BS ISO/IEC 18384-3:2016

BSI Standards Publication

Information technology — Reference Architecture for Service Oriented Architecture (SOA RA)

Part 3: Service Oriented Architecture ontology

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 18384-3:2016

National foreword

This British Standard is the UK implementation of ISO/IEC 18384-3:2016.

The UK participation in its preparation was entrusted to Technical Committee IST/38, Cloud Computing and Distributed Platforms.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2016.
Published by BSI Standards Limited 2016

ISBN 978 0 580 81295 8

ICS 35.100.05

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 July 2016.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 18384-3:2016

INTERNATIONAL STANDARD

ISO/IEC 18384-3

First edition
2016-07-01

Information technology — Reference Architecture for Service Oriented Architecture (SOA RA) —

Part 3: Service Oriented Architecture ontology

Technologie de l'information — Architecture de référence pour l'architecture orientée service (SOA RA) —

Partie 3: Ontologie de l'architecture orientée service

Reference number
ISO/IEC 18384-3:2016(E)

AVA Documents
Document Sharing Hub
© ISO/IEC 2016

Full document details: BS ISO/IEC 18384-3:2016

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2016, Published in Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub

Full document details: BS ISO/IEC 18384-3:2016

Contents

Page

Foreword ... **vi**

Introduction ... **vii**

1 Scope ... **1**

2 Normative references ... **1**

3 Terms, definitions and abbreviated terms ... **1**
3.1 Terms and definitions ... 1
3.2 Abbreviated terms ... 1

4 Notations ... **2**

5 Conventions ... **2**

6 Conformance ... **2**

7 SOA Ontology Overview ... **3**
7.1 At a Glance ... 3
7.2 Intended Use ... 5
7.3 Applications ... 5

8 System and Element ... **5**
8.1 Overview ... 5
8.2 The Element Class ... 6
8.3 The uses and usedBy Properties ... 6
8.4 Element — Organizational Example ... 7
8.5 The System Class ... 7
8.6 System — Examples ... 8
8.6.1 Organizational Example ... 8
8.6.2 Service composition Example ... 8
8.6.3 Car wash Example ... 8
8.7 The represents and representedBy Properties ... 9
8.8 The represents and representedBy Examples ... 10
8.8.1 Organizational Example ... 10
8.8.2 Car Wash Example ... 10

9 HumanActor and Task ... **11**
9.1 Overview ... 11
9.2 The HumanActor Class ... 11
9.3 HumanActor — Examples ... 12
9.3.1 The uses and usedBy Properties Applied to HumanActor ... 12
9.3.2 The represents and representedBy Properties Applied to HumanActor ... 12
9.3.3 Organizational Example ... 12
9.3.4 Car Wash Example ... 13
9.4 The Task Class ... 13
9.5 The does and doneBy Properties ... 13
9.6 Task — Examples ... 14
9.6.1 The uses and usedBy Properties Applied to Task ... 14
9.6.2 The represents and representedBy Properties Applied to Task ... 14
9.6.3 Organizational Example ... 14
9.6.4 Car Wash Example ... 15

10 Service, ServiceContract, and ServiceInterface ... **15**
10.1 Overview ... 15
10.2 The Service Class ... 16
10.3 The performs and performedBy Properties ... 16
10.4 Service Consumers and Service Providers ... 17
10.5 Service — Examples ... 17
10.5.1 The uses and usedBy properties Applied to Service ... 17

AVA Documents
Document Sharing Hub

© ISO/IEC 2016 – All rights reserved

Full document details: BS ISO/IEC 18384-3:2016

10.5.2 The represents and representedBy Properties Applied to Service 18
10.5.3 Exemplifying the Difference Between Doing a Task and Performing a Service 18
10.5.4 Car Wash Example 18
10.6 The ServiceContract Class 18
10.7 The interactionAspect and legalAspect Datatype Properties 19
10.8 The hasContract and isContractFor Properties 20
10.9 The involvesParty and isPartyTo Properties 20
10.10 The Effect Class 21
10.11 The specifies and isSpecifiedBy Properties 22
10.12 ServiceContract — Examples 22
10.12.1 Service-level Agreements 22
10.12.2 Service Sourcing 23
10.12.3 Car Wash Example 23
10.13 The ServiceInterface Class 23
10.14 The Constraints Datatype Property 24
10.15 The hasInterface and isInterfaceOf Properties 25
10.16 The InformationType Class 25
10.17 The hasInput and isInputAt Properties 26
10.18 The hasOutput and isOutputAt Properties 26
10.19 Examples 26
10.19.1 Interaction Sequencing 26
10.19.2 Car wash example 27

11 Composition and its Subclasses 27
11.1 Overview 27
11.2 The Composition Class 27
11.3 The compositionPattern Datatype Property 28
11.3.1 Overview 28
11.3.2 The Orchestration Composition Pattern 29
11.3.3 The Choreography Composition Pattern 29
11.3.4 The Collaboration Composition Pattern 29
11.4 The orchestrates and orchestratedBy Properties 31
11.5 The ServiceComposition Class 32
11.6 The Process Class 32
11.7 Service Composition and Process Examples 33
11.7.1 Simple Service Composition Example 33
11.7.2 Process Example 33
11.7.3 Process and Service Composition Example 34
11.7.4 Car Wash Example 34

12 Policy 34
12.1 Overview 34
12.2 The Policy Class 34
12.3 The appliesTo and isSubjectTo Properties 35
12.4 The setsPolicy and isSetBy Properties 35
12.5 Examples 36
12.5.1 Car Wash Example 36

13 Event 36
13.1 Overview 36
13.2 The Event Class 36
13.3 The generates and generatedBy Properties 37
13.4 The respondsTo and respondedToBy Properties 37

Annex A (informative) **Complete Car Wash Example 39**

Annex B (informative) **Internet Purchase Example 44**

Annex C (normative) **The OWL Definition of the SOA Ontology 46**

Annex D (informative) **Class Relationship Matrix 55**

Documents
Document Sharing Hub

© ISO/IEC 2016 – All rights reserved

Full document details: BS ISO/IEC 18384-3:2016

Annex E (informative) **Terms Mapping Between the SOA RA Parts** **59**
Bibliography **74**

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword — Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 38, *Cloud Computing and Distributed Platforms*.

ISO/IEC 18384 consists of the following parts, under the general title *Reference Architecture for Service Oriented Architecture (SOA RA)*:

— *Part 1: Terminology and concepts for SOA*

— *Part 2: Reference Architecture for SOA Solutions*

— *Part 3: Service Oriented Architecture Ontology*

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

Introduction

Service oriented architecture (SOA) is an architectural style in which business and IT systems are designed in terms of services available at an interface and the outcomes of these services. A service is a logical representation of a set of activities that has specified outcomes, is self-contained, it may be composed of other services but consumers of the service need not be aware of any internal structure.

SOA takes "service" as its basic element to constitute and integrate information systems so that they are suitable for a variety of solution requirements. SOA enables interactions between businesses without needing to specify aspects of any particular business domain. Using the SOA architectural style can improve the efficiency of developing information systems and integrating and reusing IT resources. In addition, using the SOA architectural style can help enable rapid response of information systems to ever-changing business needs.

This International Standard is intended to be a single set of SOA technical principles, specific norms, and standards for the world-wide market to help remove confusion about SOA and improve the standardization and quality of solutions.

This International Standard defines the terminology, technical principles, reference architecture and the ontology for SOA. ISO/IEC 18384 can be used to introduce SOA concepts, as a guide to the development and management of SOA solutions, as well as be referenced by business and industry standards.

This International Standard contains three parts:

1) ISO/IEC 18384-1 which defines the terminology, basic technical principles and concepts for SOA.

2) ISO/IEC 18384-2 which defines the detailed SOA reference architecture layers, including a metamodel, capabilities, architectural building blocks, as well as types of services in SOA solutions.

3) ISO/IEC 18384-3 which defines the core concepts of SOA and their relationships in the Ontology.

The targeted audience of this International Standard includes, but is not limited to, standards organizations, architects, architecture methodologists, system and software designers, business people, SOA service providers, SOA solution and service developers, and SOA service consumers who are interested in adopting and developing SOA.

Users of this International Standard will find it useful to read ISO/IEC 18384-1 for an understanding of SOA basics. ISO/IEC 18384-1 should be read before reading or applying ISO/IEC 18384-2. For those new to the SOA reference architecture in ISO/IEC 18384-2:2016, Clause 4 provides a high level understanding of the reference architecture for SOA solutions. The remaining clauses provide comprehensive details of the architectural building blocks and tradeoffs needed for a SOA Solution. This part of ISO/IEC 18384 contains the SOA Ontology, which is a formalism of the core concepts and terminology of SOA, with mappings to both UML and OWL. The SOA Ontology can be used independent of or in conjunction with ISO/IEC 18384-1 and ISO/IEC 18384-2.

The purpose of this part of ISO/IEC 18384 is to contribute to developing and fostering common understanding of service-oriented architecture (SOA) in order to improve alignment between the business and information technology communities and facilitate SOA adoption.

The SOA Ontology defines the concepts, terminology, and semantics of SOA in both business and technical terms, in order to

— create a foundation for further work in domain-specific areas,

— enable communications between business and technical people,

— enhance the understanding of SOA concepts in the business and technical communities,

— provide a means to state problems and opportunities clearly and unambiguously to promote mutual understanding, and

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

— provide a starting point for model-driven development of SOA solutions.

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

INTERNATIONAL STANDARD ISO/IEC 18384-3:2016(E)

Information technology — Reference Architecture for Service Oriented Architecture (SOA RA) —

Part 3: Service Oriented Architecture ontology

1 Scope

This part of ISO/IEC 18384 defines a formal ontology for service-oriented architecture (SOA), an architectural style that supports service orientation. The terms defined in this ontology are key terms from the vocabulary in ISO/IEC 18384-1.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 18384-1, *Information technology — Reference Architecture for Service Oriented Architecture (SOA RA) — Part 1 Terminology and concepts for SOA*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 18384-1 and the following apply.

3.1.1
opaque
having no internal structure that is visible to an external observer

3.1.2
ontology
model that represents a domain and is used to reason about the objects in that domain and the relations between them

Note 1 to entry: This part of ISO/IEC 18384 is high level and not meant to be used for formal reasoning.

[SOURCE: ISO/IEC/TR 24800-1:2007, 2.1.9]

3.2 Abbreviated terms

For the purposes of this document, the following abbreviated terms apply.

ABB	Architecture Building Block
BPMN	Business Process Model and Notation
EA	Enterprise Architecture
ESB	Enterprise Service Bus
IT	Information Technology

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

OWL	Web Ontology Language
RA	Reference Architecture
RDF	Resource Definition Framework
SLA	Service Level Agreement
SOA	Service Oriented Architecture
UML	Unified Modeling Language

4 Notations

The ontology is represented in the web ontology language (OWL) defined by the World Wide Web Consortium. OWL has three increasingly expressive sub-languages: OWL-Lite, OWL-DL, and OWL-Full (see Reference [10] for a definition of these three dialects of OWL). This ontology uses OWL-DL, the sub-language that provides the greatest expressiveness possible while retaining computational completeness and decidability.

The ontology contains classes and properties corresponding to the concepts of SOA. The formal OWL definitions are supplemented by natural language descriptions of the concepts, with graphic illustrations of the relations between them, and with examples of their use. For purposes of exposition, the ontology also includes UML (see Reference [8]) diagrams that graphically illustrate its classes and properties of the ontology. The natural language and OWL definitions contained in this part of ISO/IEC 18384 constitute the authoritative definition of the ontology; the diagrams are for explanatory purposes only. Some of the natural language terms used to describe the concepts are not formally represented in the ontology; those terms are meant in their natural language sense.

The availability of an OWL expression a standard RDF format allows easy loading into tools for architects and developers and allows validation.

This part of ISO/IEC 18384 uses examples to illustrate the ontology. One of these, the car-wash example, is used consistently throughout to illustrate the main concepts (see Annex A for the complete example). Other examples are used ad hoc in individual clauses to illustrate particular points.

5 Conventions

Bold font is used for OWL class, property, and instance names where they appear in clause text.

Italic strings are used for emphasis and to identify the first instance of a word requiring definition.

OWL definitions and syntax are shown in fixed-width font.

An unlabeled arrow in the illustrative UML diagrams means subclass.

The examples in this part of ISO/IEC 18384 are strictly informative and are for illustrative purposes.

6 Conformance

ISO/IEC 18384 contains three parts which have different conformance requirements:

1. terminology and concepts — conformance only to terms and adherence to the semantics in the definitions;
2. reference architecture for SOA solutions — conformance only to semantics of the metamodel and any Layers, ABBs, or capabilities that are used;
3. SOA Ontology — conformance for OWL or non-OWL applications.

Conformance to this part of ISO/IEC 18384 is defined as follows.

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

There are two kinds of applications that may conform to this ontology. One is the OWL-based ontologies (typically extensions of the SOA ontology); the other is a non-OWL application, such as a meta-model or a piece of software (see Clause 2 for the OWL version that is required).

A conforming OWL application (derived OWL-based ontology)

— shall conform to the OWL standard specified in Clause 2,

— shall include the whole of the ontology contained in Annex C,

— may add other OWL constructs, including class and property definitions, and

— may import other ontologies in addition to the SOA ontology.

This part of ISO/IEC 18384 does not use any OWL 2 (see Reference [15]) constructs; however, conforming applications may choose to use OWL or OWL 2.

A conforming non-OWL application

— shall include a defined and consistent transformation (at least semantic mapping) to a non-trivial subset of the ontology contained in Annex C,

— may add other constructs, including class and property definitions, and

— may import and/or use other ontologies in addition to the SOA ontology.

7 SOA Ontology Overview

7.1 At a Glance

A graphically compressed visualization of the entire ontology is shown in Figure 1.

The concepts illustrated in Figure 1 are described in the body.

This part of ISO/IEC 18384 starts by explaining the most basic foundational concept of elements and systems followed by explaining the elements of SOA human actor and task and then service concepts and descriptions and contracts for services and building on that to explain compositions of services. Finally, this part of ISO/IEC 18384 wraps up with Policies and Events which are relevant to all of the elements of SOA.

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

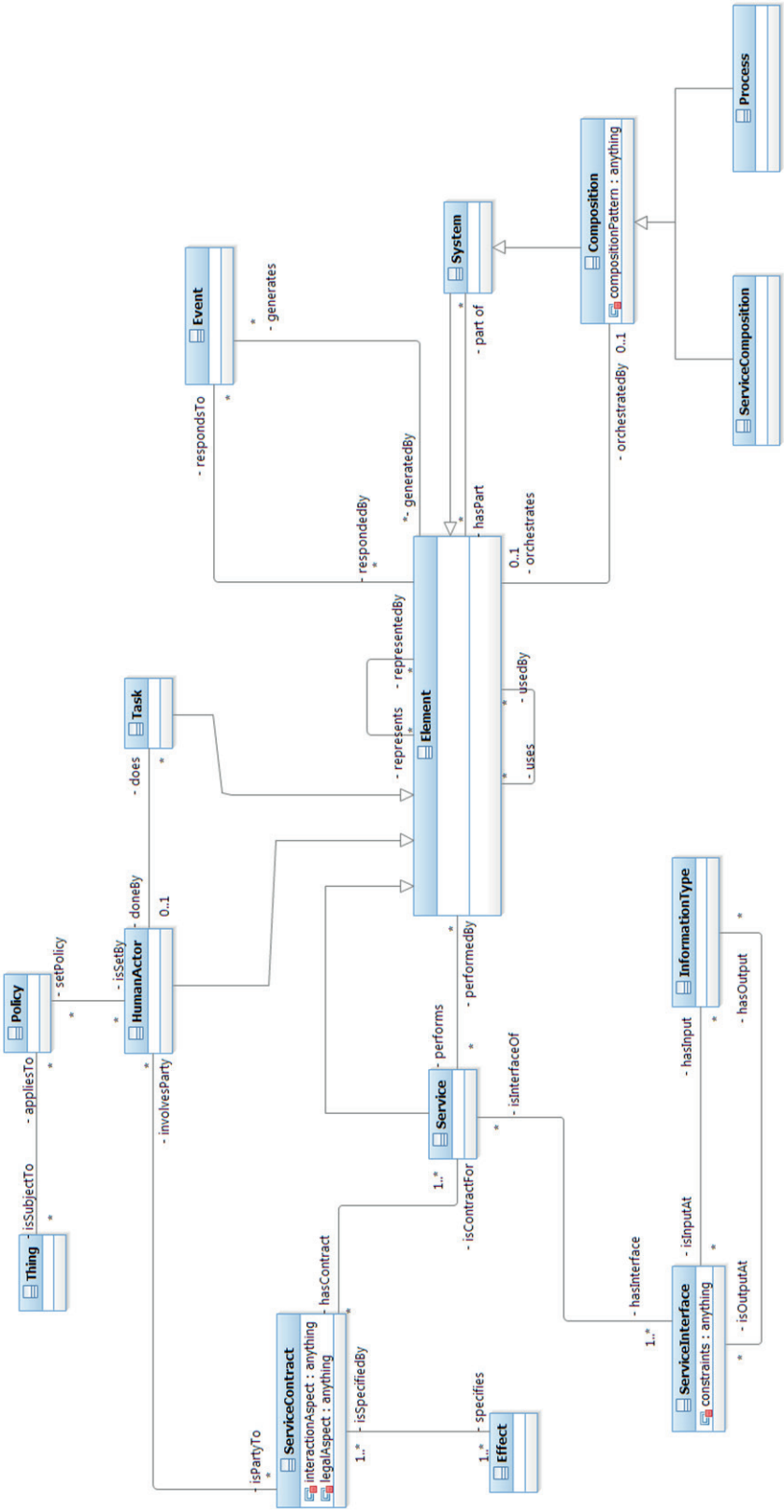

Figure 1 — SOA Ontology — Graphical Overview

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

7.2 Intended Use

This Clause describes caveats and assumptions for how this ontology should be interpreted.

— This ontology is intended for high level representation of concepts and is not intended for formal reasoning.

— This part of ISO/IEC 18384 is designed for use by business people, architects and systems and software designers to enable communications between business and technical people.

— This part of ISO/IEC 18384 focuses on a minimal set of SOA terms, modelling those terms in detail.

— This part of ISO/IEC 18384 explains relationships to other important concepts, but not at the same level of detail as the SOA terms. For example, policy is modelled, but not in great detail.

— This part of ISO/IEC 18384 restricts itself to OWL constructs, not using those introduced in OWL 2 (see Reference [15]), because the OWL constructs are sufficient for the scope of this part of ISO/IEC 18384. It is consistent with OWL 2 and does not preclude others from using it with OWL 2.

— This part of ISO/IEC 18384 elaborates on the SOA terms and relationships in ISO/IEC 18384-1 and ISO/IEC 18384-2. A separate metamodel in ISO/IEC 18384-2 provides the basis for the modeling in ISO/IEC 18384-2 and is used to describe and understand the reference architecture.

— This part of ISO/IEC 18384 defines the concepts, terminology, and semantics of SOA in both business and technical terms, in order to create a foundation for further work in domain-specific areas.

— This part of ISO/IEC 18384 provides a means to state problems and opportunities clearly and unambiguously to promote mutual understanding.

— This part of ISO/IEC 18384 may provide a starting point for model-driven development of SOA solutions.

7.3 Applications

The SOA ontology was developed in order to aid understanding and can simply be read.

It can also be used as a starting point for model-driven development, by applying it to particular usage domains and applications.

The ontology is applied to a particular usage domain by adding SOA OWL class instances of things in that domain. This is sometimes referred to as “populating the ontology.” In addition, an application can add definitions of new classes and properties, can import other ontologies, and can import the ontology OWL representation into other ontologies.

The ontology defines the relations between terms, but does not prescribe exactly how they should be applied. For explanations of what ontologies are and why they are needed, see References [11] and [14]. The examples provided in this part of ISO/IEC 18384 are describing one way in which the ontology could be applied in practical situations. Different applications of the ontology to the same situations would nevertheless be possible. The precise instantiation of the ontology in particular practical situations is a matter for users of the ontology, as long as the concepts and constraints defined by the ontology are correctly applied, the instantiation is valid.

8 System and Element

8.1 Overview

System and *element* are two of the concepts of this ontology. Both are concepts that are often used by practitioners, including the notion that systems have elements and that systems can be hierarchically combined (systems of systems). What differs from domain to domain is the specific nature of systems and elements, for instance, an electrical system has very different kinds of elements than an SOA system.

© ISO/IEC 2016 – All rights reserved

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

In the ontology, only elements and systems within the SOA domain are considered. Some SOA sub-domains use the term *component* rather than the term element. This is not contradictory, as any component of an SOA system is also an element of that (composite) system.

This Clause describes the following classes of the ontology:

Element

System

In addition, it defines the following properties:

uses and **usedBy**

represents and **representedBy**

8.2 The Element Class

```
<owl:Class rdf:about="#Element">
</owl:Class>
```

An *element* is an entity that is opaque and indivisible at a given level of abstraction. The element has a clearly defined boundary. The concept of element is captured by the **Element** OWL class, which is illustrated in Figure 2.

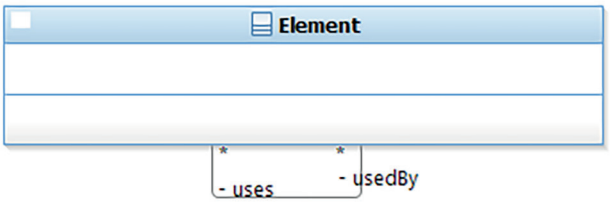

Figure 2 — The Element Class

In the context of the SOA ontology, only functional elements that belong to the SOA domain are considered in detail. There are other kinds of Elements than members of the four named subclasses (System, HumanActor, Task, and Service) described later in this ontology. Examples of such other kinds of Elements are things like software components or technology components (such as Enterprise Service Bus (ESB) implementations, etc.).

8.3 The uses and usedBy Properties

```
<owl:ObjectProperty rdf:about="#uses">
  <rdfs:domain rdf:resource="#Element"/>
  <rdfs:range rdf:resource="#Element"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#usedBy">
  <owl:inverseOf>
   <owl:ObjectProperty rdf:about="#uses"/>
  </owl:inverseOf>
</owl:ObjectProperty>
```

Elements may use other elements in various ways. In general, the notion of some element using another element is applied by practitioners for all of models, executables, and physical objects. What differs from domain to domain is the way in which such use is perceived.

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

An element uses another element if it interacts with it in some fashion. Interacts here is interpreted very broadly ranging through, for example, an element simply being a member of (used by) some system (see later for a formal definition of the **System** class), an element interacting with (using) another element (such as a service; see later for a formal definition of the **Service** class) in an *ad hoc* fashion, or even a strongly coupled dependency in a composition (see later for a formal definition of the **Composition** class). The **uses** property, and its inverse **usedBy**, capture the abstract notion of an element using another. These properties capture not just transient relations. Instantiations of the property can include "uses at this instant", "has used", and "may in future use".

For the purposes of this ontology, the multitude of different possible semantics of a *uses* relationship is not enumerated and formally defined .The semantic interpretations are left to a particular sub-domain, application or even design approach.

8.4 Element — Organizational Example

Using an organizational example, typical instances of **Element** are organizational units and people. Whether to perceive a given part of an organization as an organizational unit or as the set of people within that organizational unit is an important choice of abstraction level.

Inside the boundary of the organizational unit, as the organizational unit can in fact use the people that are members of it. Note that the same person can in fact be a member of (be used by) multiple organizational units.

Outside the boundary the internal structure of an organizational unit remains opaque to an external observer, as the enterprise wants to be able to change the people within the organizational unit without having to change the definition of the organizational unit itself.

This simple example expresses that some elements have an internal structure. In fact, from an internal perspective they are an organized collection of other simpler things (captured by the System class defined in 8.5).

8.5 The System Class

```
<owl:Class rdf:about="#System">
  <owl:disjointWith>
   <owl:Class rdf:about="#Task"/>
  </owl:disjointWith>
  <owl:disjointWith>
   <owl:Class rdf:about="#Service"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
   <owl:Class rdf:about="#Element"/>
  </rdfs:subClassOf>
</owl:Class>
```

A *system* is an organized collection of other things. Specifically, things in a system collection are instances of **Element**, each such instance being used by the system. The concept of *system* is captured by the **System** OWL class, which is illustrated in Figure 3.

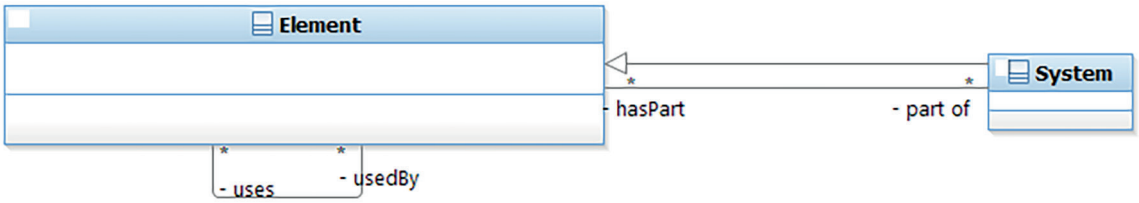

Figure 3 — The System Class

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

This definition of System is heavily influenced by ISO/IEC 42010:2011 (see Reference [13]).

In the context of the SOA ontology, only functional systems that belong to the SOA domain are considered in detail. Note that a fully described instance of **System** should have by its nature (as a collection) a *part of* relationship to at least one instance of **Element**.

Since System is a subclass of **Element**, all systems have a boundary and are opaque to an external observer (black box view). This excludes from the **System** class structures that have no defined boundary. From an SOA perspective, this is not really a loss since all interesting SOA systems do have the characteristic of being possible to perceive from an outside (consumer) perspective. Furthermore, having **System** as a subclass of **Element** allows us to naturally express the notion of systems of systems — the lower-level systems are simply elements used by the higher level system.

At the same time as supporting an external view point (black box view), all systems also support an internal view point (white box view) expressing how they are an organized collection. As an example, for the notion of a service this would typically correspond to a service specification view *versus* a service realization view (similar to the way that SoaML[9] defines services as having both a black box/specification part and a white box/realization part).

It is important to realize that even though systems using elements express an important aspect of the **uses** property, it is not necessary to "invent" a system just to express that some element uses another. In fact, even for systems it may be necessary to to express that they can use elements outside their own boundary — though this in many cases will preferably be expressed not at the system level, but rather by an element of the system using that external **Element** instance.

System is defined as disjoint with the *Service* and *Task* classes. Instances of these classes are considered not to be collections of other things. **System** is specifically not defined as disjoint with the *HumanActor* class since an organization is many cases in fact just a particular kind of system. A special intersection class to represent this fact is not defined.

8.6 System — Examples

8.6.1 Organizational Example

Continuing the organizational example from 8.5, an organizational unit can now be expressed as an instance of **System** has the people in it as members (and instances of element).

8.6.2 Service composition Example

Using a service composition example, services A and B are instances of **Element** and the composition of A and B is an instance of **System** (that uses A and B). It is important to realize that the act of composing is different than composition as a thing — it is in the latter sense that the term composition is used here.

See also Clause 11 for a formal definition of the concepts of service and service composition (and a repeat of the example in that more precise context).

8.6.3 Car wash Example

Consider a car wash business. The company as a whole is an organizational unit and can be instantiated in the ontology in the following way:

— *CarWashBusiness* is an instance of **System**.

— *Joe* (the owner) is an instance of **Element** and used by (owner of) *CarWashBusiness*.

— *Mary* (the secretary) is an instance of **Element** and used by (employee of) *CarWashBusiness*.

— *John* (the pre-wash guy) is an instance of **Element** and used by (employee of) *CarWashBusiness*.

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

— *Jack* (the washing manager and operator) is an instance of **Element** and used by (employee of) *CarWashBusiness*.

8.7 The represents and representedBy Properties

```
<owl:ObjectProperty rdf:about="#represents">
  <rdfs:domain rdf:resource="#Element"/>
  <rdfs:range rdf:resource="#Element"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#representedBy">
  <owl:inverseOf>
   <owl:ObjectProperty rdf:about="#represents"/>
  </owl:inverseOf>
</owl:ObjectProperty>
```

The environment described by an SOA is intrinsically hierarchically composite (see also 11.2 for a definition of the *Composition* class); in other words, the elements of SOA systems can be repeatedly composed to ever higher levels of abstraction. One aspect of this has already been addressed by the **uses** and **usedBy** properties in that the notion of systems of systems can be expressed. This is still a very concrete relationship though, and does not express the concept of architectural abstraction. The need for architectural abstraction is found in various places, such as a role representing the people playing that role, an organizational unit representing the people within it (subtly different from that same organizational unit using the people within it, as the **represents** relationship indicates the organizational unit as a substitute interaction point), an architectural building block representing an underlying construct (for instance, important to enterprise architects wanting to explicitly distinguish between constructs and building blocks), and an enterprise service bus (ESB) representing the services that are accessible through it (for instance, relevant when explicitly modelling operational interaction and dependencies). The concept of such an explicitly changing view point, or level of abstraction, is captured by the *represents* and *representedBy* properties illustrated in Figure 4.

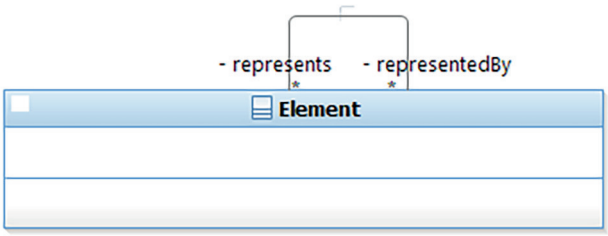

Figure 4 — The represents and representedBy Properties

It is important to understand the exact nature of the distinction between using an element (E1) and using another element (E2) that represents E1. If E1 changes, then anyone using E1 directly would experience a change, but someone using E2 would not experience any change.

When applying the architectural abstraction via the **represents** property there are three different architectural choices that can be made:

An element represents another element in a very literal way, simply by hiding the existence of that element and any changes to it. There will be a one-to-one relationship between the instance of **Element** and the (different) instance of **Element** that it represents. A simple real-world example is the notion of a broker acting as an intermediary between a seller (that does not wish to be known) and a buyer.

An element represents a particular aspect of another element. There will be a many-to-one relationship between many instances of **Element** (each of which represents a different aspect), and one (different)

© ISO/IEC 2016 – All rights reserved

AVA Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

instance of **Element**. A simple real-world example is the notion that the same person can play (be represented by) many different roles.

An element is an abstraction that can represent many other elements. There will be a one-to-many relationship between one instance of **Element** (as an abstraction) and many other instances of **Element**. A simple real-world example is the notion of an architectural blueprint representing an abstraction of many different buildings being built according to that blueprint.

Note that in most cases, an instance of **Element** will represent only one kind of thing. Specifically, an instance of **Element** will typically represent instances of at most one of the classes **System**, **Service**, Human **Actor**, and **Task** (with the exception of the case where the same thing is both an instance of **System** and an instance of **Actor**). See later clauses for the definitions of **Service**, Human **Actor**, and **Task**.

8.8 The represents and representedBy Examples

8.8.1 Organizational Example

Expanding further on the organizational example, assume that a company desires to form a new organizational unit O1. There are two ways of doing this.

Define the new organization directly as a collection of people P1, P2, P3, and P4. This means that the new organization is perceived to be a leaf in the organizational hierarchy, and that any exchange of personnel means that its definition needs to change.

Define the new organization as a higher-level organizational construct, joining together two existing organizations O3 and O4. Coincidentally, O3 and O4 between them may have the same four people P1, P2, P3, and P4, but the new organization really doesn't know, and any member of O3 or O4 can be changed without needing to change the definition of the new organization. Furthermore, any member of O3 is intrinsically *not* working in the same organization as the members of O4 (in fact need not even be aware of them) — contrary to the first option where P1, P2, P3, and P4 are all colleagues in the same new organization.

In this way, the abstraction aspect of the **represents** property induces an important difference in the semantics of the collection defining the new organization. Any instantiation of the ontology can and should use the **represents** and **representedBy** properties to crisply define the implied semantics and lines of visibility/change.

8.8.2 Car Wash Example

Joe chooses to organize his business into two organizational units, one for the administration and one for the actual washing of cars. This can be instantiated in the ontology in the following way:

— *CarWashBusiness* is an instance of **System**.

— *AdministrativeSystem* is an instance of **System**.

— *Administration* is an instance of **Element** that represents *AdministrativeSystem* (the organizational unit aspect is opaque, *aka* ignoring anything else about *AdministrativeSystem*).

— *CarwashBusiness* uses (has organizational unit) *Administration*.

— *CarWashSystem* is an instance of **System**.

— *CarWash* is an instance of **Element** that represents *CarWashSystem* (the organizational unit aspect is opaque, *aka* ignoring anything else about *CarWashSystem*).

— *CarWash* is a member of *CarWashBusiness*.

— *Joe* (the owner) is an instance of **Element** and now used by *AdministrationSystem*.

— *Mary* (the secretary) is an instance of **Element** and now used by *AdministrationSystem*.

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

— *John* (the pre-wash guy) is an instance of **Element** and now used by *CarWashSystem*.

— *Jack* (the wash manager and operator) is an instance of **Element** and now used by *CarWashSystem*.

9 HumanActor and Task

9.1 Overview

People, organizations, and the things they do are important aspects of SOA systems. *HumanActor* and *Task* capture this as another set of core concepts of the ontology. Both are concepts that are generic and have relevance outside the domain of SOA. For the purposes of this SOA ontology, specific scope is given in that tasks are intrinsically atomic [corresponding to, for instance, the business process modeling notation (BPMN) 2.0 definition of Task (see Reference [4])] and human actors are restricted to people and organizations.

This Clause describes the following classes of the ontology:

HumanActor

Task

In addition, it defines the following properties:

does and **doneBy**

9.2 The HumanActor Class

```
<owl:Class rdf:about="#HumanActor">
  <rdfs:subClassOf>
   <owl:Class rdf:about="#Element"/>
  </rdfs:subClassOf>
  <owl:disjointWith>
   <owl:Class rdf:about="#Task"/>
  </owl:disjointWith>
  <owl:disjointWith>
  <owl:Class rdf:about="#Service"/>
   </owl:disjointWith>
  </owl:Class>

<owl:Class rdf:about="#ServiceContract"/>
  </owl:disjointWith>
  <owl:disjointWith>
   <owl:Class rdf:about="#ServiceInterface"/>
  </owl:disjointWith>
</owl:Class>
```

A *human actor* is a person or an organization. The concept of *human actor* is captured by the **HumanActor** OWL class, which is illustrated in Figure 5.

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

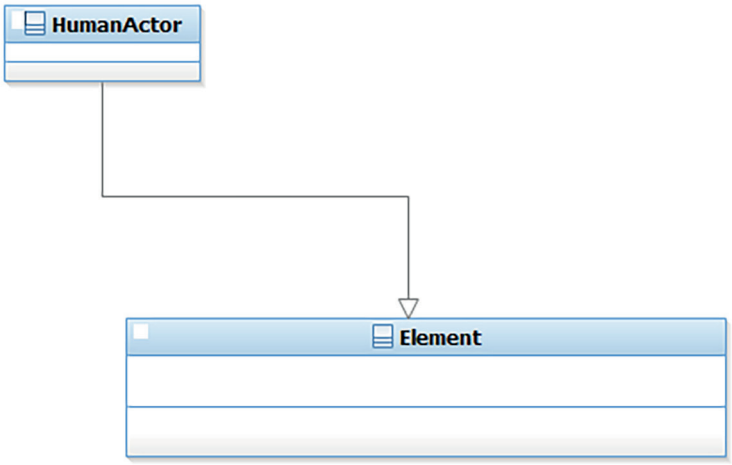

Figure 5 — The HumanActor Class

HumanActor is defined as disjoint with the *Service* and *Task* classes. Instances of these classes are considered not to be people or organizations. **HumanActor** is specifically not defined as disjoint with **System** since an organization in many cases is in fact just a particular kind of system. A special intersection class to represent this fact is not defined.

9.3 HumanActor — Examples

9.3.1 The uses and usedBy Properties Applied to HumanActor

In one direction, a human actor can itself use things such as services, systems, and other human actors. In the other direction, a human actor can, for instance, be used by another human actor or by a system (as an element within that system such as a human actor in a process).

9.3.2 The represents and representedBy Properties Applied to HumanActor

As mentioned in the introduction to this Clause, human actors are intrinsically part of systems that instantiate service oriented architectures. Yet in many cases as an element of an SOA system, the specific person or organization is not discussed, rather an abstract representation of them that participates in processes, provides services, etc. In other words, elements representing human actors are discussed

As examples, a broker (instance of **HumanActor**) may represent a seller (instance of **HumanActor**) that wishes to remain anonymous, a role (instance of **Element**) may represent (the role aspect of) multiple instances of **HumanActor**, and an organizational unit (instance of **HumanActor**) may represent the many people (all instances of **HumanActor**) that are part of it.

Note that a "role class" has not been defined, as using **Element** with the **represents** property is a more general approach which does not limit the ability to also define role-based systems. For all practical purposes there is simply a "role subclass" of **Element**, a subclass that is not defined explicitly.

9.3.3 Organizational Example

Continuing the organizational example from 8.8.1, P1 (*John*), P2 (*Jack*), P3 (*Joe*), and P4 (*Mary*) can now be expressed as instances of **Element** which are in fact (people) instances of **HumanActor**. All of O1 (*CarWashBusiness*), O3 (*CarWash*), and O4 (*Administration*) can also be expressed as (organization) human actors from an action perspective at the same time that they are systems from a collection/composition perspective.

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

9.3.4 Car Wash Example

See Annex A for the complete organizational aspect of the car wash example.

9.4 The Task Class

```
<owl:Class rdf:about="#Task">
  <owl:disjointWith>
   <owl:Class rdf:about="#System"/>
  </owl:disjointWith>
  <owl:disjointWith>
   <owl:Class rdf:about="#HumanActor"/>
  </owl:disjointWith>
  <owl:disjointWith>
   <owl:Class rdf:about="#Service"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
   <owl:Class rdf:about="#Element"/>
  </rdfs:subClassOf>
</owl:Class>
```

A *task* is an atomic action which accomplishes a defined result. Tasks are done by people or organizations, specifically by instances of **HumanActor**.

The Business Process Modeling Notation (BPMN) 2.0 defines task as follows: "A Task is an atomic Activity within a Process flow (see Reference [4]). A Task is used when the work in the Process cannot be broken down to a finer level of detail. Generally, an end-user and/or applications are used to perform the Task when it is executed." For the purposes of the ontology, precision has been added by formally separating the notion of doing from the notion of performing. Tasks are (optionally) done by human actors, furthermore (as instances of Element) tasks can use services that are performed by technology components (see details in 10.3; see also the example in Annex A).

The concept of *task* is captured by the **Task** OWL class, which is illustrated in Figure 6.

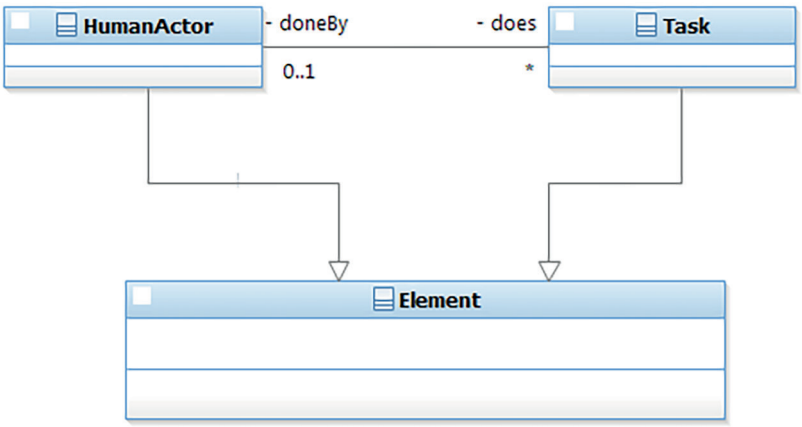

Figure 6 — The Task Class

Task is defined as disjoint with the *System*, *Service*, and *HumanActor* classes. Instances of these classes are considered not to be atomic actions.

9.5 The does and doneBy Properties

```
<owl:ObjectProperty rdf:about="#doneBy">
  <rdfs:domain rdf:resource="#Task"/>
  <rdfs:range rdf:resource="#HumanActor"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#does">
```

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

```
  <owl:inverseOf>
   <owl:ObjectProperty rdf:about="#doneBy"/>
  </owl:inverseOf>
</owl:ObjectProperty>

<owl:Class rdf:about="#Task">
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty>
     <owl:ObjectProperty rdf:about="#doneBy"/>
    </owl:onProperty>
    <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>0</owl:minCardinality>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
>1</owl:maxCardinality>
    <owl:onProperty>
     <owl:ObjectProperty rdf:about="#doneBy"/>
    </owl:onProperty>
   </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Tasks are naturally thought of as being done by people or organizations. If thinking of tasks as being the actual things done, then the natural cardinality is that each instance of **Task** is done by at most one instance of **HumanActor**. Due to the atomic nature of instances of **Task** the case is ruled out where such an instance is done jointly by multiple instances of Human**Actor**. The cardinality can be zero if someone chooses not to instantiate all possible human actors. On the other hand, the same instance of **HumanActor** can (over time) easily do more than one instance of **Task**. The **does** property, and its inverse **doneBy**, capture the relation between a human actor and the tasks it does.

9.6 Task — Examples

9.6.1 The uses and usedBy Properties Applied to Task

In one direction, the most common case of a task using another element is where an automated task (in an orchestrated process; see Clause 11 for the definition of *process* and *orchestration*) uses a service as its realization. In the other direction, a task can, for instance, be used by a system (as an element within that system, such as a task in a process).

9.6.2 The represents and representedBy Properties Applied to Task

As mentioned in the introduction to this Clause, tasks are intrinsically part of SOA systems. Yet in many cases as an element of an SOA system, the actual thing being done is not discussed, rather an abstract representation of it that is used as an element in systems, processes, etc. In other words, discuss elements representing tasks.

As a simple example, an abstract activity in a process model (associated with a role) may represent a concrete task (done by a person fulfilling that role). Note that due to the atomic nature of a task it does not make sense to talk about many elements representing different aspects of it.

9.6.3 Organizational Example

Continuing the organizational example from 8.8.1, the tasks that are done by human actors (people) P1, P2, P3, and P4 can now be expressed, and how those tasks can be elements in bigger systems that describe things such as organizational processes. Clause 11 will deal formally with the concept of *composition*, including properly defining the concept of a *process* as one particular kind of *composition*.

© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

9.6.4 Car Wash Example

As an important part of the car wash system, John and Jack perform certain manual tasks required for washing a car properly.

— *Jack* and *John* are instances of **HumanActor**.

— *WashWindows* is an instance of **Task** and is done by *John*.

— *PushWashButton* is an instance of **Task** and is done by *Jack*.

10 Service, ServiceContract, and ServiceInterface

10.1 Overview

Service is another core concept of this ontology. It is a concept that is fundamental to SOA and always used in practice when describing or engineering SOA systems, yet it is not easy to define formally. The ontology is based on the following definition of *service*:

A service is a "logical representation of a set of activities that has specified outcomes, is self-contained, may be composed of other services, and is a "black box" to consumers of the service"

This corresponds to the existing official definition of the term in the Reference architecture for SOA, ISO/IEC 18384-1.

The word activity in the definition of service is used in the general English language sense of the word, not in the process-specific sense of that same word (i.e. activities are not necessarily process activities). The ontology purposefully omits "business" as an intrinsic part of the definition of *service*. The reason for this is that the notion of business is relative to a person's viewpoint, as an example, one person's notion of IT is another person's notion of business (the business of IT). *Service* as defined by the ontology is agnostic to whether the concept is applied to the classical notion of a business domain or the classical notion of an IT domain.

Other current SOA-specific definitions of the term service include the following:

— *"A mechanism to enable access to 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*." (see Reference [3])

— *"A capability offered by one entity or entities to others using well-defined 'terms and conditions' and interfaces*." (see Reference [9])

Within the normal degree of precision of the English language, these definitions are not contradictory; they are stressing different aspects of the same concept. All three definitions are SOA-specific though, and represent a particular interpretation of the generic English language term service.

This Clause describes the following classes of the ontology:

— **Service**;

— **ServiceContract**;

— **ServiceInterface**;

— **InformationType.**

In addition, it defines the following properties:

— **performs** and **performedBy**;

— **hasContract** and **isContractFor**;

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

— **involvesParty** and **isPartyTo**;

— **specifies** and **isSpecifiedBy**;

— **hasInterface** and **isInterfaceOf**;

— **hasInput** and **isInputAt**;

— **hasOutput** and **isOutputAt**.

10.2 The Service Class

```
<owl:Class rdf:about="#Service">
  <owl:disjointWith>
    <owl:Class rdf:about="#System"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#Task"/>
  </owl:disjointWith>
  <owl:disjointWith>
    <owl:Class rdf:about="#HumanActor"/>
  </owl:disjointWith>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Element"/>
  </rdfs:subClassOf>
</owl:Class>
```

A *service* is a logical representation of a set of activities that has specified outcomes, is self-contained, may be composed of other services, and is a "black box" to consumers of the service. The concept of *service* is captured by the **Service** OWL class, which is illustrated in Figure 7.

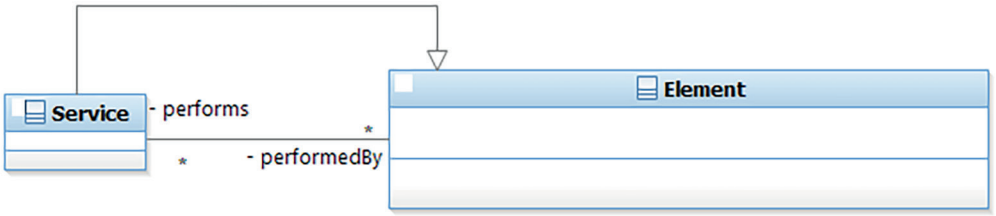

Figure 7 — The Service Class

In the context of the SOA ontology, only SOA-based services are considered. Other domains, such as integrated service management, can have services that are not SOA-based hence are outside the intended scope of the SOA ontology.

Service is defined as disjoint with the *System*, *Task*, and *HumanActor* classes. Instances of these classes are considered not to be services themselves, even though they may provide capabilities that can be offered as services.

10.3 The performs and performedBy Properties

```
<owl:ObjectProperty rdf:about="#performs">
  <rdfs:domain rdf:resource="#Element"/>
  <rdfs:range rdf:resource="#Service"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="#performedBy">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#performs"/>
  </owl:inverseOf>
</owl:ObjectProperty>
```


© ISO/IEC 2016 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016

As a service itself is only a logical representation, any service is *performed* by something. The something that *performs* a service is opaque to anyone interacting with it, an opaqueness which is the exact nature of the **Element** class. This concept is captured by the *performs* and *performedBy* properties as illustrated in the Service Class in Figure 7.

This also captures the fact that services can be performed by elements of other types than systems. This includes elements such as software components, human actors, and tasks.

Note that the same instance of **Service** can be performed by many different instances of **Element**. As long as the service performed is the same, an external observer cannot tell the difference (for contractual obligations, SLAs, etc. see the definition of the *ServiceContract* class in 10.6). Conversely, any instance of **Element** may perform more than one service or none at all.

While a service can be performed by other elements, the service itself (as a purely logical representation) does not perform other services. See the Simple Service Composition Example (11.7.1) for an example of how to represent service compositions formally in the ontology.

10.4 Service Consumers and Service Providers

Terminology used in an SOA environment often includes the notions of service providers and service consumers. There are two challenges with this terminology:

— It does not distinguish between the contractual obligation aspect of consume/provide and the interaction aspect of consume/provide. A contractual obligation does not necessarily translate to an interaction dependency, if for no other reason than because the realization of the contractual obligation may have been sourced to a third party.

— Consuming or providing a service is a statement that only makes sense in context, either a contractual context or an interaction context. These terms are consequently not well suited for making statements about elements and services in isolation.

These are the reasons why the ontology has chosen not to adopt consume and provide as core concepts, rather instead allows consume or provide terms used with contractual obligations and/or interaction rules described by service contracts; see the definition of the *ServiceContract* class in 10.6. In its simplest form, outside the context of a formal service contract, the interaction aspect of consuming and providing services may even be expressed simply by saying that some element uses (consumes) a service or that some element performs (provides) a service; see also the examples in 10.5.

10.5 Service — Examples

10.5.1 The uses and usedBy properties Applied to Service

In one direction, it does not really make sense to talk about a service that uses another element. While the thing that performs the service might very well include the use of other elements (and certainly will in the case of *Service Composition*), the service itself (as a purely logical representation) does not use other elements.

In the other direction, the most common of all interactions is found in an SOA environment: the notion that some element uses a service by interacting with it. Note that from an operational perspective this interaction actually reaches somewhat beyond the service itself by involving the following typical steps:

— picking the service to interact with (this statement is agnostic as to whether this is done dynamically at runtime or statically at design and/or construct time);

— picking an element that performs that service [in a typical SOA environment, this is most often done “inside” an enterprise service bus (ESB)];

— interacting with the chosen element (that performs the chosen) service (often also facilitated by an ESB).

© ISO/IEC 2016 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO/IEC 18384-3:2016