Reference Architecture for Service Oriented Architecture Version 0.0

working-draft, May xx,2007

Specification URIs:

This Version:

http://docs.oasis-open.org/soa-rm/ [additional path/filename] .html
http://docs.oasis-open.org/soa-rm/ [additional path/filename] .pdf

Previous Version:

http://docs.oasis-open.org/soa-rm/ [additional path/filename] .html
http://docs.oasis-open.org/soa-rm/ [additional path/filename] .pdf

Latest Version:

http://docs.oasis-open.org/soa-rm/ [additional path/filename] .html
http://docs.oasis-open.org/soa-rm/ [additional path/filename] .pdf

Latest Approved Version:

http://docs.oasis-open.org/soa-rm/ [additional path/filename] .html
http://docs.oasis-open.org/soa-rm/ [additional path/filename] .pdf

Technical Committee:

OASIS Service Oriented Architecture Reference Model TC

Chair(s):

Francis G. McCabe

Editor(s):

Jeffrey A. Estefan, Jet Propulsion Laboratory, jeffrey.a.estefan@jpl.nasa.gov
Ken Laskey, MITRE Corporation, klaskey@mitre.org
Francis G. McCabe, Genietown, frankmccabe@mac.com
Danny Thornton, danny.thornton@soamodeling.org

Related work:

This specification is related to:

- OASIS Reference Model for Service Oriented Architecture

Abstract:

This document specifies the OASIS Reference Architecture for Service Oriented Architecture. It follows from the concepts and relationships defined in the OASIS Reference Model for Service Oriented Architecture. While it remains abstract in nature, the current document describes one possible template upon which a SOA concrete architecture can be built.

Our focus in this architecture is on an approach to integrating business with the information technology needed to support it. The issues involved with integration are always present, but, we find, are thrown into clear focus when business integration involves crossing ownership boundaries.

This architecture follows the recommended practice of describing architecture in terms of models, views, and viewpoints, as prescribed in IEEE1471. The RA has three main views: the Business via Service view which lays the foundation for conducting business in the context of Service Oriented Architecture; the Realizing Services view which addresses the requirements for constructing a Service Oriented Architecture; and the Owning Service Oriented Architecture view which focuses on the governance and management of SOA systems.

Status:

This document was last revised or approved by the SOA Reference Model TC on the above date. The level of approval is also listed above. Check the “Latest Version” or “Latest Approved Version” location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee’s email list. Others should send comments to the Technical Committee by using the “Send A Comment” button on the Technical Committee’s web page at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm.

The non-normative errata page for this specification is located at http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm.

Notices

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names "OASIS", [insert specific trademarked names and abbreviations here] are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

Unified Modeling Language™, UML®, Object Management Group™, and OMG™ are trademarks of the Object Management Group.

Table of Contents

1 Introduction

Service Oriented Architecture is an important paradigm that has gained significant attention within the information technology (IT) and business communities. The OASIS Reference Model for SOA provides a common language for understanding the important features of SOA but does not address the issues involved in constructing a SOA-based system. This document focuses on this aspect of SOA; while maintaining a similarly high-level approach as the Reference Model itself.

1.1 What is a Reference Architecture

A Reference Architecture is a description of the concepts and relationships in a domain that enables stakeholders to identify how a set of requirements may be realized. It differs from a Reference Model in that a Reference Model describes the important concepts in the domain focusing on what distinguishes the elements of the domain; a Reference Architecture elaborates further on the model to show a more complete picture that includes showing what is involved in realizing the modeled entities.

We identify three broad categories of requirements: how Service Oriented Architecture fits into the life of users and stakeholders, how SOA-based systems may be realized effectively, and what is involved in owning a SOA-based system.

It is possible to define Reference Architectures at many levels of detail or abstraction. In this Reference Architecture we have followed a high-level technology neutral approach; while at the same time being fully aware of the dominant technologies likely to be employed. In fact, the degree of abstraction in modeling concepts in the Reference Architecture is very similar to that employed in the Reference Model itself. We believe that this will serve two purposes: ensuring that the true value of the SOA approach can be realized on any appropriate technology, and it permits our audience to focus on the important issues without becoming over-burdened with unnecessary detail.

1.2 Service Oriented Architecture – An Ecosystems perspective

Many systems cannot be understood by a simple decomposition of their parts into subsystems. There are too many interactions between the parts. There has been much recent research about "complex systems" because it has many applications to large systems such as the economy, or the human brain, where one cannot understand the larger picture just by putting the pieces together.

A service-oriented architecture shares many of the characteristics of such systems. From the perspective of a complex system, a SOA is a network of independent services, machines, the people who operate, affect, use, and govern those services as well as the suppliers of equipment and personnel to these people and services. This includes any entity, animate or inanimate, that may affects, or be affected by the system. With a system that large, it is clear that nobody is really "in control" or "in charge"; although there are definite stakeholders involved, each of whom has some control and influence of the whole.

We have multiple perspectives view of this SOA architecture: corresponding to the ways in which SOA must be understood by key stakeholders: by its users, by its constructors and by its owners.

1.3 Relationship to the Reference Model

This Reference Architecture takes the Reference Model as its starting point; however, it is somewhat more concrete than the Reference Model.

As a result, some of the concepts that are identified within the Reference Model are further expanded on in the Reference architecture; other concepts are ‘unpacked’ into different aspects of the eco-system/machine. Furthermore, additional concepts are introduced which were not ‘of the essence’ for describing Service but are required in order to have a Service Oriented Architecture.

One key concept that has been unpacked is that of Execution Context. Within the RM, the execution context stood for all the aspects of an information system that are needed to facilitate interaction. A large part of the goals of the Reference Architecture is to show how interaction is realized; as a result, the concept of Execution Concept is not as pertinent within the Reference Architecture.

1.4 Relationship to other Reference Architectures

It is fully recognized that other SOA reference architectures have emerged in the industry, both from the analyst community and the vendor/solution provider community. Some of these reference architectures are at a sufficient level of abstraction away from specific implementation technologies while others are based on a solution or technology “stack.” Still others use emerging middleware technologies such as the Enterprise Service Bus (ESB) as the architectural foundation.

The Reference Architecture for SOA is an abstract realization of SOA—showing how a SOA can be built while omitting any reference to specific concrete technologies. As with the Reference Model for SOA, the Reference Architecture is primarily focused on large-scale distributed IT systems where the participants may be legally separate entities. While it is quite possible for many aspects of the Reference Architecture to be realized on quite different platforms, we do not dwell on such opportunities.

1.5 Viewpoints, Views and Models

1.5.1 ANSI/IEEE Std 1471-2000

This Reference Architecture for SOA loosely follows the ANSI/IEEE Std 1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems [#]. An architectural description conforming to the ANSI/IEEE Std 1471-2000 recommended practice is described by a clause that includes the following six (6) elements:

1. Architectural description identification, version, and overview information
2. Identification of the system stakeholders and their concerns judged to be relevant to the architecture
3. Specifications of each viewpoint that has been selected to organize the representation of the architecture and the rationale for those selections
4. One or more architectural views
5. A record of all known inconsistencies among the architectural description’s required constituents
6. A rationale for selection of the architecture

The ANSI/IEEE Std 1471-2000 defines an architectural description (AD) as “a collection of products to document the architecture,” where architecture is defined as:

Architecture

> The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.

A system stakeholder is “an individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system,” where system is defined as:

System

> A collection of components organized to accomplish a specific function or set of functions.

A stakeholder concern (or care-about) should not be confused with a formal requirement. A concern is an area or topic of interest. Within that concern, system stakeholders may have many different requirements. In other words, something that is of interest or importance is not the same as something that is obligatory or of necessity.

When describing architectures, it is important to identify stakeholder concerns and associate them with viewpoints to insure that those concerns will be addressed in some manner by the models that comprise the views on the architecture. The ANSI/IEEE Std 1471-2000 defines views and viewpoints as follows:

View

> A representation of the whole system from the perspective of a related set of concerns.

Viewpoint

A specification of the conventions for constructing and using a view. A pattern or template which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis.

In other words, a view is what the stakeholders see whereas the viewpoint defines the perspective from which the view is taken.

It is important to note that viewpoints are independent of a particular system. In this way, the architect can select a set of candidate viewpoints first, or create a set of candidate viewpoints, and then use those viewpoints to construct specific views that will be used to organize the AD. A view, on the other hand, is specific to a particular system. Therefore, the practice of creating an AD involves first selecting the viewpoints and then using those viewpoints to construct specific views for a particular system or subsystem. Note that the ANSI/IEEE Std 1471-2000 requires that all views correspond to exactly one viewpoint. This helps maintain consistency among architectural views; a normative requirement of the standard (see 5 above).

A view is comprised of one or more architecture models, where model is defined as:

Model

An abstraction or representation of some aspect of a thing (in this case, a system)

Each architectural model is developed using the methods established by its associated architectural viewpoint. An architectural model may participate in more than one view.

1.5.2 UML Modeling Notation

To help visualize structural and behavioral architectural concepts, it is useful to depict them using an open standard visual modeling language. Although a myriad of architecture description languages exist in practice, we have adopted the second generation Unified Modeling Language™ (UML®) known as UML 2 managed under the auspices of the Object Management Group™ (OMG™) as the primary viewpoint modeling language. It should be noted that while UML 2 is used in this reference architecture, formalization and recommendation of a UML Profile for SOA is beyond the scope of this specification. Every attempt is made to utilize normative UML unless otherwise noted.

1.6 Viewpoints of this Reference Architecture

The Reference Architecture is partitioned into three views that conform to three primary viewpoints, reflecting the main division of concerns noted above: the Business via Services viewpoint focuses on how a SOA integrates with how people conduct their business; the Realizing a Service Oriented Architecture viewpoint focuses on the salient aspects of building a SOA, and the Owning Service Oriented Architectures viewpoint focuses on those aspects that relate to owning, managing and controlling a SOA.

The viewpoint specifications for each of the primary viewpoints of this reference architecture are summarized in Table 1. Additional detail on each of the three viewpoints is further elaborated in the following subsections. For this reference architecture, a one-to-one correspondence between viewpoints and views is assumed.

Viewpoint Element	**Viewpoint**		
	Business via Services	*Realizing a SOA*	*Owning SOAs*
Main concepts	Captures what SOA means for people using it to conduct business	Deals with the requirements for constructing an SOA	Addresses issues involved in owning an SOA vs. using or building one
Stakeholders	People (using SOA)	Service Consumers, Service Providers	Service Providers
Concerns	Conduct business safely	Effective	Processes for

	and effectively	implementation using standard technologies	engaging in an SOA are effective, equitable, and safe
Modeling Techniques	UML Class diagrams	UML Class and Sequence diagrams	UML Class diagrams

Table 1 Viewpoint specifications for the OASIS Reference

1.6.1 Business via Services Viewpoint

The Business via Services viewpoint is intended to capture what using a SOA-based system means for people using it to conduct their business. From this viewpoint, we are concerned with how SOA integrates with and supports the service model from the perspective of the people who perform their tasks and achieve their goals as mediated by Service Oriented Architectures. The Business via Services viewpoint also sets the context and background for the other viewpoints in the Reference Architecture.

The stakeholders who have key roles in or concerns addressed by this viewpoint are *people* and the primary concern for people is to ensure that they can use a SOA to conduct their business in a safe and effective way. Given the public nature of the Internet, and the intended use of SOA to allow people to access and provide services that cross ownership boundaries, it is necessary to be able to be somewhat explicit about those boundaries and what it means to cross an ownership boundary.

The modeling techniques for expressing the visual models that comprise the associated view that conforms to this viewpoint are UML models; principally, UML class diagrams.

1.6.2 Realizing a Service Oriented Architecture Viewpoint

The Realizing a Service Oriented Architecture Viewpoint focuses on the infrastructural elements that are needed in order to support the discovery and interaction with services. From this viewpoint we are concerned with the application of *normal* technologies available to system architects to realize the vision of an SOA that may cross ownership boundaries. In particular, we are aware of the importance and relevance of other standard specifications that may be used to facilitate the building of an SOA.

The modeling techniques for expressing the visual models that comprise the associated view that conforms to this viewpoint are UML models; principally, UML class, sequence, and communication diagrams.

1.6.3 Owning a Service Oriented Architectures Viewpoint

The Owning Service Oriented Architectures viewpoint addresses the issues involved in owning an SOA as opposed to using one or building one. Many of these issues are not easily addressed by automation; instead, they often involve people-oriented processes such as governance bodies.

The principal stakeholders who have key roles in or concerns addressed by this viewpoint are *service providers, service consumers* and other non-participatory organizations charged with ensuring societal goals – such as fair trade, public safety and so on. The primary concerns are that the processes for engaging in an SOA are effective, equitable, and safe.

The modeling techniques for expressing the visual models that comprise the associated view that conforms to this viewpoint are UML models; principally, UML class diagrams.

1.7 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in **[RFC2119]**.

2 Requirements

A Reference Architecture is like an abstract machine. It is built to realize some function and it, in turn, relies on a set of underlying capabilities that must be present for it to perform. In the case of the SOA RA, its purpose is to enable a system to be a Service Oriented Architecture. The underlying capabilities are the particular technologies that are used to realize the SOA; in particular technology choices such as Web services technologies, implementation technologies are not part of an abstract RA.

The purpose of the RA is reflected in the set of requirements that the RA must satisfy. We can structure these requirements into a set of goals, a set of critical success factors (CSFs) associated with these goals and a set of requirements that are connected to the CSFs that ensure their satisfaction.

Note that not all of the requirements are mapped to solutions within the scope of this RA. Indeed, the RA itself can be seen as generating a series of more explicit requirements for the realizing technology.

The overall requirements are illustrated in Figure 1.

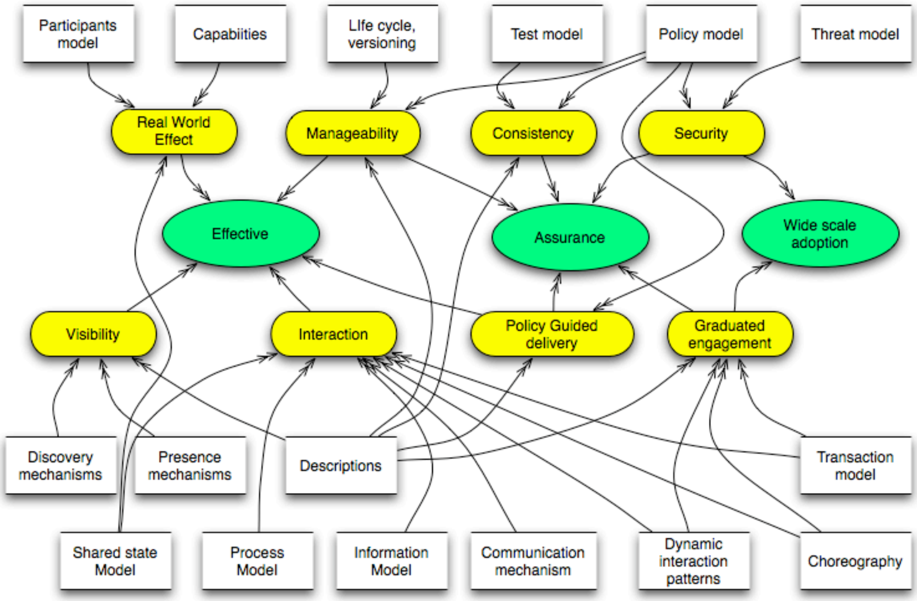


Figure 1 Critical Factors Analysis of the Reference Architecture

The critical factors analysis (CFA) requirement technique and the diagram notation is summarized in Appendix B.

2.1 Goals of the Reference Architecture

There are three principal goals of the Reference Architecture: that it shows how to build SOAs that effectively meet the requirements of the stakeholders involved, that it does so with some assurance that no harm is done, and that the architecture itself can be widespread.

2.1.1 Effectiveness

According to the RM, a SOA is a "paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains". Enabling the offering and utilizing of capabilities across ownership domains is the definition of effectiveness of Service Oriented Architecture.

For the architecture to be effective, it means that systems based on it can effectively bring the needs and capabilities of participants together.

There are a number of critical success factors that must be addressed for this goal to be satisfied: it must be possible for service providers and users to see each other, they must be able to interact in modern systems, the effects of provided and using services must similarly be communicated and the SOA-based system must itself be manageable with reasonable effort.

2.1.1.1 Visibility

Without service consumers and providers being able to "see" each other, they cannot interact with each other. The RM goes on to identify three key concepts associated with visibility: awareness, willingness and reachability. It also begins to define the basis for description which is a key to enabling visibility. This document will elaborate on the makeup and structure of description, the means by which description may be catalogued for discovery and retrieval, and the mechanisms for detecting the presence of a service.

As part of the reachability aspect of visibility, it must be possible to determine if a service is active or not. In general this is difficult to be certain of (a service might become unavailable the instant after you have been informed that its available).

2.1.1.2 Interaction

Interaction between service providers and consumers is how a service is effected. Supporting interaction is a major critical success factor for SOA. As is made clear in the rest of the architecture, interaction is achieved in many distributed systems via message-based communication.

Communication

One key requirement for interaction between service participants is communication: defined as the meaningful exchange of information that can be interpreted as part of a service interaction.

Information model

Successful interaction requires that the parties have agreement on the nature of the information that is exchanged. This is captured in the information model associated with the service.

Behavior model

The dynamics of interaction must also be effectively agreed to by the parties in the interaction.

2.1.1.3 Real World Effect

The purpose of interacting with a service partner (both the service consumer and the provider) is to achieve desired effects - the real world effect. The RWE captures the result of bringing a capability to bear as a consequence of the interaction. How the RWE is described and realized in the RA is a critical aspect of SOA.

2.1.2 Assurance

The RA should enable service providers and consumers to achieve their goals with the maximum possibility of safety in the interaction. Note that we distinguish any assurance associated with the delivery of a service from that resulting from the application of the service. The latter is beyond the scope of this RA. Assurance of service is of the essence when providers and consumers are in different ownership domains; and hence supporting the safety of that interaction is an important goal of the RA.

Some of the critical factors that concern safety of a system are how secure the system is whether participants' expectations in using the system are consistent, whether stakeholders' policies can be respected and whether the risk of interacting with the system is commensurate with expectations.

2.1.2.1 Security

In any system where there is exposure to multiple ownership domains, security is a paramount factor in its success. Security can be addressed from several perspectives: the threats that must be addressed, the mechanisms for mitigating those threats and the management of those mechanisms. In addition, legitimate access to the system needs to be easy to avoid incentives to bypass security mechanisms.

The key threats are against the privacy of interaction, against the proper identification of systems and people interacting with services, against the improper or compromised use of services, and against the possible later repudiation of previous interactions.

Security should facilitate access to content and business logic by those who are entitled, that is, continuous interruptions should not impede the functioning of the system.

2.1.2.2 Consistency

In order to effectively interact across ownership boundaries it is critical that the actual interaction matches expectations for that interaction. The key to this predictability is adequate and explicit descriptions of all facets of services; further supported by explicit policy statements.

2.1.2.3 Explicit Policies

Any machine is necessarily broad in its applicability and is often under-constrained. Policy statements define the choices that a service provider and/or service consumer (or other stakeholder) makes. Access to these policy choices is an important aspect of ensuring predictability and consistency.

A critical factor for SOA is that consistent policy can be specified, applied and enforced. Policies can be applied to many aspects of an SOA-based system; here we focus on the role of policy in delivering service functionality itself.

2.1.2.4 Participants' roles

The rights, obligations, roles and collaborative context of participants in service interactions must be accurately represented in the architecture. This is important from the perspective of the RWE — which is expressed in terms of the facts and commitments shared by service participants — and from the perspective of the security threats that any SOA is subject to.

2.1.2.5 Graduated engagement

There is a fundamental principle in interacting across ownership boundaries that there be no "unpleasant surprises". While we cannot eliminate such unexpected consequences we can insist on a model where the expectations of the parties in an interaction are commensurate with their commitments; i.e., as we gradually get deeper into an engagement with a service partner the commitments and expectations become similarly deeper.

For example, if a service receives a sequence of bytes that it cannot understand, it should not assume that its service has been validly invoked. Another example is the inadmissibility (in many jurisdictions) of the so-called drive-by license agreement: the user must perform an explicit action signaling agreement for it to be binding.

2.1.2.6 Manageability

Given that a large-scale SOA is likely to be populated with many thousands of services, managing them becomes a critical factor for the assured delivery of services.

Manageability requires clear descriptions of who is responsible for the service and what responsibility entails. This includes how a given service contributes and consumes resources as part of the SOA ecosystem.

A given service may be provided and consumed in more than one version. Version control of services is important both for service providers and service consumers (who may need to ensure certainty in the version of the service they are interacting with).

In the context of multiple ownership domains manageability may be less an issue of managing IT infrastructure components (although that is very critical) used to realize services than managing the use of services, managing the relationships between participants in their use of services, and permitting management of services across ownership boundaries.

2.1.3 Wide scale adoption

It is an explicit goal of this work that the model it promotes will find widespread acceptance in Industry. While we cannot guarantee wide-scale adoption: In addition to assurance issues mentioned above, we identify a number of factors that will enhance the adoptability of SOA-based systems: applicability, scalability, loose coupling, low cost of entry, reusability, technology independence and simplicity.

2.1.3.1 Scalability

Any architecture whose design principles are not effective across the wide range of possible scales from a small intra-Enterprise system to a full-scale Internet-wide deployment is unlikely to find general acceptance. Any given instantiation of the architecture need not scale to the full Internet; the RA itself must be capable of such scale.

On the other hand, human ownership boundaries are not limited to single legal jurisdictions or geographic regions.

- It should be possible for service providers and consumers to interact even across such boundaries.
- It should be possible for services to be offered in ways that are sensitive to the local environment; the same service may have different presentations in different locales.
- Reduced dependence on one particular natural language

This may have some surprising consequences. For example, although a natural language description of some item may be easier to understand for a native speaker of that language, the language used can become an unnecessary barrier for a non-native speaker. Formal descriptions, i.e., descriptions in a formal notation, may be more difficult to understand, however, they are more neutral that descriptions in any given natural language, and therefore are less likely to be misunderstood by people as well as by automated processors.

2.1.3.2 Loosely coupled

A loosely coupled system is one in which the constraints on the interaction between components is minimal: sufficient to permit interoperation without additional constraints that may be an artifact of implementation technology. Each non-essential constraint may have negative impacts on the scalability of systems based on the architecture.

2.1.3.3 Understandability

An architecture that is highly scalable, but which is complex and difficult to understand will not find general acceptance.

2.1.3.4 Reusability

The extent to which services are reusable, descriptions, and other artifacts are reusable will be a critical factor in promoting adoption. Without reusability there is no investment and no growth.

- Artifacts should be realized in appropriately compatible technology. There should not be unnecessary technological dependencies.
- Pre-existing capabilities should be made accessible as services with the minimum of transformation.
- Services should be combinable in appropriate forms without inconsistent effect and undue effort.

2.1.3.5 Low cost of entry

A low cost of entry is an important critical success factor for wide-scale adoption.

The technology requirements for a given system should be commensurate with the complexity of that system. Simple service systems should be simple to design and build; complex systems with complex requirements should not be made more complex by the constraints of this architecture.

2.1.3.6 Technology neutral

The extent to which the RA is technology neutral will greatly effect its applicability in different circumstances. The counterpoint to being technology neutral is the ability to operate across several different kinds of platform. Enabling service providers and consumers to occupy different platforms is a key CSF that will help to drive adoption of the RA.

2.1.3.7 Simplicity

A key factor in wide-scale adoption will be simplicity of design and realization. A complex architecture, or one that is difficult to understand, would be an impediment to wide-scale adoption.

The hallmark of good design is simplicity. This is better expressed as sufficient complexity to satisfy requirements while avoiding unnecessary complexity. History shows us that it is often better to err on the side of reduced functionality if that promotes simplicity.

- Minimal assumptions
- Compositionality

3 Business via Services View

> **No man is an island**
> *No man is an island entire of itself; every man*
> *is a piece of the continent, a part of the main;*
> *if a clod be washed away by the sea, Europe*
> *is the less, as well as if a promontory were, as*
> *well as any manner of thy friends or of thine*
> *own were; any man's death diminishes me,*
> *because I am involved in mankind.*
> *And therefore never send to know for whom*
> *the bell tolls; it tolls for thee.*
> John Donne

The *Business via Services view* focuses on how Service Oriented Architecture fits into the life of users and stakeholders. The function of SOA is to facilitate action in a community of people; where action is characterized in terms of providing services and consuming services to realize mutually desirable real world effects. Thus, our tasks in this view are to model the people involved—the participants and other stakeholders—their goals and activities and the relevant relationships between people as they affect the utility and safety of actions that are performed.

The models in this view include the Stakeholders and Participants Model, the Needs and Capabilities Model, the Resources Model, and the Social Structure Model.

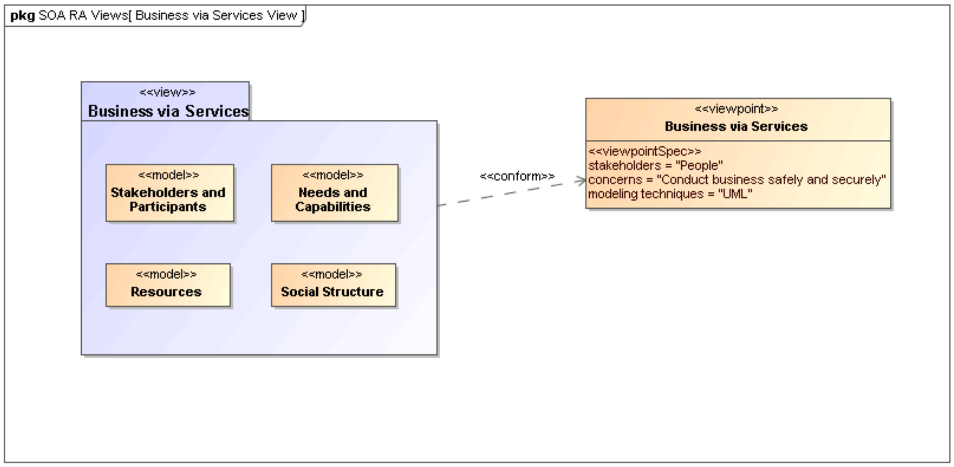


Figure 2 Model elements described in the Business via Services view

3.1 Stakeholders and Participants Model

A SOA is deployed in the context of human and non-human entities capable of action. In this section we focus on the relationship between these ultimate actors and the services that they use and deploy.

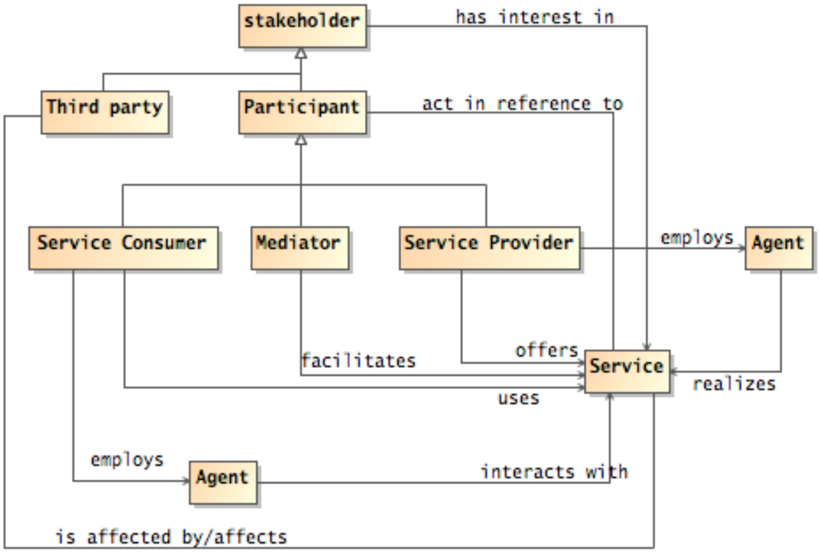


Figure 3 Service Participants

Stakeholder

> A stakeholder is a human, corporation or non-human agent that has an interest in the states of services and/or the outcomes of service interactions.

Stakeholders do not necessarily participate in service interactions. For example, a government may have an interest in the outcomes of commercial services deployed in a SOA without actively participating in the interactions (the government may collect tax from one or more participants without being part of the interaction itself).

Participant

> A participant is a human, organization or non-human agent that has the capability and requirement to act in the context of a Service Oriented Architecture.
>
> A participant is an example of a stakeholder whose interests lie in the successful use of and fulfillment of services. Note that we admit non-human agents as an extreme case: the normal situation is where participants are either human or corporations. However, human participants always require *representation* in an electronic system – they require agents.

It is convenient to classify service participants into service providers and service consumers. The reason for this is twofold: an extremely common mode of interaction is where a provider participant offers some functionality as a service and a consumer participant uses that service to achieve one of his or her goals. Secondly, it helps to illustrate the dominant situation where the participants are not truly symmetric: they each have different objectives and often have different capabilities. However, it should be noted that there are patterns of interactions where it is not clear that the distinction between service provider and consumer are valid. Give example here.

Service Provider

> A service provider is a participant that offers a service that permits some capability to be used by other participants.

In normal parlance, the service provider commonly refers to either the ultimate owner of the capability that is offered or at least an agent acting as proxy for the owner. For example, an individual may own a business capability but will enter into an agreement with another individual (the proxy) to provide SOA access to that business -- so that the owner can focus on running the business itself.

Note that several kinds of stakeholders may be involved in provisioning a service. These include the provider of the capability, an enabler that exposes it as a service, a mediator that translates and/or manages the relationship between service consumers and the service, a host that offers support for the service, a government that permits the service and/or collects taxes based on service interactions and so on.

Service Consumer

> A service consumer is a participant that interacts with a service in order to satisfy some desired need. As with service providers, several stakeholders may be involved in a service interaction supporting the consumer.

It is a common understanding that service consumers typically initiate service interactions. Again, this is not necessarily true in all situations (for example, in publish-and-subscribe scenarios, a service consumer may initiate an initial subscription, but thereafter, the interactions are initiated by publishers).

Service mediator

> A service mediator is a participant that facilitates the offering or use of services in some way without necessarily directly interacting with the service. There are many kinds of mediator, for example a registry is a kind of mediator that permits providers and consumers to find each other. Another example might be a filter service that enhances another service by encrypting and decrypting messages. Yet another example of a mediator is a proxy broker that actively stands for one or other party in an interaction.

Agent

> An agent is any person or non-human entity that is capable of acting on behalf of a person or organization.

Artificial agent

> An artificial agent is a constructed entity that is used by people to enable them to offer, consumer and otherwise participate in services. Common examples of artificial agents are software applications that make use of services, hardware devices that embody an agent with a particular mission, and enterprise systems that offer services.

In the context of SOA, since interaction between participants is mediated via the networks such as the Internet, people require proxies to participate. Artificial agents that can have direct access to electronic communications permit this.

We do not attempt to characterize artificial agents in terms of their internal architecture, computational requirements or platforms here. Within the Service as Business view, an agent stands for whatever information technology resources are required to facilitate the human use and other forms of participation involving services.

Third-party stakeholder

> A third party is any stakeholder who may be affected by the use or provisioning of services or who has an interest in the outcome of service interactions.

There are two main classes of such non-participatory stakeholders: innocent bystanders who are materially affected by someone's use or provisioning of a service, and regulatory agencies who wish to control in some way (such as by taxation) services.

At its most basic, a service is provided by a provider and used by a consumer in order to achieve a change in the real world that meets a desired goal.

However, interactions between service participants and actions undertaken as a result can only be understood in the context of other relationships between participants. I.e., in order to understand the validity and consequences of a service interaction, it is necessary to understand the relative roles of the participants (and stakeholders generally).

3.2 Needs and Capabilities Model

The motivation for participants interacting is the satisfaction of needs. Participants use and deploy services in order that they can get their needs met. From a consumer perspective, the need is often related to the role they represent in the social structure; for the provider, the need is to gain satisfaction, monetary or otherwise, for use of the service.

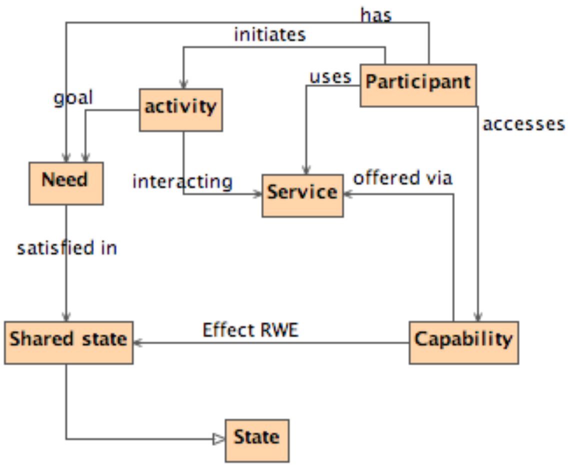


Figure 4 Needs and Capabilities

Capability

A resource that may be used by a service provider to achieve a real world effect on behalf of a service consumer.

As noted in the RM, the Real World Effect is couched in terms of changes to the state that is shared by the participants in the service; in particular the public aspects of that state. In this Reference Architecture we further refine this notion in terms of social structures.

Thus, we might refer to a capability as being able to effect facts that have meaning within a social structure; i.e., to be able to modify the state of the social structure. This does not rule out physical effects of using a service -- for all effects are ultimately rooted in the physical world -- but that the highest interpretation of a capability is in terms of the social structures in which it is embedded.

For example, a book selling service may have the capability of delivering a book to a customer. From a purely physical perspective, there is not much to distinguish one book from another, or one physical location from another. What makes the book selling service particular, however, is the ability to deliver the work of a particular author to a particular customer. Furthermore, even if the book has been successfully delivered, if the customer fails to pay for the book (if, for example, the credit card used turned out to be fraudulent), then the customer does not own the book, and is obliged to return it. The concept of transferring the ownership of the book, which is the real capability offered by the book selling service, has no direct counterpart in the physical world.

Capabilities themselves have owners, making those owners stakeholders in the SOA.

By making a capability available for use, via the Service, the owners aim to satisfy their needs as well as the needs of other participants who use the service. The extent to which a capability is exposed via a service (or via multiple services) is just one of the choices that the owners of capabilities have.

Need

> A need is a measurable requirement that a service participant is actively seeking to satisfy. The aspects of a need are that it can be measured and that it belongs to a participant (more generally any stakeholder).

A need is characterized by a proposition: an expression whose truth can be measured. However, needs themselves are often visible only to the owner of the need. Furthermore, the extent to which a need is captured in a formalizable way is likely to be very different in each situation.

Measuring expectations is not always easy, because while the actions of the service are private, only the results are public. In addition, there is always the potential for unexpected consequences of actions.

As an example, let us consider the ability to withdraw money from a bank account. Suppose as part of its implementation, the bank reports withdrawals over a certain amount to a credit agency. This potentially could alter an individual's credit report. The service description for the bank withdrawal service has to include this information so that a service consumer can accurately understand the actual real world effect. Of course this could be dynamic if the bank uses different credit reporting services that have different policies about credit ratings.

Figure 5 captures some of the key concepts and relationships involving needs. Since needs and policies share some of the same characteristics we include policy in the model; however, policies themselves are discussed in more detail in Section 4.4.

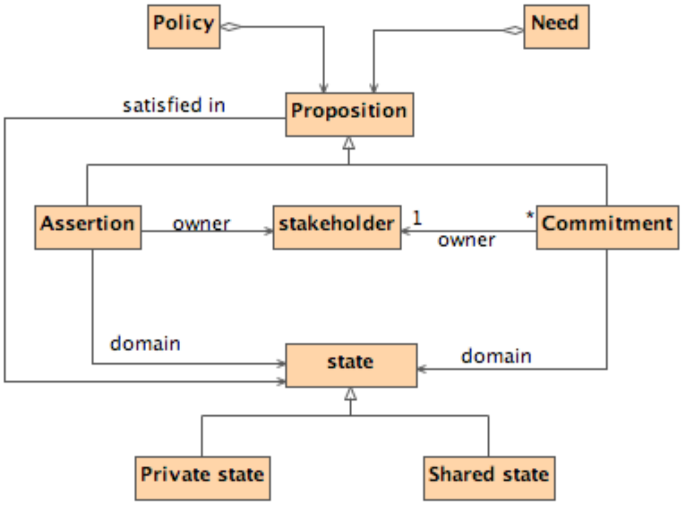


Figure 5 Needs expressed using propositions

A need is owned by a stakeholder. Many stakeholders are active participants in services, but not all.

3.3 Resources Model

As we noted above, a key relationship between many elements is that of ownership. The resource model focuses on what it is that can be owned: resources.

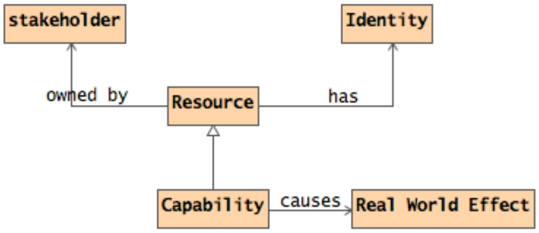


Figure 6 Resources

Our model of resources is very simple, but is the foundation for modeling many of the things that a SOA deals in: information, physical resources and so on:

Resource

> A resource is any entity of some perceived value, where the value may be in the function it performs or something intrinsic in its nature. For example, a diamond has value in its appearance, its physical properties, and its monetary worth; a data set has value in the information it carries. A resource that has identity associated with it – it can be identified – and has an owner.

Resources are not confined to physical entities: anything, real, virtual or abstract that may be owned is a resource. The key attributes of a resource are that it can be identified, and that it has an owner.

This definition of resource is a simplification and elaboration of the concept that underlies the Web architecture. Being more abstract, we do not require that the identity of a resource be in any particular form (although in practice, many resource identifiers are URIs), nor do we require resources to have representations. However, we do require resources to have owners.

One important class of resources in this architecture are the capabilities that underlie services. In this case, we can say that a capability is a resource that can cause an effect in the world. For example, a light bulb is a resource that when activated gives off light; a book is a resource that when read allows one to gain knowledge from its content. Other examples of resources are services themselves, descriptions of entities (a kind of meta-resource), IT infrastructure elements used to deliver services, contracts and policies, and so on.

3.4 Social Structure Model

The actions undertaken by participants, whether mediated by services or in some other way, are normally performed in the context of a social context which defines the meaning of the actions themselves. We can formalize that context as a **social structure**: the embodiment of a particular social context.

The social structure model is important to defining and understanding the implications of crossing ownership boundaries; it is the foundation for an understanding of security in SOA and also provides the context for determining how SOAs can be effectively managed and governed.

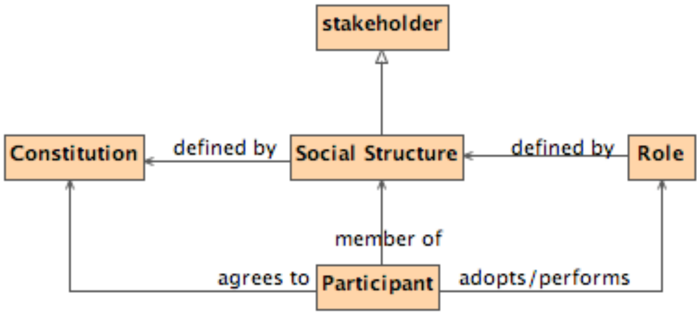


Figure 7 Social Structure

Social Structure

> A social structure (sometimes identified as social institutions) embodies some of the cultural aspects that characterize the relationships and actions among a group of participants.

Social structures are often, but not necessarily, aligned with organizations. For example, a meeting of like-minded fellows by a watering hole may be an informally defined social structure (with its own rules) that is not connected with any organization per se.

The richness of social structures reflects the richness of human culture itself. However, in the context of the Reference Architecture, we are concerned primarily with social structures that are embodied in legal and quasi-legal frameworks; i.e., they have some rules that are commonly understood.

So, for example, a corporation is a common kind of social structure, as is a fishing club. At the other extreme, the legal frameworks of entire countries and regions also count as social structures.

It is not necessarily the case that the social structures involved in a service interaction are explicitly identified by the participants. For example, when a customer buys a book over the Internet, the social structure that defines the validity of the transaction is often the legal framework of the region that the book vendor belongs to. This legal jurisdiction qualification is typically buried in the fine print of the service description.

3.4.1 Shared state and social facts

Most of the actions performed by people and most of the important aspects of a person's state are inherently social in nature. The social context of an action is what gives it much of its meaning. We call actions in society social actions and those facts that are understood in a society social facts. It is often the case that social actions give rise to social facts.

Social facts are inherently public or shared: they only have meaning in the context of the social structure and to the participants in the social structure.

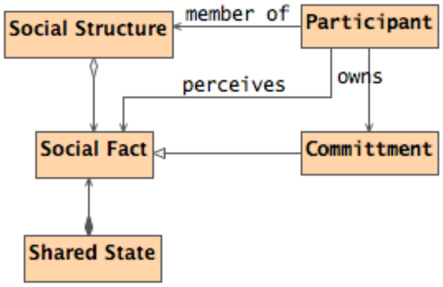


Figure 8 Shared state and social facts

Shared state

> The set of facts and commitments that manifest themselves to service participants as a result of interacting with a service.

Note that a participant has only a partial view of the shared state in a system. Furthermore, the participant will have internal state that is not accessible to other participants directly. However, elements of the shared state are in principle accessible to participants even if a given participant does not have access to all elements at any given time.

Social fact

> A social fact is an element of the state of a social structure that is sanctioned by that social structure. For example, the existence of a valid purchase order with a particular customer has a meaning that is defined primarily by the company itself.

Commitment

> A commitment is a social fact about the future: in the future some fact will be true and a participant has the current responsibility of ensuring that that fact will indeed be true. A commitment to deliver some good is a classic example of a fact about the future.

Other important classes of social facts include the policies adopted by an organization, any agreements that it is holding for participants, and the assignment of participants to roles within the organization. The social facts that are understood in the context of a social structure define the shared state that is referenced in Figure 5.

Social facts and commitments are inherently abstract; however, facts have the property of being verifiable (technically, a social fact can be verified to determine if it is satisfied in the social context). If as a result of interacting with a service, a buyer incurs the obligation of paying for some good or service, this obligation (and the discharge of it) is measurable (perhaps by further interactions with the same or other services).

3.4.1.1 Measuring social facts

We make considerable use of the term **proposition** and related concepts. Propositions are the basis of policies, agreements and contracts, ownership, social facts, shared state and many other elements of the architecture.

Proposition

> A proposition is an expression, normally in a language that has a well-defined written form, that expresses some property of the world from the perspective of a stakeholder. The truth of the proposition may be measured – using a decision procedure – by examining the world and checking that the proposition and the world are consistent with each other.

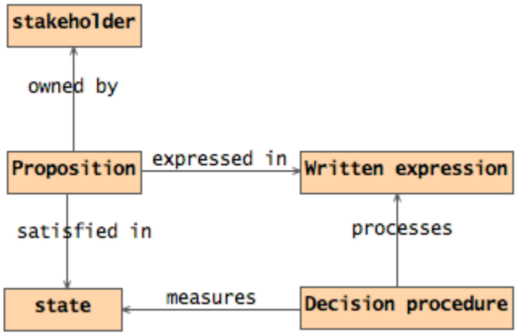


Figure 9 Propositions

There are two kinds of propositions that relate to needs (and policies), assertions and promises.

Assertion

An assertion is a proposition that is held to be true by a stakeholder. It is essentially a claim about the state of the world.

Promise

A promise is a proposition regarding the future state of the world by a stakeholder. In particular, it represents a commitment by the stakeholder to ensure the truth of the proposition.

For example, an airline may report its record in on-time departures for its various flights. This is a claim made by the airline which may (or may not be) verified. The same airline may promise that some percentage of its flights depart within 5 minutes of their scheduled departure. The truth of this promise depends on the effectiveness of the airline in meeting its commitments.

Another way of contrasting assertions and promises is to see what happens when the propositions fail: a stakeholder that makes a false assertion about the world might be classified as a liar; a stakeholder that makes a false promise is said to break its promises.

3.4.2 Acting in a social context

The essence of SOA is *action at a distance*: service participants interact with each other, possibly remotely, in order to act. There is always a desire to achieve an effect, an effect in the real world. Of course, there are many possible effects that are desirable and undesirable; we cannot, in general, completely characterize all of the effects of interacting with services.

In the context of SOA, actions are primarily social in nature -- one participant is asking another to do something -- and goal oriented -- the purpose of interacting with a service is to satisfy a need; by attempting to ensure that a remote entity applies its capabilities to the need.

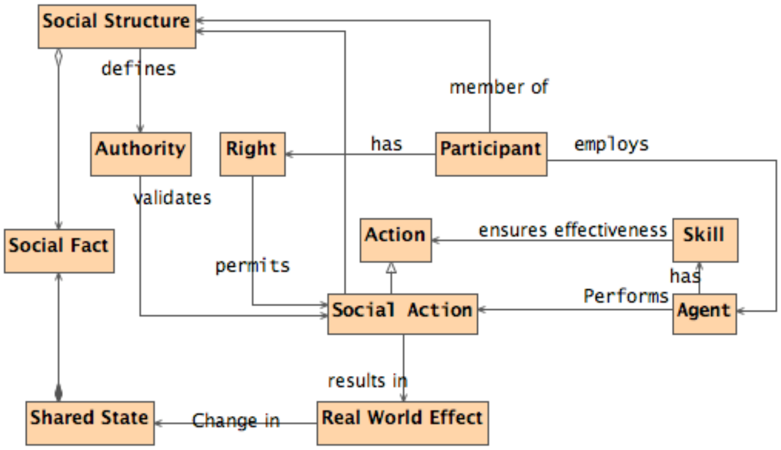


Figure 10 Acting within Social Structures

Real World Effect

The result of a participant performing an action in response to a service interaction.

Action

An action is the application of a capability on a target resource by an active entity (otherwise known as the agent of the action) in order to achieve an effect

The application of intent by a participant (or agent) to achieve a real world effect.

This important concept is simultaneously one of the fulcrums of the Service Oriented Architecture and a touch point for many other aspects of the architecture: such as policies, service descriptions, management, security and so on.

When participants interact with each other they are performing **joint actions**:

Joint Action

> The application of intent by two or more participants to achieve a real world effect.

Intent is at the heart of many social activities: it represents an agent's relationship to one or more of its goals:

Intent

> The relationship between an agent and its goals that signifies a commitment by the agent to achieve that goal.

Many, if not most, instances of Real World Effect involve acting in the context of a social structure; i.e., the effect desired is the establishment of one of more social facts.

More formally, we can model the concept of a social action:

Social action

> A social action is an action that results in a change in the state of a social structure by establishing one or more new social facts. A social action consists of a physical action together with an appropriate authority.

Social actions are always contextualized by a social structure: the organization gives meaning to the action, and often defines the requirements for an action to be recognized as having an effect within the organization.

Social facts typically require some kind of ritual to establish: the action itself is physical, its interpretation is social. For example, the existence of an agreed contract typically requires both parties to sign papers and to exchange those papers. If the ritual is not performed correctly, or if the parties are not properly empowered to perform the ritual, then it is as though nothing happened.

In the case of agreements reached by electronic means, this involves the exchange of electronic messages; often with special tokens being exchanged in place of a hand-written signature.

For example, the hiring of a new employee is an action that is defined by the hiring company (and not, for example, by the president of another company). For a hiring to be valid, it is often the case that specific business processes must be followed, with key actions to be performed only by suitably authorized personnel (such as the company CEO).

Right

> A right is a predetermined permission that permits the role player to perform some action or adopt a stance in relation to the social structure and other role players. For example, in most circumstances, sellers have a right to refuse service to potential customers; but may only do so based on certain criteria.

Authority

> The right to act on behalf of an organization or another person. Usually, this is constrained terms of the kinds of actions that are authorized, and in terms of the necessary skills and qualifications of the persons invoking the authority.

In fact, any entity may authorize another entity to act as its agent. Often the actions that are so authorized are restricted in some sense. In the case of human organizations, the only way that they can act is via an agent.

One of the primary benefits of formalizing the relationships between people in terms of groups, corporations, legal entities and so on, is that it allows greater efficiencies in the operation of society. However, corporations, governments and even society, are abstractions: a government is not a person that can perform actions -- only people can actually do things.

For example, a fishing club is an abstraction that is important to its members. The club, however, cannot act physically in the world. On the other hand, a person who is appropriately empowered by the fishing club can, and so, when that person writes a cheque and mails it to the telephone company, that action counts as though the fishing club has paid its bills.

An artificial agent is somewhere between a person and a corporation: non-human agents definitely can perform actions; however, given foreseeable technology, it would not be reasonable to expect an agent to take responsibility for its actions: instead some combination of the designer, builder, deployer, owner of the agent is ultimately responsible for the actions of non-human agents.

Skill

> A skill is a competence or capability to achieve some real world effect. Skills are typically associated with roles in terms of requirements: a given role description may require that the role player has a certain skill.

3.4.3 Transactions and exchanges model

An important class of joint action is the **business transaction**, or **contract exchange**.

Business transaction

> A business transaction is a joint action engaged in by two or more participants in which **resources** are exchanged.

A classic business transaction is buying some good or service, but there is a huge variety of kinds of possible business transactions.

Key to the concept of business transaction is the contract or agreement to exchange. The form of the contract can vary from a simple handshake to an elaborately drawn contract with lawyers giving advice from all sides.

A completed transaction establishes a set of social facts relating to the exchange; typically to the changes of ownerships of the resources being exchanged.

Business agreement

> A business agreement is an agreement entered into by two or more partners that constrains their future behaviors and permitted states. A business agreement is typically associated with business transactions: the transaction is guided by the agreement and an agreement can be the result of a transaction.

Business transactions often have a well defined life-cycle: a negotiation phase in which the terms of the transaction are discussed, an agreement action which establishes the commitment to the transaction, an action phase in which the agreed-upon items are exchanged (they may need to be manufactured before they can be exchanged), and a termination phase in which there may be long-term commitments by both parties but no particular actions required (e.g., if the exchanged goods are found to be defective, then there is likely a commitment to repair or replace them).

From an architectural perspective, the business transaction often represents the top-most mode of interpretation of service interactions. When participants interact in a service, they exchange information, perform actions that have an effect in the world, an so on. These exchanges can often (always?) be interpreted as realizing part of, and in support of, business transactions.

Business process

> A business process is a description of the tasks, participants' roles and information needed to fulfill a business objective.

Business processes are often used to describe the actions and interactions that form business transactions. This is most clear when the business process defines an activity involving parties external to the organization; however, even within an enterprise, a business process typically involves multiple participants and stakeholders.

In the context of transactions mediated and supported by electronic means, business processes are often required to be defined well enough to permit automation. The forms of such definitions are often referred to as choreographies:

Process Choreography

> The description of the possible interactions that may take place between two or more participants to fulfill an objective.

A choreography is, in effect, a description of what the forms of permitted joint actions are when trying to achieve a particular result. Joint actions are by nature formed out of the individual actions of the participants; a choreography can be used to describe those interlocking actions that make up the joint action itself.

3.4.4 Roles in Social Structures

Participants' actions within a social structure are often defined by the roles that they adopt.

Role

> A role is an identified relationship between a participant and a social structure that defines the rights, responsibilities, qualifications, and authorities of that participant within the context of the social structure.

For many scenarios, the roles of participants are easily identified: for example, a buyer uses the service offered by the seller to achieve a purchase. However, in particular in situations involving delegation, the role of a participant may be considerably more complex.

Role player

> In the discussions below we refer to the role player as a participant that is acting in a given role. We refer to this as the participant adopting a role.

A role player may adopt one or more roles; and have zero or more skills and qualifications.

Note that, while many roles are clearly identified, with appropriate names and definitions of the responsibilities, it is also entirely possible to separately bestow rights, responsibilities and so on; usually in a temporary fashion. For example, when a CEO delegates the responsibility of ensuring that the company accounts are correct to the CTO, this does not imply that the CTO is adopting the full role of CFO.

Rights, authorities, responsibilities and roles form the foundation for the security architecture of the Reference Architecture. We should be able to trace back any particular security policy to the appropriate relationships between the participants involved, the actions they are performing (or states that they are in) and the Social Structure governing the policy.

Responsibility

> A responsibility is an obligation on a role player to perform some action or to adopt a stance in relation to other role players.

For example, a role player adopting the role of secretary of a standards group is obliged to ensure that all the minutes of the various meetings are properly recorded; and members of certain standards groups are obliged to declare any pre-existing IP claims that may be relevant to the work of the groups.

Rights and responsibilities have similar structure to permissive and obligation policies; except that the focus is from the perspective of the constrained participant rather than the constrained actions.

In order for a person to act on behalf of some other person or on behalf of some legal entity, it is required that they have the power to do so and the authority to do so.

For example, what actually happens when an issuing agency determines the status of some stakeholder (i.e., person, corporate entity or non-human agent) is that some person performs some action (such as signing a certificate) that has the effect of qualifying the stakeholder. For that action to be valid, the person signing the certificate has to be empowered by the agency and must be acting within his or her authority.

Qualification

> A qualification is a public determination by an issuing authority that a stakeholder has achieved some state. The issuing authority may require some successful actions on the part of the

> stakeholder (such as demonstrating some skills). The qualification may have constraints attached to it; for example, the certification may be time limited.

For example, someone may have the skills to fly an airplane but not have a pilot's license. Conversely, someone may have a pilot license, but because of some temporary cause be incapable of flying a plane (they may be ill for example).

Qualifications are often used as constraints on roles: any entity adopting a role within an organization (or other social structure) must have certain qualifications.

3.4.5 Governance and Social Structures

Given that SOA mediates an important aspect of people's relationships, it follows that there are commitments entered into by participants that require enforcement by the community and that the SOA itself must reflect the requirements of the community itself.

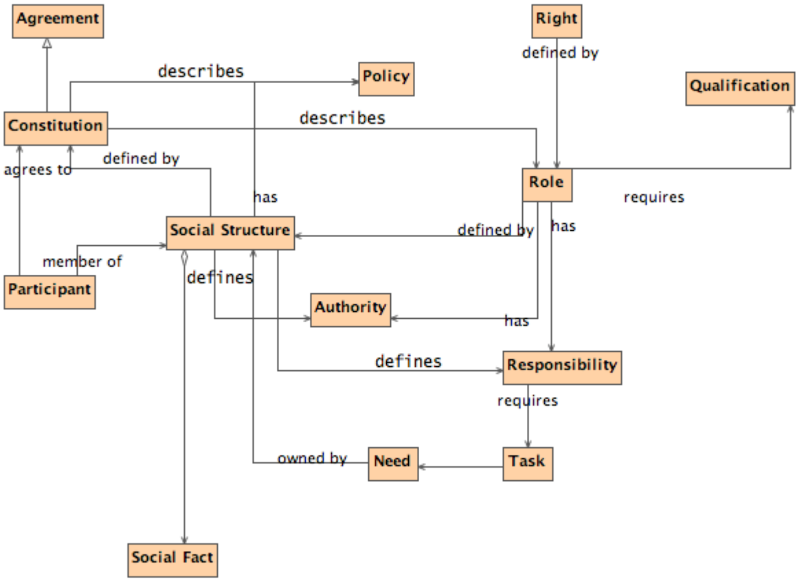


Figure 11 Social Structures and Governance

Both of these are aspects of the governance of Service Oriented Architecture.

The key elements of our model that relate to governance are the constitution of the social structure, the policies of the social structure, authority in a social structure, and the associated mechanisms of enforcement.

Constitution

> A constitution is an agreement which defines a social structure. The primary purpose of the constitution is to define the roles of participants in the institution, and how to establish the regulations that define the legal actions. The regulations of the social structure effectively define how those assertions and commitments that are relevant to the social structure are created.

For example, a company's constitution is normally called the "Articles of Association". A company's articles define the officers of the company, their rights and responsibilities and the purpose of the company. It will often also declare what the rules are for resolving conflicts.

A constitution is an agreement, and is also a social fact itself. It is agreed to by the participants in the social structure. For example, when a new employee joins a company, he or she is often required to sign an employment contract. That contract defines key aspects of the relationship between the new employee and the company.

With few exceptions, social structures are embedded in other social structures. One result of this is that the institution's constitution is often viewable as a social fact in one or more outer social structure. For example, the Articles of Association of a company is considered a legal document that supports the legal fact of existence of the company -- by the legal jurisdiction of the company.

The main exception to this is, of course, the agreement that defines the constitution of a country. Notably, for most people who are born into the country, its constitution is one that they often do not explicitly agree to. However, it is universal for people who are naturalizing their citizenship to be required to explicitly agree to the constitution of their new country.

4 Realizing a Service Oriented Architecture View

Make everything as simple as possible but no simpler.
Albert Einstein

The *Realizing a Service Oriented Architecture view* focuses on the infrastructural elements that are needed in order to support the discovery and interaction with services. The key questions asked are "What are services, what support is needed and how are they realized?"

The models in this view include the Service Description Model, the Visibility of Services Model, the Interacting with Services Model, the Realization of Policies Model, and the Policies and Contracts Model.

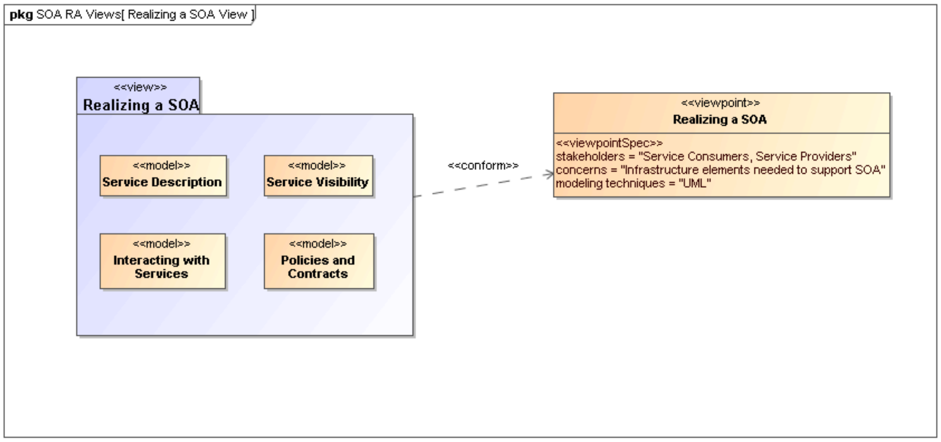

pkg SOA RA Views[Realizing a SOA View]
<<view>>
Realizing a SOA
<<model>>
Service Description
<<model>>
Service Visibility
<<model>>
Interacting with Services
<<model>>
Policies and Contracts
<<conform>>
<<viewpoint>>
Realizing a SOA
<<viewpointSpec>>
stakeholders = "Service Consumers, Service Providers"
concerns = "Infrastructure elements needed to support SOA"
modeling techniques = "UML"

Figure 12 Model elements described in the Realizing a Service Oriented Architecture view

4.1 Service Description Model

SOA depends on a wide variety of descriptions to characterize the needs and capabilities it can facilitate connecting. Description elements, such as those indicating the real world effects produced by a service and those desired by the consumers, provide the basis for determining the match between consumers and providers. Policies and attributes that are needed to evaluate policy compliance are also important elements to determine the conditions under which interactions may be initiated and continue to completion, and description can inform as to which policies may or must be applied.

For SOA to enable efficient connectivity between providers and consumers, descriptions must provide sufficient information to achieve visibility between the provider and consumer and to support continued interaction. The information provided by description may be augmented during the interaction. For example, the interaction may reach a point where message exchanges must be encrypted; it may or may not be important that the description indicate that at some point encrypted messages may be required. The critical point is that this additional information becomes available during the interaction and neither the provider nor the consumer is required to have undocumented a priori details about the other, including details of their needs and capabilities, in order for interaction to be initiated or proceed.

Several points to make:

- This model currently focuses on the description of services but it is equally important to consider the descriptions of the consumer and possibly other participants.
- Descriptions are inherently incomplete. The necessary elements of description depend on the context. The intent of "standard" description sets is to capture "essential" information, i.e. that most

likely to be needed. It should be understood that what is considered essential will change over time as, for example, the ingredients and nutrition information for food labeling. A requirement for transparency of transactions may require additional description for those associated contexts.

- Description always proceeds from a basis of what is considered "common knowledge". This may be social conventions that are commonly expected or possibly codified in law. It is impossible to describe everything and it can be expected that a mechanism as far reaching as SOA will also connect entities where there is inconsistent "common" knowledge.
- The basis for determining when a description is *sufficient* is quite simple: is it possible for the intended audience of participants to use the descriptions to provide and access services that are offered and used by them. This means that, at one end of the spectrum, a description along the lines of "*That service on that machine*" may be sufficient for the intended audience. On the other extreme, a service description with a machine-process-able description of the semantics of its operations and real world effect may be required for services accessed via automated service discovery and planning systems. Generally, somewhere in between these extremes is effective.
- Descriptions of the provider and consumer are the essential building blocks for establishing the execution context of an interaction.

4.1.1 Components of Service Description

A service description is an artifact, usually document-based, that defines or references the information needed to use a service. This includes not only the information and behavior models associated with a service and is needed to define the service interface but also includes information needed to decide whether the service is appropriate for the current needs of the service consumer. Thus, the description will also include information on data associated with service reachability, service functionality, and the policies and contracts associated with a service. More specifically, a service description must convey the following:

- Service Reachability - The ability for service participants to locate and interact with one another. Reachability includes the address to access the service (i.e., the endpoint), an indication of whether the service is currently available (i.e., the service presence), and the protocols needed to communicate with the service. The service presence may include a static representation of conditions, a representation that is updated at regularly defined intervals, or a dynamic means to assess the current service availability.
- Service Functionality - An unambiguous expression of service function(s) and the real world effects of invoking the function. The Functions may be expressed as natural language text, reference to an existing taxonomy of functions, or reference to a more formal knowledge capture providing richer description and context. This portion of description should also include technical assumptions, dependencies, or limitations that underlie the effects that can result. [THERE IS OBVIOUSLY A CONNECTION BETWEEN FUNCTIONS (VERBS) AND RWE (NOUNS). DO WE NEED BOTH FUNCTIONS AND EFFECTS EXPLICITLY DESCRIBED?]
- Interaction Policies & Contracts - Information for prospective consumers pertaining to conditions or constraints when interacting with a service. Whereas technical assumptions, dependencies, and limitations described under the Service Functionality are statements of "physical" fact, policies are subjective assertions made by the service provider (sometimes as passed on from higher authorities) and contracts are agreements on policies between the service provider and some consumer or consumer community. For example, a policy can be that the consumer must have purchased a particular license to use the service, but a contract can identify a specific consumer community who operates under a separately negotiated license. Policies & Contracts are often associated with Performance Metrics through Service Level Agreements, as discussed below.
- Information Model - A definition of the data model required to exchange information with a service and invoke actions defined in the behavior model. The Information Model includes both the Structure and Semantics of the information exchange. Messages are constructed that conform to the structures defined in the Information Model, and the pattern for using individual messages follows that defined by the Message Exchange Pattern.

- Behavior Model - The characterization of the responses to and the temporal dependencies between the actions that can be performed on a service. The Behavior Model is discussed in more detail below.
  - Action Model - The description of actions that may be performed against a service.
  - Process Model - The temporal relationships between service actions and events

    [ACTIONS SEEM TO CORRESPOND TO WSDL OPERATIONS AND I THINK I NEED TO MAKE THIS CONNECTION. WSDL DOESN'T HAVE THE EQUIVALENT OF A PROCESS MODEL, AND I THINK THIS WOULD BE EXPECTED TO BE HANDLED BY SOME REFERENCED BPEL ORCHESTRATION. SO THE DIFFICULTY IS HOW TO KEEP THIS FROM GETTING TOO DEEPLY INTO WS AND HOW TO AVOID THE FACT THAT THERE ARE TWO RELEVANT OVERLAPPING BUT NOT IDENTICAL WSDL VERSIONS.]

4.1.2 The Model for Service Description

These aspects of description and their relationships are among the elements of description are shown in Figure 13. In this figure, Service Description is shown as a subclass of a general Description class. Participant description is another subclass that is not elaborated here. The mechanism by which values are associated with description is illustrated in Figure 14.

Any description should have associated identity – so a description instance can be referenced – and provenance – so the entity responsible for the subject being described can be identified. In addition, the description may associate its subject with predefined keywords or classification taxonomies that derive from reference-able formal definitions and vocabularies.

The general description instance may also reference associated documentation that is in addition to that considered necessary in this model. For example, the owner of a service may have documentation on best practices for using the service. Alternately, a third party may certify a service based on their own criteria and certification process, but such a certification may be vital information to other prospective consumers if they were willing to accept the certification in lieu of having to perform another certification themselves. Note, while the examples of Associated Documentation are related to services, the concept applies equally to description of other entities.

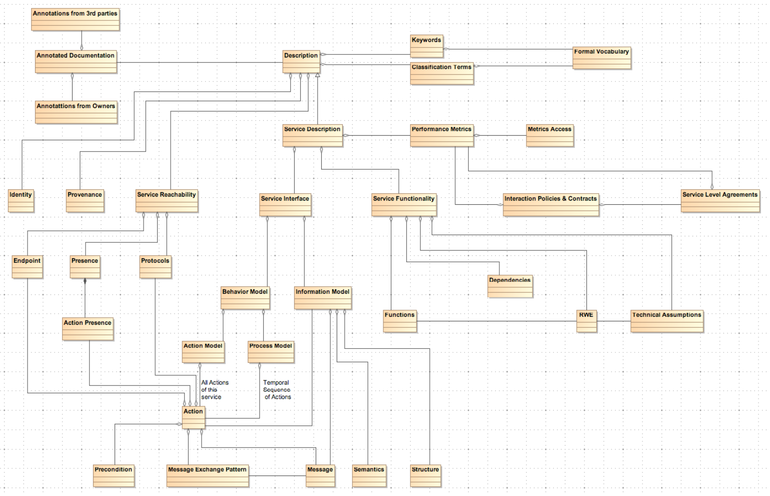


Figure 13 Service Description

The Service Description subclass may also be composed of information related to Performance Metrics. As with many quantities, the performance metrics are not themselves defined by the Service Description but metric values or the means to access such values may be an important part of description. It is from such performance metrics that Service Level Agreements may be defined.

4.1.3 Service Description in support of Service Interaction

If we assume we have awareness, i.e. access to relevant descriptions, the service participants must still establish willingness and presence to ensure full visibility [ref visibility section] and interact with the service. The agreements that establish conditions for willingness are collected in the execution context of the interaction. The execution context can be thought of as a series of answers to the questions of why would the participants be willing to interact and whether such interaction is possible.

From a description standpoint, a consumer would show interest in a service if the service functionality is what is needed and the service policies are at least worth pursuing if not immediately acceptable. By saying functionality is of interest, we are saying the (business) functions and real world effects (RWE) are of interest and there is nothing in the dependencies or technical assumptions that is a showstopper. Note at this level, the business functions are not concerned with the action or process models. These models get into the nuts and bolts of making the business function happen and will be dealt with at that level later.

A service can result in more than one RWE from a business function. In addition, there can be multiple dependencies for the service to successfully complete its functions and there can be numerous policies that provide conditions that may affect willingness. For a service with a single business function (see extension to multiple business function variations below) identifying interaction policies and applicable contracts for the service as a whole and dependencies, technical assumptions, and real world effects as part of service functionality is sufficient.

At this point, let us assume the descriptions were sufficient to establish willingness; the details of working this out are considered elsewhere. Figure 12 indicates the service endpoint establishes where to go to actually carry out the interaction. This is where we have to start considering the action and process models.

We may have multiple actions a user can perform against a service and the user would perform these in the context of the process model. For a given business function, there is a corresponding process model, where any process model may involve multiple actions. Each action has its own endpoint and also its own protocols associated with the endpoint[1] and whether there is presence for the action through that endpoint. How presence through any endpoint relates to presence of the service is still an open question but likely presence of a service is an aggregation of the presence of the service's actions, and the service level may aggregate to some degraded or restricted presence if some action presence is not confirmed. For example, if error processing actions are not available, the service can still provide required functionality if no error processing is needed. This implies Reachability in some local sense for each action and Reachability also applying at the service/business function level.

An action may have preconditions where a precondition is something that needs to be in place before an action can occur, e.g. confirmation of a precursor action. Whether preconditions are satisfied is evaluated when someone tries to perform the action and not before. Presence for an action means someone can initiate it and is independent of whether the preconditions are satisfied.

A service may have dependencies. As stated above, the presence of a service is some aggregate of the presence of its actions. A dependency does not affect the presence of a service although it may affect whether the business function successfully completes.

In summary to this point, (1) actions has reachability information, including endpoint and presence, (2) presence of service is some aggregation of presence of its actions, (3) action preconditions and service dependencies do not affect presence although these may affect successful completion.

Having established visibility, the interaction can proceed. Given a business function, the consumer knows what will be accomplished (the service functionality), the conditions under which interaction will proceed (service policies and contracts), and the process that must be followed (the process model). Given the process model, the consumer knows which actions need to be performed; given the action, the consumer knows the endpoint and protocol to be used and whether there is presence for the action. The remaining question is how does the description information for structure and semantics enable interaction.

In the discussion above, we indicate the importance of the process model in identifying relevant actions and their sequence. Interaction with the actions are through messages and thus it is the syntax and semantics of the messages with which we are concerned. There seems to be a number of ways to approach this but the common way now[2] is to define the structure and semantics that can appear as part of a message and then assemble the pieces into messages and associate messages with actions. Actions make use of structure and semantics as defined in the information model to describe its legal messages. In addition, MEP defines sequencing and use of messages for a given action.

So to continue from above, the process model identifies actions I need to perform against a service and the action sequence. I check to see what protocol bindings are available and then check for the endpoint and (possibly) whether there is presence at that endpoint. The interaction with actions is through messages that conform to the structure and semantics defined in the information model and the message sequence conforming to the action's identified MEP. The result is some portion of the RWE initially examined in the service description (e.g. only that part that covers the processing error generated).

[1] This is analogous to a WSDL 2.0 interface operation (WSDL 1.1 portType) having one or more defined bindings and the service identifies the endpoints (WSDL 1.1 ports) corresponding to the bindings. [While I do not intend to make this WSDL specific, it should have an obvious mapping.]

[2] WSDL defines syntax through <types> and SAWSDL proposes to add a pointer to semantics at various places in a WSDL document.

Ken Laskey 5/16/07 5:17 PM
Formatted: Font:9 pt

Ken Laskey 5/16/07 5:29 PM
Formatted: Font:9 pt

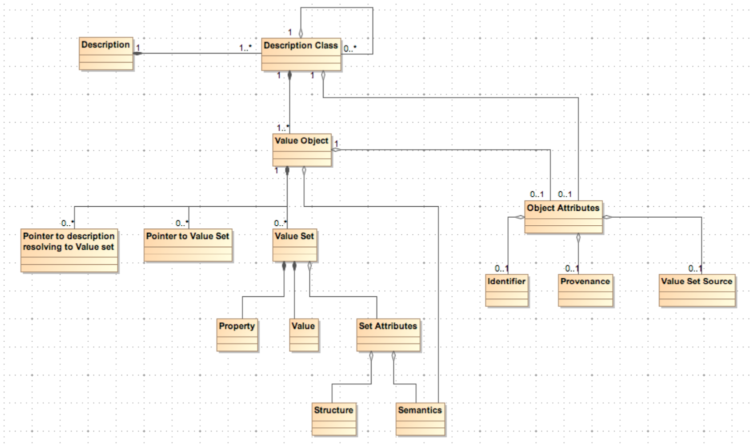
4.1.3.1 The question of multiple business functions

It is assumed a service provides a well-defined business function. That is simple. Could (should) it also optionally provide variations of the business functions, e.g. different qualities of service. [I HAVE PREVIOUSLY ARGUED NOT BUT LET'S SEE WHERE THIS TAKES US. I WOULD BE INTERESTED TO HAVE OTHER VARIATION EXAMPLES OTHER THAN QOS FOR WHICH VARIATIONS MAKE MORE SENSE THAN SEPARATE SERVICES (BUT DRAWING A BLANK AT THE MOMENT).]

As noted, a service can have more than one RWE from a business function. There can also be multiple dependencies. If there is more than one business function, then we probably need to identify whether a given dependency or RWE is connected with the service as a whole or one of the business functions. (If there is one business function, relating to the service or its business function is one and the same.) WOULD WE SAY THAT BY DEFINITION ALL DEPENDENCIES, RWE, AND TECHNICAL ASSUMPTIONS RELATE TO ALL BUSINESS FUNCTION VARIATIONS AND SO CAN BE CONNECTED AT SERVICE FUNCTIONALITY? ARE THERE REASONABLE COUNTEREXAMPLES?

We also have policies connected with a service. If we have multiple business functions, does it make sense that some policies relate to only a subset of the functions? One could would say the QoS variation does not require this because we can have a single QoS policy that encompasses the variations. [ARE THERE COUNTER-EXAMPLES TO SAYING POLICIES RELATE TO THE SERVICE AND THE EXPRESSION OF POLICY AT THE SERVICE LEVEL CAN ENCOMPASS POLICY VARIATIONS ASSOCIATED WITH THE BUSINESS FUNCTION VARIATIONS?]

As noted above, we may have multiple actions a user can perform against a service and this does not change with multiple functions. The idea of presence, especially an aggregated determination of presence, would likely be affected. A business function corresponds to a process model, so multiple business functions imply multiple process models because either the process is different or the specific



ısed in multiple process

s function because the

rocess model, (2) its own

Figure 14 Representation of a Description Class

Figure 13 shows the template for a service description but individual description instances depend on the ability to associate meaningful values; this is described through the structures shown in Figure 14. As shown, each class is represented by a value object (N.B. this is a temporary term that should be changed to something less computer code oriented) or is made up by components that will eventually resolve to a value object. For example, Description has several components, one of which is Identity. Identity will be represented by a value object.

A value object consists of

- a collection of value sets with associated property-value pairs, pointers to such value sets, or pointers to descriptions that eventually resolve to value sets that describe the component; and
- attributes that qualify the value sets.

The qualifying attributes include

- an optional identifier that would allow the value set to be defined, accessed, and reused elsewhere;
- provenance information that identifies the party (individual, role, or organization) that has responsibility for assigning the value sets to any description component;
- an optional source of the value set, if appropriate and meaningful, e.g. if a particular data source is mandated.

If the value object is contained within a higher-level component, (such as Service Description containing Service Functionality), the component may inherit values for the attributes from its container.

Note, provenance as a qualifying attribute of a value object is different from provenance as part of a general instance of Description or, more specifically, a service description. Provenance for a service identified who "owns" the service, i.e. who is responsible for its creation, maintenance, and provisioning. Provenance for a value object identifies who is responsible for choosing and assigning values to the value sets that comprise the value object. It is assumed that granularity at the value object level is sufficient and provenance is not required for each value set.

The value set also has attributes that define its syntax and semantics.

- The semantics of the value set property should be associated with a semantic model conveying the meaning of the property within the context for use, where the semantic model could vary from a free text definition to a formal ontology.
- For numeric values, the syntax would provide the numeric format of the value and the "semantics" would be conveyed by a dimensional unit with an identifier to an authoritative source defining the dimensional unit and preferred mechanisms for its conversion to other dimensional units of like type.
- For nonnumeric values, the syntax would provide the data structure for the value representation and the semantics would be an associated semantic model.
- For pointers, architectural guidelines would define the preferred addressing scheme.

The value object may indicate a default semantic model for its component value sets and the individual value sets may provide an override.

4.1.5 Relationship to Other Description Models

While the representation shown in Figure 14 is derived from considerations related to service description, it is acknowledged that other metadata standards are relevant and should, as possible, be incorporated into this work. Two standards of particular relevance are the Dublin Core Metadata Initiative (DCMI) and ISO 11179, especially Part 5.

When the service description (or even the general description class) is considered as the DCMI "resource", Figure 14 aligns nicely with the DCMI resource model. While some differences exist, these are mostly in areas where DCMI goes into detail that is considered beyond the scope of the current Reference Architecture. For example, DCMI defines classes of "shared semantics" whereas for the Reference Architecture, it is sufficient to prescribe that an identification of relevant semantic models is sufficient. Likewise, the DCMI "description model" goes into the details of possible syntax encodings whereas for the Reference Architecture it is sufficient to identify the relevant formats.

With respect to ISO 11179 Part 5, the metadata fields defined in that reference may be used without prejudice as the properties in Figure 14 above. Additionally, other defined metadata sets may be used by the service provider is the other sets are considered more appropriate, i.e. it is fundamental to this Reference Architecture to identify the need and the means to make vocabulary declarations explicit but it is beyond the scope to specify which vocabularies are to be used. In addition, the identification of domain of the properties and range of the values has not been included in the current Reference Architecture discussion, but the text of ISO 11179 Part 5 can be used consistently with the model prescribed in this document.

4.1.6 Implications of the Description Model

There are numerous implications that follow from a consideration of the description model shown in Figure 13.

1. The overall description model is applied to the service and not the components of the service. For example, the Action Model identifies numerous actions that can be performed against a service and the Process Model defines the order in which the actions are performed, but the real world effects are defined for the service and not the individual actions. Similarly, numerous policies may be associated with a service, but individual policies are not associated with each action. Thus, a SOA service must represent an identifiable business function to which policies can be applied and from which desired business effects can be obtained.

1a. Specifying a model where policies are not associated at the action level but only at the service level goes against many of the discussions of SOA use but it is not obvious that there is really a significant body of practice that effectively uses such distributed information. If I apply policies at the action level, how do I describe the service for use when it may have hidden policies embedded anywhere in its structure? Can I ever describe a service or must I describe every operation/action/endpoint (o/a/e)? Does this say every o/a/e should be its own service? Does it say the service description has to be structured in such a way that it describes details of its every o/a/e? Does a service description basically become the structure that holds all the o/a/e descriptions? What does this say about discovery? Is there any value in discovering a service or do I discover o/a/e? Does this get me back to every o/a/e is a service?

1b. For this model, actions are assumed to correspond to WSDL operations – that is not necessary but it is sometimes useful. An o/a/e is an action with which some external entity must interact in order for the RWE to be realized. The sequence of such actions is defined in the process model. Only actions needed to realize the RWE are actions of that service. The RWE should be kept well scoped so a service does not do a number of unrelated, albeit useful, things. For example, while stock quotes are nice and weather reports are nice, you wouldn't include these as o/a/e for one service. I would argue that different QoS are different services. Note, more than one service can access the same capability, and this is appropriate if a different RWE is provided. I would argue that getting a response in one minute rather than one hour is more than a QoS difference; it is a fundamental difference in the business function I am receiving.

1c. That all said, RWE in Figure 13 connects to both Functionality and Action, and Policies connect to Actions. What does that mean? For RWE, it is simple: the functionality is described in terms of the RWE realized and Actions against the service result in the RWE. However, the RWE can be described at the service level and the consumer typically does not need to know which actions in particular are needed for which RWE. This, however, may need to be discussed in regard to availability as questioned in item 6 following. HOWEVER, IF IN GENERAL THE CONSUMER DOESN'T NEED TO KNOW WHICH ACTION(S) LEAN TO A PARTICULAR RWE, IS THIS LINE NEEDED IN THE SERVICE DESCRIPTION DIAGRAM?

1.d Back now to policies and their connection with actions, policy alternatives should result in different RWE and this relationship should be unambiguously expressed at the service level. Analogous to what was noted in 1.c., the specific policy relationship to a particular action is not something the consumer typically needs to know, although there may be an availability issue to consider. SO, IF IN GENERAL THE CONSUMER DOESN'T NEED TO KNOW WHICH POLICIES RELATE TO WHICH ACTION(S), IS THE LINE NEEDED IN THE SERVICE DESCRIPTION DIAGRAM?

1.e The next question comes in at item 4 below when we take up preconditions.

3.1. A few words on identity and privacy [likely to be moved elsewhere]

Identity needs to be sufficient so one can evaluate provenance when deciding on use of a service but not so complete as to violate privacy. For example, I may make my office phone number publicly available but I do not want to do the same with my cell phone number. For a participant description, there will be the publicly available information, and Web Services would provide an interesting mechanism for attribute-based access control of additional information. So my cell phone number could be accessed by someone in the role of emergency medical personnel. In a less dramatic fashion, someone who is a registered user of my service could get access to me for priority support.

4.1.7 Identity and versioning

Identity requires an unambiguous identifier. In some cases, a new identifier could be assigned for each version, but this would get tedious, would break interfaces when there were updates, and violates Web architecture which states there should be no proprietary decomposition of a URI to extract additional information. Assuming a resource has one identifier that covers multiple versions, then the description should also unambiguously identify the version value and the version definition from which the values derives. For example, a versioning scheme of (i.j.k) can indicate

* increment i when enhancements do not guarantee backward compatibility;

* increment j for enhancements for which backward compatibility is guaranteed for any previous j for the same i;

* increment k for bug fixes with no compatibility impacts.

From the perspective of a flexible SOA architecture, it is not important and should probably not be codified within the general SOA description as to what are the details of an applied versioning scheme but it should be clear what scheme is being used and what is available to the consumer to decipher a value derived from the scheme. Similar principles will be used when discussing attributes such as status.

<A discussion of how version designation/value should be impacted by change of version of any of its components (or any descriptions?) still needs to be discussed.>

4.1.8 Service Description and Service Level Agreements.

Service Level Agreements (SLAs) receive a great deal of attention from many of those interested in SOA because they allude to metrics that can be the basis for charging models and thus the economic models for SOA. SLAs also have a comfortable feel as the business agreements through which business services can be procured. In its extreme, SLAs encompass all aspects of service description because it would prescribe the details of service use and the conditions under which performance metrics are obtained and analyzed. Unfortunately, the expanded role of SLA as service description does not support the primary discriminators of services for which contracts of use are not in place and results in massive redundancies if the characteristics of the service must be repeated for the enforcement functions surrounding such SLAs.

In the current model, SLAs have a more limited function that derives from the idea of associated policies and agreements on policies that eventually form the basis for service interactions proceeding within an execution context. To begin, providers and consumers both have policies and possibly technical assumptions (i.e. physical, representational, or system-level constraints) that must be aligned in order for interactions to proceed among service participants. The agreements which encapsulate the necessary alignment form the basis upon which interactions may proceed – in the SOA Reference Model, this collection of agreements and the necessary environmental support establish the execution context. Note that the policies and constraints are properties of the participants and are the basis for any future interactions among for new or alternate participants but the final contracts/agreements are less reusable and more specific to a particular execution context.

Where then do SLAs fit in? In the current model, SLAs are the agreed upon values that performance metrics are intended to satisfy for any contract. The metrics may apply to a single instance of an execution context, over several reuses of a single or prescribed set of execution contexts, or at the extreme, an average over all interactions. The metrics can be cumulative or scoped to a particular period of time. The SLA is the prescribed value associated with an identified Performance Metric, where the Performance Metrics are fundamental to the service. Using such a definition, SLAs become valuable as measurable targets and can be modified as appropriate without affecting more permanent and more broadly relevant elements of the service description.

4.1.9 Consumer Description

Illustration - Consumer Description

What's not listed in the SOA RM but has to be done for an implementation is Service Description versioning and version compatibility.

<DETAILS OF CONSTRUCTING AN INFORMATION MODEL GOES ELSEWHERE BUT NOT SURE WHERE. HOW TO REFERENCE INSTANCES OF MODELS OR MODEL DEFINITIONS GOES IN DESCRIPTION.>

4.1.10 Information Model

Within the context of SOA, the information model is a characterization of the information that is associated with the use of a service. The information model describes the structure, format, and meaning of information and data that may be exchanged with a service as well as prescribing what information needs to be provided to the service in order to access its capabilities and interpret responses.

4.1.10.1 Data-Level Information Model

Layering

Data Formats, Elements and Definitions

Schema

4.1.10.2 Message Level Information Model

The message level information architecture can be divided into several areas—the content type of messages exchanged within the architecture, the type of packages in which they are enclosed, the metadata associated with messages, the topic space in which they are exchanged, and the security information carried by them.

4.1.10.2.1 Message Types

Messages should be classified into types based upon their content. The type of a message will usually indicate how it is processed by the receiver. Messages of differing types are most commonly distinguished by the format and syntax of their content, but can also be distinguished by the format of the message envelope, a type name, and, for example, a URL endpoint, an applicable XML schema, etc., or even by source or destination applications.

The taxonomy of message type names should be defined such that it aids in interpretation of the message content. Message types generated by a particular service or subsystem should begin with a common prefix or namespace specifier. Some message types may fall into a natural hierarchy, for example, [SPECIFY AN EXAMPLE]. The taxonomy for message types could embed this hierarchy within the message type name.

The metadata accompanying a message should indicate its type. The message type name should be adequate for locating and interpreting all registry, schema, and data dictionary information describing the message content. This information should be published in the metadata registry-repository.

<THIS SHOULD BE HANDLED IN INTERACTION. AGAIN, HOW TO REFERENCE INSTANCES OF MODELS OR MODEL DEFINITIONS GOES IN DESCRIPTION.>

4.1.10.2.2 Message Topic

In some circumstances it may be relevant to identify the topic of subject of a message. For example, *this message is about your invoice #324510*. This allows processing of messages based on topic identifiers rather than functional end-point: for example, *invoices go here and purchase requests go there*.

4.1.10.2.3 Topic Space Division

In publish-and-subscribe (pub/sub) messaging, topics, queues, and/or messaging endpoints divide up the subject space. For the purposes of this reference architecture, all such divisions will be referred to as topic divisions. NOTE: Message exchange patterns (MEPs) such as pub/sub are described in the Interacting with Services model of the Realizing Services view.

Topics need names that uniquely identify them to the participants and logically divide up the message space. The granularity of these is also important. Too many divisions can be hard to manage and may require subscribers to listen to many topics, or require publishers to duplicate messages on many topics. Too few divisions can result in subscribers getting far more messages than they need. (Metadata keywords in the message header should be used to mitigate the latter problem by allowing subscribers to filter messages). In general, the following guidelines apply:

Topic division should be consumer-oriented. If all the primary consumers of messages generated by a given publisher divide messages up by country ID and by no other classification, then topics should be created for the various countries involved, even if this violates other guidelines for topic division.

- The fewer different types of messages delivered on a single topic, the easier the consumer’s job tends to be.
- Topic divisions should be chosen such that they will balance load without resulting in unnecessary topic proliferation.
- All topics, queues, and endpoints specific to a particular organization or common service should be named using a common prefix or top-level namespace, for example “ORG2.”

The first division of topics within the common top-level namespace should be by message type. In other words, topic names should follow the naming pattern <ORG-prefix>.<ORG-message-type> If further qualification is required, more specific topic divisions can be made, and additional suffixes appended to the topic name.

- Temporary endpoints should be used wherever possible and appropriate.

4.1.10.2.4 Topic Discovery

As more messaging topics are made available across organizational boundaries as well as the enterprise, a means to topic discovery is required. Topic discovery will follow the other information discovery techniques described in the Service Visibility model. That is, information about the topic will be made discoverable in agreed upon-compliant metadata catalogs that are searchable using standardized mechanisms.

Depending on the type of messaging employed, the metadata used to describe topics could be specific to an organization or more general across multiple organizations. For instance, if it is a topic that is used to communicate information about business-to-business transactions for say, purchase orders, the topic metadata could provide information first about the organization itself and further break down the information to describe the specific information for that organization, in this case purchase orders.

The metadata catalog will provide information on the topic of interest including a number of other pieces of descriptive metadata that can be searched for including for example, quality of service, security requirements, etc.

4.1.10.2.5 Metadata Strategy

Messages should be accompanied by, or have references to, appropriate metadata. If references are used, they should be globally resolvable, for example, references to the metadata registry-repository or globally available URIs. Metadata should describe the message type and pedigree, as well as allowing subscribers to filter messages appropriately. The primary means for achieving this is through message headers. Again, this type of metadata should be registered in the metadata registry-repository.

4.1.10.2.6 Message Headers and Pedigree

A transport-independent metadata header for each message should identify

- The message type
- The source and destination of the message
- The creation or last modification time of the message
- The security properties of the message
- The purpose of the message
- The expiration time of the message
- Any other properties used for filtering by subscribers

While some of this information may be placed into transport specific headers, a standard content header should also either be present in the “body” of the message, for transport independence, or extractable from the transport-specific header in a non-transport dependent format. Destination should be included only as required by the addressing mechanism of the messaging transport, and should then be included only in the message header and not in the body. This is because, in pub/sub messaging systems, decoupling of senders from receivers is a key feature—publishers do not have to know where subscribers are.

The following guidelines apply to metadata content:

- Standard metadata keywords and/or tags (e.g., XML tags) should be defined and registered for use in message content headers
- The standard service description documentation should indicate which metadata items are required in the message header, and which are optional
- Message producers may need to place producer or purpose-specific keywords/tags in the message header as well. These may be metadata keywords used for a particular type of message, event, organization, subsystem, etc. To avoid naming collisions, these metadata keywords should adhere to a naming standard that uses prefixes or namespaces to identify their purpose
- For maximum transport independence, keywords/tags should be alphanumeric, may include “_”, and should start with a letter.
- It is best if metadata values are confined to simply data types supported by the various schema and messaging technology standards available (e.g., W3C XML schema or otherwise).

4.1.10.2.7 Message Format

A message should generally have the structure shown in the UML package diagram shown below. Specifics of the syntax and semantics are determined by the transport protocol, and the type of the message envelope itself is also determined by the transport.

Messages are divided into: 1) the transport header, 2) the transport body, and message attachments. The transport header includes transport housekeeping, routing, and security information, as well as some metadata keywords/tags. The transport body contains the data to be transmitted. The transport body is further divided into two blocks: 3) the transport independent content header and 4) the producer content.

The following guidelines are best practices for formatting messages:

- The transport header should harbor routing and security information, and all metadata required by the specific transport.

- The content header should contain required metadata keywords or tags, although they may be duplicated in the transport header if needed (for instance, for message filtering).
- The producer content should be interpretable given the message type indicator included in the metadata.
- Producer content that is text should use a universal data standard whenever possible (e.g., XML). Producer content that is standard should be defined by a standard schema (e.g., W3C XML schema).
- Attachments should be identified using a standard designation that will allow the receiver to interpret the attachment (e.g., IETF Multipurpose Internet Mail Extensions)(MIME)).
- Attachments that are not self-describing should be described by the metadata in the content and/or transport headers.

4.2 Service Visibility Model

One of the key requirements for participants interacting with each other in the context of an SOA is achieving visibility: before you can talk to someone, you have to know that they exist.

4.2.1 Visibility to Business

The relationship of visibility to the SOA ecosystem encompasses both human social structures and automated IT mechanisms. It is within the social structure where governance of SOA visibility takes place, the automated mechanisms for SOA visibility provide greater flexibility in how SOA service visibility is attained.

Figure 15 depicts a business setting that is a basis for visibility as related to the Business Via Services View (see Section 3.2). Service consumers have goals specified by needs. Service providers have capabilities. Mediated or direct visibility between consumers and producers facilitates interactions that lead to satisfaction of needs.   The forms of mediation for visibility provide increased flexibility for consumer/producer interoperability within complex social structures.

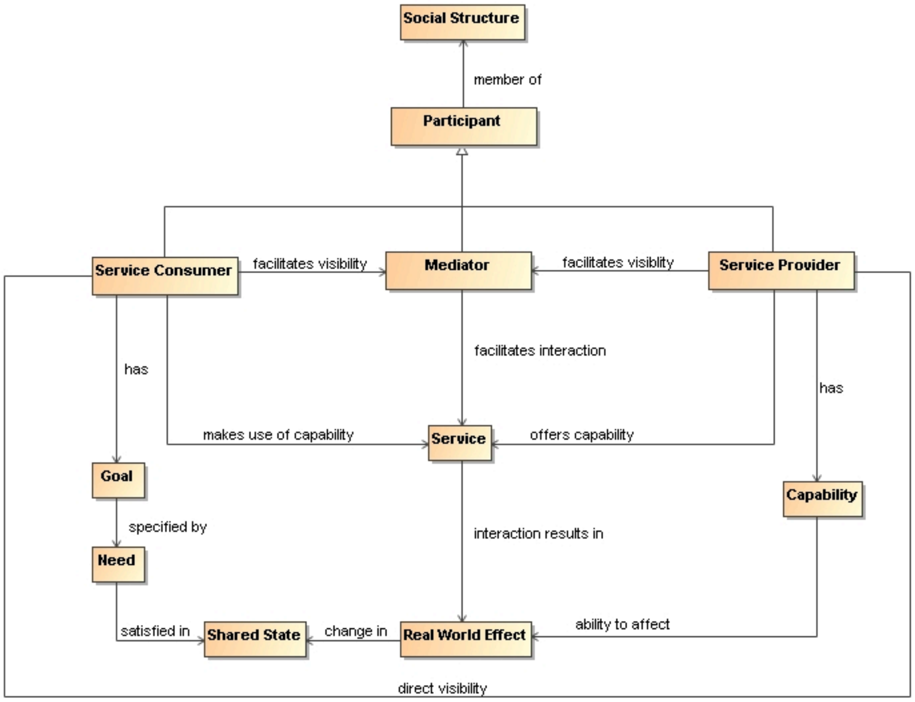


Figure 15 Visibility to Business

The IT mechanisms for SOA visibility can play a much more prominent role in the social context of computing interoperability than previous computing architectures allowed. Visibility and interoperability in a SOA ecosystem requires more than location and interface information, or the traditional Application Programming Interface (API). A meta-model for this broader view of visibility is depicted in the Section 4.1 Service Description Model. In addition to providing improved awareness of service capabilities the Service Description can also be valuable for determination of willingness to interact, references to policies and contracts in the Service Description for example.

Figure 16 provides a distinction between direct visibility and mediated visibility where direct is a one to one relationship between the provider and consumer and mediated is a one to many relationship between providers and consumers.

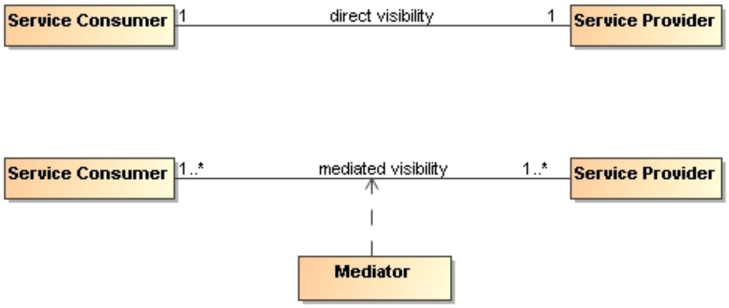


Figure 16 Direct and mediated visibility

Another important business capability in a SOA environment is the ability to narrow visibility to trusted members within a social structure, often referred to as Communities of Interest (COI) in government sectors. SOA standards and vendor products have been developed and continue to be developed to help automate the process of becoming aware, determining willingness, and establishing reachability for service interoperability between complex government and business social structures.

4.2.2 Attaining Visibility

Attaining visibility is described in terms of steps that lead to visibility. While there can be many contexts for visibility within a single social structure, the same general steps can be applied to each of the contexts to accomplish visibility.

Attaining SOA visibility requires

- service description creation and maintenance,
- putting description in a place where others can become aware of it,
- mechanisms for achieving awareness of description,
- mechanisms and/or processes for establishing willingness of participants,
- mechanisms to determine reachability.

4.2.3 Achieving Awareness

A service participant is aware of another participant if it has access to a description of that participant with sufficient completeness to establish the other requirements of visibility.

Awareness is a joint activity between two or more participants. Awareness is a joint activity between two or more participants. Awareness can be decomposed into the creation of description, advertisement of description, and discovery of description. Awareness is often discussed in terms of consumer awareness of providers but the concepts are equally valid for provider awareness of consumers. Prior to awareness in a SOA, a participant creates a description that captures qualifications to advertise to other participants. Discovery in the Service Visibility Model is the act of a consumer discovering a service description or a service provider discovering a likely consumer’s description. Discovery can be initiated or it can be by notification. Initiated discovery for business may require formalization of the required capabilities and resources to achieve business goals. Figure 17 depicts the activities for achieving awareness.

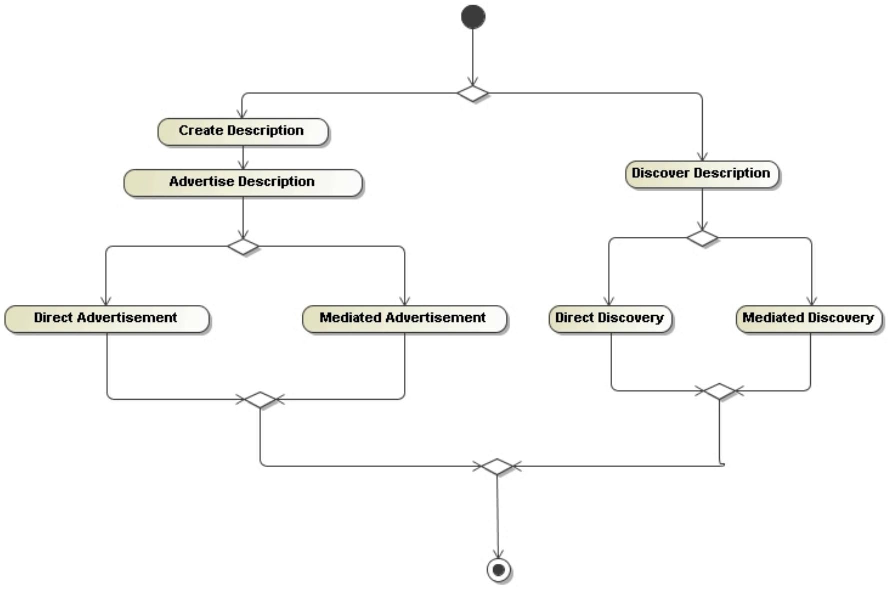


Figure 17 Achieving Awareness

Descriptions may be formal or informal. This SOA RA provides a comprehensive Service Description that can be applied to formal registry/repositories used to mediate visibility. Using consistent description taxonomies and standards based mediated visibility helps provide more effective awareness.

4.2.3.1 Awareness in Complex Social Structures

Joint awareness applies to one or more communities within one or more social structures where a community consists of at least one description provider and one description consumer. In Figure 17, joint awareness can be between a single community, multiple communities, or all communities in the social structure. The social structure can provide governance of awareness where the governance rules translate into policies and/or contracts which can then be incorporated into human processes and automated IT policy/contract mechanisms. The IT policy/contract mechanisms can be used by visibility access mechanisms to provide awareness between communities. The IT mechanisms for awareness may incorporate trust mechanisms to assure awareness between trusted communities. For example, government organizations will often want to limit awareness of an organization's services to specific communities of interest.

Another common business model for awareness is maximizing awareness to communities within the social structure, the traditional market place business model. A centralized mediator often arises as a provider for this global visibility, a gatekeeper of visibility so to speak. For example, Google is a centralized mediator for accessing information on the web. As another example, television networks have centralized entities providing a level of visibility to communities that otherwise could not be achieved without going through the television network.

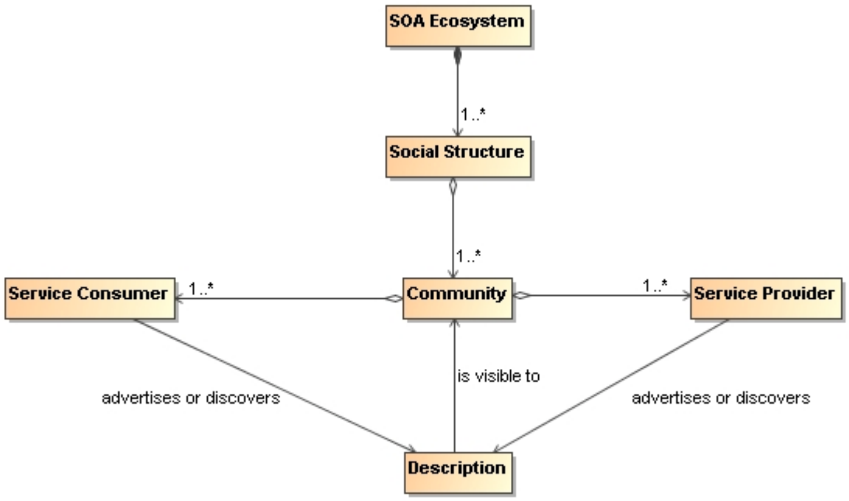


Figure 18 Joint Awareness

4.2.3.2 Establishing Willingness

Having achieved awareness, participants use descriptions to help determine their willingness to interact with another participant. Both awareness and willingness are established prior to consumer/provider interaction. The activities in Figure 19, or a subset there of, can be performed to help establish willingness;

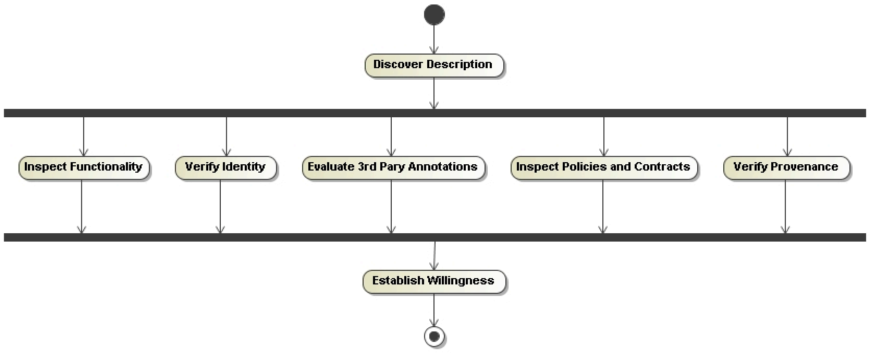


Figure 19 Establishing Willingness

Figure 20 relates elements of the Business via Services View, and elements from the Service Description Model to willingness. By having a willingness to interact within a particular social structure, the social structure provides the participant access to capabilities that help satisfy the participant's goals and objectives as specified by needs. In Figure 20, Information used to establish willingness is defined by Description. Information referenced by Description may come from many sources. For example, a mediator for descriptions may provide 3rd party annotations for reputation. Another source for reputation may be a participant's own history of interactions with another participant.

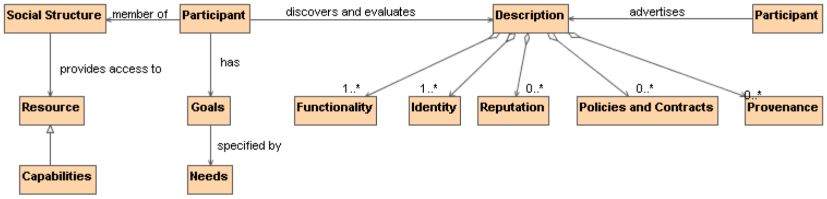


Figure 20 Business, Description and Willingness

Walking through elements referenced by Description, a participant will inspect functionality for potential satisfaction of needs. Identity is associated with any participant, however, identity may or may not be verified. If available, participant reputation may be a deciding factor for unwillingness to interact. Policies and contracts referenced by the description may be particularly important to determine the agreements and commitments required for business interactions. Provenance may be used for verification of authenticity of a resource.

4.2.3.3 Determining Reachability

Reachability involves knowing the service location, service interface, and availability of a service. Figure 21 lists activities involved to determine reachability.

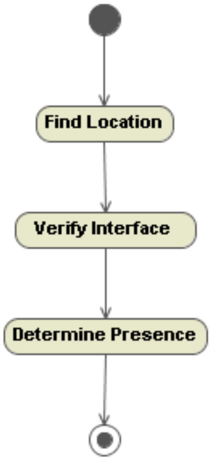


Figure 21 Determining Reachability

Location

A Location is the electronic address to where messages are sent. It is the information needed by a participant’s message delivery mechanism to send a message. Location is referenced from the Service Description in the Service Description Model.

Interface

Interface verification involves determination of compatible communication protocols, compatible message exchange capabilities, and service interface version. Interface is referenced from the Service Description in the Service Description Model.

Presence

Presence is determined when a service can be reached at a particular point in time. Presence may not be known in many cases until the act of interaction begins. To overcome this problem, IT mechanisms may make use of presence protocols to provide the current up/down status of a service.

4.2.3.3.1 Re-establishing Reachability

After reachability has been established, there may be times when participants need to re-establish reachability such as when a service fails and a new location and version for the service needs to be determined. For SOA, both location and version are important for re-establishing reachability. Multiple versions of a service may be in operation for backward compatibility. A Domain Name Service (DNS) lookup for service location may not be sufficient for re-establishing service reachability after a failure.

4.2.4 Mechanisms for Attaining Visibility

Attaining visibility in a SOA can range from word of mouth to formal Service Descriptions in a standards based registry/repository.   Another example of attaining visibility in a SOA is the use of a web page containing service description information. In this case, the web page is the mediator for visibility. To gain the greatest degree of flexibility and interoperability in an ecosystem of services, standards based mediated visibility will most likely be employed for the advertisement and discovery of standards based service descriptions.

4.2.4.1 Mediated Visibility

Mediation promotes loose coupling by keeping the consumers and services from explicitly referring to each other and the descriptions. Mediation lets interaction vary independently. Rather than all potential service consumers being informed on a continual basis about all services, there is a known or agreed upon facility or location that houses the service description.

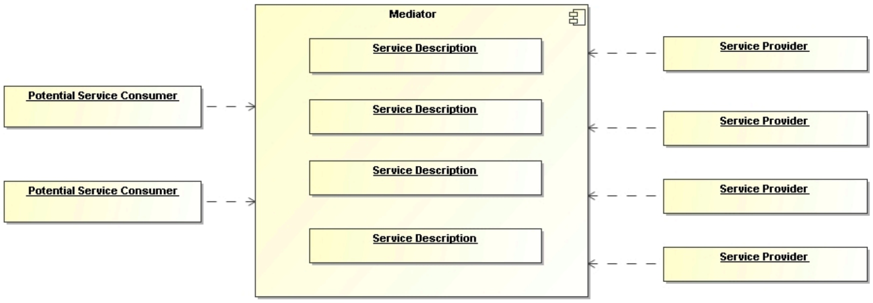


Figure 22 Mediated Service Visibility

In Figure 22, the potential service consumers perform queries or are notified in order to locate those services that satisfy their needs. As an example, the telephone book is a mediated registry where individuals perform manual searches to locate services (i.e. the yellow pages). The telephone book is also a Mediated Registry for solicitors to find and notify potential customers (i.e. the white pages).

In Mediated Service Visibility for large and dynamic numbers of service consumers and service providers, the benefits typically far outweigh the management issues associated with it. Some of the benefits of Mediated Service Visibility are

- Potential service consumers have a known location for searching thereby eliminating needless and random searches
- Typically a consortium of interested parties (or a sufficiently large corporation) signs up to host the mediation facility
- Standardized tools and methods can be developed and promulgated to promote interoperability and ease of use.

However, mediated visibility can have some risks associated with it:

- A single point of failure. If the central visibility service fails then a potentially large number of service providers and consumers will be adversely affected.
- A single point of control. If the central visibility service is owned by, or controlled by, someone other than the service consumers and/or providers then the latter may be put at a competitive disadvantage based on policies of the discovery provider.

Mediation can also apply to service description metadata. There may be a standard vocabulary or mediated interaction among known metadata vocabularies.

4.2.4.2 Service Registries and Repositories

While there can be several mechanisms for service visibility in a SOA, a common mechanism for mediation in the industry is a registry. Figure 23 depicts a mediation facility containing a registry and a repository. The registry stores links or pointers to service description artifacts. The repository in this example is the storage location for the service description artifacts. Service descriptions can be pushed (publish/subscribe for example) or pulled from the register-repository mediator.

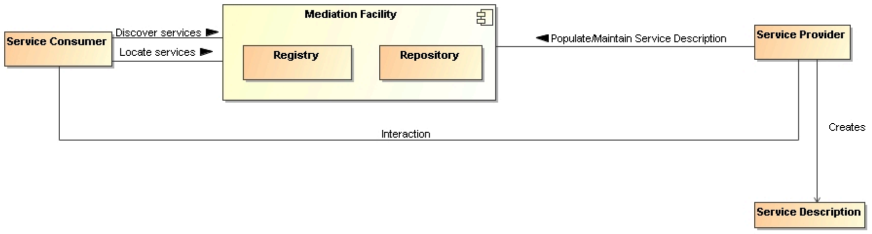


Figure 23 Mediated Registry/Repository

The registry is like a card catalog at the library and a repository is like the shelves for the books. Standardized metadata describing repository content can be stored as registry objects in a registry and any type of content can be stored as repository items in a repository.

4.3 Interacting with Services Model

Interaction is the activity involved in making use of a capability offered in order to achieve a particular desired "'real world effect'", where real world effect is the actual "result" of using a service (as opposed to merely the "capability" offered by a service provider). An activity can be characterized by a sequence of actions. Consequently, interacting with a service involves performing "actions" against the service, usually through a series of information exchanges (e.g., messages), although other modes of interaction are possible such as modifying the shared state of a resource. Note that a participant (or agent acting on behalf of the participant) can be the sender of a message, the receiver of a message, or both.

4.3.1 Action Model

For purposes of this SOA reference architecture, the authors have committed to the use of "message" exchange between service participants to denote actions against the services that "cause" a real world effect, and to denote events that "report" on real world effects that arise from those actions.

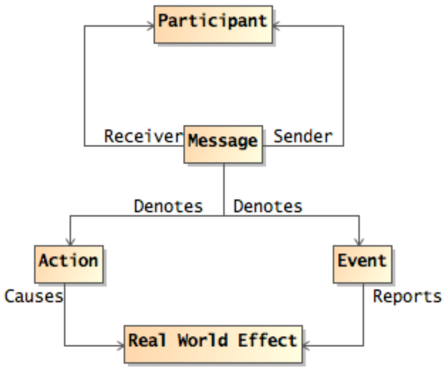


Figure 24 A "message" denotes either an action or an event.

A "message" denotes either an action or an event. In other words, actions and events are realized through messages.

4.3.1.1 Message Exchange

Message exchange is the "means" by which service participants (or their agents) interact with each other. There are two primary modes of interaction: joint actions (see Section 3.4.2) and communicating real world effects.

A message exchange is used to effect an action when the messages contain the appropriately formatted content that should be interpreted as an action and the agents involved interpret the message appropriately (you have to have a speaker and a listener).

A message exchange is also used to communicate event notifications. An event is a report of an occurrence that is of interest to some participant; in our case when some real world effect has occurred. Just as action messages will have formatting requirements, so will event notification messages.

When a message is interpreted as an action, the correct interpretation typically requires the receiver to perform a set of operations. These *operations* represent the sequence of (private) actions a service must perform in order to validly participate in a given joint action.

Similarly, the correct consequence of realizing a real world effect may be to initiate the reporting of that real world effect via an event notification.

Message Exchange

> The means by which joint actions and event notifications are coordinated by service participants (or agents).

Operations

> The sequence of (private) actions a service must perform in order to validly participate in a given joint action.

4.3.1.2 Message Exchange Patterns (MEPs)

As stated earlier, this reference architecture commits to the use of message exchange to denote actions against the services, and to denote events that report on real world effects that arise from those actions.

Because message exchange denotes actions against services, service interaction patterns can be characterized by a common set of message exchange patterns (MEPs):

- Request/response to represent action
- Event notification to represent event

This is by no means a complete list of all possible MEPs but it does represent those that are most commonly used in exchange of information both within organizations and across organizational boundaries. At design time, the MEPs must be specified as part of the action model, which is referenced

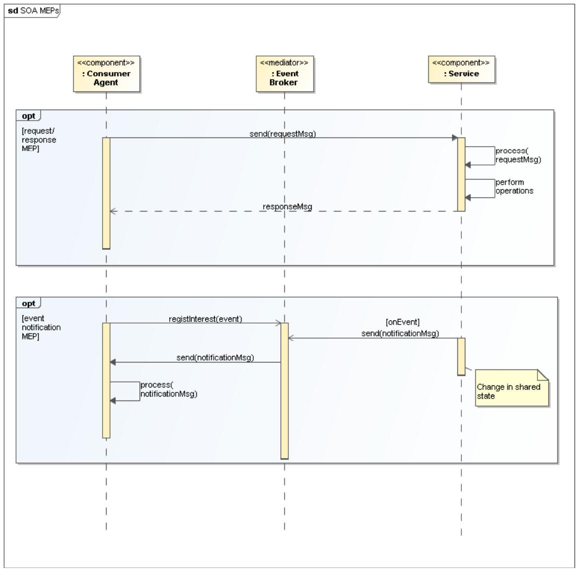


Figure 25 Set of Common SOA Message Exchange Patterns (MEPs)

The interaction (sequence) diagram reflected in Figure 25 shows four different service interaction “options” based on the stated guard condition (e.g., [request/response MEP]). Introduced in UML 2, these fragments are known as "interaction fragments" and the "'opt'" keyword on the interacting fragment frame is known as an "interaction operand". In this case, the interaction operand opt stands for “option.” The way to read these options is by reviewing the guard condition, which is shown in between brackets "[guard]" (e.g., [request/response MEP]), and if the condition is true, then this particular interaction fragment is executed.

In the interaction diagram shown in Figure 25, it is assumed that the service participants (consumer and provider) have delegated message handling to hardware or software agents acting on their behalf. The message interchange model illustrated represents a logical view of the MEPs and not a physical view. In other words, specific hosts, network protocols, and underlying messaging system are not shown as these tend to be implementation specific. While such implementation-specific elements are typically considered outside the scope of reference architecture such as this SOA-RA, they are important considerations in modeling the "'execution context'", a subject to be addressed in additional detail in a subsequent section.

4.3.1.2.1 Request/Response MEP

In a request/response MEP, the Consumer Agent sends a request message to the Provider Agent. The Provider Agent (which is the hardware or software component that actually implements the service) then processes the request message. Based on the content of the message, the Provider Agent performs the service operations. Following the completion of these operations, a response message is returned to the Consumer Agent. This type of MEP is considered a "synchronous interaction" because the sender of the request message (i.e., Consumer Agent) is blocked from continued processing until a response is returned from the Provider Agent.

4.3.1.2.2 Event Notification MEP

An event is realized by means of an event notification message exchange. The basic Event Notification MEP takes the form of a one-way message sent by a notifier agent and received by agents with an interest in the event. Often the sending agent may not be fully aware of all the agents that will receive the notification; particularly in so-called pub/sub situations.

4.3.1.3 Effects of actions

Have to connect the message with the change of state that represents the real world effect of an action.

4.3.2 Process Model

The Process Model could be translated into a machine processable artifact that could be used like a schema for message flow validation of the single service. The Process Model contains the Message Exchange Patterns (MEPS) for the service. The Process Model could also be used by Service Orchestration for the automation of higher order service interactions.

Getting things done sometimes requires performing more than one action.

From WS-CDL :

- "Business Process Languages layer": describes the execution logic of Web Services based applications by defining their control flows (such as conditional, sequential, parallel and exceptional execution) and prescribing the rules for consistently managing their non-observable data

- "Choreography layer": describes collaborations of participants by defining from a global viewpoint their common and complementary observable behavior, where information exchanges occur, when the jointly agreed ordering rules are satisfied.

Beyond the need to specify how to combine services to accomplish some business is the need for the SOA infrastructure to run its own internal processes. An implemented SOA needs a "traffic cop" process/service to

- accept requests,
- decide what needs to be done with the request,
- route the request or derived requests to other services,
- collect the results of other services,
- decide on next actions based on behavior model and responses to routed requests,
- continue this until the initial request is satisfied or terminated,
- package and send results to receiver designated by original requester (where receiver possibly *not* the requester).

The traffic cop will make use of known compositions to make sure routing is done properly and for error recovery. Whatever the traffic cop decides, it will probably be captured internally as a service composition and corresponding behavior model. This is likely a useful format for logging and future audits. It also gives a basis for evaluating levels of service and identifying bottlenecks.

4.3.3 Orchestration and Service Composition

Composition – a service visible to a service consumer via a single interface and described via a single service description is composed of more than one component service, where each component service is visible to the containing service via a single interface and described via a single service description. Figure 1 shows a service A that relies on two other services in its implementation. The service consumer does not know that Services B and C are used, or whether they are used in serial or parallel, or their operations succeed or fail. The consumer only cares about the success or failure of Service A.

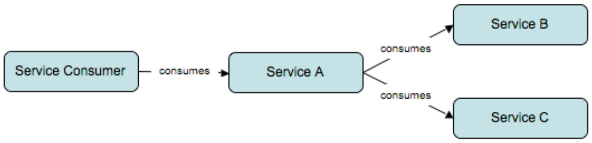


Figure 26 Service Composition

"[need to define business processes and their execution (typically called “orchestration” but the term is not used in the 5 May 2006 WS-BPEL draft) and choreography]"

The combining of services needs to be looked at in the context of not just how you would specify the combination of more atomic services into a higher level one, but also the behavior models and the orchestration/choreography that results in generating real world effects. The behavior models must be consistent (NEEDS TO BE DEFINED) or the combination of services would not make sense.

4.3.4 Dependencies between services

Transitive effects of policies

Preconditions

Configuration dependencies

Not all dependencies may be explicitly listed

Service parameters

What needs to be done about dependencies

2. The Service Interface is made up of the Action Model, i.e. things you can do with a service, and the Process Model, i.e. order you must follow. Here, the actions are only those that result in service’s documented real world effects and not general maintenance/management functions, such as update and delete. As noted in Figure 1, the action model comprises all the available actions internal to the service, however, the Process Model may refer to actions of other services. In such a case, the external services would be listed under Dependencies.

3. The process model implies a maintaining of state. [QUESTION: WHERE IN THE RA WILL WE TALK ABOUT STATE? TO WHAT EXTENT DO WE FOLLOW A WEB MODEL AND PASS ALL STATE INFORMATION AROUND VS. HAVING AN INSTANCE OF THE PROCESS TO WHICH THE CONSUMER NEEDS TO RECONNECT TO MAKE USE OF SAVED STATE?] State will be considered in the model of Interactions and any appropriate references will be added to Service Description.

4. Whereas dependencies are identified at the service level as part of Service Functionality, preconditions may be associated with individual actions. Some preconditions are likely satisfied internal to the service by complying with the process model, but other preconditions may depend on external conditions that will need to be checked before the action can be successfully performed. [QUESTION: IS THE RELATIONSHIP BETWEEN ACTIONS IN THE PROCESS MODEL AND PRECONDITIONS JUST WHETHER INTERNALLY OR EXTERNALLY SATISIFIED? DOES THE PROCESS MODEL AND ACTIONS MADE AVAILABLE TO BE PART OF IT PRIMARILY CONCENTRATE ON WHAT THE SERVICE FINDS SO IMPORTANT THAT IT NEEDS TO BE HANDLED INTERNALLY BUT WITH CONSUMER INTERACTION THROUGH THE VARIOUS ACTIONS?] A Dependency is what has to be

there to do something and a Precondition is exactly what that something had to do or accomplish. For example, a Dependency could be I need access to a service to do currency conversion; the Precondition for another action is that I got a value from that service so I can proceed with my business. Handling of preconditions will also be part of the Interaction model and any appropriate references will be added to Service Description. In response to the question here, Rex noted we will probably need to talk to both internally satisfied and externally satisfied preconditions. But as with policies and RWE, it is unclear if service description needs to have the granularity of specifying this to the action level. Again, look at this from the perspective of discovery: if you specify detail at the action level and you feel this detail is really necessary, how does it reflect at the service level or is it just a surprise you find when you try to use the service? FOR SERVICE DESCRIPTION, IT MAY BE MOST APPROPRIATE TO HAVE ACTION AS A LEAF NODE AND HAVE RWE ATTACH TO THE PROCESS MODEL AS THE SEQUENCE OF WHAT YOU DO TO GET EACH RWE.

5. [likely to require significant massaging, most likely elsewhere] There are numerous complications with the way Web Services are often described. A service that supports more than one independent action or set of actions appears to correspond to a Web Service with multiple operations. We have already said that functionality, policies, and the like are service descriptors and not action descriptors. Thus, we are restricting the concept of WS operations to actions that support the overall function and not different ways of doing the same function. For example, the common WS scenario of different operations providing different qualities of services would be invalid, and these would be different services rather than operations of the same service. However, having the service checks (through a consumer action) to see if the requestor is eligible for enhanced QoS before allowing the action to make use of the enhanced QoS is valid. THIS IS BEATEN TO DEATH IN WHAT ALREADY BEEN ADDED IN THE ITEMS ABOVE. IN SUMMARY, IT NEEDS TO BE CONSISTENT WITH THE POLICY AND RWE LINES TO ACTION OR PROCESS MODEL.

6. [QUESTION: WE HAVE RESOLVED THAT ACTIONS (AND BY EXTENSION, WS OPERATIONS) ARE COVERED BY DESCRIPTION AT THE SERVICE LEVE. HOW DOES AN ACTION NOT BEING AVAILABLE AFFECT THE SERVICE'S OVERALL AVAILABILITY? WHAT DOES THIS INDICATE ABOUT PRESENCE AS INCLUDED UNDER SERVICE REACHABILITY?]

4.4 Policies and Contracts

A policy is the representation of a constraint or condition on the use, deployment, or description of an owned entity as defined by any participant. A contract is a representation of an agreement between two or more participants.

Core aspects of contracts and policies are the constraint assertions, the owners, the measurement and enforcement of the policy or contract. An assertion may be an expression of a policy and/or a contract. Assertions are enforceable and measurable statements about the way a service is realized.

In Section **Error! Reference source not found.**, measurable assertions are characterized as propositions - an expression of some property of the world whose truth can be measured by examining the world and checking that the expression and the world are consistent with each other.

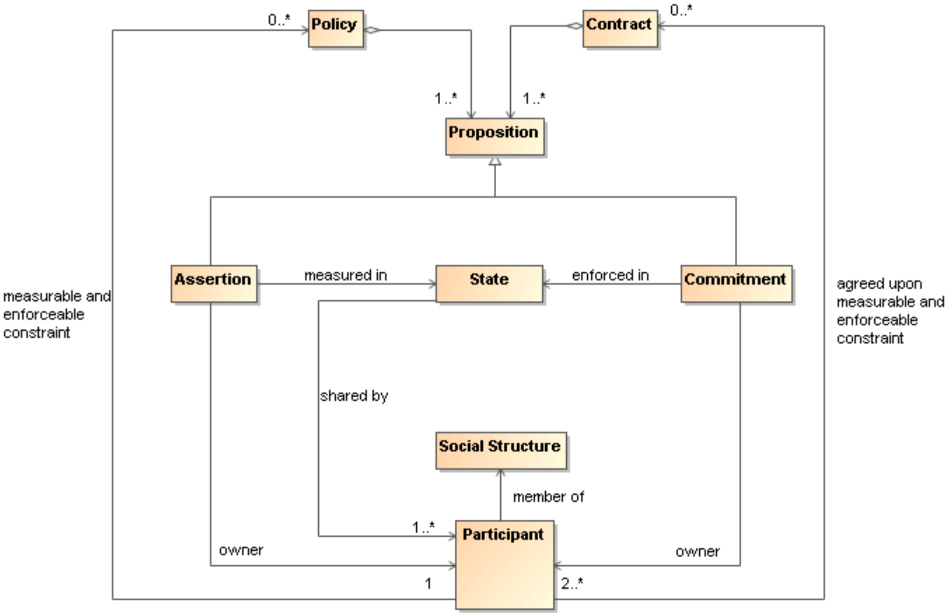


Figure 27 Distinguishing between policies and contracts

Figure 27 derives from the Business via Services View. Policies and contracts are an aggregation of propositions. Both are measurable and enforceable, however contracts are agreed upon by two or more participants while a policy may not have been agreed to.

In a business context, contracts are legally binding agreements between two or more parties. A contract is formed when there is an offer that is duly made and the offer is accepted and there is evidence that indicates there was a tangible exchange of value between the two parties.

The measurability and enforcement of propositions may include many indirectly related participants within the social structure. Dispute resolutions, for example, may involve courts. Policy and contract IT mechanisms support automated governance and management within the SOA ecosystem to improve governance and management efficiency. Advances in IT standards and technologies dictate the level of automation achieved in support of SOA governance and management.

Providing automated mechanisms to enforce policies and contracts can help the social structure operate more efficiently and also enable the social structure to operate at higher levels of abstraction.

Understanding the complete environment which policies and contracts apply in a SOA requires understanding of the processes surrounding policies and contracts in the social structure, the IT mechanisms that support automated enforcement of policies and contracts, and the traversal from/to the social structure to/from the IT policy automation mechanisms.

From the IT perspective, high level policies and contracts are translated into low level rules and measurable properties. For low level rules and measurable properties, both contracts and policies are likely to be enforced by the same type of IT policy mechanisms.

Policies and contracts have wide applicability within the Reference Architecture. They are used to express security policies, service policies, relationships and constraints within the social structures that encapsulate service participants, management of services and many other instances. The enforcement of a policy or contract may be a part of the SOA computing environment or it may be handled outside of the SOA computing environment. The RA is concerned with the underlying principles and IT mechanisms that support enforceable and measurable contracts and policies in the widest range of situations for a SOA.

4.4.1 Policy and Contract Relationships

Figure 4-17 depicts relationships between policies and contracts. A contract may include references to policies and other contracts while a policy may include references to contracts and other policies. For example, a contract may reference a set of policies and a policy may prioritize certain contracts over others.

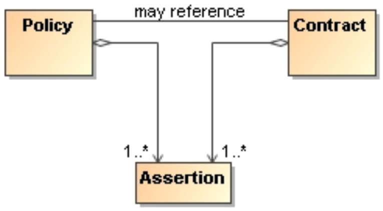


Figure 4-17 Policy/Contract Relationship

Policy

For IT, policies are measurable and enforceable rules/assertions for IT mechanisms that define the choices in behavior of a system.

Contract

Contracts are the set of rules/assertions that define the agreements under which service functionality is delivered.

A policy may result in the application of different choice of behavior at a particular juncture when compared to a contract. However, this could be captured in a process model that makes use of the same policy/contract IT mechanisms. For example, a contract between an Internet Service Provider (ISP) and a consumer specifying maximum bandwidth usage may result in an automatic increase in the consumer's bill if that bandwidth is exceeded. An Internet Service Provider (ISP) policy specifying maximum bandwidth usage may result in the purchase of additional resources by the ISP if monitored bandwidth is exceeded.

4.4.2 Policies and Contracts Life Cycle

The genesis of a policy or contract will likely start with a governance or management process, see section 5.1 Governance of Service Oriented Architectures and section 5.3 Services as Managed Entities Model. The governance and management process may use formal and standardized policy languages and employ the use of common IT mechanisms to maximize computing interoperability and compliance. IT mechanisms for policies and contracts are discussed further in section 4.4.2 IT Mechanisms Supporting Policies and Contracts.

4.4.3 Policy Types

When discussing IT policies, two prevalent policy types are **access control policies** and **event-condition-action policies**. Access control policies are constraints applied to authenticated users. Access control policies ask permission to perform an action. There can be many access control policy models applied in a SOA, the definition of specific access control models is beyond the scope of this reference architecture. Event-condition-action policies specify actions to be taken when certain events occur. Event-condition-action policies are also commonly referred to as obligation policies.

Policies and contracts can contain a mix of permissions and obligations, but policies tend to be permission oriented and contracts tend to be obligation oriented. The mechanisms for enforcing a permission-oriented constraint is prevention at the point of action. The mechanisms for enforcing obligations is assurance of compliance at the point of action. For example, there may be a policy for the types of customer transactions logged. At the point of action for logging customer transactions, a request is made to determine if logging should be done. There may be a contract for the types of customer transactions audited. In this case, auditing requires both logging of the transaction as well as assurance of compliance with the auditing requirements.

4.4.4 Policy and Contract Specification

The language used to describe policies and contracts inevitably constrains the forms and types of policies and contracts expressible in the description. Formal policy language definitions are outside the scope of this specification. For formal policy languages, standard specifications such as XACML and WS-Policy may be referenced. Policy/Contract descriptions may be associated with a service through the Service Description as defined the section 4.1 Service Description Model. Policy enforcement points and policy decision points can interpret policy descriptions expressed in a consumer/provider agreed upon language to make policy decisions.

Regardless of the language used to describe policies and contracts, there are certain aspects that must be captured in any system for representing policies and contracts: how to describe atomic policy constraints, how to handle the composition of policies, how to resolve conflict between policies and how to realize enforcement of policy constraints.

4.4.4.1 Policy Composition

Multiple policies may be defined for one or more services in one or more ownership domains. The application of policies and contracts over distributed services requires the ability to compose one or more policies into an overarching policy. The composition of policies may be implemented as a hierarchy or nesting and/or it can be implemented as intersections and unions of sets.

4.4.4.2 Conflict Resolution

The analysis of policy rules may result in conflicts between the policy rules. There can be many causes for policy conflicts such as conflicting policy rules between ownership domains or policy language specifications that do not convert to first order predicate logic for IT policy mechanisms. This can cause policy decision results to be indeterminate. Policy administration mechanisms may provide conflict resolution capabilities prior to the storage/distribution of policies. At run time, conflicts may resolve to higher authorities inside and outside the SOA IT mechanisms.

4.4.4.3 Delegation of Policy

Policy authorization may be delegated to agents acting on behalf of a client to enable decentralized policy administration and/or policy enforcement. This allows policies to be administered and/or enforced in a hierarchical fashion. Policies may also be transferred to an agent or resource to effectively allow that agent or resource to separate from an ownership domain. The agent or resource may join another ownership domain or rejoin the same ownership domain at a later time.

4.4.5 IT Mechanisms Supporting Policies and Contracts

The common policy architectural elements that are provided in this section are based on the minimal mechanisms required to provide policy guided delivery across distributed services within an ownership domain and across ownership domains. The same mechanisms can provide compliance assurance and/or auditing of contractual obligations between participants.

4.4.5.1 Basic Standards Based Policy and Contract Elements

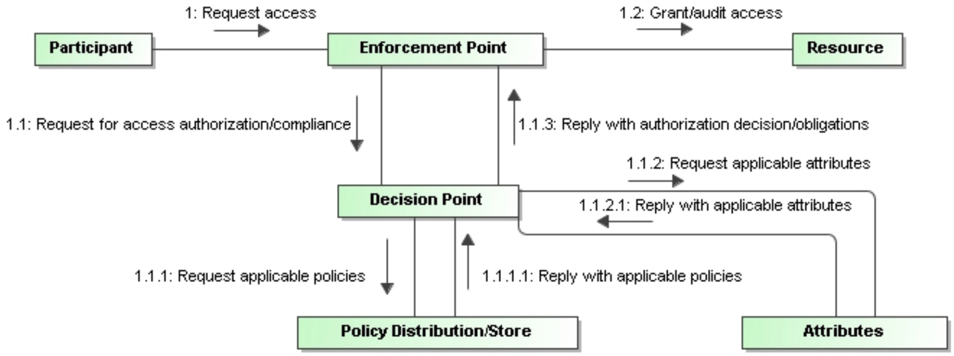

1: Request access
1.2: Grant/audit access
Participant
Enforcement Point
Resource
1.1: Request for access authorization/compliance
1.1.3: Reply with authorization decision/obligations
1.1.2: Request applicable attributes
Decision Point
1.1.2.1: Reply with applicable attributes
1.1.1: Request applicable policies
1.1.1.1: Reply with applicable policies
Policy Distribution/Store
Attributes

Figure 28 Basic Standards-based Policy/Contract IT Elements

Figure 29 Interacting agents in the context of a communications policy

Resource

A resource is any entity that has a name and an owner; see Section 3.3.

Attributes

Attributes are named values that define characteristics of participants, resources, actions, or the environment.

Decision Point

The decision point evaluates participant requests against relevant policies/contracts and attributes to render an authorization decision. The decision point provides a measurement for an assertion. The decision point renders an authorization decision in the form of permit, deny, indeterminate, not applicable, or a set of obligations. A decision point may obtain an authorization decision from a computing mechanism or from outside the computing system, decisions by humans through workflow for example.

Enforcement point

The enforcement point enforces and assures the decision point decisions. In a Service Oriented Architecture, one policy or contract may be applicable to multiple distributed services. Due to the distributed nature of a SOA, the enforcement or auditing of authorizations is attributed to an enforcement point that is separate from the decision point. The enforcement point is responsible for protecting access and determining access compliance to one or more resources. When attempting to access a resource, the enforcement point sends a description of the attempted access to a decision point. The decision point evaluates the request against its available policies/contracts and produces an authorization decision that is returned to the enforcement point. Like the decision point, an enforcement point may require a means of enforcement outside the computing system.

Policy Distribution/Store

The Policy Distribution/Store distributes policy to decision points or stores policies for retrieval by decision points.

4.4.5.2 Policy/Contract Administration

A Policy/Contract administration point can provide many enterprise SOA policy/contract administration capabilities but the end result of the administration point is to store or distribute policy/contract updates.

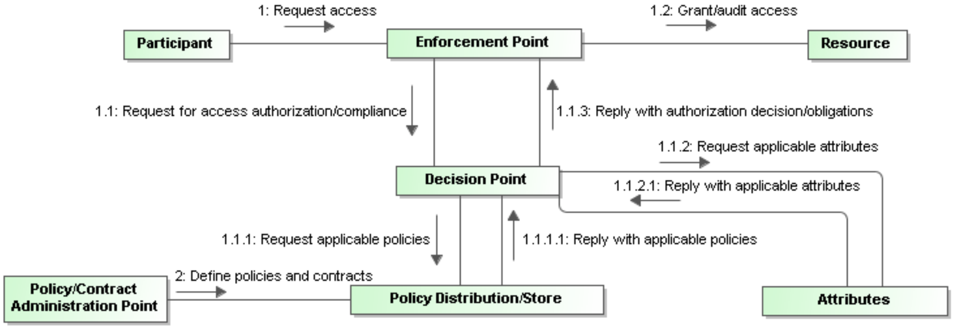

1: Request access
1.2: Grant/audit access
Participant
Enforcement Point
Resource
1.1: Request for access authorization/compliance
1.1.3: Reply with authorization decision/obligations
1.1.2: Request applicable attributes
Decision Point
1.1.2.1: Reply with applicable attributes
1.1.1: Request applicable policies
1.1.1.1: Reply with applicable policies
2: Define policies and contracts
Policy/Contract Administration Point
Policy Distribution/Store
Attributes

Figure 30 Policy/Contract Administration Point

4.4.5.3 Policies/Contracts and Attributes

There are many possible approaches to the management and application of policy/contract attributes. A commonality of the application of attributes is an attribute collection point that collects and forwards attributes to a Decision Point. The SOA RA references this collection point as an Attribute Information Point. Illustration 4 depicts handling resource, environment, and participant attributes.

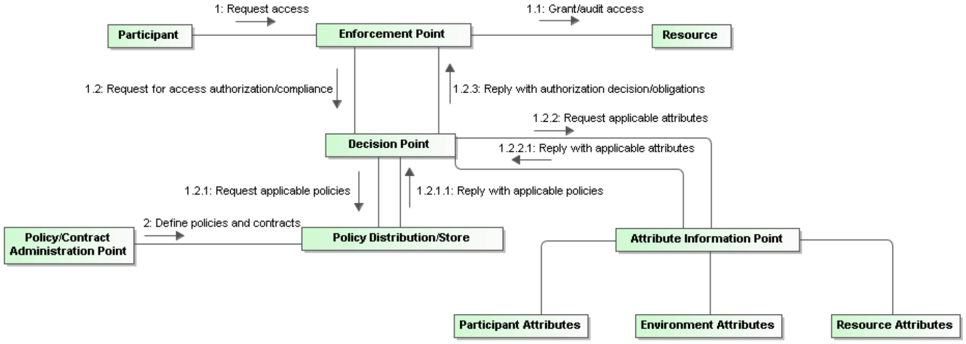

1: Request access
1.1: Grant/audit access
Participant
Enforcement Point
Resource
1.2: Request for access authorization/compliance
1.2.3: Reply with authorization decision/obligations
1.2.2: Request applicable attributes
Decision Point
1.2.2.1: Reply with applicable attributes
1.2.1: Request applicable policies
1.2.1.1: Reply with applicable policies
2: Define policies and contracts
Policy/Contract Administration Point
Policy Distribution/Store
Attribute Information Point
Participant Attributes
Environment Attributes
Resource Attributes

Figure 31 Policies/Contracts and Attributes

5 Owning Service Oriented Architectures View

In the absence of policy-based governance,
organizations will operate as unruly collection of
factions that pull in opposing directions.
Paul A. Strassmann

The Owning Service Oriented Architectures view focuses on what is involved in owning a SOA-based system. [NEED A BIT MORE DESCRIPTIVE TEXT HERE]

The key models in this view are the SOA Governance Model, Security Model, and Services as Managed Entities Model.

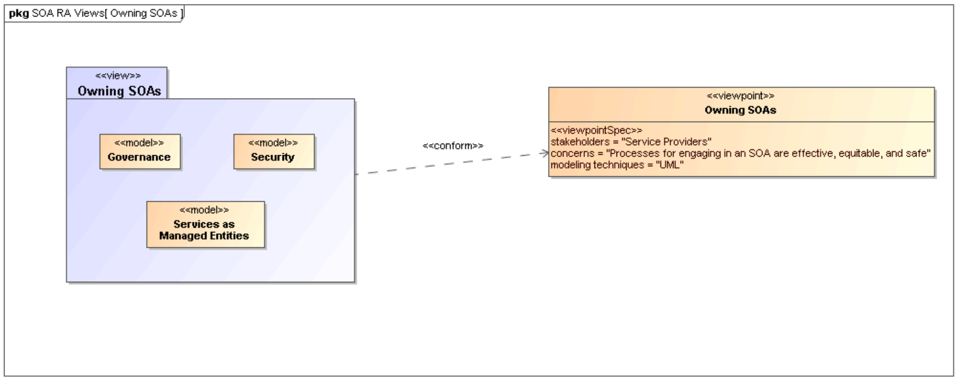

pkg SOA RA Views[Owning SOAs]
<<view>>
Owning SOAs
<<model>>
Governance
<<model>>
Security
<<model>>
Services as Managed Entities
<<conform>>
<<viewpoint>>
Owning SOAs
<<viewpointSpec>>
stakeholders = "Service Providers"
concerns = "Processes for engaging in an SOA are effective, equitable, and safe"
modeling techniques = "UML"

Figure 32 Model elements described in the Owning Service Oriented Architectures view

5.1 Governance of Service Oriented Architectures

Given the importance that people attach to the outcomes of their actions, including those mediated by an SOA, it becomes important to be able to manage the relationship that stakeholders have with their SOA. Governance is fundamentally about how decisions that are pertinent to the adoption, use and evolution of an SOA are arrived at and who has the decision rights to make such decisions. As such, we view the enactment of decisions as being primarily a management concern rather than a governance concern.

5.1.1 Why is explicit Governance important to SOA?

Just as anarchy is also a form of government, so an SOA without an explicit governance structure is also subject to governance. An explicit model for managing the relationship promotes SOA as an equitable platform for people to conduct their work.

One of the hallmarks of SOA as compared to other paradigms for distributed computing is the acknowledgment of the importance of ownership boundaries. Ownership, and issues around it, is one of the primary topics for governance.

No technology platform is static; likewise, an SOA is also subject to change and evolution. Managing that evolution, establishing strategies for change, resolving disputes that arise, and ensuring that the SOA continues to fulfill the goals of the business are all reasons why governance is important to SOA.

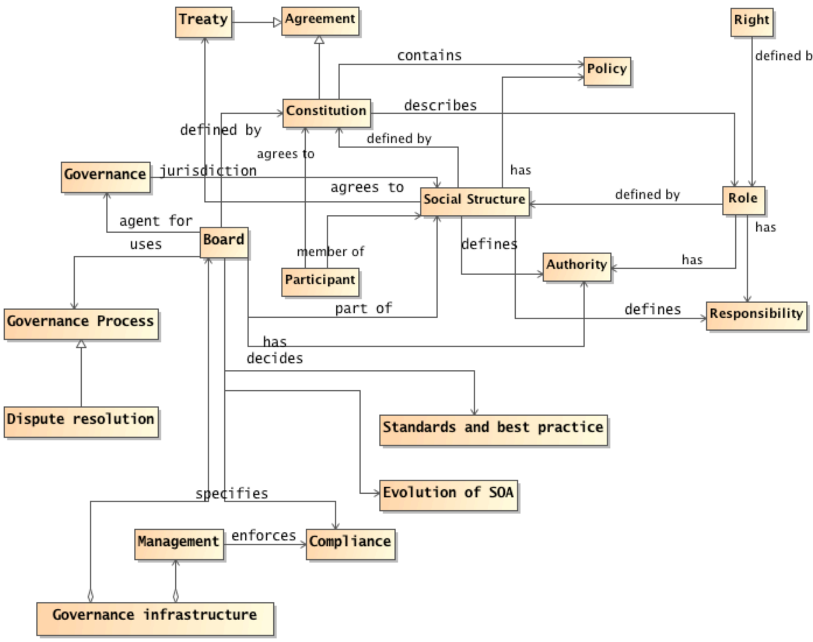


Figure 33 Governance Model

5.1.1.1 Who are the stakeholders in SOA Governance?

As noted earlier, a stakeholder is a human, corporation or non-human agent that has an interest in the states of services and/or the outcomes of service interactions. Stakeholders in SOA governance include the owners of the business who define the high level goals, participants who seek to reach the goals via a SOA and interested third parties, such as regulatory bodies.

5.1.2 What are the concerns of the stakeholders?

The concerns of the stakeholders will be driven by their unique needs and requirements. In particular the mechanisms for managing relationships across ownership boundaries and even within them will tend to differ based on the different types of governance archetypes that are involved.

At a high level, Weill and Ross in “IT Governance” define six governance archetypes:

1. Business Monarchy – Senior business exectives make the decisions
2. IT Monarchy – IT Executives make the decisions
3. Fedudal – Local leadership makes the decsions
4. Federal – Coordinated decisions involving both a central organization and individual units
5. IT Duopoly – IT Executives and one other group
6. Anarchy – Each individual user makes the decision

When applied to SOA, it is more than likely that ownership domains, each of which could well be characterized by a particular archetype, need to interact in order to accomplish the goals that have been established for a SOA.

As such, there is not a one-size-fits-all governance but a need to understand the types of things governance will be called on to do in the context of the goals of SOA. It is likely that some communities will initially desire and require very stringent governance policies and procedures while other will see need for very little. Over time, best practices will evolve, likely resulting in some consensus on a sensible minimum and, except in extreme cases where it is demonstrated to be necessary, a loosening of strict governance toward the best practice mean.

<I was looking over Bob Ellinger's paper on Governance and would be interested to find out how the organizational structure described there maps to one of the archetypes above.. Seemed like the closest fit was the IT Duopoly. My only concern was that that we do need to factor in that all orgs involved in a SOA implementation may not fit that mold>

Ownership boundaries

* Dispute resolution capability

* Ongoing management of environment

* Dealing with issues that go bump in the night

* Providing guidance on the "right" way of doing things

* Who makes the decisions?

* Guidelines vs. Best Practices vs. Laws

* How do you handle exceptions to polices/guidelines?

* What is the feedback loop via which current exceptions may end up becoming future policy/guideline?

* Who tests to make sure that policies are being conformed to?

* Distinguishing between manual vs. automated processes

* How are the contracts between providers and consumers managed?

* More...

5.1.3 Inputs to the decision process

* Policies and Contracts

* Best practices

* Architectural principles

* Government regulations

* Laws

* Organizational rules

5.1.4 Implementing SOA Governance

Establish processes used by stakeholders

Establish specific groups, committees and boards with responsibilities for different aspects of governance

Setting standards

Courts, juries and executioners

Processes for making decisions

Establishing a constitution

The realization of what is considered "right" for participants in a SOA will be as varied as the participants themselves and will be based on their shared expectations as well as factors such as architectural principles, best practices, government regulations, laws, organizational rules. At a high level, it requires the stakeholders to:

- Define stakeholder goals and strategies
- Create the organizational structure with the appropriate the decision rights (The Consitution)
- Formulate polices and decision making processes that are appropriate to the domain
- Define the standards to be applied to the domain
- Define the metrics that need to be collected to ensure that policy enforcement

- Put mechanisms into place that provides for the enforcement of policies and the ability to collect metrics
- Implement feedback and adjudication mechanisms that can adjust the existing policies as needed
- Execute and refine on an ongoing basis

5.1.5 Governance Bone Yard

<I am not sure if we have adequately emphasized the incentive/dis-incentive aspect of goverance in this>

<One of the items that has come up on various conversations is the concept of a Center of Excellence when it comes to SOA. I am leery of associating it with the "Board" as that association has a tendency to imbue it with a significant amount of authority. In certain circumstannces it simply may not have any but may simply be a place for that has influence (based on expertise etc.) and not authority. Not sure if this is something that should be addressed under the governance umbrella>

* Governance must address the entire lifecycle of services

  * Creation

  * Testing

  * Provisioning

  * Utilization

  * Operation

  * Changes/Enhancements

  * Versioning

  * Retirement

* What IT infrastructure is needed to support SOA Governance?

   * Repository

   * Policy Management

   * Metadata Management

   * Contract Management

   * Service Management Systems

   * Service Mediation Systems

5.1.6 Ken's notes on Governance

What does it mean to have governance across ownership boundaries? One aspect is management, and per an earlier email suggestion, a management section has been added to the wiki draft to begin to reflect this separation. In general, governance reflects what some authority wants to happen, e.g. policies, and management provides the details and mechanisms by which policy becomes reality. Much of the management-focused material in the current governance write-up will likely migrate to the management section, including the appropriate parts dealing with life cycle considerations.

In general, governance should be a function of what one wants to accomplish, and thus while SOA should leverage existing structures and best practices, it should not adopt approaches developed for single systems in a single ownership domain if these would significantly inhibit or even preclude the benefits we expect from SOA. Governance for SOA, both development and enforcement, is likely to parallel governance for traditional commerce. This leads to the following conclusions:

1. There will be a range of governance depending on the perceived needs of the participants. In a free market, a dominant mechanism is the satisfaction of the consumer, i.e. if consumers do not find sufficient value in an offering, the product is not used and will either be modified to better serve its intended audience or it will disappear. One can see this with numerous consumer products and with shareware on

the Web. There is little if any governance, and this will likely serve similar situations for SOA where experience with fitness for use is the dominant governance mechanism.

Even with the market, there are situations where market feedback is not considered sufficient in terms of speed, precision, or need to mitigate effects. This is seen where there are health and safety considerations, such as advance approval of new drugs. Alternately, there are intermediate situations, for example the automobile industry, where the market is the dominant governance mechanism but a third party, i.e. some level of government, intervenes where health or safety is an issue. Further discussion on this is included under item 3 below.

The conclusion then is there is not a one-size-fits-all governance but a need to understand the types of things governance will be called on to do in the context of the goals of SOA. It is likely that some communities will initially desire and require very stringent governance policies and procedures while other will see need for very little. Over time, best practices will evolve, likely resulting in some consensus on a sensible minimum and, except in extreme cases where it is demonstrated to be necessary, a loosening of strict governance toward the best practice mean.

2. Whatever level of governance is chosen, it must have effective enforcement, including collection of and access to information needed for enforcement. At a basic level, this requires a relatively free flow of information on consumer experience so prospective consumers can determine whether a given resource available through a SOA implementation provides described functionality and robustness consistent with consumer expectations. Again, the need for enforcement depends on the importance, e.g. life criticality, of the resource. If a resource is free of charge and is generally available to provide some useful but non-critical function, its mere availability may be sufficient and little if any governance or associated enforcement is necessary. For something where there is advance arrangement for the service, e.g. a subscription service, information is likely needed to document availability and form the basis for penalties as prescribed by applicable enforceable policy. If the service is important, reporting may be more critical than penalties because there will be an imperative for understanding and fixing any problem that occurs.

The conclusion then is enforcement is likely dependent on available information (metrics?) and the enforcement mechanism should be consistent with the level of governance perceived as needed. Some aspects of enforcement fall under management.

3. Regulatory governance likely to evolve to reflect perceived needs of stakeholders, including non-participatory stakeholders and regulatory governance of the Commons.

Governance as grounds for mediation of differences between participants.

Governance to codify consensus behavior between participants.

Governance to protect participants.

Governance to protect non-participants from side effects.

Governance to protect the Commons.

SOA provides an interesting example where we are trying to prescribe governance for something that in many cases does not yet exist. We are trying to use past experience to deal with anticipated needs and requirements. From a historical context, governance concepts evolved first to provide grounds for mediation of differences between participants, and later to codify consensus behavior between participants and protect the participants from damaging behavior of one of the parties. The enforcement mechanism could be any agreed upon third party who had was given authority over the participants or later some governmental body that also generated the policy to be enforced. In any case, the enforcement mechanism was answerable to that body's stakeholders. When government is the enforcement mechanism, the stakeholders include not only the immediate participants but other non-participatory stakeholders who may be affected by side effects of the primary interaction. The RM gives examples of real world effects that go beyond the immediate public actions, such as a change in credit rating after getting a loan for a large purchase. In a society, perceived effects on the well being of the general population (the non-participatory stakeholders) often lead to additional policies and enforcement, such as environmental standards. The intent of such governance is to protect and regulate behavior that affects the Commons, the resources under the ownership or protection of society as a whole.

The conclusion then is governance by third parties, whether through government or other agreed upon organization, is likely to develop and have significant effect on the overall governance of SOA.

5.2 Security Model

Providing for security in the context of Service Oriented Architecture is not especially different to other contexts. However, the fact that SOA embraces crossing ownership boundaries and the fact that we aim to explicitly relate the IT architecture with the human architecture (see Business via Services) makes it possible to give a more complete accounting of security.

Any comprehensive security solution must take into account the people that are using, maintaining and managing the SOA. Furthermore, the relationships between them must also be incorporated: any security assertions that may be associated with particular interactions originate in the people that are behind the interaction.

Concepts such as constitutions, roles, and authority within social structures play an important part in the establishment of ownership and trust boundaries within and between social structures. These in turn provide a sound rationale for determining what security policies should be applied and how.

Security is associated with a *threat model* and a *security response model*. The threat model identifies the kinds of threats that are the concern of security specialists, and the response model is the basis for responding to those threats to provide assurance in the safety and integrity of the system.

5.2.1 Security Concepts

Security is one aspect of assurance – the confidence in the integrity and reliability of the system. In particular, security focuses on those aspects of assurance that involve the malign intent of other people. We can characterize security in terms of five key concepts: confidentiality, integrity, authentication, authorization, and non-repudiation.

Confidentiality

Confidentiality concerns the protection of privacy of participants in their interactions. Confidentiality refers to the assurance that unauthorized entities are not able to read messages or documents that are interchanged.

Note that confidentiality has degrees: in a completely confidential exchange, third parties would not even be aware that a confidential exchange has occurred. In a partially confidential exchange, the identities of the participants may be known but the content of the exchange obscured.

Integrity

Integrity concerns the protection of information that is exchanged from corruption. Integrity refers to the assurance that information that has been exchanged has not been tampered with.

Integrity is different from confidentiality in that messages that are sent from one participant to another may be obscured to a third party, but the third party may still be able to introduce his own content into the exchange without the knowledge of the participants.

Authentication

Authentication concerns the identity of the participants in an exchange. Authentication refers to the means by which one participant can be assured of the identity of other participants.

Authorization

Authorization concerns the legitimacy of the interaction. Authorization refers to the means by which one participant may be assured that the information and actions that are exchanged are valid and may be acted on.

Non-repudiation

Non-repudiation concerns the accountability of participants. Non-repudiation refers to the means by which a participant may not, at a later time, successfully deny having participated in the interaction or having performed the actions as reported by other participants.

Availability

Availability concerns the ability of systems to use and offer the services for which they were designed. One of the threats against availability is the so-called denial of service attack in which attackers attempt to prevent legitimate access to the system.

We differentiate here between general availability – which includes aspects such as systems reliability – and availability as a security concept where we need to respond to active threats to the system.

Note that these security concepts are never absolute: it is not possible to guarantee 100% confidentiality, non-repudiation, etc. *However, a well designed and implemented security response model can ensure that the costs of abrogating security are greater than the potential benefits of having done so.* For example, using a well-designed cipher to encrypt messages may make the cost of breaking communications so great and so lengthy that the information obtained is valueless.

While confidentiality and integrity can be viewed as primarily the concerns of the direct participants in an interaction, authentication and authorization and non-repudiation imply the participants are acting within a broader social structure.

5.2.2 Threat Model

There are a number of ways in which an attacker may attempt to compromise the security of a system. The two primary sources of attack are third parties attempting to subvert interactions between legitimate participants and an entity that is participating but attempting to subvert its partner(s). The latter is particularly important in an SOA where there may be multiple ownership boundaries and trust boundaries.

Message alteration

If an attacker is able to modify the content (or even the order) of messages that are exchanged without the legitimate participants being aware of it then the attacker has successfully compromised the security of the system. In effect, the participants may unwittingly serve the needs of the attacker rather than their own.

An attacker may not need to completely replace a message with his own to achieve his objective: replacing the identity of the beneficiary of a transaction may be enough.

Message interception

If an attacker is able to intercept and understand messages exchanged between participants, then the attacker may be able to gain advantage. This is probably the most commonly understood security threat.

Man in the middle

In a man in the middle attack, the legitimate participants believe that they are interacting with each other; but are in fact interacting with the attacker. The attacker attempts to convince each participant that he is their correspondent; whereas in fact he is not.

In a successful man-in-the-middle attack, legitimate participants will often not have a true understanding of the state of the other participants. The attacker can use this to subvert the intentions of the participants.

Spoofing

In a spoofing attack, the attacker convinces a participant that he is really someone else – someone that the participant would normally trust.

Denial of service attack

In a denial of service attack, the attacker attempts to prevent legitimate users from making use of the service. A DoS attack is easy to mount and can cause considerable harm: by preventing legitimate interactions, or by slowing them down enough, the attacker may be able to simultaneously prevent legitimate access to a service and to attack the service by another means.

A variation of the DoS attack is the **Distributed Denial of Service** attack. In a DDoS attack the attacker uses multiple agents to the attack the target. In some circumstances this can be extremely difficult to counteract effectively.

One of the features of a DoS attack is that it does not require valid interactions to be effective: responding to invalid messages also takes resources and that may be sufficient to cripple the target.

Replay attack

In a replay attack, the attacker captures the message traffic during a legitimate interaction and then replays part of it the target. The target is persuaded that a similar transaction to the previous one is being repeated and it will respond as though it were a legitimate interaction.

A replay attack may not require that the attacker understand any of the individual communications; the attacker may have different objectives (for example attempting to predict how the target would react to a particular request).

Repudiation

In a repudiation attack, the attacker completes a normal transaction and then later attempts to deny that the transaction occurred. For example, a customer may use a service to buy a book using a credit card; then, when the book is delivered, refuse to pay the credit card bill claiming that *someone else* must have ordered the book.

5.2.3 Mitigation Model

Responding to security threats in a coherent and consistent way is key to mitigating those threats in a cost-effective way. We identify a few elements of the architecture that may form the basis of a comprehensive security response model for SOA.

We structure the security mitigation model into a number of different elements: an association between policies and security elements, mechanisms intended to support privacy and integrity, mechanisms intended to support authority, mechanisms intended to support obligation-style policies and mechanisms intended to resist DoS attacks.

5.2.3.1 Policies for security

Mechanisms are not the same as solutions; a combination of security mechanisms and their control via explicit policies can form the basis of a solution. Elsewhere in the architecture policies are used to express routing constraints, business constraints and information processing constraints. Security policies are used to marry stakeholders' choices with mechanisms to enforce security.

Security policies are not equivalent to security. However, they are very important as the expression of choices that can be used by security mechanisms to enforce security.

The role of a machine readable security policy is to permit, on the one hand, stakeholders to express their choices; and, on the other hand, to act as instructions for security enforcement mechanisms.

5.2.3.2 Privacy Enforcement

The most efficient mechanism to enforce privacy is the encryption of information. Encryption is particularly important when messages must cross trust boundaries; especially the Internet. Note that encryption need not be limited to the content of messages: it is possible to obscure even the existence of messages themselves through encryption and ‘white noise’ generation in the communications channel.

The specifics of encryption are beyond the scope of this architecture. However, we are concerned about how the connection between privacy-related policies and their enforcement is made. In Section XX, we show how policies in general are enforced using a combination of Policy Decision Points (PDP) and Policy Enforcement Points (PEP).

A PEP for enforcing privacy may take the form of an automatic function to encrypt messages as they leave a trust boundary; or perhaps simply ensuring that such messages are suitably encrypted. If it is important to completely disguise message traffic then some central function to generate encrypted ‘white noise’ when no messages are being transmitted does typically require a centralized facility.

Any policies relating to the level of encryption being used would then apply to these centralized messaging functions.

5.2.3.3 Integrity

To protect against tampering, and to allow the receiver of a message to authenticate the sender, messages may be accompanied by a digital signature. Digital signatures provide a means to detect if signed data has been altered.

A digital signature can be generated with the use of a private key that is associated with a public key and a digital certificate. The private key of some entity in the system is used to create a digital signature for some set of data. Other entities in the system can check the integrity of the signed data set via signature verification algorithms. Any changes to the data that was signed will cause signature verification to fail, which indicates that integrity of the data set has been compromised.

A party verifying a digital signature must have access to the public key that corresponds to the private key used to generate the signature. A digital certificate contains the public key of the owner, and is itself protected by a digital signature created using the private key of the issuing Certificate Authority (CA).

5.2.3.4 Message Replay Protection

To protect against replay attacks, messages may contain information that can be used to detect replayed messages. The simplest requirement to prevent replay attacks is that each message that is ever sent is unique. For example, a message may contain a message ID, a timestamp, the intended destination.

By caching message IDs, and comparing each new message with the cache, it becomes possible to verify whether a given message has been received before (and therefore should be discarded).

The timestamp may be included in the message to help check for message freshness. Messages that arrive after their message ID could have been cleared (after receiving the same message some time previously) may also have been replayed. A common means for representing timestamps is a useful part of an interoperable replay detection mechanism.

The destination information is used to determine if the message was misdirected or replayed. If the replayed message is sent to a different endpoint than the destination of the original message, the replay could go undetected if the message does not contain information about the intended destination.

In the case of messages that are replies to prior messages, it is also possible to include seed information in the prior messages that is randomly and uniquely generated for each message that is sent out. A replay attack can then be detected if the reply does not embed the random number that corresponds to the original message.

5.2.3.5 Trust, Social Structures and Identity

Trust is an assertion as to the behavior of participants in relation to each other. In terms of security assurance, trust often refers to the confidence that target systems may have as to the identity and validity of a participant as they interact with the system. However, in general, trust is a far larger topic.

There are various kinds of trust domain: at the infrastructure level, a trust domain may refer to the networking equipment that is under the control of the owners of a SOA and is used to propagate communication. At an application level, a trust domain may refer to a social structure (see Section 3.4) within which members have previously established a certain degree of trust.

Generally, there are special procedures necessary to communicate across trust domains: for example, participants may need to present credentials to participate in a trust domain. Once authenticated, credentials would typically not be needed to continue within that trust domain.

The connection between policies and trust domains is similar to that for privacy: when a participant wishes to perform an action that requires access to a trust domain, depending on the policies that are in place, he/she must provide suitable credentials to the PEP before continuing the interaction.

In the context of a SOA that is used by many people, there may not be a single repository for information that can justify trust. Often different aspects of trust are managed by different entities. For example, a corporate directory might be used to verify the employment of an individual, whereas a bank would be used to verify their credit worthiness and a government agency used to verify their residency.

Together, the various entities that provide corroboration of an individual's identity and trustworthiness form a *web of trust*. Webs of trust need not be functionally organized: third parties who are known to both may also be used to facilitate trust. Webs of trust have some promise in permitting the efficient scaling of large SOA-based systems. Of course, a complex and long *trust chain* is likely to be more fragile and less trustworthy (sic) than a simple one.

5.2.3.6 Authority, Social Structures and Authorization

The authority held by that participant often determines the validity of actions that a participant engages in. As noted in Section 3.4.2, that authority is always in relation to a particular social structure.

In the context of SOA, the meeting point of action, policy, and validity is often at the service itself. When a participant attempts an action against another participant, the latter may require that the former is properly authorized. (For example, when opening a bank account, it is often required that the customer has appropriate residency status in the bank's country.)

A PEP for enforcing authorization policies would normally be attached to the service that offers the capability. Note that for other reasons, it may not be advisable to *embed* such a PEP within the service capability itself.

The core task of any authorization PEP is to verify that a requested action is valid for the participant; given the identified role that the participant has within the social structure that validates the action.

5.2.3.7 Auditing and logging

A non-repudiation attack involves a participant denying that it authorized a previous interaction. An effective strategy for responding to such a denial is to maintain careful and complete audits of interactions. The more detailed and comprehensive an audit trail is, the less likely it is that a false repudiation would be successful.

Unlike many of the security responses discussed here, it is likely that the scope for automation in rejecting a repudiation attempt is limited to careful logging.

5.2.3.8 Graduated engagement

The key to managing and responding to DoS attacks is to be careful in the use of resources when responding to interaction. Put simply, a system has a choice to respond to a communication or to ignore it. In order to avoid vulnerability to DoS attacks a service provider should not commit to any interaction to a significantly greater extent than the service consumers.

5.3 Services as Managed Entities Model

Managing systems that may be used across ownership boundaries raises issues that may not normally be present in managing a system within a single ownership domain. For example, how is the management of a service to be arranged when the owner of the service, the provider of the service, the host of the service and access mediators to the service may all belong to different stakeholders. Furthermore, how may a service customer communicate his or her requirements to the service provider so that they are satisfied in a timely manner.

In fact, managing a service has quite a few similarities to using a service: suggesting that we can use the service oriented model to manage SOA systems as well as provide them. A management service would be distinguished from a non-management service more by the nature of the capabilities involved (i.e., capabilities that relate to managing services) than by any intrinsic difference.

In this model we show how the SOA framework may apply to managing services as well as using and offering them. There are, of course, some special considerations that apply to service management which we bring out: namely that we will be managing the life-cycle of services, managing any service level attributes, managing dependencies between services and so on.

As in other views of SOA, policies and contracts also are important in managing systems. It has long been known that a systematic management policy framework may be very helpful in managing distributed systems, and SOA is no exception.

Critical SOA Management Topics:

Distributed, independently generated parts that need to interact.

Notions of Mgt:

- management infrastructure (manage things/resources) *current emphasis (more technical)
- manage participant relationships (and other services used to do this) (manage actions/uses) (more social)

Service lifecycle (see telemanagement group as a reference).

Hook into governance (day-to-day activities of gov.)

Governance proposes, management disposes

Service Level Agreements

Policies and contracts

Policy generation Is a governance function, policy implementation is a management function

Services may be managed. In a service oriented management scheme we may use services to manage other services.

A managed service has a number of its aspects that are managed. Furthermore, a management service may offer management of certain aspects of other services. Collectively, these aspects are known as manageability capabilities. (Ed. Note: something of a mouthful).

In a deployed system, it may well be that different aspects of the management of a given service are managed by different management services. For example, the life-cycle management of services often involves managing dependencies between services and resource requirements. Managing quality of service is often very specific to the service itself; for example, quality of service attributes for a video streaming service are quite different to those for a banking system.

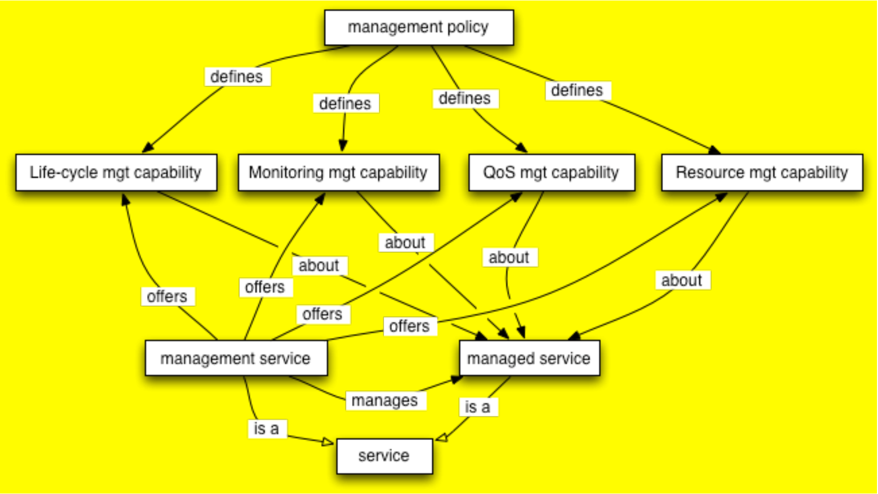


Figure 34 Services and Managed Services

5.3.1 Definitions

Managed service

A service that may be managed

Management service

A service that manages other services

Management Policy

A policy whose topic is a management topic

Manageability capability

An aspect of a service that may be managed by a management service

Lifecycle Manageability

A manageability capability that permits a management service to control the lifecycle of a managed service. Lifecycle management actions include starting services, stopping them, pausing them and so on.

Quality of Service Manageability

A manageability capability that permits a management service to control quality of service aspects of a management service.

Monitoring Manageability

A manageability capability

SOA is by definition a "distributed" paradigm; therefore, from an information technology (IT) perspective, a managed distributed system architecture is needed to fully realize the potential of SOA. Distributed capabilities facilitated by the SOA paradigm may be under different ownership domains. This suggests that there is no theoretical limit as to how widely dispersed—geographically or otherwise—participants in a SOA environment can be so long as there exists a means for service participants to communicate. The prospect of having to support a highly distributed system architecture poses significant challenges from a systems and network management point of view, and requires the introduction of a specialized form of management known as "service management". The bottom line is that a SOA must be managed to be effective.

Systems management

Systems management refers to enterprise-wide maintenance and administration of distributed computer systems.

Network management

Network management refers to the maintenance and administration of large-scale networks such as computer networks and telecommunication networks. Systems and network management execute a set of functions required for controlling, planning, deploying, coordinating, and monitoring the distributed computer systems and the resources of a network.

Service management

Service management, which is the subject of this reference architecture view, refers to the management and administration of service-based resources through a set of activities and capabilities that continuously monitor, control, coordinate, and report on the qualities and usage of these resources. Examples of service qualities include health qualities, or common Qualities of Service (QoS) attributes such as availability and performance, and accessibility. Examples of service usage that may be monitored or controlled include frequency, duration, scope, functional extent, and access authorization. Ultimately, service management is about insuring that acceptable levels of service quality meet the needs of the service consumer.

5.3.2 Management Capabilities

Historically, systems management capabilities have been organized by the following functional groups known as “FCAPS” functions (based on the ITU-T Rec. X.700 | ISO/IEC 7498-4:1989(E) standard):

- Fault Management
- Configuration Management
- Accounting Management
- Performance Management
- Security Management

From a service management perspective, each of these functional groups can be leveraged and defined for purposes of this SOA reference architecture as follows (in concert with ITU-T Rec. X.700 | ISO/IEC 7498-4:1989(E)):

Fault Management

Encompasses fault detection, isolation and the correction of abnormal operation of the SOA environment. Faults cause SOA distributed systems to fail to meet their operational objectives and they may be persistent or transient. Faults manifest themselves as particular events (e.g., errors) in the operation of a distributed system. Error detection provides capabilities to recognize faults. Fault management includes functions to a) maintain and examine error logs, b) accept and act upon error detection notifications, c) trace and identify faults, d) carry out sequences of diagnostic tests, and e) correct faults. For purposes of this reference architecture, monitoring functions such as service status and alerting are included in this functional group.

Accounting Management

Enables charges to be established for the use of resources in the SOA environment, and for costs to be identified for the use of those resources. Accounting management includes functions to a) inform service consumers of costs incurred or resources consumed, b) enable accounting limits to be set and tariff schedules to be associated with the use of resources, and c) enable costs to be combined where multiple resources are invoked to achieve a given objective (resulting in a real-world effect). For purposes of this reference architecture, related accounting functions such as metering and billing fall into this category.

Configuration Management

Identifies, exercises control over, collects data from and provides data to SOA distributed systems for the purpose of preparing for, initializing, starting, providing for the continuous operation of, and terminating services. Configuration management includes functions to a) set the parameters that control the routine operation of the SOA distributed system, b) associate names with managed resources and sets of managed resources, c) initialize and close down managed resources, d) collect information on demand about the current condition of the SOA distributed system, e) obtain announcements of significant changes in the condition of the SOA distributed system, and f) change the configuration of the SOA distributed system. For purposes of this reference architecture, related configuration management functions of service versioning and service provisioning (i.e., supplying of services) is included in this functional category.

Performance Management

Enables the behavior of resources in the SOA environment and the effectiveness of service-oriented activities to be evaluated. Performance management includes functions to a) gather statistical information, b) maintain and examine logs of system state histories, c) determine system performance under natural and artificial conditions, and d) alter system modes of operation for the purpose of conducting performance management activities. Measurements gathered as part of performance management are used to compare against service level agreements (SLAs).

Security Management

Support the application of security policies by means of functions which include a) the creation, deletion and control of security services and mechanisms, b) the distribution of security-related information, and c) the reporting of security-relevant events. A more detailed treatment on the topic of security is provided in the Security View of this SOA reference architecture.

5.3.3 Management Contracts and Policies

5.3.3.1 SLA management (ex of contract), etc.

QoS Attributes:

Quality of Service capabilities will enable graceful degradation, fault tolerance, high reliability, and bounded deterministic behavior. Typical QoS attributes are:

Availability

Probability for service availability including such factors as MTBF (H/W and S/W) and MTTR

Accessibility

Probability of successful service instantiation when required

Scalability

Probability to successfully serve requests independent of load

Integrity

Measurement of interaction correctness with respect to the source versus probabilistic requirement

Performance

Measurement of round trip service request throughput and latency versus requirement

Reliability

The probability of being able to maintain a service at specified service quality

Regulatory

The probability of conforming to rules, standards, service level agreements

5.3.3.2 Policies

"Although provision of management capabilities enables a service to become manageable, the extent and degree of permissible management are defined in management policies that are associated with the services. Management policies are used to define the obligations for, and permissions to, managing the service." [WSA]

Will come back to this...

Relate to policies, i.e., "policies are also intended as a vehicle to express SLAs."

5.3.4 Manageability & Instrumentation

5.3.5 Management Infrastructure

Elements of a basic service management infrastructure should include the following characteristics:

- Integrate with existing security services
- Monitoring
- Heartbeat and Ping
- Alerting
- Pause/Restore/Restart Service Access
- Logging, Auditing, Non-Repudiation
- Runtime Version Management
- Complement other infrastructure services (discovery, messaging, mediation)

 * Message Routing and Redirection
   * Failover
   * Load-balancing

 * QoS, Management of Service Level Objects and Agreements
   * Availability
   * Response Time
   * Throughput

- Fault and Exception Management

5.3.6 Service Life-cycle

5.3.7 Service Provisioning

Requirements on a management system should be to manage the services and not the infrastructure.

6 References

6.1 Normative References

[ANSI/IEEE Std 1471-2000] *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, American National Standards Institute/Institute for Electrical and Electronic Engineers, September 21, 2000.

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[SOA-RM] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, and R. Metz, (editors), “Reference Model for Service Oriented Architecture 1.0, OASIS Open, October 12, 2006.

[WSA] David Booth, et al., "Web Services Architecture", W3C Working Group Note, World Wide Web Consortium (W3C) (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University), February, 2004

6.2 Non-Normative References

[COX] D. E. Cox and H. Kreger, “Management of the service-oriented architecture life cycle,” ''IBM Systems Journal'' '''44''', No. 4, 709-726, 2005

[DEEPAK] Deepak Kakadia, et al., “Enterprise Management Systems Part I: Architectures and Standards,” '''Sun BluePrints™ Online''', Sun Microsystems, Inc., Santa Clara, CA, April, 2002.

[ITU-T Rec. X.700 | ISO/IEC 10746-3:1996(E)] Information processing systems—Open Systems Interconnection—Basic Reference Model—Part 4: Management Framework", International Telecommunication Union, International Organization for Standardization and International Electrotechnical Commission, Geneva, Switzerland, 1989.

[OECD] Organization for Economic Cooperation and Development, Directorate for Financial, Fiscal and Enterprise Affairs, OECD Principles of Corporate Governance, SG/CG(99) 5 and 219, April 1999.

[TOGAF] ***The Open Group Architecture Framework (TOGAF) 8.1 Enterprise Edition*, The Open Group, Doc Number: G051, December 19, 2003.**

[WEILL] Harvard Business School Press, IT Governance: How Top Performers Manage IT Decision Rights for Superior Results, Peter Weill and Jeanne W. Ross, 2004

A. Acknowledgements

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

[Participant Name, Affiliation | Individual Member]
[Participant Name, Affiliation | Individual Member]

B. Critical Factors Analysis

A critical factors analysis (CFA) is an analysis of the key properties of a project. A CFA is analyzed in terms of the goals of the project, the critical factors that will lead to its success and the measurable requirements of the project implementation that support the goals of the project.

B.1 Goals

A goal is an overall target that you are trying to reach with the project. Typically, goals are hard to measure by themselves. Goals are often directed at the potential consumer of the product rather than the technology developer.

B.1.1 Critical Success Factors

A critical success factor (CSF) is a property, sub-goal that directly supports a goal and there is strong belief that without it the goal is unattainable. CSFs themselves are not necessarily measurable in themselves.

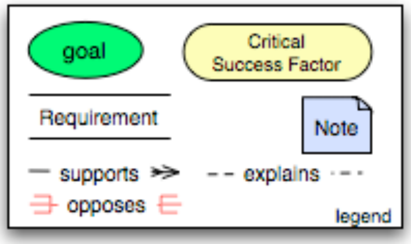
B.1.2 Requirements

A requirement is a specific measurable property that directly supports a CSF. The key here is measurability: it should be possible to unambiguously determine if a requirement has been met. While goals are typically directed at consumers of the specification, requirements are focused on technical aspects of the specification.

B.1.3 CFA Diagrams

It can often be helpful to illustrate graphically the key concepts and relationships between them. Such diagrams can act as effective indices into the written descriptions of goals etc., but is not intended to replace the text.

The legend:



illustrates the key elements of the graphical notation. Goals are written in round ovals, critical success factors are written in round-ended rectangles and requirements are written using open-ended rectangles. The arrows show whether a CSF/goal/requirement is supported by another element or opposed by it. This highlights the potential for conflict in requirements.

C. Revision History

[optional; should not be included in OASIS Standards]

Revision	Date	Editor	Changes Made
Disp	Dis	Disp	DisplayTe