



# Reference Model for Service Oriented Architecture 1.0

## Public Review Draft 1.0, 10 February 2006

**Document identifier:**

wd-soa-rm-cd1

**Location:**

[http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=soa-rm](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm)

**Editors:**

C. Matthew MacKenzie, Adobe Systems Incorporated, [mattm@adobe.com](mailto:mattm@adobe.com)  
Ken Laskey, MITRE Corporation, [klaskey@mitre.org](mailto:klaskey@mitre.org)  
Francis McCabe, Fujitsu Limited, [frank.mccabe@us.fujitsu.com](mailto:frank.mccabe@us.fujitsu.com)  
Peter Brown, [peter@justbrown.net](mailto:peter@justbrown.net)  
Rebekah Metz, Booz Allen Hamilton, [metz\\_rebekah@bah.com](mailto:metz_rebekah@bah.com)

**Abstract:**

This Reference Model for Service Oriented Architecture is an abstract framework for understanding significant entities and relationships between them within a service-oriented environment, and for the development of consistent standards or specifications supporting that environment. It is based on unifying concepts of SOA and may be used by architects developing specific service oriented architectures or in training and explaining SOA. A reference model is not directly tied to any standards, technologies or other concrete implementation details. It does seek to provide a common semantics that can be used unambiguously across and between different implementations.

While service-orientation may be a popular concept found in a broad variety of applications, this reference model focuses on the field of software architecture. The concepts and relationships described may apply to other "service" environments; however, this specification makes no attempt to completely account for use outside of the software domain.

**Status:**

This document is updated periodically on no particular schedule. Send comments to the editor(s).

Committee members should send comments on this specification to the [soa-rm@lists.oasis-open.org](mailto:soa-rm@lists.oasis-open.org) list. Others should visit the SOA-RM TC home page at [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=soa-rm](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm), and record comments using the web form available there.

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 SOA-RM TC web page at:

[http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=soa-rm](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm)

The errata page for this specification is at:

[http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=soa-rm](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm).

---

## Notices

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's procedures with respect to rights in OASIS specifications can be found at 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 specification, can be obtained from the OASIS Executive Director.

OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights, which may cover technology that may be required to implement this specification. Please address the information to the OASIS Executive Director.

Copyright © OASIS Open 2006. *All Rights Reserved.*

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 paragraph are included on all such copies and derivative works. However, this document itself does not be modified in any way, such as by removing the copyright notice or references to OASIS, except as needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights defined in the OASIS Intellectual Property Rights document 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 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

---

# Table of Contents

1	Introduction.....	4
1.1	What is a reference model.....	4
1.2	A Reference Model for Service Oriented Architectures.....	4
1.3	Audience .....	5
1.4	Guide to using the reference model.....	6
1.5	Notational Conventions .....	6
1.6	Relationships to Other Standards .....	6
2	Service Oriented Architecture.....	7
2.1	What is Service Oriented Architecture?.....	7
2.1.1	A worked Service Oriented Architecture example.....	8
2.2	How is Service Oriented Architecture different? .....	9
2.3	The Benefits of Service Oriented Architecture .....	9
3	The Reference Model.....	10
3.1	Service.....	10
3.2	Dynamics of Services.....	11
3.2.1	Visibility .....	11
3.2.2	Interacting with services .....	13
3.2.3	Real World Effect.....	16
3.3	About services .....	17
3.3.1	Service description .....	18
3.3.2	Policies and Contracts.....	20
3.3.3	Execution context.....	22
4	Conformance Guidelines.....	23
5	References .....	24
5.1	Normative .....	24
5.2	Non-Normative.....	24
A.	Glossary .....	25
B.	Acknowledgments .....	28

---

# 1 Introduction

The notion of Service Oriented Architecture (SOA) has received significant attention within the software design and development community. The result of this attention is the proliferation of many conflicting definitions of SOA. Whereas SOA architectural patterns (or *reference architectures*) may be developed to explain and underpin a generic design template supporting a specific SOA, a **reference model** is intended to provide an even higher level of commonality, with definitions that should apply to *all* SOA.

## 1.1 What is a reference model

A reference model is an abstract **framework** for understanding significant relationships among the entities of some environment. It enables the development of specific architectures using consistent standards or specifications supporting that environment. A reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details.

As an illustration of the relationship between a reference model and the architectures that can derive from such a model, consider what might be involved in modeling what is important about residential housing. In the context of a reference model, we know that concepts such as eating areas, hygiene areas and sleeping areas are all important in understanding what goes into a house. There are relationships between these concepts, and constraints on how they are implemented. For example, there may be physical separation between eating areas and hygiene areas.

The role of a **reference architecture** for housing would be to identify abstract solutions to the problems of providing housing. A general pattern for housing, one that addresses the needs of its occupants in the sense of, say, noting that there are bedrooms, kitchens, hallways, and so on is a good basis for an abstract reference architecture. The concept of eating area is a reference model concept, a kitchen is a realization of eating area in the context of the reference architecture.

There may be more than one reference architecture that addresses how to design housing; for example, there may be a reference architecture to address the requirements for developing housing solutions in large apartment complexes, another to address suburban single family houses, and another for space stations. In the context of high density housing, there may not be a separate kitchen but rather a shared cooking space or even a communal kitchen used by many families.

An actual – or concrete – architecture would introduce additional elements. It would incorporate particular architectural styles, particular arrangements of windows, construction materials to be used and so on. A blueprint of a particular house represents an instantiation of an architecture as it applies to a proposed or actually constructed dwelling.

The reference model for housing is, therefore, at least three levels of abstraction away from a physical entity that can be lived in. The purpose of a reference model is to provide a common conceptual framework that can be used consistently across and between different implementations and is of particular use in modeling specific solutions.

## 1.2 A Reference Model for Service Oriented Architectures

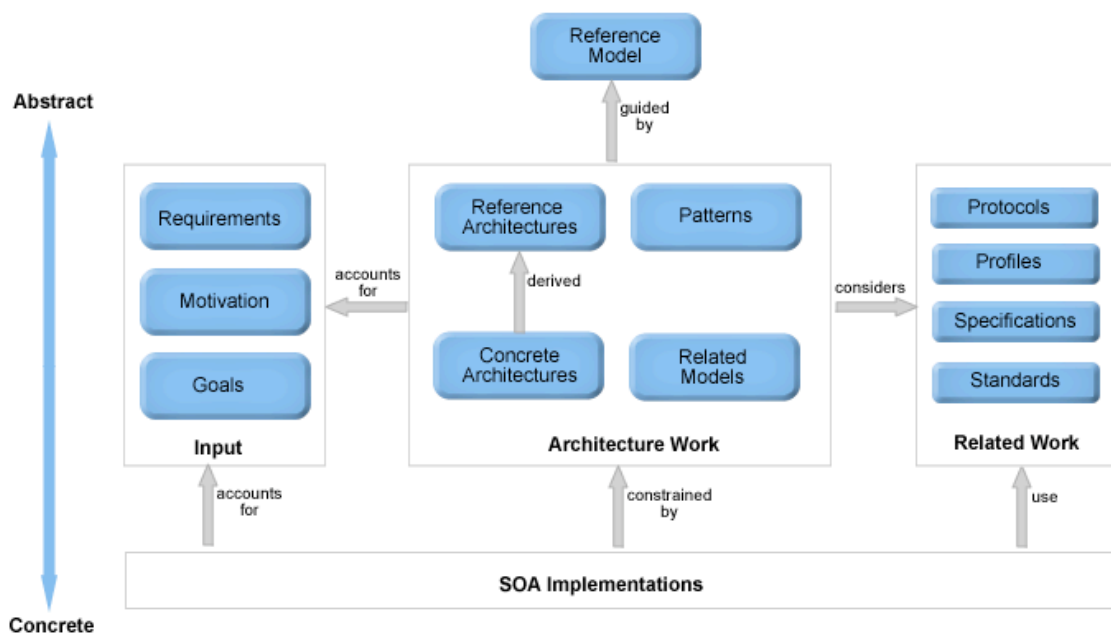
The goal of this reference model is to define the essence of service oriented architecture, and emerge with a vocabulary and a common understanding of SOA. It provides a normative reference that remains relevant for SOA as an abstract and powerful model, irrespective of the various and inevitable technology evolutions that will influence SOA deployment.

47 Figure 1 shows how a reference model for SOA relates to other distributed systems architectural  
 48 inputs. The concepts and relationships defined by the reference model are intended to be the  
 49 basis for describing references architectures and patterns that will define more specific categories  
 50 of SOA designs. Concrete architectures arise from a combination of reference architectures,  
 51 architectural patterns and additional requirements, including those imposed by technology  
 52 environments.

53 Architecture is not done in isolation but must account for the goals, motivation, and requirements  
 54 that define the actual problems being addressed. While reference architectures can form the  
 55 basis of classes of solutions, concrete architectures will define specific solution approaches.

56 Architecture is often developed in the context of a pre-defined environment, such as the  
 57 protocols, profiles, specifications, and standards that are pertinent.

58 SOA implementations combine all of these elements, from the more generic architectural  
 59 principles and infrastructure to the specifics that define the current needs, and represent specific  
 60 implementations that will be built and used in an operational environment.



61  
 62 Figure 1 How the Reference Model relates to other work

### 63 1.3 Audience

64 The intended audiences of this document include non-exhaustively:

- 65 • Architects and developers designing, identifying or developing a system based on the
- 66 service-oriented paradigm.
- 67 • Standards architects and analysts developing specifications that rely on service oriented
- 68 architecture concepts.
- 69 • Decision makers seeking a "consistent and common" understanding of service oriented
- 70 architectures.
- 71 • Users who need a better understanding of the concepts and benefits of service oriented
- 72 architecture.

## 73 1.4 Guide to using the reference model

74 New readers are encouraged to read this reference model in its entirety. Concepts are presented  
75 in an order that the authors hope promote rapid understanding.

76 This section introduces the conventions, defines the audience and sets the stage for the rest of  
77 the document. Non-technical readers are encouraged to read this information as it provides  
78 background material necessary to understand the nature and usage of reference models.

79 Section 2 introduces the concept of SOA and identifies some of the ways that it differs from  
80 previous paradigms for distributed systems. Section 2 offers guidance on the basic principles of  
81 service oriented architecture. This can be used by non-technical readers to gain an explicit  
82 understanding of the core principles of SOA and by architects as guidance for developing specific  
83 service oriented architectures.

84 Section 3 introduces the Reference Model for SOA. In any framework as rich as SOA, it is difficult  
85 to avoid a significant amount of cross referencing between concepts. This makes presentation of  
86 the material subject to a certain amount of arbitrariness. We resolve this by introducing the  
87 concept of service itself, then we introduce concepts that relate to the dynamic aspects of service  
88 and finally we introduce those concepts that refer to the meta-level aspects of services such as  
89 service description and policies as they apply to services.

90 Section 4 addresses compliance with this reference model.

91 The glossary provides definitions of terms that are relied upon within the reference model  
92 specification but do not necessarily form part of the specification itself. Terms that are defined in  
93 the glossary are marked in **bold** at their first occurrence in the document.

94 Note that while the concepts and relationships described in this reference model may apply to  
95 other "service" environments, the definitions and descriptions contained herein focus on the field  
96 of software architecture and make no attempt to completely account for use outside of the  
97 software domain. Examples included in this document that are taken from other domains are  
98 used strictly for illustrative purposes.

## 99 1.5 Notational Conventions

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

103 References are surrounded with **[square brackets and are in bold text]**.

## 104 1.6 Relationships to Other Standards

105 Due to its nature, this reference model may have an implied relationship with any group that:

- 106 • Considers its work "service oriented";
- 107 • Makes (publicly) an adoption statement to use the Reference Model for SOA as a base or  
108 inspiration for their work; and
- 109 • Standards or technologies that claim to be service oriented.

110 The reference model does not endorse any particular service-oriented architecture, or attest to  
111 the validity of third party reference model conformance claims.

---

## 112 2 Service Oriented Architecture

### 113 2.1 What is Service Oriented Architecture?

114 **Service Oriented Architecture** (SOA) is a paradigm for organizing and utilizing distributed  
115 **capabilities** that may be under the control of different ownership domains.

116 In general, entities (people and organizations) create capabilities to solve or support a solution for  
117 the problems they face in the course of their business. It is natural to think of one person's needs  
118 being met by capabilities offered by someone else; or, in the world of distributed computing, one  
119 computer agent's requirements being met by a computer agent belonging to a different owner.

120 There is not necessarily a one-to-one correlation between needs and capabilities; the granularity  
121 of needs and capabilities vary from fundamental to complex, and any given need may require the  
122 combining of numerous capabilities while any single capability may address more than one need.  
123 The perceived value of SOA is that it provides a powerful framework for matching needs and  
124 capabilities and for combining capabilities to address those needs.

125 Visibility, interaction, and effect are key concepts for describing the SOA paradigm. **Visibility**  
126 refers to the capacity for those with needs and those with capabilities to be able to see each  
127 other. This is typically done by providing descriptions for such aspects as functions and technical  
128 requirements, related constraints and policies, and mechanisms for access or response. The  
129 descriptions need to be in a form (or can be transformed to a form) in which their syntax and  
130 **semantics** are widely accessible and understandable.

131 Whereas visibility introduces the possibilities for matching needs to capabilities (and vice versa),  
132 **interaction** is the activity of using a capability. Typically mediated by the exchange of messages,  
133 an interaction proceeds through a series of information exchanges and invoked actions. There  
134 are many facets of interaction; but they are all grounded in a particular **execution context** – the  
135 set of technical and business elements that form a path between those with needs and those with  
136 capabilities. This permits service providers and consumers to interact and provides a decision  
137 point for any policies and contracts that may be in force.

138 The purpose of using a capability is to realize one or more **real world effects**. At its core, an  
139 interaction is “an act” as opposed to “an object” and the result of an interaction is an effect (or a  
140 set/series of effects). We are careful to distinguish between *public* actions and *private* actions;  
141 private actions are inherently unknowable by other parties. On the other hand, public actions  
142 result in changes to the *state* that is shared at least between those involved in the current  
143 execution context and possibly shared by others. Real world effects are, then, couched in terms  
144 of changes to this shared state.

145 The expected real world effects form an important part of the decision on whether a given  
146 capability matches similarly described needs. At the interaction stage, the description of real  
147 world effects establishes the expectations of those using the capability. Note, it is not possible  
148 to describe every effect from using a capability, a cornerstone of SOA is that we can use  
149 capabilities without needing to know all the details.

150 To this point, this description of SOA has yet to mention what is usually considered the central  
151 concept: the **service**. The noun “service” is defined in dictionaries as “The performance of work  
152 (a function) by one for another.” However, service, as the term is generally understood, also  
153 combines the following related ideas:

- 154 • The capability to perform work for another
- 155 • The specification of the work offered for another
- 156 • The offer to perform work for another

157 These concepts emphasize a distinction between a capability and the ability to bring that  
158 capability to bear. While both needs and capabilities exist independently of SOA, **in SOA,**  
159 **services are the mechanism by which needs and capabilities are brought together.**

160 SOA is a means of organizing solutions that promotes reuse, growth and interoperability. It is not  
161 itself a solution to domain problems but rather an organizing and delivery paradigm that enables  
162 one to get more value from use both of capabilities which are locally “owned” and those under the  
163 control of others. It also enables one to express solutions in a way that makes it easier to modify  
164 or evolve the identified solution or to try alternate solutions. SOA does not provide any domain  
165 elements of a solution that do not exist without SOA.

166 The concepts of visibility, interaction, and effect apply directly to services in the same manner as  
167 these were described for the general SOA paradigm. Visibility is promoted through the **service**  
168 **description** which contains the information necessary to interact with the service and describes  
169 this in such terms as the service inputs, outputs, and associated semantics. The service  
170 description also conveys what is accomplished when the service is invoked and the conditions for  
171 using the service.

172 In general, entities (people and organizations) offer capabilities and act as **service providers**.  
173 Those with needs who make use of services are referred to as **service consumers**. The service  
174 description allows prospective consumers to decide if the service is suitable for their current  
175 needs and establishes whether a consumer satisfies any requirements of the service provider.

176 (Note, service providers and service consumers are sometimes referred to jointly as **service**  
177 **participants**.)

178 In most discussions of SOA, the terms “loose coupling” and “coarse-grained” are commonly  
179 applied as SOA concepts, but these terms have intentionally not been used in the current  
180 discussion because they are subjective trade-offs and without useful metrics. In terms of needs  
181 and capabilities, granularity and coarseness are usually relative to detail for the level of the  
182 problem being addressed, e.g. one that is more strategic vs. one down to the algorithm level, and  
183 defining the optimum level is not amenable to counting the number of interfaces or the number or  
184 types of information exchanges connected to an interface.

185 Note that although SOA is commonly implemented using Web services, services can be made  
186 visible, support interaction, and generate effects through other implementation strategies. Web  
187 service-based architectures and technologies are specific and concrete. While the concepts in the  
188 Reference Model apply to such systems, Web Services are too solution specific to be part of a  
189 general reference model.

## 190 **2.1.1 A worked Service Oriented Architecture example**

191 An electric utility has the capacity to generate and distribute electricity (the underlying capability).  
192 The wiring from the electric company’s distribution grid (the service) provides the means to supply  
193 electricity to support typical usage for a residential consumer’s house (service functionality), and  
194 a consumer accesses electricity generated (the output of invoking the service) via a wall outlet  
195 (service interface). In order to use the electricity, a consumer needs to understand what type of  
196 plug to use, what is the voltage of the supply, and possible limits to the load; the utility presumes  
197 that the customer will only connect devices that are compatible with the voltage provided and load  
198 supported; and the consumer in turn assumes that compatible consumer devices can be  
199 connected without damage or harm (service technical assumptions).

200 A residential or business user will need to open an account with the utility in order to use the  
201 supply (service constraint) and the utility will meter usage and expects the consumer to pay for  
202 use at the rate prescribed (service policy). When the consumer and utility agree on constraints  
203 and polices (service contract), the consumer can receive electricity using the service as long as  
204 the electricity distribution grid and house connection remain intact (e.g. a storm knocking down  
205 power lines would disrupt distribution) and the consumer can have payment sent (e.g. a check by  
206 mail or electronic funds transfer) to the utility (reachability).



207 Another person (for example, a visitor to someone else's house) may use a contracted supply  
208 without any relationship with the utility or any requirement to also satisfy the initial service  
209 constraint (i.e. reachability only requires intact electricity distribution) but would nonetheless be  
210 expected to be compatible with the service interface.

211 In certain situations (for example, excessive demand), a utility may limit supply or institute rolling  
212 blackouts (service policy). A consumer might lodge a formal complaint if this occurred frequently  
213 (consumer's implied policy).

214 If the utility required every device to be hardwired to its equipment, the underlying capability  
215 would still be there but this would be a very different service and have a very different service  
216 interface.

## 217 **2.2 How is Service Oriented Architecture different?**

218 Unlike Object Oriented Programming paradigms, where the focus is on packaging data with  
219 operations, the central focus of Service Oriented Architecture is the task or business function –  
220 getting something done. This is a more viable basis for large scale systems because it is a better  
221 fit to the way human activity itself is managed – by delegation.

222 How does this paradigm of SOA differ from other approaches to organizing and understanding  
223 Information Technology assets? Essentially, there are two areas in which it differs both of which  
224 shape the framework of concepts that underlie distributed systems.

225 First, SOA reflects the reality that ownership boundaries are a motivating consideration in the  
226 architecture and design of systems. This recognition is evident in the core concepts of visibility,  
227 interaction and effect. However, SOA does not itself address all the concepts associated with  
228 ownership, ownership domains and actions communicated between legal peers. To fully account  
229 for concepts such as trust, business transactions, authority, delegation and so on – additional  
230 conceptual frameworks and architectural elements are required. Within the context of SOA,  
231 these are likely to be represented and referenced within **service descriptions** and **service**  
232 **interfaces**.

233 Second, SOA applies the lessons learned from commerce to the organization of IT assets to  
234 facilitate the matching of capabilities and needs. That two or more entities come together within  
235 the context of a single interaction implies the exchange of some type of value. This is the same  
236 fundamental basis as trade itself, and suggests that as SOAs evolve away from interactions  
237 defined in a point-to-point manner to a marketplace of services; the technology and concepts can  
238 scale as successfully as the commercial marketplace.

## 239 **2.3 The Benefits of Service Oriented Architecture**

240 The main drivers for SOA-based architectures are to facilitate the manageable growth of large-  
241 scale enterprise systems, to facilitate Internet-scale provisioning and use of services and to  
242 reduce costs in organization to organization cooperation.

243 The value of SOA is that it provides a simple scalable paradigm for organizing large networks of  
244 systems that require interoperability to realize the value inherent in the individual components.  
245 Indeed, SOA is scalable because it makes the fewest possible assumptions about the network  
246 and also minimizes any trust assumptions that are often implicitly made in smaller scale systems.

247 An architect using SOA principles is better equipped, therefore, to develop systems that are  
248 scalable, evolvable and manageable. It should be easier to decide how to integrate functionality  
249 across ownership boundaries. For example, a large company that acquires a smaller company  
250 must determine how to integrate the acquired IT infrastructure into its overall IT portfolio.

251 Through this inherent ability to scale and evolve, SOA enables an IT portfolio which is also  
252 adaptable to the varied needs of a specific problem domain or process architecture. The  
253 infrastructure SOA encourages is also more agile and responsive than one built on an  
254 exponential number of pair-wise interfaces. Therefore, SOA can also provide a solid foundation  
255 for business agility and adaptability.

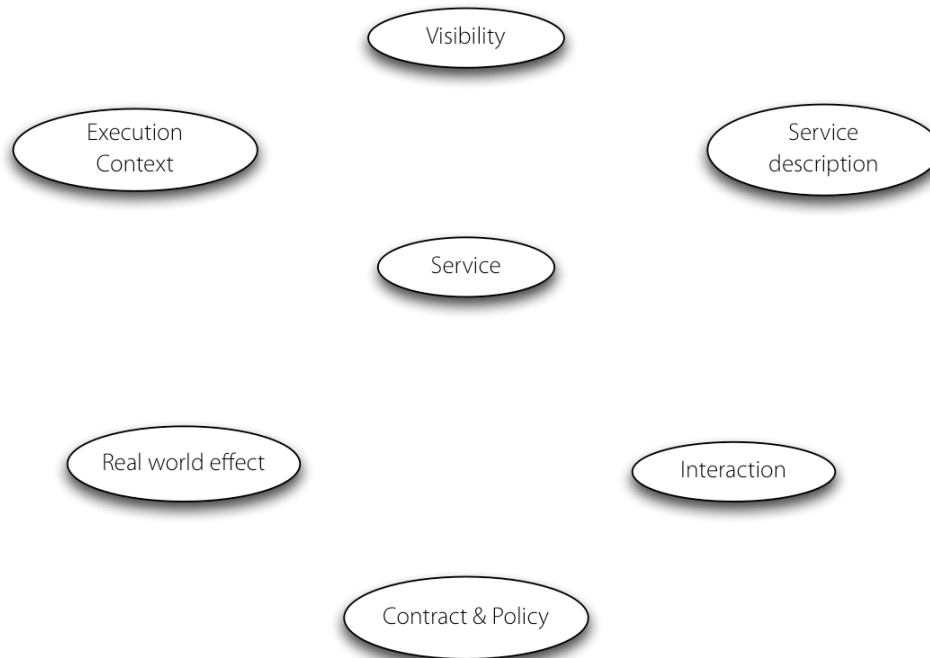
256

## 3 The Reference Model

257

Figure 2 illustrates the principal concepts this reference model defines. The relationships between them are developed as each concept is defined in turn.

258



259

260

Figure 2 Principal concepts in the Reference Model

261

### 3.1 Service

262

A **service** is a mechanism to enable access to a set of one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description. A service is provided by one entity – the **service provider** – for use by others, but the eventual consumers of the service may not be known to the service provider and may demonstrate uses of the service beyond the scope originally conceived by the provider.

268

A service is accessed by means of a service interface (see Section 3.3.1.4), where the interface comprises the specifics of how to access the underlying capabilities. There are no constraints on what constitutes the underlying capability or how access is implemented by the service provider. Thus, the service could carry out its described functionality through one or more automated and/or manual processes that themselves could invoke other available services.

273

A service is opaque in that its implementation is typically hidden from the **service consumer** except for (1) the information and behavior models exposed through the service interface and (2) the information required by service consumers to determine whether a given service is appropriate for their needs.

277

The consequence of invoking a service is a realization of one or more real world effects (see Section 3.2.3). These effects may include:

279

280

1. information returned in response to a request for that information,

281

2. a change to the shared state of defined entities, or

282 3. some combination of (1) and (2).

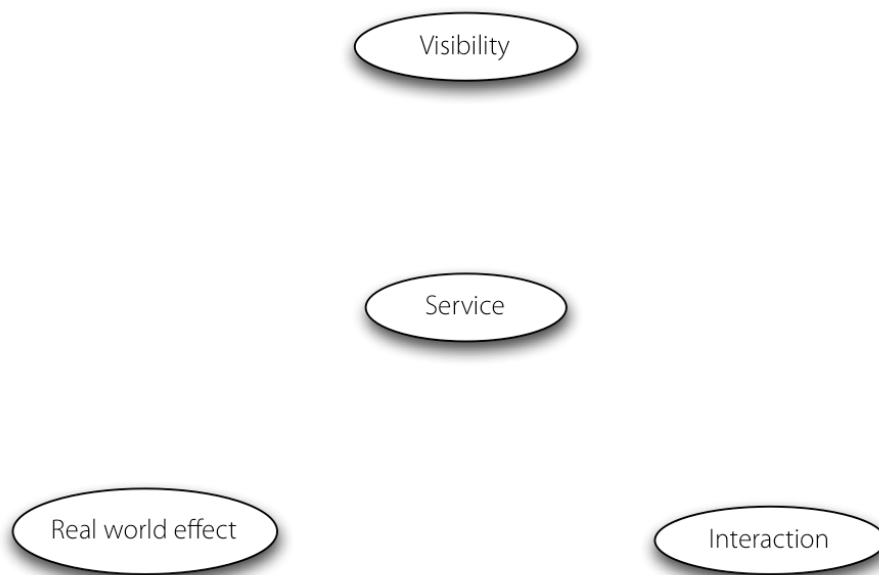
283

284 Note, the service consumer in (1) does not typically know how the information is generated, e.g.  
285 whether it is extracted from a database or generated dynamically; in (2), it does not typically know  
286 how the state change is effected.

287 The service concept above emphasizes a distinction between a capability that represents some  
288 functionality created to address a need and the point of access to bring that capability to bear in  
289 the context of SOA. It is assumed that capabilities exist outside of SOA. In actual use,  
290 maintaining this distinction may not be critical (i.e. the service may be talked about in terms of  
291 being the capability) but the separation is pertinent in terms of a clear expression of the nature of  
292 SOA and the value it provides.

## 293 3.2 Dynamics of Services

294 From a dynamic perspective, there are three fundamental concepts that are important in  
295 understanding what is involved in interacting with services: the visibility between service providers  
296 and consumers, the interaction between them, and the real world effect of interacting with a  
297 service.

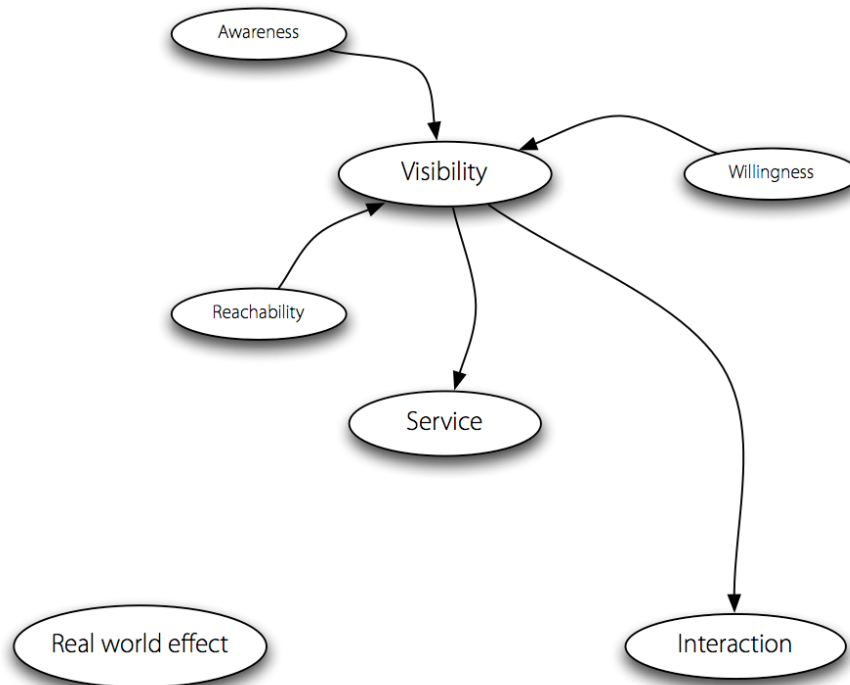


298

299 Figure 3 Concepts around the dynamics of service

### 300 3.2.1 Visibility

301 For a service provider and consumer to interact with each other they have to be able to 'see' each  
302 other. This is, in fact, true for any consumer/provider relationship – including in an application  
303 program where one program calls another: without the proper libraries being present the function  
304 call cannot complete. In the case of SOA, visibility needs to be emphasized because it is not  
305 necessarily obvious how service participants *can* see each other.



306

307 Figure 4 Concepts around Visibility

308 Visibility is the relationship between service consumers and providers that is satisfied when they  
 309 are able to interact with each other. Preconditions to visibility are **awareness**, **willingness** and  
 310 **reachability**. The initiator in a service interaction **MUST** be aware of the other parties, the  
 311 participants **MUST** be predisposed to interaction, and the participants **MUST** be able to interact.

### 312 3.2.1.1 Awareness

313 Both the service provider and the service consumer **MUST** have information that would lead them  
 314 to know of the other's existence. Technically, the prime requirement is that the *initiator* of a  
 315 service interaction has knowledge of the responder. The fact of a successful initiation is often  
 316 sufficient to inform the responder of the other's existence.

317 Awareness is a state whereby one party has knowledge of the existence of the other party.  
 318 Awareness does not imply willingness or reachability. Awareness of service offerings is often  
 319 effected by various *discovery* mechanisms. For a service consumer to discover a service, the  
 320 service provider must be capable of making details of the service (notably service description and  
 321 policies) available to potential consumers; and consumers must be capable of becoming aware of  
 322 that information. Conversely, the service provider may want to discover likely consumers and  
 323 would need to become aware of the consumer's description. In the following, we will discuss  
 324 awareness in terms of service visibility but the concepts are equally valid for consumer visibility.

325 Service awareness requires that the **service description** and **policy** – or at least a suitable  
 326 subset thereof – be available in such a manner and form that, directly or indirectly, a potential  
 327 consumer is aware of the existence and capabilities of the service. The extent to which the  
 328 description is “pushed” by the service provider, “pulled” by a potential consumer, subject to a  
 329 probe or another method, will depend on many factors.

330 For example, a service provider may advertise and promote their service by either including it in a  
 331 service directory or broadcasting it to all consumers; potential consumers may broadcast their  
 332 particular service needs in the hope that a suitable service responds with a proposal or **offer**, or a  
 333 service consumer might also probe an entire network to determine if suitable services exist.  
 334 When the demand for a service is higher than the supply, then, by advertising their needs,

335 potential consumers are likely to be more effective than service providers advertising offered  
336 services.

337 One way or another, the potential consumer must acquire sufficient descriptions to evaluate  
338 whether a given service matches its needs and, if so, the method for the consumer to interact  
339 with the service.

### 340 **3.2.1.2 Willingness**

341 Associated with all service interactions is intent – it is an intentional act to initiate and to  
342 participate in a service interaction. For example, if a service consumer discovers a service via its  
343 description in a registry, and the consumer initiates an interaction, if the service provider does not  
344 cooperate then there can be no interaction. In some circumstances it is precisely the correct  
345 behavior for a service to fail to respond – for example, it is the classic defense against certain  
346 denial-of-service attacks.

347 The extent of a service participant's willingness to engage in service interactions may be the  
348 subject of policies. Those policies may be documented in the service description.

349 Of course, willingness on the part of service providers and consumers to interact is not the same  
350 as a willingness to perform requested actions. A service provider that rejects all attempts to cause  
351 it to perform some action may still be fully willing and engaged in interacting with the consumer.

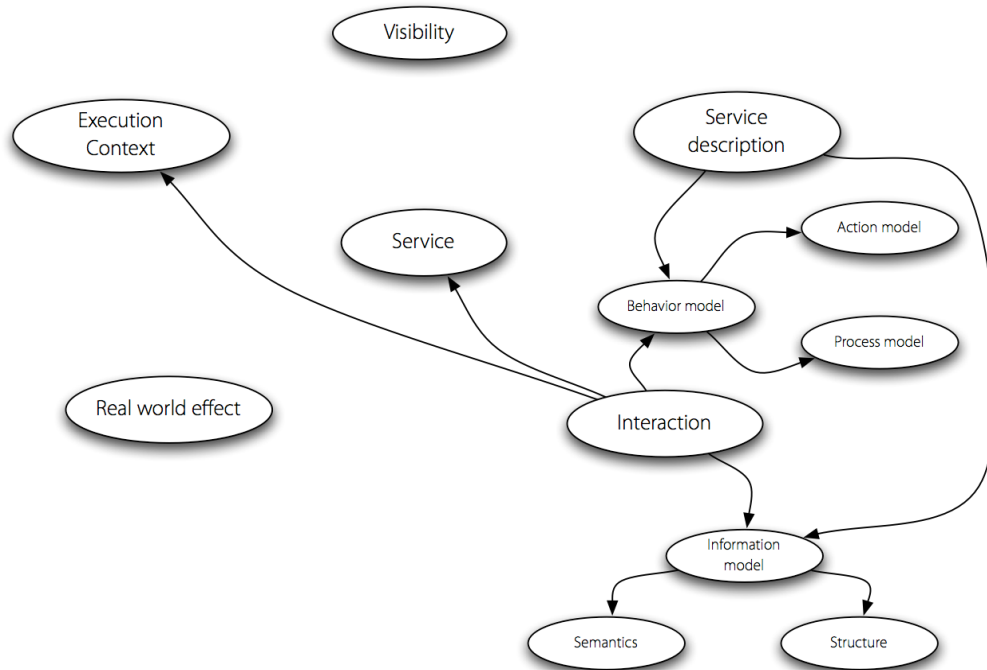
### 352 **3.2.1.3 Reachability**

353 Reachability is the relationship between service participants where they are able to interact;  
354 possibly by exchanging information. Reachability is an essential pre-requisite for service  
355 interaction – participants **MUST** be able to communicate with each other.

356 A service consumer may have the intention of interacting with a service, and may even have all  
357 the information needed to communicate with it. However, if the service is not reachable, for  
358 example if there is not a communication path between the consumer and provider, then,  
359 effectively, the service is not visible to the consumer.

### 360 **3.2.2 Interacting with services**

361 Interacting with a service involves performing actions against the service. In many cases, this is  
362 accomplished by sending and receiving messages, but there are other modes possible that do  
363 not involve explicit message transmission. For example, a service interaction may be effected by  
364 modifying the state of a shared resource. However, for simplicity, we often refer to message  
365 exchange as the primary mode of interaction with a service.



366

367 Figure 5 Service Interaction concepts

368 Figure 5 illustrates the key concepts that are important in understanding what it is involved in  
 369 interacting with services; these revolve around the service description – which references a  
 370 **information model** and a **behavior model**.

### 371 3.2.2.1 Information model

372 The information model of a service is a characterization of the information that may be exchanged  
 373 with the service. Only information and data that are potentially exchanged with a service are  
 374 generally included within that service's information model.

375 The scope of the information model includes the format of information that is exchanged, the  
 376 structural relationships within the exchanged information and also the definition of terms used.

377 Particularly for information that is exchanged across an ownership boundary, an important aspect  
 378 of the service information model is the consistent interpretation of strings and other tokens in the  
 379 information.

380 The extent to which one system can effectively interpret information from another system is  
 381 governed by the **semantic engagement** of the various systems. The semantic engagement of a  
 382 system is a relationship between the system and information it may encounter. This is highly  
 383 variable and application dependent; for example an encryption service interprets all information  
 384 as a stream of bytes for it to encrypt or decrypt, whereas a database service would attempt to  
 385 interpret the same information stream in terms of requests to query and/or modify the database.

386 Loosely, one might partition the interpretation of an informational block into structure (syntax) and  
 387 meaning (semantics); although both are part of the information model.

#### 388 3.2.2.1.1 Structure

389 Knowing the representation, structure, and form of information required is a key initial step in  
 390 ensuring effective interactions with a service. There are several levels of such structural  
 391 information; including the encoding of character data, the format of the data and the structural  
 392 data types associated with elements of the information.

393 A described information model typically has a great deal to say about the form of messages.  
394 However, knowing the type of information is not sufficient to completely describe the appropriate  
395 interpretation of data. For example, within a street address structure, the city name and the street  
396 name are typically given the same data type – some variant of the string type. However, city  
397 names and street names are not really the same type of thing at all. Distinguishing the correct  
398 interpretation of a city name string and a street name string is not possible using type-based  
399 techniques – it requires additional information that cannot be expressed purely in terms of the  
400 structure of data.

### 401 **3.2.2.1.2 Semantics**

402 The primary task of any communication infrastructure is to facilitate the exchange of information  
403 and the exchange of intent. For example, a purchase order combines two somewhat orthogonal  
404 aspects: the description of the items being purchased and the fact that one party intends to  
405 purchase those items from another party. Even for exchanges that do not cross any ownership  
406 boundaries, exchanges with services have similar aspects.

407 Especially in the case where the exchanges are across ownership boundaries, a critical issue is  
408 the interpretation of the data. This interpretation **MUST** be consistent between the participants in  
409 the service interaction. Consistent interpretation is a stronger requirement than merely type (or  
410 structural) consistency – the tokens in the data itself must also have a shared basis.

411 There is often a huge potential for variability in representing street addresses. For example, an  
412 address in San Francisco, California may have variations in the way the city is represented: SF,  
413 San Francisco, San Fran, the City by the Bay are all alternate denotations of the same city. For  
414 successful exchange of address information, all the participants must have a consistent view of  
415 the meaning of the address tokens if address information is to be reliably shared.

416 The formal descriptions of terms and the relationships between them (e.g., an ontology) provides  
417 a firm basis for selecting correct interpretations for elements of information exchanged. For  
418 example, an ontology can be used to capture the alternate ways of expressing the name of a city  
419 as well as distinguishing a city name from a street name.

420 Note that, for the most part, it is not expected that service consumers and providers would  
421 actually exchange descriptions of terms in their interaction but, rather, would reference existing  
422 descriptions – the role of the semantics being a background one – and these references would be  
423 included in the service descriptions.

424 Specific domain semantics are beyond the scope of this reference model; but there is a  
425 requirement that the service interface enable providers and consumers to identify unambiguously  
426 those definitions that are relevant to their respective domains.

### 427 **3.2.2.2 Behavior model**

428 The second key requirement for successful interactions with services is knowledge of the actions  
429 invoked against the service and the process or temporal aspects of interacting with the service.  
430 This is characterized as knowledge of the actions on, responses to, and temporal dependencies  
431 between actions on the service.

432 For example, in a security-controlled access to a database, the actions available to a service  
433 consumer include presenting credentials, requesting database updates and reading results of  
434 queries. The security may be based on a challenge-response protocol. For example, the initiator  
435 presents an initial token of identity, the responder presents a challenge and the initiator responds  
436 to the challenge in a way that satisfies the database. Only after the user's credentials have been  
437 verified will the actions that relate to database update and query be accepted.

438 The sequences of actions involved are a critical aspect of the knowledge required for successful  
439 use of the secured database.

#### 440 3.2.2.2.1 Action model

441 The **action model** of a service is the characterization of the actions that may be invoked against  
442 the service. Of course, a great portion of the behavior resulting from an action may be private;  
443 however, the expected public view of a service surely includes the implied effects of actions.

444 For example, in a service managing a bank account, it is not sufficient to know that you need to  
445 exchange a given message (with appropriate authentication tokens), in order to use the service. It  
446 is also necessary to understand that using the service may actually affect the state of the account  
447 (for example, withdrawing cash); that dependencies are involved (for example, a withdrawal  
448 request must be less than the account balance); or that the data changes made have different  
449 value in different contexts (for example, changing the data in a bank statement is not the same as  
450 changing the actual data representing the amount in an account).

#### 451 3.2.2.2.2 Process Model

452 The **process model** characterizes the temporal relationships between and temporal properties of  
453 actions and events associated with interacting with the service.

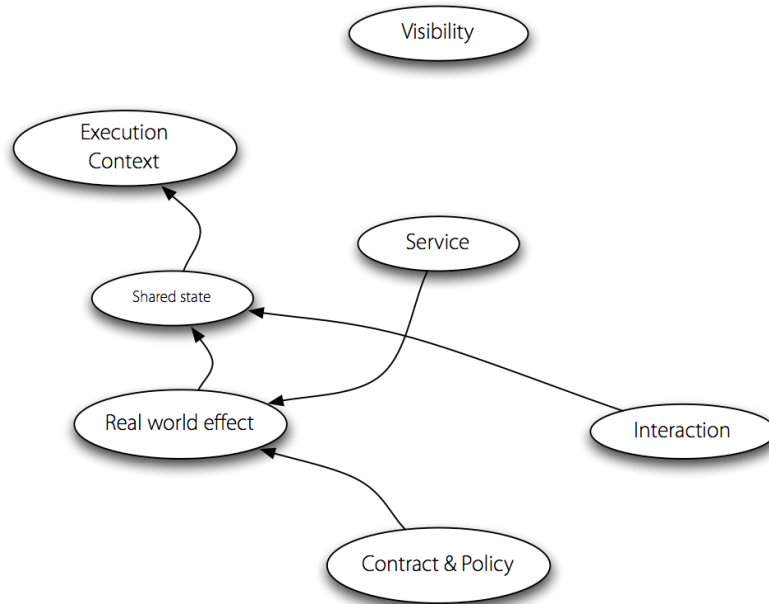
454 Note that although the process model is an essential part of this Reference Model, its extent is  
455 not completely defined. In some architectures the process model will include aspects that are not  
456 strictly part of SOA – for example, in this Reference Model we do not address the orchestration of  
457 multiple services, although orchestration and choreography may be part of the process model of  
458 a given architecture. At a minimum, the process model **MUST** cover the interactions with the  
459 service itself.

460 Beyond the straightforward mechanics of interacting with a service there are other, higher-order,  
461 attributes of services' process models that are also often important. These can include whether  
462 the service is **idempotent**, whether the service is **long-running** in nature and whether it is  
463 important to account for any **transactional** aspects of the service.

#### 464 3.2.3 Real World Effect

465 There is always a particular purpose associated with interacting with a service. Conversely, a  
466 service provider (and consumer) often has a priori conditions that apply to its interactions. The  
467 service consumer is trying to achieve some result by using the service, as is the service provider.  
468 At first sight, such a goal can often be expressed as “trying to get the service to do something”.  
469 This is sometimes known as the real world effect of using a service. For example, an airline  
470 reservation service can be used in order to book travel – the desired real world effect being a seat  
471 on the right airplane.





472

473 Figure 6 Real World Effect and shared state

474 The internal actions that service providers and consumers perform as a result of participation in  
 475 service interactions are, by definition, private and fundamentally unknowable. By unknowable we  
 476 mean both that external parties cannot see others' private actions and, furthermore, SHOULD  
 477 NOT have explicit knowledge of them. Instead we focus on the set of facts shared by the parties  
 478 – the *shared state*. Actions by service providers and consumers lead to modifications of this  
 479 shared state; and the real world effect of a service interaction is the accumulation of the changes  
 480 in the shared state.

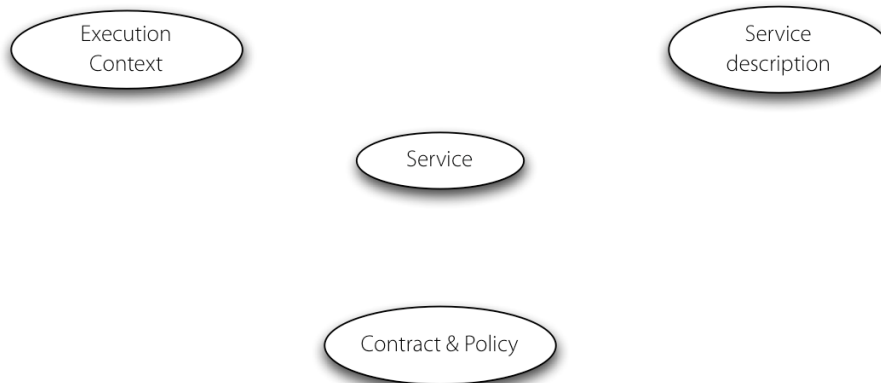
481 There is a strong relationship between the shared state and the interactions that lead up to that  
 482 state. The elements of the shared state SHOULD be inferable from that prior interaction together  
 483 with other context as necessary. In particular, it is not required that the state be recorded;  
 484 although without such recording it may become difficult to audit the interaction at a subsequent  
 485 time.

486 For example, when an airline has confirmed a seat for a passenger on a flight this represents a  
 487 fact that both the airline and the passenger share – it is part of their shared state. Thus the real  
 488 world effect of booking the flight is the modification of this shared state – the creation of the fact of  
 489 the booking. Flowing from the shared facts, the passenger, the airline, and interested third  
 490 parties may make inferences – for example, when the passenger arrives at the airport the airline  
 491 confirms the booking and permits the passenger onto the airplane (subject of course to the  
 492 passenger meeting the other requirements for traveling).

493 For the airline to know that the seat is confirmed it will likely require some private action to record  
 494 the reservation. However, a passenger should not have to know the details of the airline internal  
 495 procedures. The passenger's understanding of the reservation is independent of how the airline  
 496 maintains its records.

### 497 3.3 About services

498 In support of the dynamics of interacting with services are a set of concepts that are about  
 499 services themselves. These are the service description, the execution context of the service and  
 500 the contracts and policies that relate to services and service participants.

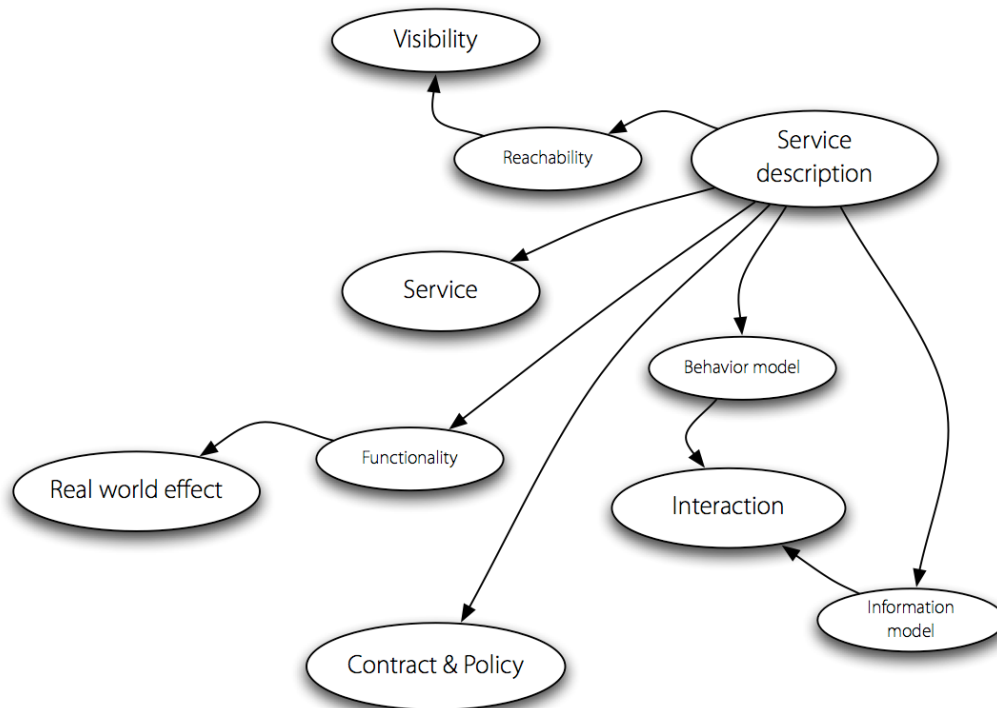


501  
502 Figure 7 About services

### 503 3.3.1 Service description

504 One of the hallmarks of a Service Oriented Architecture is the large amount of associated  
505 documentation and description.

506 The service description represents the information needed in order to use a service. In most  
507 cases, there is no one “right” description but rather the elements of description required depend  
508 on the context and the needs of the parties using the associated entity. While there are certain  
509 elements that are likely to be part of any service description, most notably the information model,  
510 many elements such as function and policy may vary.



511  
512 Figure 8 Service description

513 The purpose of description is to facilitate interaction and visibility, particularly when the  
514 participants are in different ownership domains, between participants in service interactions. By  
515 providing descriptions, it makes it possible for potential participants to construct systems that use  
516 services and even offer compatible services.

517 For example, descriptions allow participants to discriminate amongst possible choices for service  
518 interaction; such as whether the service provides required capabilities, how to access the service,  
519 and negotiate over specific service functionality. In addition, descriptions can be used to support  
520 the management of services, both from the service provider's perspective and the service  
521 consumer's perspective.

522 Best practice suggests that the service description SHOULD be represented using a standard,  
523 referenceable format. Such a format facilitates the use of common processing tools (such as  
524 discovery engines) that can capitalize on the service description.

525 While the concept of a SOA supports use of a service without the service consumer needing to  
526 know the details of the service implementation, the service description makes available critical  
527 information that a consumer needs in order to decide whether or not to use a service. In  
528 particular, a service consumer needs to possess the following items of information:

- 529 1. That the service exists and is **reachable**;
- 530 2. That the service performs a certain function or set of functions;
- 531 3. That the service operates under a specified set of constraints and policies;
- 532 4. That the service will (to some implicit or explicit extent) comply with policies as prescribed  
533 by the service consumer;
- 534 5. How to interact with the service in order to achieve the required objectives, including the  
535 format and content of information exchanged between the service and the consumer and  
536 the sequences of information exchange that may be expected.

537 While each of these items SHOULD be represented in any service description, the details can be  
538 included through references (links) to external sources and are NOT REQUIRED to be  
539 incorporated explicitly. This enables reuse of standard definitions, such as for functionality or  
540 policies.

541 Other sections of this document deal with these aspects of a service, but the following  
542 subsections discuss important elements as these relate to the service description itself.

### 543 **3.3.1.1 Service Reachability**

544 Reachability is an inherently pairwise relationship between service providers and service  
545 consumers. However, a service description SHOULD include sufficient data to enable a service  
546 consumer and service provider to interact with each other. This MAY include metadata such as  
547 the location of the service and what information protocols it supports and requires. It MAY also  
548 include dynamic information about the service, such as whether it is currently available.

### 549 **3.3.1.2 Service Functionality**

550 A service description SHOULD unambiguously express the function(s) of the service and the real  
551 world effects (see Section 3.2.3) that result from it being invoked. This portion of the description  
552 SHOULD be expressed in a way that is generally understandable by service consumers but able  
553 to accommodate a vocabulary that is sufficiently expressive for the domain for which the service  
554 provides its functionality. The description of functionality may include, among other possibilities,  
555 a textual description intended for human consumption or identifiers or keywords referenced to  
556 specific machine-processable definitions. For a full description, it MAY indicate multiple  
557 identifiers or keywords from a number of different collections of definitions.

558 Part of the description of functionality may include underlying technical assumptions that  
559 determine the limits of functionality exposed by the service or of the underlying capability. For  
560 example, the amounts dispensed by an automated teller machine (ATM) are consistent with the  
561 assumption that the user is an individual rather than a business. To use the ATM, the user must  
562 not only adhere to the policies and satisfy the constraints of the associated financial institution  
563 (see Section 3.3.1.3 for how this relates to service description and Section 3.3.2 for a detailed  
564 discussion) but the user is limited to withdrawing certain fixed amounts of cash and a certain  
565 number of transactions in a specified period of time. The financial institution, as the underlying

566 capability, does not have these limits but the service interface as exposed to its customers does,  
567 consistent with its assumption of the needs of the intended user. If the assumption is not valid,  
568 the user may need to use another service to access the capability.

### 569 **3.3.1.3 Policies Related to a Service**

570 A service description MAY include support for associating policies with a service and providing  
571 necessary information for prospective consumers to evaluate if a service will act in a manner  
572 consistent with the consumer's constraints.

### 573 **3.3.1.4 Service Interface**

574 The service interface is the means for interacting with a service. It includes the specific protocols,  
575 commands, and information exchange by which actions are initiated that result in the real world  
576 effects as specified through the service functionality portion of the service description.

577 The specifics of the interface SHOULD be syntactically represented in a standard referenceable  
578 format. These prescribe what information needs to be provided to the service in order to access  
579 its capabilities and interpret responses. This is often referred to as the service's information  
580 model, see Section 3.2.2.1. It should be noted that the particulars of the interface format are  
581 beyond the scope of the reference model. However, the existence of interfaces and accessible  
582 descriptions of those interfaces are fundamental to the SOA concept.

583 While this discussion refers to a standard referenceable syntax for service descriptions, it is not  
584 specified how the consumer accesses the interface definition nor how the service itself is  
585 accessed. However, it is assumed that for a service to be usable, its interface MUST be  
586 represented in a format that allows interpretation of the interface information by its consumers.

### 587 **3.3.1.5 The Limits of Description**

588 There are well-known theoretic limits on the effectiveness of descriptions – it is simply not  
589 possible to specify, completely and unambiguously, the precise semantics of and all related  
590 information about a service.

591 There will always be unstated assumptions made by the describer of a service that must be  
592 implicitly shared by readers of the description. This applies to machine processable descriptions  
593 as well as to human readable descriptions.

594 Fortunately, complete precision is not necessary – what is required is sufficient scope and  
595 precision to support intended use.

596 Another kind of limit of service descriptions is more straightforward: whenever a repository is  
597 searched using any kind of query there is always the potential for *zero or more* responses – no  
598 matter how complete the search queries or the available descriptions appear to be. This is  
599 inherent in the principles involved in search.

600 In the case that there is more than one response, this set of responses has to be converted into a  
601 single choice. This is a private choice that must be made by the consumer of the search  
602 information.

## 603 **3.3.2 Policies and Contracts**

604 A **policy** represents some constraint or condition on the use, deployment or description of an  
605 owned entity as defined by any participant. A **contract**, on the other hand, represents an  
606 agreement by two or more parties. Like policies, agreements are also about the conditions of use  
607 of a service; they may also constrain the expected real world effects of using a service. The  
608 reference model is focused primarily on the concept of policies and contracts as they apply to  
609 services. We are not concerned with the form or expressiveness of any language used to  
610 express policies and contracts.

### 611 3.3.2.1 Service Policy

612 Conceptually, there are three aspects of policies: the policy assertion, the policy owner  
613 (sometimes referred to as the policy subject) and policy enforcement.

614 For example, the assertion: “All messages are encrypted” is an assertion regarding the forms of  
615 messages. As an assertion, it is measurable: it may be true or false depending on whether the  
616 traffic is encrypted or not. Policy assertions are often about the way the service is realized; i.e.,  
617 they are about the relationship between the service and its execution context, see 3.3.3.

618 A policy always represents a participant’s point of view. An assertion becomes the policy of a  
619 participant when they adopt the assertion as their policy. This linking is normally not part of the  
620 assertion itself. For example, if the service consumer declares that “All messages are encrypted”,  
621 then that reflects the policy of the service consumer. This policy is one that may be asserted by  
622 the service consumer independently of any agreement from the service provider.

623 Finally, a policy may be enforced. Techniques for the enforcement of policies depend on the  
624 nature of the policy. Conceptually, service policy enforcement amounts to ensuring that the policy  
625 assertion is consistent with the real world. This might mean preventing unauthorized actions to be  
626 performed or states to be entered into; it can also mean initiating compensatory actions when a  
627 policy violation has been detected. An unenforceable constraint is not a policy; it would be better  
628 described as a wish.

629 Policies potentially apply to many aspects of SOA: security, privacy, manageability, Quality of  
630 Service and so on. Beyond such infrastructure-oriented policies, participants MAY also express  
631 business-oriented policies – such as hours of business, return policies and so on.

632 Policy assertions SHOULD be written in a form that is understandable to, and processable by, the  
633 parties to whom the policy is directed. Policies MAY be automatically interpreted, depending on  
634 the purpose and applicability of the policy and how it might affect whether a particular service is  
635 used or not.

636 A natural point of contact between service participants and policies associated with the service is  
637 in the service description – see Section 3.3.1. It would be natural for the service description to  
638 contain references to the policies associated with the service.

### 639 3.3.2.2 Service Contract

640 Whereas a policy is associated with the point of view of individual participants, a contract  
641 represents an agreement between two or more participants. Like policies, contracts can cover a  
642 wide range of aspects of services: quality of service agreements, interface and choreography  
643 agreements and commercial agreements. Note that we are not necessarily referring to legal  
644 contracts here.

645 Thus, following the discussion above, a service contract is a measurable assertion that governs  
646 the requirements and expectations of two or more parties. Unlike policy enforcement, which is  
647 usually the responsibility of the policy owner, contract enforcement may involve resolving  
648 disputes between the parties to the contract. The resolution of such disputes may involve appeals  
649 to higher authorities.

650 Like policies, contracts may be expressed in a form that permits automated interpretation. Where  
651 a contract is used to codify the results of a service interaction, it is good practice to represent it in  
652 a machine processable form. Among other purposes, this facilitates automatic service  
653 composition. Where a contract is used to describe over-arching agreements between service  
654 providers and consumers, then the priority is likely to make such contracts readable by people.

655 Since a contract is inherently the result of agreement by the parties involved, there is a *process*  
656 associated with the agreement action. Even in the case of an implicitly agreed upon contract,  
657 there is logically an agreement action associated with the contract, even if there is no overt action  
658 of agreement. A contract may be arrived at by a mechanism that is not directly part of an SOA –  
659 an out of band process. Alternatively, a contract may be arrived at during the course of a service  
660 interaction – an in-band process.

### 661 3.3.3 Execution context

662 The **execution context** of a service interaction is the set of infrastructure elements, process  
663 entities, policy assertions and agreements that are identified as part of an instantiated service  
664 interaction, and thus forms a path between those with needs and those with capabilities. The  
665 consumer and provider can be envisioned as separate places on a map and, for a service to  
666 actually be invoked, a path must be established between those two places. This path is the  
667 execution context. As with a path between places, it can be a temporary connection (e.g. a  
668 tenuous footbridge of an ad hoc exchange) or a well-defined coordination (e.g. a super highway)  
669 that can be easily reused in the future.

670 The execution context is not limited to one side of the interaction; rather it concerns the totality of  
671 the interaction – including the service provider, the service consumer and the common  
672 infrastructure needed to mediate the interaction. While there may be third parties, for example,  
673 government regulators, who set some of the conditions for the execution context, this merely  
674 increases the conditions and constraints needing to be coordinated and may require additional  
675 information exchange to complete the execution context.

676 The execution context is central to many aspects of a service interaction. It defines, for example,  
677 a decision point for policy enforcement relating to the service interaction. Note that a policy  
678 decision point is not necessarily the same as an enforcement point: an execution context is not by  
679 itself something that lends itself to enforcement. On the other hand, any enforcement mechanism  
680 of a policy is likely to take into account the particulars of the actual execution context.

681 The execution context also allows us to distinguish services from one another. Different instances  
682 of the same service – denoting interactions between a given service provider and different service  
683 consumers for example – are distinguished by virtue of the fact that their execution contexts are  
684 different.

685 Finally, the execution context is also the context in which the interpretation of data that is  
686 exchanged takes place. A particular string has a particular meaning in a service interaction in a  
687 particular context – the execution context.

688 An execution context often evolves during a service interaction. The set of infrastructure  
689 elements, the policies and agreements that apply to the interaction, may well change during a  
690 given service interaction. For example, at an initial point in an interaction, it may be decided by  
691 the parties that future communication should be encrypted. As a result the execution context also  
692 changes – to incorporate the necessary infrastructure to support the encryption and continue the  
693 interaction.

---

## 694 4 Conformance Guidelines

695 The authors of this reference model envision that architects may wish to declare their architecture  
696 is conformant with this reference model. Conforming to a Reference Model is not generally an  
697 easily automatable task – given that the Reference Model’s role is primarily to define concepts  
698 that are important to SOA rather than to give guidelines for implementing systems.

699 However, we do expect that any given Service Oriented Architecture will reference the concepts  
700 outlined in this specification. As such, we expect that any design for a system that adopts the  
701 SOA approach will

- 702 • Have entities that can be identified as services as defined by this Reference Model;
- 703 • Be able to identify how visibility is established between service providers and consumers;
- 704 • Be able to identify how interaction is mediated;
- 705 • Be able to identify how the effect of using services is understood;
- 706 • Have descriptions associated with services;
- 707 • Be able to identify the execution context required to support interaction; and
- 708 • It will be possible to identify how policies are handled and how contracts may be modeled  
709 and enforced.

710 It is not appropriate for this specification to identify *best practices* with respect to building SOA-  
711 based systems. However, the ease with which the above elements can be identified within a  
712 given SOA-based system could have significant impact on the scalability, maintainability and  
713 ease of use of the system.

---

714 **5 References**

715 **5.1 Normative**

716 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,  
717 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.  
718

719 **5.2 Non-Normative**

720 [W3C WSA] W3C Working Group Note "Web Services Architecture",  
721 <http://www.w3.org/TR/ws-arch/> , 11 February 2004



---

## 722 A. Glossary

723 The glossary contains a concise definition of terms used within this specification, but the full  
724 description in the text is the normative description.

725

### 726 Action Model

727 The characterization of the permissible actions that may be invoked against a service.  
728 See Section 3.2.2.2.1.

### 729 Architecture

730 A set of artifacts (that is: principles, guidelines, policies, models, standards and  
731 processes) and the relationships between these artifacts, that guide the selection,  
732 creation, and implementation of solutions aligned with business goals.

733 Software architecture is the structure or structures of an information system consisting of  
734 entities and their externally visible properties, and the relationships among them.

### 735 Awareness

736 A state whereby one party has knowledge of the existence of the other party. Awareness  
737 does not imply willingness or reachability. See Section 3.2.1.1.

### 738 Behavior Model

739 The characterization of (and responses to, and temporal dependencies between) the  
740 actions on a service. See Section 3.2.2.2.

### 741 Capability

742 A real-world effect that a service provider is able to provide to a service consumer. See  
743 Section 2.1.

### 744 Execution context

745 The set of technical and business elements that form a path between those with needs  
746 and those with capabilities and that permit service providers and consumers to interact.  
747 See Section 3.3.3.

### 748 Framework

749 A set of assumptions, concepts, values, and practices that constitutes a way of viewing  
750 the current environment.

### 751 Idempotency/Idempotent

752 A characteristic of a service whereby multiple attempts to change a state will always and  
753 only generate a single change of state if the operation has already been successfully  
754 completed once. See Section 3.2.2.2.2.

### 755 Information model

756 The characterization of the information that is associated with the use of a service. See  
757 Section 3.2.2.1.

### 758 Interaction

759 The activity involved in making using of a capability offered, usually across an ownership  
760 boundary, in order to achieve a particular desired real-world effect. See Section 3.2.3.

### 761 Offer

762 An invitation to use the capabilities made available by a service provider in accordance  
763 with some set of policies.

- 764 **Policy**
- 765 A statement of obligations, constraints or other conditions of use of an owned entity as  
766 defined by a participant. See Section 3.3.2.
- 767 **Process Model**
- 768 The characterization of the temporal relationships between and temporal properties of  
769 actions and events associated with interacting with the service. See Section 3.2.2.2.2.
- 770 **Reachability**
- 771 The ability of a service consumer and service provider to interact. Reachability is an  
772 aspect of visibility. See Section 3.2.1.3.
- 773 **Real world effect**
- 774 The actual result of using a service, rather than merely the capability offered by a service  
775 provider. See Section 3.2.3.
- 776 **Reference Architecture**
- 777 A reference architecture is an architectural design pattern that indicates how an abstract  
778 set of mechanisms and relationships realizes a predetermined set of requirements. See  
779 Section 1.1.
- 780 **Reference Model**
- 781 A reference model is an abstract framework for understanding significant relationships  
782 among the entities of some environment that enables the development of specific  
783 architectures using consistent standards or specifications supporting that environment.
- 784 A reference model consists of a minimal set of unifying concepts, axioms and  
785 relationships within a particular problem domain, and is independent of specific  
786 standards, technologies, implementations, or other concrete details. See Section 1.1.
- 787 **Semantics**
- 788 A conceptualization of the implied meaning of information, that requires words and/or  
789 symbols within a usage context. See Section 3.2.2.1.2.
- 790 **Semantic Engagement**
- 791 The relationship between an agent and a set of information that depends on a particular  
792 interpretation of the information. See Section 3.2.2.1.
- 793 **Service**
- 794 The means by which the needs of a consumer are brought together with the capabilities  
795 of a provider. See Section 3.1.
- 796 **Service Consumer**
- 797 An entity which seeks to satisfy a particular need through the use capabilities offered by  
798 means of a service.
- 799 **Service description**
- 800 The information needed in order to use, or consider using, a service. See Section 3.3.1.
- 801 **Service Interface**
- 802 The means by which the underlying capabilities of a service are accessed. See Section  
803 3.3.1.4.
- 804 **Service Oriented Architecture (SOA)**
- 805 Service Oriented Architecture is a paradigm for organizing and utilizing distributed  
806 capabilities that may be under the control of different ownership domains. It provides a  
807 uniform means to offer, discover, interact with and use capabilities to produce desired  
808 effects consistent with measurable preconditions and expectations. See Section 2.1.

809 **Service Provider**

810 An entity (person or organization) that offers the use of capabilities by means of a  
811 service.

812 **Visibility**

813 The capacity for those with needs and those with capabilities to be able to interact with  
814 each other. See Section 3.2.1.

815 **Willingness**

816 A predisposition of service providers and consumers to interact. See Section 3.2.1.2.

---

817

## B. Acknowledgments

818 The following individuals were members of the committee during the development of this  
819 specification and are gratefully acknowledged:

820 **Participants:**

821 Christopher Bashioum, Mitre Corporation  
822 Prasanta Behera, Individual Member  
823 Kathryn Breininger, The Boeing Company  
824 Rex Brooks, HumanMarkup.org, Inc.  
825 Al Brown, FileNet Corporation  
826 Peter Brown, Individual Member  
827 Joseph Chiusano, Booz Allen Hamilton  
828 Jeff Estefan, Propulsion Laboratory  
829 Don Flinn; Steve Jones, Capgemini  
830 Gregory Kohring, NEC Europe Ltd.  
831 Ken Laskey, Mitre Corporation  
832 C. Matthew MacKenzie (**secretary**), Adobe Systems  
833 Francis McCabe (**secretary**), Fujitsu Laboratories of America Ltd.  
834 Wesley McGregor, Treasury Board of Canada, Secretariat  
835 Tom Merkle, Lockheed Martin Information Technology  
836 Rebekah Metz, Booz Allen Hamilton  
837 Oleg Mikulinsky, WebLayers, Inc.  
838 Jyoti Namjoshi, Patni Computer Systems, Ltd.  
839 Duane Nickull (**chair**), Adobe Systems  
840 George Ntinolazos, Strata Software Ltd  
841 Joseph Pantella, Individual Member  
842 Ron Schuldt, Lockheed Martin  
843 Michael Stiefel, Reliable Software, Inc.  
844 Danny Thornton, Individual Member