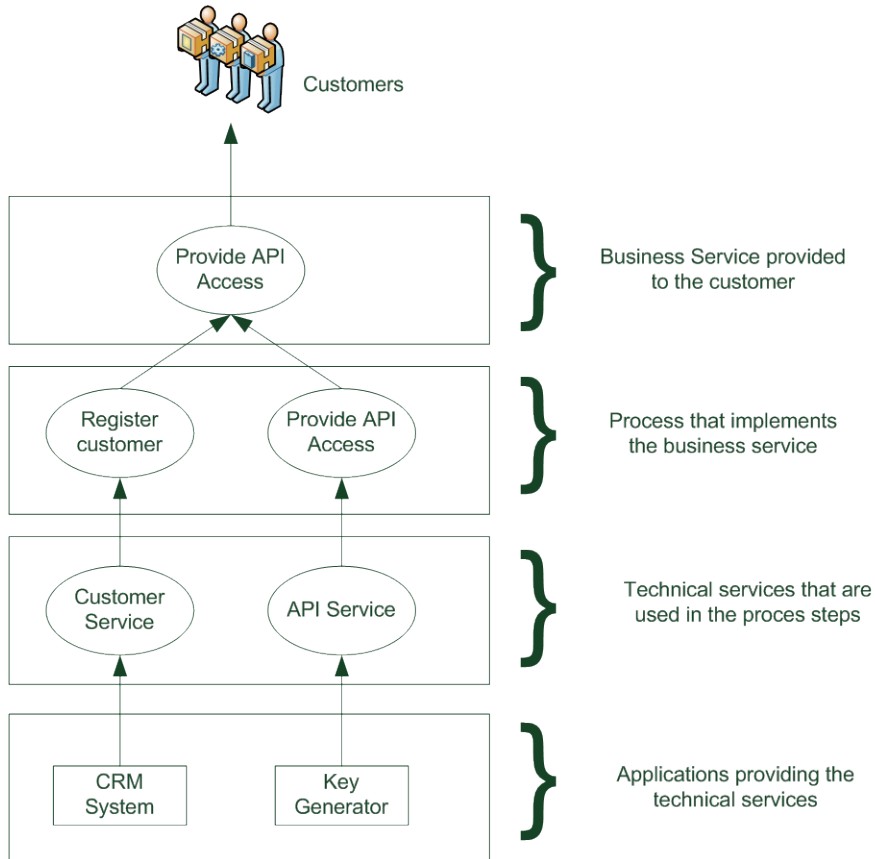


Figure 1.1: This figure shows how the various views on SOA combined to provide a product to a customer

You can compare this with the process you have to go through to get an API key for the services Google provides. The business point of view here is that they want to provide this functionality to their users to get as many users as possible. More users means more custom mashups, and in the case of Google ultimately more advertisement revenue. From the IT point of view they want to provide a simple set of reusable services. So that next time the business wants some small changes or provide another 'product' to their customer they can

do this as quickly and efficiently as possible. And finally, from a technical viewpoint, for this scenario we need to provide the actual implementation of the services provided. And of course we want to do this following best practices and standards.

So let's summarize what SOA is. SOA defines a set of technical and non-technical guidelines and best practices you should follow when developing services. If you follow these guidelines you create services that allow the business to better serve their customers and help the IT in bringing down costs and improve the reuse of existing functionality.

Before we look a bit deeper at the Governance part, let's quickly look closer at what the advantages are that we can gain by correctly applying SOA. The following list shows some of the advantages a SOA offers:

#### **ADVANTAGES OF SOA**

- **Business agility/reduced time to market:** This is of course one of the main advantages a company hopes to achieve when applying SOA principles. With more agility a company can better respond to changes in the market and quickly launch new products and services. Note that this doesn't only apply to internal applications and services, with all the REST and cloud services available nowadays it's much easier for businesses to quickly get products and functionality.
- **Reduced costs:** One of the other main 'business' reasons. When everything is going well, for instance during the .com boom, money wasn't that hot an issue. Technology companies and IT departments received all the funding they wanted, whether the business, or the venture capitalists, really knowing what they expect back from it. With SOA businesses want to reduce costs by re-use, standards based development and a clear view of what services are out there and the functionality they provide.
- **Improve reuse of services:** If the services are better defined, and a clear inventory of the services is kept, it's much easier to start reusing the services out there. This is once again an example where we're not just talking about internal services, but also about reusing existing services on the web. In this last category you can think about the cloud based services provided by Amazon, Microsoft's Azure platform, or any of the services out there. By the way a nice overview of available services can be found at <http://www.programmableweb.com/>.
- **Improving software quality:** Within a SOA we've got a set of defined standards and best practices. In other words we've learned how to build services, what to do, and what not to do. This will lead to a higher quality of software. Another advantage is that since we're reusing existing services we don't have to reinvent the wheel every time, assuming the service we're reusing is being well maintained.
- **Better interoperability:** Whether you're building a REST based service or a WS-\* based service, in both cases you've got a well defined contract, based on standards to help you in the interoperability department.

Now that we've look at what SOA is, let's look at the governance part

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=778>

### **1.1.2 Introducing governance**

Most people have probably heard the term governance in one way or the other. Usually when people talk about governance they talk about corporate governance. Corporate governance defines a set of rules, laws, policies and regulations that affect how a corporation should be run. Corporate governance should make sure that corporations are run correctly, efficiently and responsibly. Well executed corporate governance makes sure that all the stakeholders in a corporation are represented properly.

#### **CORPORATE GOVERNANCE**

When you look back at the last couple of years though we've seen a lot of things go wrong in this area. The crisis in the financial market, various stock market scandals and large corporations going bankrupt are all examples of this. This however, doesn't mean we should stop with corporate governance, what this means is that even though you can define all the processes, regulations, laws and what more. You still need some way to enforce and control the policies you've made.

#### **The Enron scandal**

One of the main reasons governance has become an important part of how a business operates is because of the scandals at the beginning of last decade. The most prominent one of these scandals was the Enron one. Enron, which was an energy corporation from Houston, at its height, had a value of \$111 billion, a year later it filed for bankruptcy. In the nineties the energy market in California was deregulated and Enron quickly became one of the largest energy companies in the US. However in 2001 investigations were started to look into the financial position of Enron, and all kinds of fraudulent practices were revealed. Enron, for instance, stored its debts in foreign accounts and used its political influence to raise the price of energy. To make matters even worse, high ranking Enron executives sold most of their stock when the shares were at \$90, the highest the shares reached. They did this, since they knew Enron was making massive losses. On the other hand the public was encouraged to buy stocks, which within a few months, dropped to thirty cents. As a result the Enron executives were charged with bank fraud, securities fraud, wire fraud, money laundering, conspiracy and insider trading. As a result of the Enron scandal, the federal government passed the Sarbanes-Oxley act (SOX for short) which forces companies to follow a set of policies with regards to reporting information to their investors and that companies have strict internal financial control mechanisms in place.

#### **IT GOVERNANCE**

Another area where governance has become more important the last decade or so, is in the area of IT Governance. During the big .com bubble and the Y2K problems IT spending went through the roof. It was very hard for the business to see where the money was going to, and what the IT was actually doing. Basically the goal of IT Governance is to minimize the risks of IT projects and make sure that IT provides actual business value. If you consider

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=778>

that, depending on who you believe, almost two thirds of all IT projects fail, you'll understand the need for a good governance body. A more reasonable percentage was given by "Standish Group International", and is shown in Figure 1.2.

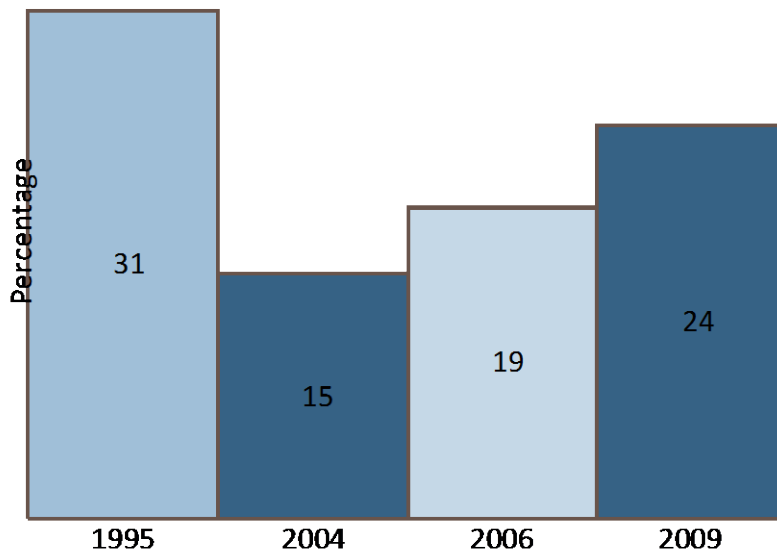


Figure 1.2. Failed projects in 1995, 2004, 2006 and 2009

### IT projects failures

In the previous sidebar I used Enron as an example why governance within a company is needed. IT projects also have a tendency to go seriously wrong if there isn't a good governance or control mechanism in place. One of the most talked about examples is the Denver Airport Baggage System. This baggage system, created for the new to build Denver Airport, would route all the passenger's bags to and from aircrafts without any human intervention. Even though this system was eventually completed (\$55M over budget), it basically didn't work. The carts which were used to automatically transfer the bags couldn't cope with sharp corners, sensors lost track of where bags were in the system, and more problems were present in the system. All this caused the Denver Airport to open 16 months later, while the system was being fixed, and added a total of \$560M to the cost of the airport. Another example is the automated fulfillment system developed for Sainsbury's, a British supermarket. Sainsbury's wanted a new system for its main distribution center. This barcode-based system would save a huge amount of money and increase efficiency since a lot of human tasks could be automated. In the end though, the system, installed in 2003, failed because of apparent barcode-reading errors.

After two years of bug fixing Saintsbury's wrote that the system worked as intended. In 2007 however the complete system was scrapped. Total write off: £150M.

If you ever need to start with IT Governance, there are a number of frameworks that can guide you in implementing IT Governance. I've listed a couple of them in table 1.1, and provided a link to additional information.

**Table 1.1. IT Governance frameworks**

<b>Name</b>	<b>Description</b>
ITIL	<p>The <b>Information Technology Infrastructure Library</b> defines a set of best practices and concepts you can use to setup the IT Governance processes within your organization. It for instance defines best practices for security management, ICT Infrastructure Management, Software Asset Management and much more.</p> <p>For more information can be found on the official ITIL website: <a href="http://www.itil-officialsite.com/home/home.asp">http://www.itil-officialsite.com/home/home.asp</a></p>
CMM	<p>The <b>Capability Maturity Model</b> defines the level of maturity on organization is on with regards to software development. It defines five levels, where on level one (called initial) software development is done without any process and control and on level five (called optimizing) the software development process is already very mature and only small parts can be optimized. Although not specifically a IT Governance framework. You can use CMM to measure the maturity of your current governance related processes and best practices.</p> <p>Information on CMM can be found at: <a href="http://www.sei.cmu.edu/cmml/start/">http://www.sei.cmu.edu/cmml/start/</a></p>
COBIT	<p><b>Control Objectives for Information and related Technology</b> is a framework that can help you setup IT Governance for your organization. It provides tools, models and methodologies for this.</p> <p>More information on COBIT can be found at the official COBIT website: <a href="http://www.isaca.org/Knowledge-Center/COBIT/Pages/Overview.aspx">http://www.isaca.org/Knowledge-Center/COBIT/Pages/Overview.aspx</a></p>

What you see in both corporate governance and IT governance is that governance will fail if not all the stakeholders are involved with the critical decision making. That is the main reason scandals such as Enron happen, and why so many IT projects go wrong.

### 1.1.3 Defining SOA Governance

The goal of applying governance on SOA is to get the most out of your SOA and make sure that you create high quality services. We do this by listening to the stakeholders and, based on their input, define a number of policies.

Basically what we need to do, is take the following steps:

1. Define the policies we want to apply.
2. Apply these policies during design time.
3. Monitor and enforce the policies during runtime.

It's important to know that SOA Governance should be applied as an iterative approach. When you've executed these steps you aren't done. Based on the information you learn from step three, and other inputs, you might want to adjust the policies that you have defined.

Let's look a bit closer at the first item on the list.

#### DEFINE THE POLICIES WE WANT TO APPLY

This step is mostly an organizational step where you get all the stakeholders together (for instance in a SOA Governance board) and based on the strategy and goals of the company coordinate and control all the various SOA efforts. The organizational part of SOA, which is what we're talking about here, is an important part of SOA Governance. If there is no backing from your stakeholders it's very hard to apply SOA Governance effectively and define the correct policies to implement and enforce. What we mean by this is that besides the technical aspect of applying the policies we define we also need to take into account the role the process and the people play in regards to SOA Governance. These concepts are sometimes called the three P's: People, Process and Policy.

- **People:** To effectively apply SOA Governance you need to know who the business owners of your services are. Who is using your services, why is your service being used and who is technically responsible for keeping your service up and running?
- **Processes:** What processes are in place to define your policies? Do you have lifecycle processes in place for your services? What business processes are depending on your service? Is there a process in place to determine whether you services implement the defined policies?
- **Policies:** Which policies are defined for your service and how are they applied during design and runtime.

A number of books have been written on these specific topics that dive into the details of the process and people part of SOA Governance. This book however focuses on the practical approach of SOA Governance. We do look at the lifecycle of a service and the lifecycle of a policy, but won't dive into the details of the Processes and People part of the three P's.

Once the policies have been defined, we can look at how we apply those during design time. For instance let's assume your organization has defined the following policy regarding the documentation of your services:

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=778>

*"All the services that are provided to external clients must have documentation explaining all the service operations. This documentation must explain what the operation does, must explain all the arguments the operation takes, and must describe the results of the operation. Furthermore, if there is a logical sequence in which operation need to be called, this flow should be described as well"*

### What is a policy?

In this book we'll often talk about policies. When we talk about a policy, we mean a policy as defined by OASIS in their SOA Reference model (you can find more on this model here: [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=soa-rm](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm)). You can read the complete definition if you follow this link, I'll just summarize the most important parts here. A policy consists out of three parts: the policy assertion, the policy owner and policy enforcement. The assertion of a policy for instance is "All the messages are encrypted". This is an assertion which can be measured; it's either true or false. The second part of a policy is its owner. In the example above, a service consumer can make this assertion and enforce it independently from the provider. Finally, a policy may be enforced, this can be done technically, but can also be done for instance by having reviews.

In the second section of the book we'll elaborate on specific policies regarding documentation, but for now this small summary will suffice.

#### APPLY THE POLICIES DURING DESIGN TIME

During design time you have to take this, and other policies into account and provide an adequate solution. Let's look back at the aviation example from the introduction to this chapter. Design time policies also apply to the aviation industry. When an airplane is being designed it has to comply to all different kinds of government legislation and safety protocols. For instance it must have multiple backups for the primary system, it should only emit so much CO2 and it must be able to land on just two engines.

Let's assume you're working at the IT department of your hometown and you're asked to create a service which allows the clients to retrieve a list of all the provided building permits for a specific area. Since this is a public service you decide to use a REST based service for this (the technical type of service to provide in a specific scenario could also be a specific policy). Now you need to make sure you can fill in the requirements of the policy for this service. An example of the supplied documentation could be the following (which could be provided on the city's website as a simple HTML document):

**Name:** City of Seaford: Building Permits service.

**Description:** This document describes the operations provided by the City of Seaford to their residents. This service can be used to retrieve information about the currently approved building permits for a specific region within the city limits.

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=778>

**URI:** {serviceURI}/permits?postalCode=?{postalCode}&range={range}  
**Method:** GET  
**Example:** http://api.seaford.org/services/public/permits?postalCode=90210&range=300  
**Description:** This URI can be used to find a set of permits that match the provided search criteria.  
**Arguments:** {postalCode} the postal code that serves as the center of the search region. If no, or a postal code outside the city is provided, this search will yield no results. {range} the range in yards to search for. If a negative range is used no results will be returned. If no range is provided, the search will default to a one mile radius.  
**Result:** The result of this operation will be a list of permit resources. The media type of this resource is application/vnd.seaford.org.permit+xml.  
**Links:** In the returned list of permits you'll find a number of links to resources. These possible links are described below:  
*self:* Points to the permit resource itself. This resource is of the type application/vnd.seaford.org.permit+xml  
*Location:* Points to the exact location of this permit. This resource is of the type: application/vnd.seaford.org.location+xml  
*owner:* Points to the owner of the permit. This resource is of the type application/vnd.seaford.org.permit.owner+xml  
*status:* Shows the current status of this permit. This resource is of the type application/vnd.seaford.org.permit.status+xml

If we had decided to do this service as a WS-\* based service, we would have annotated the WSDL with the correct information on the provided operations. You can find more on this subject starting from chapter 4.

#### **MONITOR AND ENFORCE THE POLICIES DURING RUNTIME**

The third part of SOA Governance deals with enforcing and checking the policies at runtime. If we just spend time defining policies but have no means of checking whether they are followed, it's not much use defining the policies in the first place. For this we need a mechanism to check whether the policies we defined are followed. For an airplane you want to measure the fuel consumption to see whether it's within defined parameters, you check whether the back-up systems are functioning and so on.

To make it clearer we'll have a quick look at a simple security policy:

*"All calls to the publicly provided services should be done over a secure channel."*

This is a very simple security policy and you'll probably know how to comply with this service. If we look back at the previous service we discussed, the service providing information on permits, we see that this service should comply with the policy we've defined. At design time we don't really have to worry about this policy, whether we're running securely or not, our service interface and implementation doesn't have to chance to comply with this policy. This is however a policy we have to enforce at runtime. Following this particular policy isn't that hard. If we can force all the calls to our service to be done over

HTTPS, we'll comply with this policy. What you can see in figure 1.3 is that by using Apache as a filter we can make sure all calls are done over HTTPS.

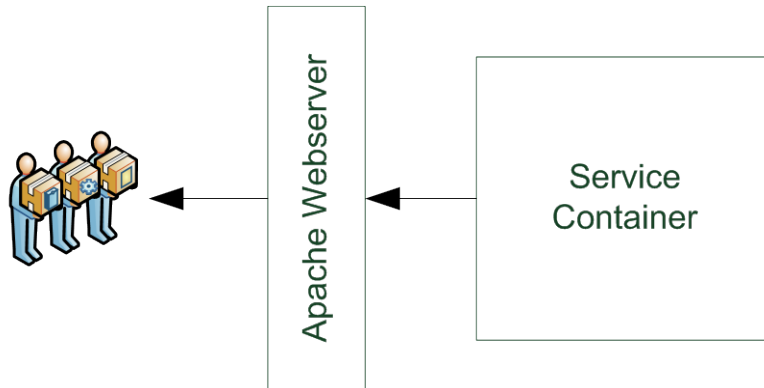


Figure 1.3 Show a basic implementation how you can make sure all the calls to the publicly provided service are done over a secure channel

We could also configure Apache in such a way that calls made over normal HTTP are redirected to HTTPS, thus making sure we comply with the requirements set out by the policy.

In section three of the book we'll dive a whole lot deeper into the details of runtime policy enforcement and show you the tools you can use to enforce and check policies at runtime.

In this first section we've looked at SOA, at Governance and finally at what SOA Governance entails. In the next section we'll look at more detail why you need SOA Governance and the advantages it offers.

## 1.2 How using SOA Governance can help

As we've seen in the previous section, the goal of SOA Governance is getting the most out of your SOA. We've already touched upon some of the reasons why applying SOA Governance is a good thing. In this section we'll give an overview of reasons why you need to apply SOA Governance. We'll keep away from the business reasons such as total cost of ownership, time to market and other buzzwords, and look at some of the most important reasons why SOA Governance is needed. As a software developer or architect you're faced with a lot of different challenges when designing and implementing services. Whether you're creating a public REST based service that provides social networking functionality or you're building an internal WS-\* based service to provide accounting information to another department, you have a number of challenges you have to deal with. The following sections give you an

overview of a couple of these challenges, and explain how applying SOA Governance can help you in solving these challenges.

### **1.2.1 Keeping track of how services are used**

The first one we'll look at is when your service isn't used in the way you imagined. When you've created a service and other people start using this service, they expect a certain level of performance and reliability. When you designed this service you probably took this into account, but it's hard to plan for everything.

#### **The service that couldn't keep up**

I remember that we created a service that served as a web service facade to a document management system. During our tests everything was fine, during the customer's test some small issues were found, but nothing that we couldn't quickly fix. During production though, we noticed that the usage pattern of our service wasn't as we expected. Instead of small documents being added, very large documents were added instead. Because of this change of usage, our service was becoming unusable, not just for this client, but this also had impact on the other clients of our service.

On the other hand, it's possible that your team has created a great service, but no one is using it. How can SOA Governance help in these scenarios? It can help by providing you with the following tools:

- **Define a life-cycle for your service:** part of SOA Governance is defining the life-cycle for your service. This means describing the phases a service goes through from inception to retirement. Including in the life cycle are, amongst others, processes defined that describe how the availability of your service is communicated to the other departments or possible clients.
- **Apply and enforce runtime policies:** Without metrics it's impossible to determine if your service is being used as you imagined, how much it's being used and whether you provide the performance your clients expected. When you apply policies at runtime you can use those metrics to quickly find out if your service is struggling.

Another important part of developing services is making sure you have uniformity amongst your services. In the next section we'll look at how SOA Governance can help you in maintaining a good level of quality when you're developing your services.

### **1.2.2 Keeping uniformity amongst services**

If five services are written by five different teams they should follow the same principles with regards to documentation, message design, interoperability and security. SOA Governance can help you in this area:

- **Defining design time policies:** If you define a set of design time policies and create a reference architecture based on these policies, you provide the developers with the

information they need to define consistent services.

- **Setup service review boards:** Just defining these policies however isn't enough, you have to setup regular review sessions, to make sure the services that are designed follow the principles defined through the policies.
- **Standardize messages and facilitate reuse:** An important tool to support SOA Governance is a service repository. Within this repository you can for instance define the messages for your domain, the canonical data model and register the services that are available.
- **Enforce policies at runtime:** With runtime SOA Governance you can enforce certain policies at runtime. You can make sure the correct security levels are used, add additional input validation etc.

Besides the advantages mentioned above, SOA Governance also introduces a number of common pitfalls.

### **1.3 Common pitfalls when introducing SOA Governance**

The following is a list of issues you'll see at a lot of companies who introduce SOA Governance.

- **Introduce too complex governance processes:** If you've look at any of the other SOA Governance books out there you've probably have noticed that they all focus on the governance processes and on the organizational part of SOA Governance. Even though very important, many SOA Governance initiatives get buried under too many rules, governance boards and regulations. It's often easier to start small, be successful and work from that. The information in this book can help you with this.
- **Introduce too simple governance processes:** On the other hand, there are also enough organizations where there are almost no governance processes or governance boards present. They might have some standards and do an occasional service review, but the organizations don't a structure in place. Just as doing too much won't work, doing too little also won't work. You need some sort of structure and processes to at least handle the reviews and that allow you to check whether the policies set out are followed.
- **Too much reliance on tools:** If you listen to the big tool vendors, and even some open source ones, you can buy SOA Governance. Just buy their SOA Registry and you've got a SOA Governance solution. Unfortunately, that isn't the case. Tools can help immensely in applying SOA Governance, but they will always be supportive to the policies that have been designed, the reference architecture that has been defined and the processes that have been put into place.

Quickly looking back at these sections we've seen that applying SOA Governance provides a lot of advantages. It will help you create better software, allow you to better control how your services are used, and promotes reuse and standardization. What we've also seen is

that SOA Governance isn't the silver bullet and that there isn't a tool you can just buy to implement it. It takes effort, both on the technical as organization level. In the next section we'll look at how to get started with SOA Governance.

## 1.4 Requirements of a SOA Governance solution

So far we've looked at what SOA Governance is and why it's important. In this section we'll look at what a complete SOA Governance solution should do and how this can help you can in applying SOA Governance in practice.

Basically a SOA Governance solution should help you in:

1. creating and maintaining a set of policies;
2. applying these policies at design time;
3. applying these policies at runtime

In figure 1.4 you can see an overview of the functionality a SOA Governance solution should provide. Here you have stakeholders defining policies which need to be stored and managed by the SOA Governance solution. Once the policies have been defined, they are consumed by various other parties. You've got developers who need to be able to access the policies so they know what the services they are developing need to comply to. You've got system admins who access the runtime information from the SOA Governance solution to see if everything is running according to the policies that have been defined and finally you've got management who wants to see if, for instance, orders are completed within the defined time.

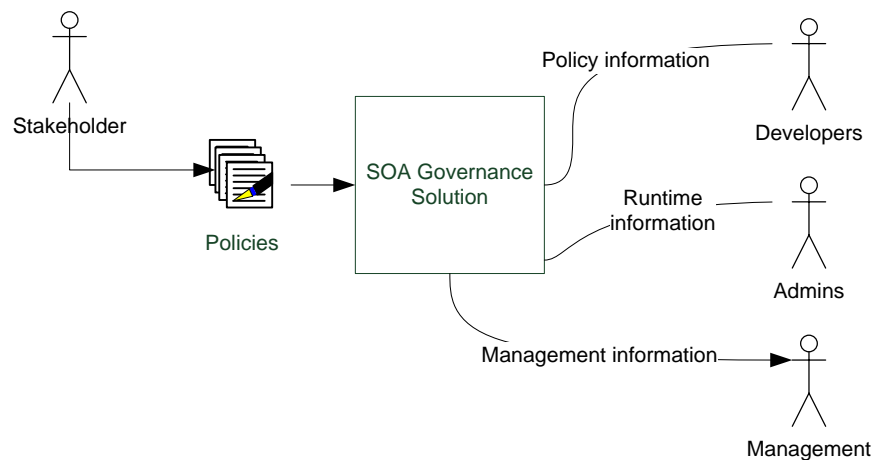


Figure 1.4: Figure showing the information a SOA Governance solution should provide to the various actors

Let's look a bit closer at these three points we've set out at the beginning of this section. We'll start by looking at how such a solution should help you in creating and maintaining your policies.

### **1.4.1 *Creating and maintaining policies***

The first thing a complete SOA Governance solution should provide is a way to register your policies. Often when I talk about policies, people automatically think about WS-Policy and automatic enforcement. This however isn't a requirement when you talk about policies or when you want to register them. In section 1.1 I already mentioned the OASIS' SOA Reference Model which defines a policy in the context of a SOA. In its definition it mentions the following with regards to how they should be written:

#### **DEFINITION**

"Policy assertions SHOULD be written in a form that is understandable to, and processable by, the parties to whom the policy is directed. Policies MAY be automatically interpreted, depending on the purpose and applicability of the policy and how it might affect whether a particular service is used or not. A natural point of contact between service participants and policies associated with the service is in the service description. It would be natural for the service description to contain references to the policies associated with the service." Reference Model for Service Oriented Architecture 1.0

What you can read in this quote is that policies should be written in a format that can be understood and processed by the parties to whom the policy is directed. In most cases the party at which a policy is directed is a human party. It's directed at developers, at administrators at information analysts etc.

There are many different tools out there that you can use to register your policies in. You've got commercial tools that allow you to register your policy, but, because we're talking about human consumers, a shared Word document or a Wiki is also a good, maybe even better, option depending on the environment you are in. If you have only a small set of service consumers from within your own organization, a shared document or Wiki will probably be more accessible than a (complex) commercial or open source tool.

### **1.4.2 *Applying policies at design time***

Besides helping in defining the policies, the solution should also help in communicating these policies. You can see this in figure 1.1 where the designers and administrators need information regarding the policies, and the already available services. For the policies part we need the same functionality as we did in the previous section; a tool to register and view the policies. However we also need access to the list of services that are already defined. We need this to determine whether functionality requires a new service, but also to determine the functionality we can reuse. For this we need a central place where we can store the contract of our service and its other metadata. In the most basic sense this can once again

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=778>

be a word document or a wiki page describing the service, but usually a service repository is used for this purpose.

### Repository versus registry

The words SOA repository and SOA registry are often used for the same thing; a place where you can register your services so that all the information regarding those services can be easily found and the services themselves can be easily located. When you really look at the definitions though, there is a difference between a repository and a registry. A repository is the place where you register a service and its metadata and where you can also register other artifacts. A registry on the other hand helps in locating a specific service. In the beginning of web services and SOAP, people often also mentioned UDDI. A UDDI registry provided a phone book / yellow pages like interface which you could use to lookup a specific service, either at runtime or design time. It didn't allow you to register other metadata or artifacts. That makes it a registry, which often used a repository as their backend. Most of the current repositories out there started there live as a registry. They however gained the ability to also register other metadata and artifacts and that made them into repositories. Nowadays these terms are still used both ways.

Figure 1.5 shows an example of a SOA repository with a service registered. Note that in the screenshot you see that the product calls itself a registry. For this specific case there is a good reason. The WSO2 Governance registry provides all the specific features of a registry. It provides a Governance API, support a "yellow pages" like interface and artifact classification. On the other hand it also presents itself as a repository, since you're pretty much free to register everything you want. The reason to call this product a registry instead of a repository is because, even though it provides a resource oriented repository interface, it provides all the features and APIs to also serve as a full fledged repository.

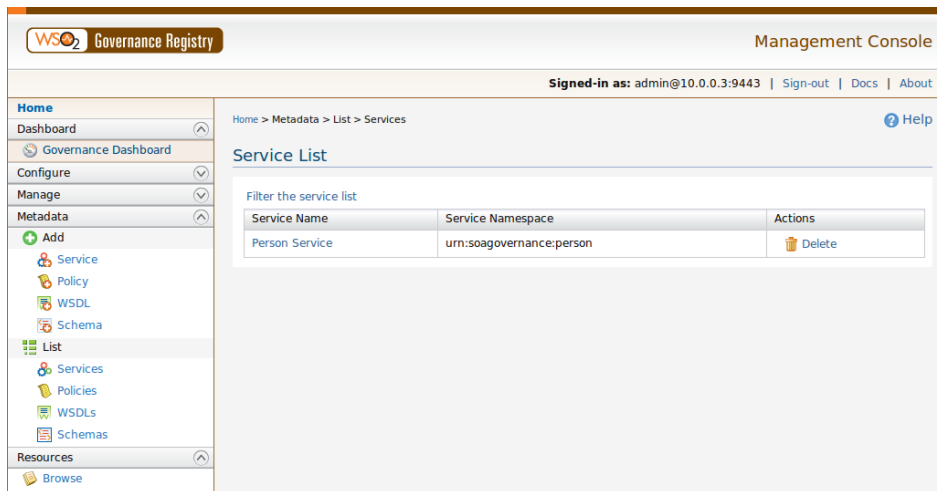


Figure 1.5 showing an example of a SOA Repository. In this repository you can register services, policies and also other structured and unstructured content.

The final item a SOA Governance solution should help us with is in applying these policies at runtime.

### 1.4.3 Applying policies at runtime

The final aspect a SOA Governance solution should cover is how to apply and enforce the policies we've created at runtime. Not all the policies lend themselves to be enforced automatically but a number of them do. For instance monitoring whether the minimum response time is less than two seconds can be easily done. Just register the time the service was called and register the time the service responded. Calls that take longer than these two seconds are logged and reported for further analysis.

Once again, there are many different tools out there that can help you with this. On the one hand you've got active monitoring tools (gateways) that monitor in and outgoing data and can check whether the data complies with your policies. Those tools can restrict or deny calls that don't follow your policies. On the other hand you've got the more reactive tools. They monitor the events and the performance of your service and provide reports on their usage. These reports can then be analyzed to determine whether the service follows the policies defined for it. In this latter category for instance falls WSO2 Business Activity Monitor, which is shown in figure 1.6.

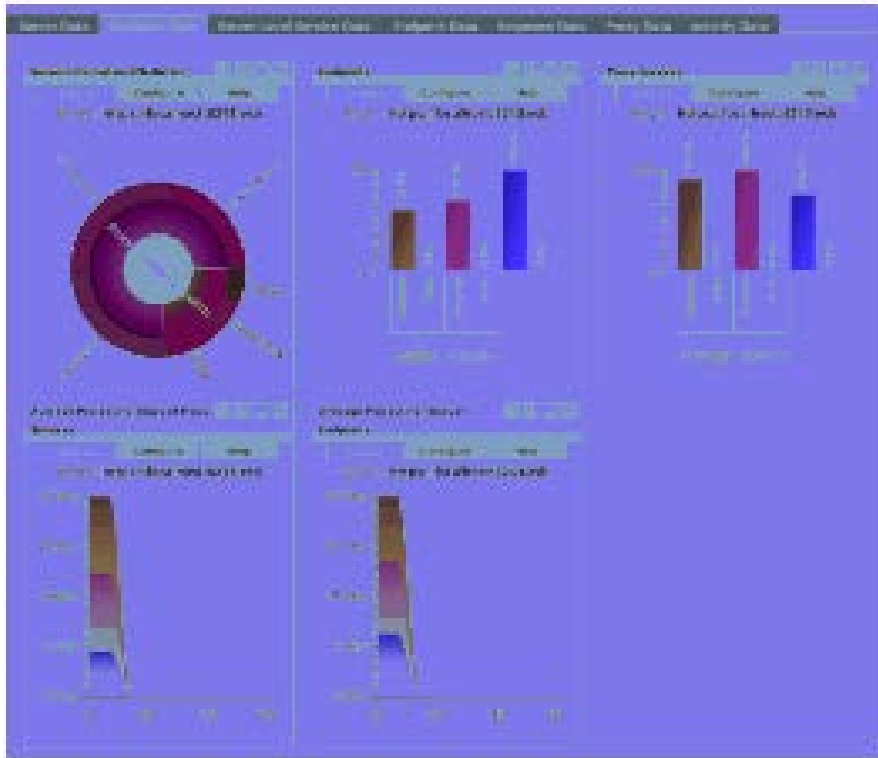


Figure 1.6 showing an example of reactive monitoring. With this information the IT or Business can determine whether the service complies with the policies set out for it.

In this section we've looked at which parts make up a SOA Governance solution. Now let's get started and look at some simple policies you can apply to get started with SOA Governance.

## 1.5 Getting started with SOA Governance

As you've read so far SOA Governance encompasses a large area. You've got to get a set of policies defined, need a reference architecture that can serve as an example for the services that need to be created and you need the setup all kinds of different control mechanisms to make sure the policies you've set out are really followed. All this can be a bit overwhelming and might make it seem impossible to get started with SOA Governance.

To get a feeling for SOA Governance and create awareness within your organization you can start with some small and simple policies within your own department and let the results speak for themselves. If you're successful within your department, which results in better services, lower costs, and more predictability, other departments will follow suit.

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=778>

So how do you start with this? The first thing to do, is define a couple (two or three) simple policies, which you'll follow throughout the process of creating new services. A good starting point would be at least one policy you follow during design time, and one policy you can enforce during runtime. This way you'll get experience with following and enforcing policies and you'll quickly see the advantages you get from applying SOA Governance.

The following table shows a number of good design time policies to follow when you're just getting started. In the other parts of this book, we'll dive deeper into the details of these policies.

**Table 1.2: Basic design time policies to get started with**

<b>Name</b>	<b>Description</b>
No duplicate services	This might seem like a “no-brainer”, but it's a good policy to start with. Even within a department many functionality is duplicated between services. Formalizing this in a policy, and checking this policy before you start work on a new service can save you a lot of duplicate work.
Technology standardization	This policy can sound something like this: “All services provided to the general public e.g. over the internet, must follow the REST paradigm. Services offered exclusively for B2B purposes must use a WS-* architecture”. Once again a very basic policy, but one that will avoid projects doing what they think is best, without looking at the bigger picture.
You must be able to validate message content.	This policy can help you in creating a uniform service landscape. When you look deeper at this policy you'll probably also need to define the type of validation that is required. This could for instance mean that all the message formats of this service should be XML accompanied by a schema. But this could also imply that all the JSON messages that are send, can be validated using a JSON Schema.

Let's also look at a couple of simple runtime policies that can form a good starting point for SOA Governance.

**Table 1.3: Basic runtime policies to get started with**

<b>Name</b>	<b>Description</b>
Services should be available from nine to five.	With the tools we'll be using in this book you can check whether your service is available. This is one of the easiest runtime policies to start with. Set up the tools and you get a nice graphic of when your service

was available and when not.

Services calls should be made over a secure channel. Another basic one to get started. If one of your policies is that all communications should be over a secure channel, you can for instance easily enforce this with apache.

Services should be self describing. When a service is deployed it should be self-describing. In other words you shouldn't need a big book describing the service and how to interact with it. For REST we've already shown an example in section 1.1.

In the next section an overview of the policies we'll be discussing in this book is given.

## 1.6 *Getting an overview of the available policies*

For this book we divided the policies into three main categories:

- **Design and documentation policies:** These policies define certain rules you must follow when you're designing and implementing your services.
- **Security policies:** Security is always a hot topic. Our security policies deal with how a service should be accessed, audited and secured.
- **Testing and performance policies:** The last policy category we use, are the testing and performance related policies. Here you'll, for instance, find policies that require execution of specific operation within a certain time.

Let's look a bit closer at these categories:

### DESIGN AND DOCUMENTATION POLICIES

Table 1.4 shows an overview of the policies we'll discuss in this book that fall into the "Design and documentation" category.

Table 1.4: Design and documentation policies

Name	Policy description
Self documenting service	All services should be self-documenting. You should be able to use a service without having to check a manual.
Canonical model	If there is a standard or de-facto model that is used within your organization or business domain, you should use that model in your services.
Reusable service	Before starting work on a new service, you must make sure that there isn't a service available already which provides the same functionality, or which can be easily altered to provide the required functionality.
Versionable service	It must be possible to support multiple versions of the same service next to each other. Changes in versions should affect the consumer as little as possible.

Refactor a service	It must be possible to refactor a service with minimal impact to the consumers of this service.
Service dependencies	When a new service is deployed or obsoleted it must be clear what the impact is on other services. Dependencies between services must be transparent.

In part two of this book we'll look at how you can take these policies into account during design time, and how you monitor your compliance at runtime.

### SECURITY POLICIES

Table 1.5 gives an overview of the security related policies we'll cover in this book.

**Table 1.5: This table provides an overview of the security policies discussed in this book**

<b>Name</b>	<b>Policy description</b>
Secure communication channel	All communication with our service must be done over a secure channel.
Message integrity and non-repudiation	We must make sure we can guarantee a message's integrity and origin of data.
Only authenticate once	When a consumer needs to authenticate for our service he should only have to authenticate one time, not separately for each service.
Reuse existing authorization provider	All services and application must reuse the existing authorization provider to avoid a proliferation of authorization schemes
Comply to auditing rules	All services must comply to the rules set out by external parties (e.g. governments) regarding the auditing of messages

The last set of policies we'll look at are those defined for the "Testing and performance" category.

### TESTING AND PERFORMANCE POLICIES

Table 1.6 gives a short description of the policies we discuss for this category.

**Table 1.6: Testing and performance policies**

<b>Name</b>	<b>Policy description</b>
Enforce code quality and test coverage	All our services must comply to a specific percentage of code coverage and have a pre-defined minimal level of quality
Must be able to run in a cloud environment	All our services must be designed in such a way that they can easily be deployed in a cloud environment
Must be horizontally	All the services that are developed must be horizontally scalable, this to

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=778>

scalable	avoid high investments in hardware and allow linear growth
Prevent service abuse	We must prevent our services being abused. For this we must be able to limit the amount of calls a client can make to our service for a specific time interval.
Maximum response time	A service should always respond within a minimum amount of time.

We'll show you throughout the following chapters how to implement these policies and how you can set up and configure a runtime environment to monitor and check whether your services comply with these policies. For now let's continue with this introduction and dive somewhat deeper in what open source can offer in this area.

## **1.7 SOA Governance and open source**

In this book we'll focus on using open source tools to help you with applying SOA Governance. In this section we'll look a bit deeper at the open source developments in the SOA Governance area and how these tools can help in creating your own open source based SOA Governance solution. Before we look at the SOA Governance specific developments, we'll first have a quick look at open source itself.

### **1.7.1 Where is open source at the moment**

We don't think we need to explain to the buyers of this book what Open Source is, but for completeness sake let's look at the official definition from the Open Source Initiative (OSI). OSI defines Open Source using the following criteria:

- **Free redistribution:** This criterion is what most people associate with Open Source. You are free to redistribute the open source product. Either in its whole or as an aggregate of another product.
- **Source code:** Together with the free redistribution criterion, this is the other one criterion most people think about when talking about Open Source. This criterion states that you must be able to get access to the source code of the product.
- **Derived Works:** The license attached to the software product must also allow you to make derived works or modifications to the original from it.
- **Integrity of The Author's Source Code:** The license may require that any derived works use a different name and version number from the original one. This is for instance the case for projects from the Apache foundation.
- **No Discrimination against Persons or Groups:** You can't say that left handed people with red hair can only use your software. Everybody can use your software.
- **No Discrimination Against Fields of Endeavor:** Pretty much the same as the previous criterion. You can't specify in your license that your software can only be used by financial institutions. Regardless of the "Field of Endeavor", anybody can use

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=778>

your software.

- **Distribution of License:** If a work is redistributed the original license still applies.
- **License Must Not Be Specific to a Product:** This criterion means that you can't say in your license that it may only be distributed part of a specific software distribution. For instance you can't say that your piece of software may only be distributed part of Fedora.
- **License Must Not Restrict Other Software:** The license can't put any restrictions on other software along which it's distributed.
- **License Must Be Technology-Neutral:** You can't say that your product may only be used together with a specific technology or on a specific platform.

If a piece of software fulfills the above mentioned criteria it can be classified as Open Source. The tools which we'll be using in this book all follow these criteria.

If we look at Open Source these last couple of years we see that the type of Open Source projects have been slowly moving up the stack. In figure 1.x we show a software stack for open source.

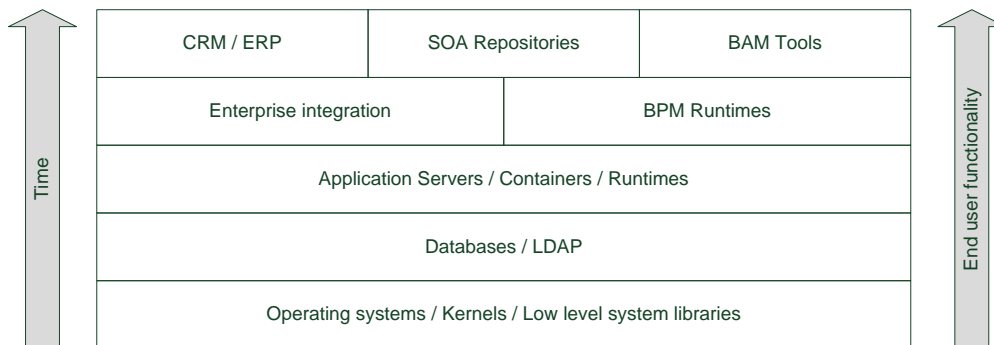


Figure 1.7 Open Source software in its early days focused on offering technical functionality and low level services. In recent years Open Source software has been offering products that also offer lots of business and end-user functionality.

In figure 1.7 you can see that in the beginning of Open Source software most software was low level software such as operating systems and databases. In the last couple of decades Open Source has matured and you can also see this in the type of applications that can be found in the Open Source community. In the last couple of years a number of tools appeared in the Open Source community that can also help us in applying SOA Governance. In the following sections we'll look at these tools.

## 1.7.2 Open source tools

In this section we'll look at two different types of tools. We'll be looking at the open source SOA Repositories that are out there and we'll be looking at the BAM tools that are made available by the open source community.

### BASED SOA REPOSITORIES

When you look at the open source SOA Repositories out there your choices at the moment are still a bit limited. There are various open source UDDI registries out there, but as we've mentioned before, a registry is not a repository. So which open source SOA Repositories are out there?

- **Mule Galaxy:** Mule Galaxy is a mature SOA Registry brought to you by the people who created the great open source ESB Mule. Mule Galaxy provides all the features an SOA Repository requires. You can manage all different kinds of artifacts and apply a life-cycle to these artifacts. Besides this Mule Galaxy also provides you with service discovery functionality, flexible reporting on services and easy customization. If you're already using Mule as your ESB, using Mule Galaxy as a SOA Repository is a very good option. More information on Mule Galaxy can be found at the Mulesource site: <http://www.mulesoft.org/documentation/display/GALAXY/Home>
- **WSO2 SOA Registry:** The SOA Registry from WSO2 is also a standalone product with which you can manage different kinds of artifacts and apply a life-cycle to these artifacts. This is the registry we'll be using in our examples for this book. More information on this product can be found at the WSO2 website: <http://wso2.com/products/governance-registry/>. Besides a registry, WSO2 also offers a complete set of other products, for instance a great ESB, that integrate with this repository.
- **Petals Master:** Petals Master calls itself a complete SOA Governance solution. It allows you to categorize your services and endpoints based on an UDDI registry. Just as the previous two registries Petals Master allows you to add comments to your artifacts and share these with all the users of the registry. Besides the registry oriented functionality Petals Master also allows you to manage your organization, since it includes an UDDI based identification system. More information on this solution can be found at ow2s website: <http://petalsmaster.ow2.org/>

For a list of commercial SOA Repositories see Appendix X. Besides the registries we've looked at here, for a complete SOA Governance solution we also need some way to monitor and manage our runtime environment

### OPEN SOURCE BAM TOOLS

We mentioned that for runtime monitoring we need to be able to monitor service actively (e.g. through an XML Gateway) and reactively (using some sort of BAM tool). For actively monitoring services there are many different options available and this heavily depends on the architecture you use for your services. If you've got an ESB based architecture, your ESB will normally enforce these policies and if you run inside a service container, this container

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=778>

will often provide this functionality. We won't list all the open source ESBs and service containers in this section, but in the later chapter of this book we'll show, using various open source ESBs and containers, how you can use the specific functionality of your container or ESB to enforce certain policies.

So in this section let's focus a bit more on the BAM tools. Normally when you start looking for a certain piece of software in the open source community, you'll quickly find many different mature solutions. However, this isn't the case for BAM tools. There are one or two tools, which I'll show you in a second, but the main issue with these tools is their inflexibility. The currently available BAM tools are bound to a specific technology stack, are limited in their reports or only cover a specific framework. So what is out there?

- **WSO2 Business Activity Monitor:** We've already shown an example in this chapter of what this application looks like. This tool provides beautiful diagrams of usage stats, error reports, business diagrams etc. The problem is, that customizing the graphs or monitoring non-WSO2 servers is nigh impossible. This tool uses fixed reports and if you've got a WSO2 only landscape this tool is a good way to go. More information can be found on WSO2s website: <http://wso2.com/products/business-activity-monitor/>
- **Esper:** Esper is a tool which calls itself a "Complex Event Processor". In other words it's great for handling, analyzing and correlating events. Since a BAM tool needs to be able to correlate events and show the end user only useful data, this is a very great feature. The issue, however, is that that is all Esper is. It doesn't provide a reporting environment with which you can monitor or view these correlated events. More information on Esper can be found at: <http://esper.codehaus.org/>

So within the open source community there really isn't a great BAM tool that allows us to monitor (and analyze) events and show them in a customizable way to the end user. Luckily though all the various components that we need to build such a solution are available. In the following chapters we'll show you how you can combine tools such as Esper and WSO2s gadget server to create an easy to use and very flexible BAM tool. And of course we'll also show you how you can use this tool, whose code is of course also made available as open source, to monitor your services so that you can easily check whether you comply to the specified run time policies.

## 1.8 Summary

- SOA Governance is all about defining the policies that are important for the various stakeholders within your organization, applying these policies during design time and monitoring and enforcing these policies at runtime.
- SOA Governance isn't scary. The goal of SOA Governance isn't to set up an mammoth like process you need to follow, the goal is to make sure your services are of certain level of quality and follow the policies set out by your company. If you follow the guidelines set out in this book, you'll see how easy it is to get started with SOA Governance and get the advantages from applying SOA Governance.

- SOA isn't just exposing services using a specific technology. If you want to get the most out of your SOA you need to set the business goals first. The business goals should lead to the specific services that need to be developed, and when services are developed, reuse needs to be taken into account. With SOA Governance you're presented with a set of tools and best practices that can help you in getting the most out of the services you've defined.
- SOA Governance can be divided into two distinct areas. SOA Governance during design time and SOA Governance during runtime. The first one focuses on making sure all the services are created and defined consistently and follow a set of predefined policies. The second one makes sure that the policies you've defined, and those that can be monitored, are enforced during runtime.
- Tools can help you in making sure policies are followed. Besides many commercial offerings there are also a number of mature open source tools that can help you in applying SOA Governance. In this book we'll show a pragmatic approach using open source tools. There is however not a good open source BAM tool, this we'll create ourselves.