OASIS

Reference Model for Service Oriented Architectures

Working Draft 11ED, 15 December 2005

Editors:
>   C. Matthew MacKenzie, Adobe Systems Incorporated, mattm@adobe.com
>   Ken Laskey, MITRE Corporation, klaskey@mitre.org
>   Francis McCabe, Fujitsu Limited, fgm@fla.fujitsu.com
>   Peter Brown, Individual, peter@justbrown.net
>   Rebekah Metz, Booz Allen Hamilton, metz_rebekah@bah.com

Abstract:
>   This Reference Model for Service Oriented Architectures 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. While the concepts and relationships described may apply to other "service" environments, 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 list. Others should visit the SOA-RM TC home page at 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

42   The errata page for this specification is at:

43   http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm.

44

Table of Contents

Draft

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. We know that concepts such as eating areas, 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 to address the requirements for developing housing solutions in large apartment complexes, suburban single family houses, and 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 Architecture

Figure 1 shows how a reference model for SOA relates to other distributed systems architectural inputs.

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 impact SOA.
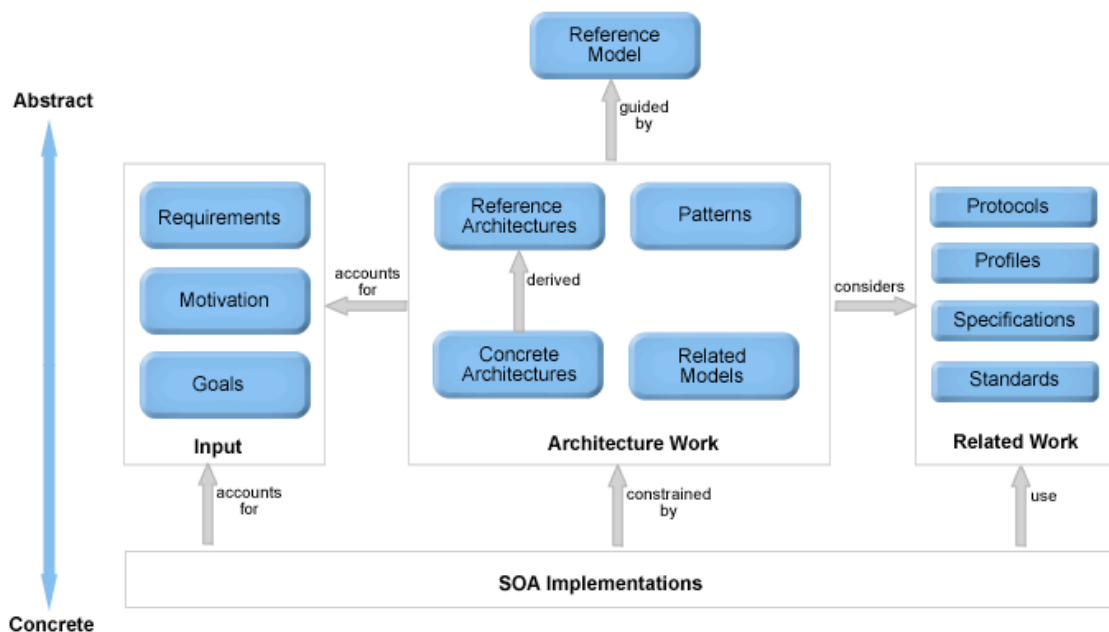
121

122   *Figure 1 How the Reference Model relates to other work*

1.3 Audience

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

125   •   Architects and developers designing, identifying or developing a system based on the
126   service-oriented paradigm.
127   •   Standards architects and analysts developing specifications that rely on service oriented
128   architecture concepts.
129   •   Decision makers seeking a "consistent and common" understanding of service oriented
130   architecture.
131   •   Users who need a better understanding of the concepts and benefits of service oriented
132   architecture.

1.4 How to use the reference model

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

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

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

144   Section 3 introduces the Reference Model for SOA. In any framework as rich as SOA, it is difficult
145   to avoid a significant amount of cross referencing between concepts. This makes presentation of
146   the material subject to a certain amount of arbitrariness. We resolve this by introducing the
147   concept of *service* itself, then we introduce concepts that relate to the dynamic aspects of service

148 and finally we introduce those concepts that refer to the meta-level aspects of services such as
149 service description and policies as they apply to services.

150 Section 4 addresses compliance with this reference model.

151 The glossary provides definitions of terms which are relied upon within the reference model
152 specification but do not necessarily form part of the specification itself.

1.5 Notational Conventions

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

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

1.6 Relationships to Other Standards

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

160 • Considers its work "service oriented";

161 • Makes (publicly) an adoption statement to use the Reference Model for SOA of this TC
162 as a base or inspiration for their work; and

163 • Standards or technologies that claim to be service oriented.

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

166 2 Service Oriented Architecture

167 2.1 What is Service Oriented Architecture?

168 Service Oriented Architecture (SOA) is a paradigm for organizing and utlizing distributed
169 capabilities that may be under the control of different ownership domains.

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

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

179 Visibility, interaction, and effect are key concepts for describing the SOA paradigm. **Visibility**
180 refers to the capacity for those with needs and those with capabilities to be able to see each
181 other. This is typically done through providing descriptions for such aspects as functions and
182 technical requirements, related constraints and policies, and mechanisms for access or response.
183 The descriptions need to be in a form (or can be transformed to a form) in which its syntax and
184 semantics are widely accessible and understandable.

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

192 The purpose of using a capability is to realize one or more **real world effects**. At its core, an
193 interaction is "an act" as opposed to "an object" and the result of an interaction is an effect (or a
194 set/series of effects). We are careful to distinguish *public* actions and *private* actions; private
195 actions are inherently unknowable by other parties. On the other hand, public actions result in
196 changes to the *state* that is shared (at least) between those involved in the current execution
197 context. Real world effects are, then, couched in terms of changes to this shared state.

198 The expected effects should be made visible as part of the capability description and form an
199 important part of the decision on whether a given capability matches similarly described needs.
200 At the interaction stage, the description of real world effects establishes the expectations of those
201 using the capability. Note, it is not possible to describe every effect from using a capability and,
202 in fact, a cornerstone of SOA is that one using a capability does not need to know all the details.

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

207 • The capability to perform work for another

208 • The specification of the work offered for another

209 • The offer to perform work for another

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

213 SOA is not itself a solution to domain problems but rather an organizing and delivery paradigm
214 that enables one to get more value from use both of capabilities which are locally "owned" and
215 those under the control of others. It also enables one to express solutions in a way that makes it
216 easier to modify or evolve the identified solution or to try alternate solutions. SOA does not
217 provide any domain elements of a solution that do not exist without SOA.

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

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

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

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

237 Note that although SOA is commonly implemented using Web services, services can be made
238 visible, support interaction, and generate effects through other implementation strategies. Web
239 service-based architectures and technologies are specific and concrete and while the concepts in
240 the Reference Model apply to such systems, they are too solution specific to be part of a general
241 reference model.

2.1.1 A worked Service Oriented Architecture example

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

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

259 Another person (say, a visitor to someone else's house) may use a contracted supply without any
260 relationship with the utility or any requirement to also satisfy the initial service constraint (i.e.

261 reachability only requires intact electricity distribution) but would nonetheless be expected to be
262 compatible with the service interface.

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

266 In this example, the underlying capability would still exist and be usable even if every device were
267 required to be hard-wired to the utility's equipment, but this would result in a very different service
268 and service interface.

2.2 How is Service Oriented Architecture different?

269

270 How does this paradigm of Service Oriented Architecture differ from other approaches to
271 organizing and understanding IT assets? Essentially, there are two areas in which SOA shapes
272 the framework of concepts that underlie distributed systems.

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

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

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

2.3 The Benefits of Service Oriented Architecture

291

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

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

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

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

308

3 The Reference Model

309

310 Figure 2 illustrates the principal concepts this reference model defines. The relationships between
311 them are developed as each concept is defined in turn.
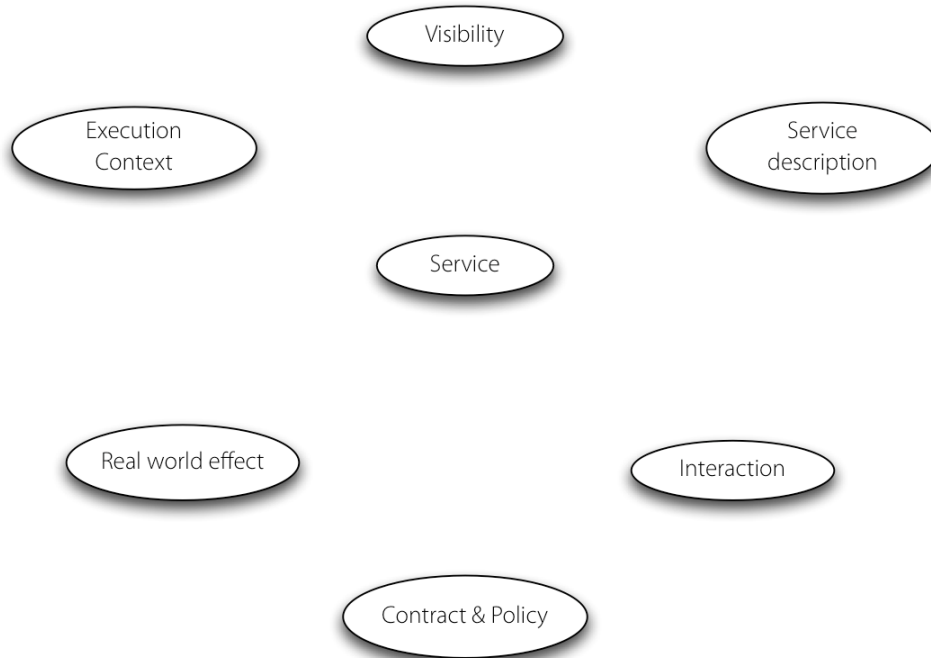
312
313 *Figure 2 Principal concepts in the Reference Model*

3.1 Service

314

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

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

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

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

332

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

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

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

336

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

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

346 3.2 Dynamics of Services

347 From a dynamic perspective, there are three fundamental concepts that are important in
348 understanding what is involved in interacting with services: the **visibility** between service
349 providers and consumers, the **interaction** between them, and the **real world effect** of interacting
350 with a service.
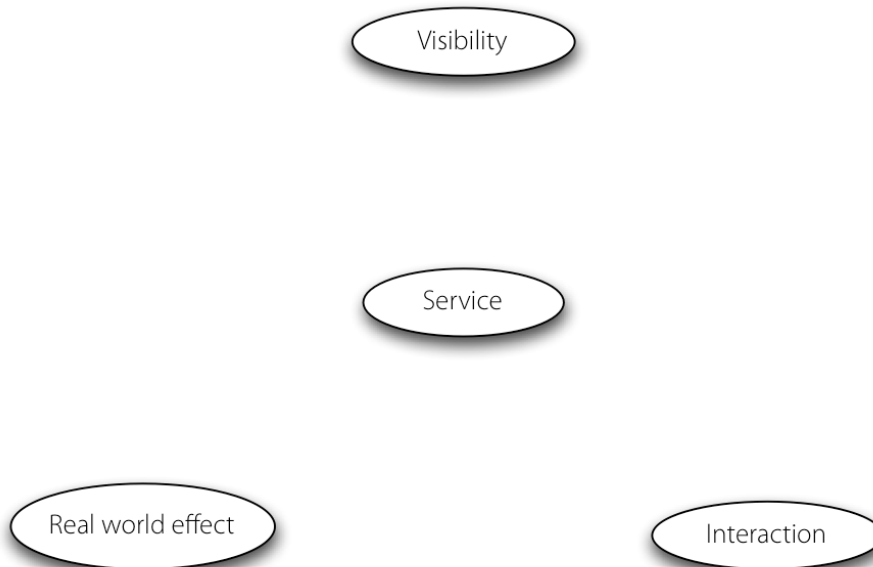
351

352 *Figure 3 Concepts around the dynamics of service*

353 3.2.1 Visibility

354 For a service provider and consumer to interact with each other they have to be able to 'see' each
355 other. This is, in fact, true for any consumer/provider relationship – including in an application
356 program where one program calls another: without the proper libraries being present the function
357 call cannot complete. In the case of SOA, visibility needs to be emphasized because it is not
358 necessarily obvious how service participants *can* see each other.
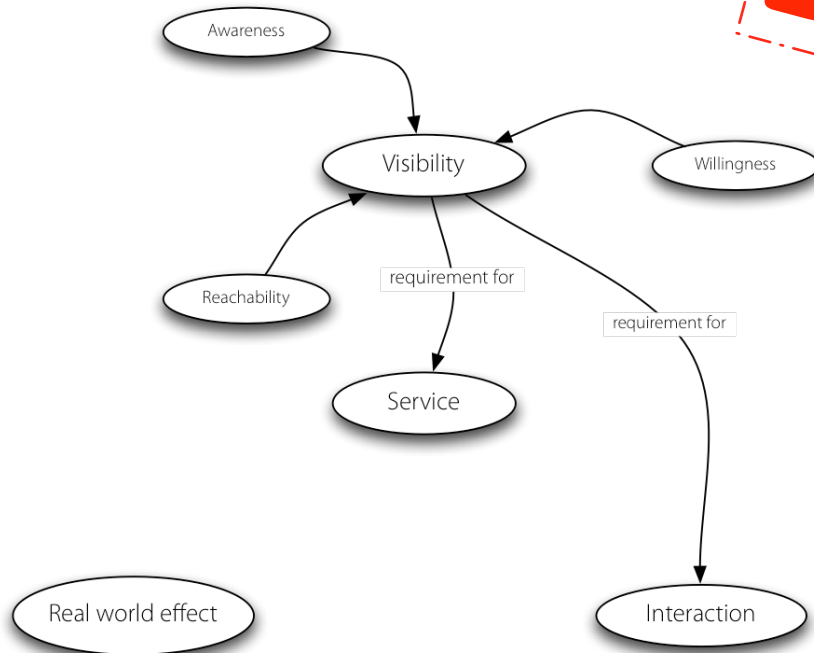
359

360   *Figure 4 Concepts around Visibility*

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

3.2.1.1 Awareness

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

370   Awareness of service offerings is often effected by various *discovery* mechanisms. For a service
371   consumer (say) to discover a service, the service provider must be capable of making details of
372   the service (notably service description and policies) available to potential consumers; and
373   consumers must be capable of becoming aware of that information.

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

379   For example, a service provider may advertise and promote their service by either including it in a
380   service directory or broadcasting it to all consumers; potential consumers may broadcast their
381   particular service needs in the hope that a suitable service responds with a proposal or offer or a
382   service consumer might also probe an entire network to determine if suitable services exist.
383   When the demand for a service is higher than the supply, then, by advertising their needs,
384   potential consumers are likely to be more effective than service providers advertising offered
385   services.

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

3.2.1.2 Willingness

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

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

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

3.2.1.3 Reachability

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

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

3.2.2 Interacting with services

Interacting with a service involves performing actions against the service. In many cases, this is accomplished by sending and receiving messages, but there are other modes possible that do not involve explicit message transmission. However, for simplicity, we often refer to message exchange as the primary mode of interaction with a service.
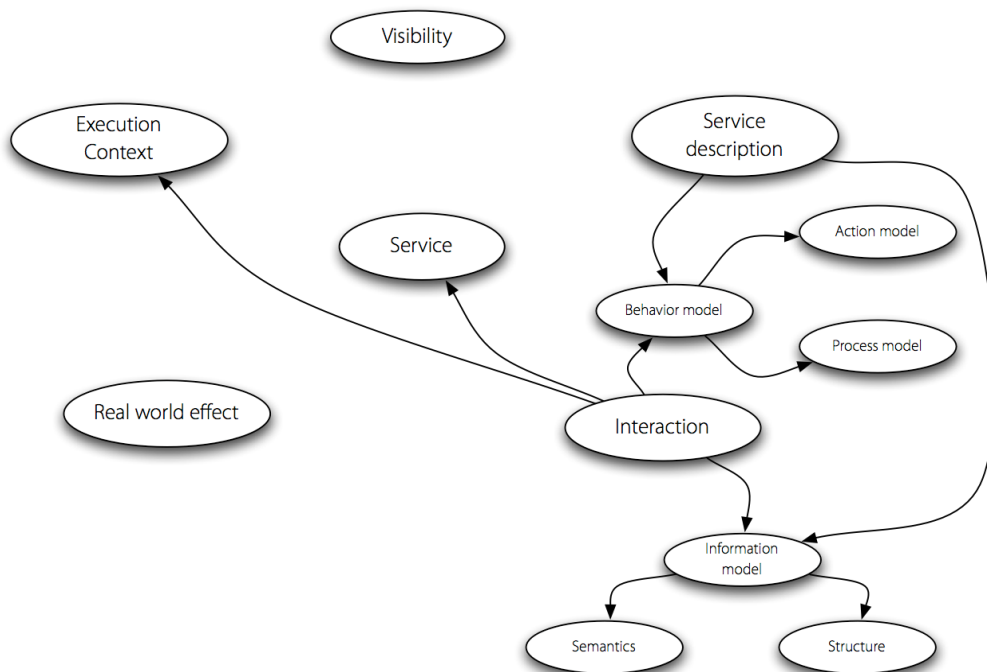
Figure 5 Service Interaction concepts

416 Figure 5 illustrates the key concepts that are important in understanding what it is involved in
417 interacting with services; these revolve around the **service interface** – which is composed of a
418 **information model** and a **behavior model**.

3.2.2.1 Information model

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

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

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

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

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

3.2.2.1.1 Structure

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

441 A described information model typically has a great deal to say about the form of messages.
442 However, pure "typed" information is not sufficient to completely describe the appropriate
443 interpretation of data. For example, within a street address structure, the city name and the street
444 name are typically given the same data type – some variant of the string type. However, city
445 names and street names are not really the same type of thing at all. Distinguishing the correct
446 interpretation of a city name string and a street name string is not possible using type-based
447 techniques – it requires additional information that cannot be expressed purely in terms of the
448 structure of data.

3.2.2.1.2 Semantics

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

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

459 For example, there is often a huge potential for variability in representing street addresses. For
460 example, an address in San Francisco, California may have variations in the way the city is
461 represented: SF, San Francisco, San Fran, the City by the Bay are all alternate denotations of the
462 same city. For successful exchange of address information, all the participants must have a

463 consistent view of the meaning of the address tokens if address information is to be reliably
464 shared.

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

469 Note that, for the most part, it is not expected that service consumers and providers would
470 actually exchange descriptions of terms in their interaction but, rather, would reference existing
471 descriptions – the role of the semantics being a background one – that are mostly to be found in
472 **service descriptions**.

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

3.2.2.2 Behavior model

477 The second key requirement for successful interactions with services is knowledge of the actions
478 invoked against the service and the process or temporal aspects of interacting with the service.
479 Loosely, this can be characterized as knowledge of the actions on, responses to and temporal
480 dependencies between actions on the service.

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

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

3.2.2.2.1 Action model

490 The **action model** of a service is about the individual actions that may be invoked against the
491 service. Of course, a great portion of the behavior resulting from an action may be private;
492 however, the expected public view of a service surely includes the implied effects of actions.

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

3.2.2.2.2 Process Model

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

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

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

3.2.3 Real World Effect

514 There is always a particular purpose associated with interacting with a service. Conversely, a
515 service provider (and consumer) often has a priori conditions that apply to its interactions. The
516 service consumer is trying to achieve some result by using the service, as is the service provider.
517 At first sight, such a goal can often be expressed as "trying to get the service to do something".
518 This is sometimes known as the **real world effect** of using a service. For example, an airline
519 reservation service can be used in order to book travel – the desired real world effect being a seat
520 on the right airplane.
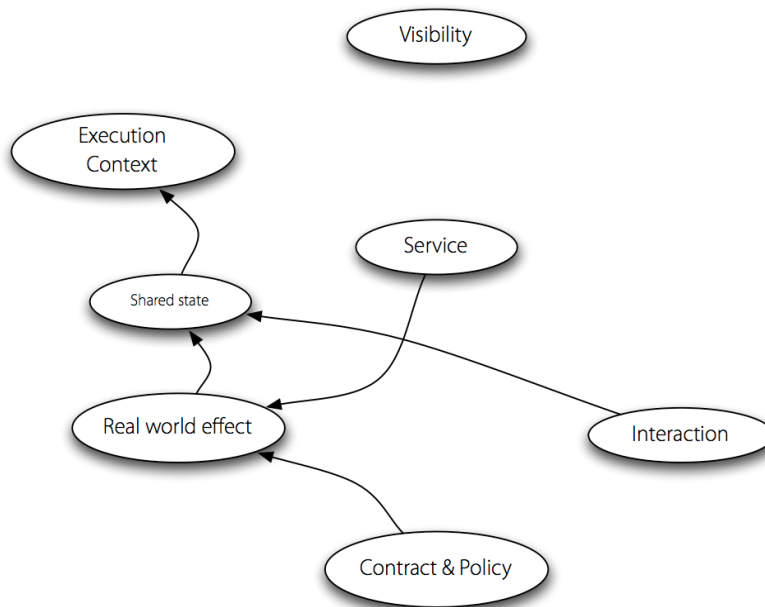
521

522 *Figure 6 Real World Effect and shared state*

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

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

535 For example, when an airline has confirmed a seat for a passenger on a flight this represents a
536 fact that both the airline and the passenger share – it is part of their shared state. Thus the real
537 world effect of booking the flight is the modification of this shared state – the creation of the fact of
538 the booking. Flowing from the shared facts, the passenger, the airline, and interested third
539 parties may make inferences – for example, when the passenger arrives at the airport the airline

540 confirms the booking and permits the passenger onto the airplane (subject of course to the
541 passenger meeting the other requirements for traveling).

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

3.3 About services

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


550

551 *Figure 7 About services*

3.3.1 Service description

553 One of the hallmarks of a Service Oriented Architecture is the degree of documentation and
554 description associated with it.

555 The service description represents the information needed in order to use a service. In most
556 cases, there is no one "right" description but rather the description depends on the context and
557 the needs of the parties using the associated entity. While there are certain elements that are
558 likely to be part of any service description, most notably the information model, many elements
559 such as function and policy may vary.

560
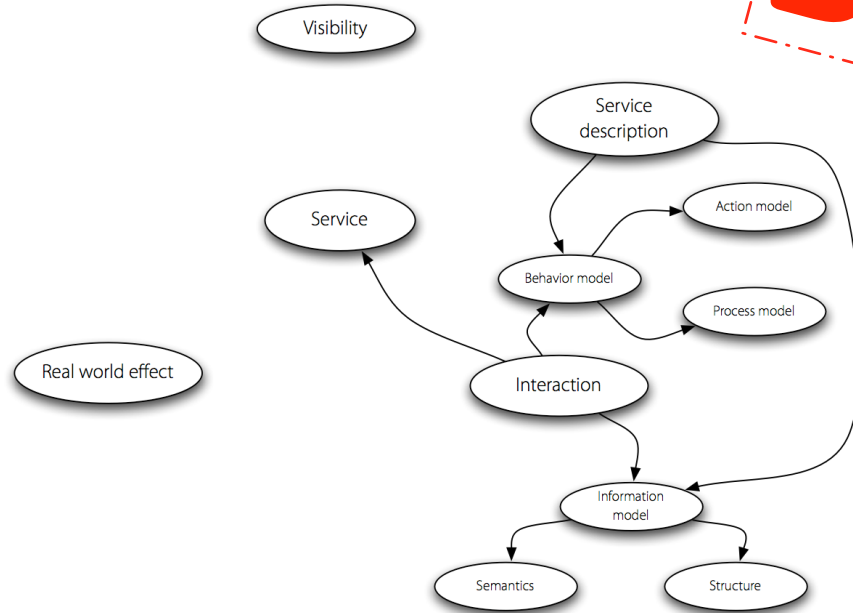
The purpose of description is to facilitate interaction and visibility, particularly across ownership domains, between participants in service interactions. By providing descriptions, it makes it possible for potential participants to construct systems that use services and even offer compatible services.

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

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

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

1. That the service exists and is **reachable**;

2. That the service performs a certain function or set of functions;

3. That the service operates under a specified set of constraints and policies;

4. That the service will (to some implicit or explicit extent) comply with policies as prescribed by the service consumer;

5. How to interact with the service in order to achieve the required objectives, including the format and content of information exchanged between the service and the consumer and the sequences of information exchange that may be expected.

While each of these items should be represented in any service description, the details can be included through reference (links) to external sources and are not required to be incorporated explicitly. This enables reuse of standard definitions, such as for functionality or policies.

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

3.3.1.1 Service Reachability

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

3.3.1.2 Service Functionality

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

Part of the description of functionality may include underlying technical assumptions that determine the limits of functionality exposed by the service or of the underlying capability. For example, the amounts dispensed by an automated teller machine (ATM) are consistent with the assumption that the user is an individual rather than a business. To use the ATM, the user must not only adhere to the policies and satisfy the constraints of the associated financial institution (see Section 3.3.1.3 for how this relates to service description and Section 3.3.2 for a detailed discussion) but the user is limited to withdrawing certain fixed amounts of cash and a certain number of transactions in a specified period of time. The financial institution, as the underlying capability, does not have these limits but the service interface as exposed to its customers does, consistent with its assumption of the needs of the intended user. If the assumption is not valid, the user may need to use another service to access the capability.

3.3.1.3 Policies Related to a Service

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

3.3.1.4 Service Interface

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

The specifics of the interface SHOULD be syntactically represented in a standard referenceable format. These prescribe what information needs to be provided to the service in order to access its capabilities and interpret responses. This is often referred to as the service's information model (see Section 3.2.2.1). It should be noted that the particulars of the interface format are beyond the scope of the reference model. However, requiring that mechanisms be available (in order to define and retrieve such definitions) is fundamental to the SOA concept.

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

3.3.1.5 The Limits of Description

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

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

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

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

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

3.3.2 Policies and Contracts

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

3.3.2.1 Service Policy

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

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

A policy always represents a participant's point of view. An assertion becomes the policy of a participant when they make it their policy. This linking is normally not part of the assertion itself. For example, if the service consumer declares that "All messages are encrypted", then that reflects the policy of the service consumer. This policy is one that may be asserted by the service consumer independently of any agreement from the service provider.

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

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

Policy assertions SHOULD be written in a form that is understandable to, and processable by, the parties to whom the policy is directed. Policies may need to be automatically interpreted,

682 depending on the purpose and applicability of the policy and whether it might affect whether a
683 particular service is used or not.

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

3.3.2.2 Service Contract

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

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

698 Like policies, contracts may be expressed in a form that permits automated interpretation. Where
699 a contract is used to codify the results of a service interaction, it is good practice to represent it in
700 a machine processable form. This facilitates automatic service composition, for example. Where
701 a contract is used to describe over-arching agreements between service providers and
702 consumers, then the priority is likely to make such contracts readable by people.

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

3.3.3 Execution context

710 The **execution context** of a service interaction is the set of infrastructure elements, process
711 entities, policy assertions and agreements that are identified as part of an instantiated service
712 interaction. The consumer and provider can be envisioned as separate places on a map and, for
713 a service to actually be invoked, a path must be established between those two places. This path
714 is the execution context. As with a path between places, it can be a temporary connection (e.g. a
715 tenuous footbridge of an ad hoc exchange) or a well-defined coordination (e.g. a super highway)
716 that can be easily reused in the future.

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

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

728 The execution context also allows us to distinguish services from one another. Different instances
729 of the same service – denoting interactions between a given service provider and different service

730 consumers for example – are distinguished by virtue of the fact that their execution contexts are
731 different.

732 Finally, the execution context is also the context in which the interpretation of data that is
733 exchanged takes place – it is where the *symbol grounding* happens. A particular string has a
734 particular meaning in a service interaction in a particular context – the execution context.

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

4 Conformance Guidelines

741

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

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

749 • Have entities that can be identified as services as defined by this Reference Model;

750 • Be able to identify how visibility is established between service providers and consumers;

751 • Be able to identify how interaction is mediated;

752 • Be able to identify how the effect of using services is understood;

753 • Have descriptions associated with services;

754 • Be able to identify the execution context required to support interaction; and

755 • It will be possible to identify how policies are handled and how contracts may be modeled
756   and enforced.

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

5 References

5.1 Normative

[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.

5.2 Non-Normative

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

Appendix A. Glossary

Terms that are used within this Reference Model are often also found in other specifications. In order to avoid potential ambiguity, this glossary locally scopes the definitions of those terms for the purpose of this Reference Model and thus overrides any other definitions.

Action Model

> The characterization of the permissible actions that may be invoked against a service.

Architecture

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

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

Awareness

> A state whereby one party has knowledge of the existence of the other party. Awareness does not imply willingness or reachability.

Behavioral Model

> The characterization of (and responses to, and temporal dependencies between) the actions on a service.

Capability

> A real-world effect that a service provider is able to provide to a service consumer.

(Service) Consumer

> An entity which seeks to satisfy a particular need through the use capabilities offered by means of a service.

Execution context

> The set of technical and business elements that form a path between those with needs and those with capabilities and that permit service providers and consumers to interact.

Framework

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

Idempotency/Idempotent

> A characteristic of a service whereby multiple attempts to change a state will always and only generate a single change of state if the operation has been already been successfully completed once.

Information model

> The characterization of the information that is associated with the use of a service.

808 Interaction

809 The activity involved in making using of a capability offered, usually across an ownership
810 boundary, in order to achieve a particular desired real-world effect.

811 Interface

812 The means by which the underlying capabilities of a service are accessed.

813 Offer

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

816 Opaqueness

817 The extent to which an agent is able to interact successfully with a service without
818 detecting how the service is implemented.

819 Policy

820 A statement of obligations, constraints or other conditions of use of an owned entity as
821 defined by a participant.

822 Process Model

823 The characterization of the temporal relationships between actions and events
824 associated with interacting with a service.

825 (Service) Provider

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

827 Reachability

828 The ability of a service consumer and service provider to interact. Reachability is an
829 aspect of visibility.

830 Real world effect

831 The actual result of using a service, rather than merely the capability offered by a service
832 provider

833 Reference Model

834 A reference model is an abstract framework for understanding significant relationships
835 among the entities of some environment that enables the development of specific
836 architectures using consistent standards or specifications supporting that environment.

837 A reference model is based on a small number of unifying concepts. A reference model is
838 not directly tied to any standards, technologies or other concrete implementation details,
839 but it does seek to provide a common semantics that can be used unambiguously across
840 and between different implementations.

841 Semantics

842 A conceptualization of the implied meaning of information, that requires words and/or
843 symbols within a usage context.

844 Semantic Engagement

845 The relationship between an agent and a set of information that depends on a particular
846 interpretation over the information.

847 Service

848 The means by which the needs of a consumer are brought together with the capabilities
849 of a provider.

850   Service description

851   The information needed in order to use, or consider using, a service.

852   Service Oriented Architecture (SOA)

853   A software architecture of services, policies, practices and frameworks in which
854   components can be reused and repurposed rapidly in order to achieve shared and new
855   functionality. It provides a uniform means to offer, discover, interact with and use
856   capabilities to produce desired effects consistent with measurable preconditions and
857   expectations.

858   Visibility

859   The capacity for those with needs and those with capabilities to be able to interact with
860   each other.

861   Willingness

862   A predisposition of service providers and consumers to interact.

Appendix B. Acknowledgments

The following individuals were members of the committee during the development of this specification:

[TODO: insert cte. Members]

Appendix C. Notices

868

869 OASIS takes no position regarding the validity or scope of any intellectual property or other rights
870 that might be claimed to pertain to the implementation or use of the technology described in this
871 document or the extent to which any license under such rights might or might not be available;
872 neither does it represent that it has made any effort to identify any such rights. Information on
873 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS
874 website. Copies of claims of rights made available for publication and any assurances of licenses
875 to be made available, or the result of an attempt made to obtain a general license or permission
876 for the use of such proprietary rights by implementers or users of this specification, can be
877 obtained from the OASIS Executive Director.

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

881 Copyright © OASIS Open 2005. *All Rights Reserved.*

882 This document and translations of it may be copied and furnished to others, and derivative works
883 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
884 published and distributed, in whole or in part, without restriction of any kind, provided that the
885 above copyright notice and this paragraph are included on all such copies and derivative works.
886 However, this document itself does not be modified in any way, such as by removing the
887 copyright notice or references to OASIS, except as needed for the purpose of developing OASIS
888 specifications, in which case the procedures for copyrights defined in the OASIS Intellectual
889 Property Rights document must be followed, or as required to translate it into languages other
890 than English.

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

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