FFI-notat 2013/00131

On the roles and synergies of the service oriented architecture- and the semantic technologies paradigms in an NNEC context

Eli Gjørven and Audun Stolpe

Norwegian Defence Research Establishment (FFI)

16 January 2013

FFI-notat 2013/00131

1176

Keywords

Nettverksbasert forsvar

Semantiske teknologier

Tjenesteorientert arkitektur

Approved by

Rolf Rasmussen	Project manager
Anders Eggen	Director

English summary

This document compares the Service-oriented Architecture (SOA) with Semantic Technologies with regard to applications within the context of NNEC. We attempt to find out to what degree these two paradigms are overlapping, and/or complementary as technologies within one information infrastructure.

The document focus on the basic conceptual differences between the two paradigms, and does not pretend to give a complete description over theory and practice in the respective research communities. Rather, we look at the consequences for potential synergies between SOA and Semantic Technologies based on the intentions of these two paradims.

We argue that SOA and Semantic Technologies are not conflicting, but rather orthogonal. Furthermore, vi discuss how the respective capabilities of the paradigms could be utilized in an information infrastructure that requires predefined, composable, and interoperable SOA services, as well as the analytic capabilties of Semantic Technologies.

We illustrate our propositions with presumably realistic examples from the tactical military domain where the analysis functionality provided by Semantic Technologies, are combined with data collection, processing, and dissemination in a secure way using SOA technology.

Sammendrag

Dette notatet sammenlikner den Tjenesteorienterte Arkitekturen med Semantiske Teknologier med hensyn på anvendelser innenfor en NNEC kontekst. Hensikten er anslå i hvilken grad disse to paradigmene er overlappende og/eller gjensidig utfyllende som teknologier i samme informasjonsinfrastruktur.

Notatet fokuserer på de grunnleggende konseptuelle forskjellene mellom disse to paradigmene, og pretenderer ikke å gi en komplett oversikt over oppfatninger og praksis i de respektive fagmiljøene. Hensikten er snarere å trekke enkelte konsekvenser for det potensielle samspillet mellom Tjenesteorientering og Semantiske Teknologier, basert på hvordan disse paradigmene er tenkt og hva de er designet for.

Vi argumenterer for at Tjenesteorientering og Semantiske Teknologier er ortogonale teknologier. Videre diskuterer vi hvordan samspillet mellom dem fruktbart kan utnyttes i en NNEC-konform infrastruktur som krever både pre-definerte, kombinerbare, og interoperable tjenester, og de analysefunksjonaliteten som semantiske teknologier tilbyr. Vi illustrerer dette med et par presumtivt realistiske eksempler fra det taktiske militære domenet hvor analysefunksjonaliteten som Semantiske Teknologier tilbyr, kombineres med datainnsamling, prosessering og disseminering på en sikker måte med Tjenesteorientert teknologi.

Contents

1 Introduction

1.1 Background

Informed decisions making in a military context—be it at the tactical or strategic level—requires extensive sharing of timely and up-to-date information that induces a high degree of shared situational awareness. This is particularly the case when the military conducts its operations according NATO's Network Enabled Capability concept (henceforth NNEC). The NNEC feasibility study (Booth et al. 2005) states that:

"The NATO Network-Enabled Capability (NNEC) will be composed of a dynamic networked coalition of military forces; cooperatively sharing information to progressively improve coordination, collaboration and coherency across the full spectrum of military activity."

The realization of NNEC depends on a common Communication and Information System (CIS) infrastructure, called Networking and Information Infrastructure (NII). The NII must facilitate interaction between the technical solutions deployed by different military forces, constituting different administrative domains. This type of cross-domain interaction requires the NII to satisfy the following two requirements:

1. The NII must enable both software functions and data to be discovered and accessed across domains and networks.

2. The NII must facilitate re-usability through static and/or dynamic composition and integration of existing software functions and data.

Two different integration paradigms, well-known from the civil world, have continued to receive attention as means to fulfill different aspects of these requirements, namely Service Oriented Architectures (SOA) and Semantic Technologies (ST).

1.2 Perspective of this Note

While SOA is focused on making software functions available through standardized interfaces suitable for remote access, composition, and reuse, Semantic Technologies aim to make data from different domains available and useful through formal information models, automated reasoning, and standardized query languages.

The purpose of this note is to compare the above mentioned paradigms with an eye to areas of overlap, conflict, and potential synergy. We have deliberately preferred clarity over completeness, that is, we have chosen to try to draw a few instructive contrasts, rather than to give a complete survey of opinions and practices.

More specifically, we have chosen to let the document be guided principally by the fundamental *conceptual* difference between SOA and semantics, which we consider to be that of *service-orientedness* vs. *data-orientedness*. These principles may be said to capture the key intentions behind each paradigm, and to indicate to which uses each is most naturally put.

In this note we argue that it ought to be possible to utilize the respective capabilities of the SOA and ST paradigms in an information infrastructure that requires predefined, composable, and interoperable SOA services, as well as the analytic capabilities of ST. In such an infrastructure one would for example include semantic technologies for data integration, analysis and interpretation—possibly on the fly—into SOA workflows consisting of services performing data collection, processing, and transportation.

This document is organized as follows: Key ideas behind the SOA and ST paradigms are presented in Section 2 and 3 respectively. Each section ends with a list of general features that is meant to highlight some important consequences of applying the principles in question. This material in turn feeds into Section 4, which is a brief point for point contrasting of general characteristics and natural application areas. In Section 5 we discuss how these contrasts open for potential synergies, illustrated by applications where SOA and ST components fulfill complementary functional roles.

These thought experiments are sufficiently realistic, we maintain, to serve as evidence that the union of the two aforementioned paradigms satisfy requirements 1) and 2) above, and that they are both key enablers for the realization of the NNEC concept.

2 Service-oriented Architectures

In this section, we present the main concepts and principles of SOA with definitions that we find useful for the discussions found later in this document. The discussion below is based on the Service-oriented Architecture reference model (OASIS 2006*a*), the W3C glossary (W3C 2004*b*), and (Erl 2005) as widely accepted authorities in the SOA literature. We also briefly describe current SOA technologies.

2.1 Service Orientation

In (Erl 2005) service orientation is described as follows:

Service orientation presents an ideal vision of a world in which resources are cleanly partitioned and consistently represented. (...) By adhering to this vision, past technical and philosophical disparities are blanketed by layers of abstraction that introduce a globally accepted standard for representing logic and information.

As the term “service oriented” implies, from a SOA perspective this visionary world materialize as a set of “services”, as opposed to for example programmable objects in the object oriented paradigm, or relational databases in the database world. In real life, a service is often understood as a capability of assistance provided by one to another, or work performed by one for another. By (OASIS 2006*a*), a service is defined as *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*

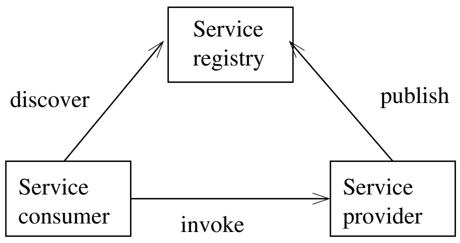

Figure 2.1 SOA pattern of interaction

specified by the service description. This definition separates the service, as “access to one or more capabilities”, from the entities, such as software components or databases, actually implementing the service. As an example, consider the service `addition a b`. The `addition` service represents the capability of performing the task of adding two numbers `a` and `b`. Obviously, we need an actual computer program to be able to submit two numbers, and get the result back.

The above definition of a service allows a service to be specified, reasoned about, and even invoked, in a standardized, technology-independent language without any knowledge about the service implementation technologies. Furthermore, it allows a service to be referenced before a concrete service implementation has been identified.

When a system is comprised by services matching our definition of service, we call it a *service-oriented system.*

2.2 The Service-oriented Architecture

A software architecture normally describes the structure or style of a software system, concentrating on the software elements in the systems, and the dependencies between them. SOA does not define one such architecture. Rather, SOA is a conceptual approach to systems’ interaction and interoperation, which SOA systems realize. SOA is defined by (OASIS 2006*a*) as *a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains.*

The SOA concept is often illustrated as shown in Figure 2.1. Fundamentally, a SOA system is based on three roles: The *service provider*, the *service consumer*, and the *service registry*. Service providers make services available for service consumers to invoke. However, the two roles are decoupled in the sense that a service consumer should not depend on the existence of a particular service provider, and that a service provider should not assume any particular service consumer. Service providers publish information about their provided services in a service registry, where arbitrary service consumers may discover them. Based on information found in the registry, a service consumer should be able to discover, locate, and invoke services as needed. To this end, we can also consider SOA as a pattern for interaction, defined by its participating roles and the interaction between them.

To further describe SOA, it is common to refer to a set of *service-orientation principles* which characterizes the ideal SOA system:

Contract: Service providers and consumers share a contract which defines a set of operations supported by the service, as a set of input and output messages, and the rules and characteristics of these operations.

Abstraction: Beyond what is described in the service contract, services hide their implementation. The service consumer must not make any assumptions about implementation details such as implementation logic, type of implementation resources (implementation classes and data) etc.

Reusability: As services expose their contracts and abstract their implementations, services promote reuse.

Loose coupling: As service providers expose, and service consumers perceive, only the service contract, services can interact without the need for tight, cross-service dependencies beyond the contract.

Discoverability: Service contracts can be published to the registries or brokers, so that the services can be statically or dynamically found and assessed by available discovery mechanisms.

Statelessness: Services should minimize the amount of state information they manage, and the duration they hold it, in order to remain available to other requester. Services are stateless in the sense that state information is specific to the current activity, and ideally to the currently processed operation. Between operations, state information should be externalized from the processing of service operations, into external components such as databases or file systems.

Autonomy: A service is autonomous if its implementation, i.e. implementation logic and resources, reside within an explicit boundary and under unified control and self-governance. Service autonomy improves the availability and robustness of the service.

Composability: Reusable services that allow loose coupling are inherently suitable for composition. Furthermore, as for part services, statelessness and service autonomy improves the availability and robustness of the composed service.

As these principles describe the ideal SOA system, we do not require actual SOA systems to satisfy the all the principles entirely. However, if we find that a system only to a small degree satisfies the SOA principles, we would normally not call it a SOA system.

2.3 SOA Fosters System Interoperability

Interoperable software systems are able to operate together, even though they are realized by different technologies. Interoperable systems must at least share a common protocol for specifying message exchange. (Erl 2005) and others claim that SOA systems fosters interoperability by their nature. We can easily accept that services as defined above, and the principles described above, does contribute to interoperability. As an example, the contract and abstraction principles by definition enable developers to program service clients without considering the technologies used to implement the services. As the service implementation is hidden from the client, the client system can operate together with the server system by applying the operations specified by the service contract only, given that they share a language for contract specification and common message exchange protocol.

The international standards organization for the Internet, World Wide Web Consortium (W3C) has recommended a set of standards for service description and message exchange, including the Web Services Description Language (WSDL) (W3C 2001) and the Simple Object Access Protocol (SOAP) (W3C 2000). Based on these standards, W3C defines a "web-service" as *a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.*

As web-services are broadly used to realize SOA, Web-services provides a technology platform for SOA systems.

2.4 Web-services - WSDL and Accompanying Standards

WSDL specifies a web-service interface as a set of operations with input and output parameters. A concrete web-service can be identified as a specific endpoint (i.e. IP host and port), potentially available over the Internet. The web-service can be accessed by sending a SOAP message containing the name of the operation to invoke, and input parameters, to this endpoint. If the service specifies an output parameter, a SOAP message containing operation output may be returned.

Several other standards exist, that manage different aspects of distribution in web-service technologies. For example, WS-Discovery (OASIS 2009) specifies a discovery protocol for locating services based on their type. WS-Policy (W3C 2007) defines a framework for expressing different types of policies for Web-services, such as Quality of Service (QoS) or security policies. WS-Policy is often applied together with WS-Security (OASIS 2006*c*), which standardizes enhancements to SOAP messaging that provide secure exchange of web-service messages. WS-Notification (OASIS 2006*b*) is a group of standards that specify how web-services can interact through notifications, also called events, and how producers and consumers of such events can be loosely coupled using a publish-subscribe pattern. These standards have been recommended by standardization organizations such as W3C and OASIS. However, there are many other proposals, and it is generally a challenge for the community to agree upon new standards.

Web-services use Extensible Markup Language (XML) namespaces to uniquely name resources, such as service descriptions, service endpoints, and policies. Normally, Uniform Resource Locators (URLs) are used to identify a namespace. As an example, the URL `http://www.w3.org/2002/ws/policy/ns/ws-policy` always points to the latest XML schema for the WS-Policy 1.5 namespace.

2.5 Workflows and Processes

As discussed above, the SOA principles state that services are inherently suitable for composition, into sequences of services called "workflows". When processed, the output of each executed service in the workflow provides input to the next service in the sequence. Often, such workflows reflects business processes in an organization. A much used example in civilian context, is travel planning; A workflow consisting of services provided by a travel agency which customers can use to book flights, hotel, car rental, and other related services, in one process. However, in a military tactical domain, we may expect that workflows will model smaller scale processes of work, such as the information flow from a particular sensor on a tactical unit, to a decision maker.

Specifying a workflow from web-services is called “web-service orchestration”. Several orchestration languages have been proposed, such as the Web Services Business Process Execution Language (WS-BPEL) (OASIS 2007), which has been much used both in the academia and in e-commerce.

2.6 SOA Technology

Commonly, SOA is realized by web-services running on top of the Enterprise Service Bus (ESB) (Manes 2007). The ESB is a SOA-based software architecture in the traditional interpretation of the term: It defines a framework and roles in which software can be added to build running SOA systems based on web-services. The primary task of the ESB is to provide a message bus for message passing between Web-services. Secondary, an ESB provides mechanisms related to service management and execution, such as service discovery, composition, transactions, etc. Thus, an ESB can be seen as a tool-chest which has a default set of tools, and which can be extended with more tools as needed. There are many implementations available, suitable for different purposes.

As a reaction to what many see as the complexity of web-services and the ESB architecture, and the overhead of XML-based protocols such as SOAP, WS-Notification, and the like, the Representational State Transfer (REST) architectural style has become a popular alternative for implementing web applications over recent years (Fielding 2000, Rodriguez 2008). So-called RESTful web-services use the Hypertext Transfer Protocol (HTTP) and the HTTP verbs (POST, GET, PUT, and DELETE) to deal with data structures and the transfer of their state. However, RESTful web-services’ compatibility with certain SOA principles, such as contract, composability, and in fact the service concept as understood as “capacity of work”, is at best unclear. Thus, in the rest of this document we focus our discussions around web-services as defined by W3C.

2.7 General Features and Observations of SOA

SOA systems generally has the following characteristics:

Service oriented: Services, as defined in this document, provide a capability of performing work for a client. Thus, SOA is suitable for tasks where the processing of input generates new output.

Composability: Services can be composed into workflows, or processes, given that the output from each service matches the input to the next.

Dominated by standards: SOA is largely realized through implementation of web-service standards such as WSDL, SOAP, and accompanying standards and standard proposals, which fosters interoperability.

Finally, we observe that web-service standard proposals and recommendations, and current SOA research, have ambitions for SOA systems to develop into more dynamic systems, where automated reasoning can be applied to service discovery and composition. This capacity has not been widely exploited so far in civilian context. A possible reason may be that as most web-service based systems in civilian context are statically and manually administered, they require little automated behavior. However, as we discuss in the next section, dynamism and automation could be more interesting in a military context, in particular when considering military tactical networks, where resources are scarce and variable.

3 Semantic Technologies and the Semantic Web Paradigm

Whereas service orientation most aptly focus on operations, their input and corresponding outputs, and compositions of operations into workflows, semantic technologies revolves around describing the relationship between static objects—or in Semantic Web terminology *resources*. In this section we try to spell out precisely what this means by describing the building blocks and the philosophy behind the so-called Semantic Web. At the possible expense of nuance on behalf of both paradigms, we are explicitly opting for a contrasting exposition that will yield a clear picture of the respective discipline's idiosyncratic capabilities.

3.1 Semantic Technologies = Knowledge Representation + Reasoning

The term 'semantic technologies' is most aptly considered a generic term that covers a wide range of techniques based on *knowledge representation* and *reasoning*. Knowledge representation is concerned with encoding qualitative information in machine-processable form, whereas automated reasoning aims at providing an algorithmic description of how to draw conclusions from a corpus of knowledge in an acceptable amount of time.

In this classical rule-based AI-paradigm the term 'knowledge' is best understood as a technical term. It is used chiefly as an attribute of systems that are based on some declarative language for stating facts which induces a computable relation of inference. More specifically, a system is a knowledge-based system if a) facts are declaratively encoded in a formal language, and b) this language gives rise to a notion of logical entailment that enables a machine to verify that a new fact is implicit in the set of facts the system already knows. Such a language is by definition a logic, whence the term 'logic-based system' may be regarded as a synonym.

3.2 The Semantic Web

The Semantic Web may be considered an adaptation of the knowledge representation paradigm described in the previous section to HTTP-based network environments. By encouraging the inclusion of formally encoded information in web pages and in web-oriented databases, the Semantic Web aims at converting the current web, which is dominated by unstructured and semi-structured documents into a "web of data".

Today, data is to a large extent stored in databases or files that feed *into* the Web but are not *part* of it. By the time the data reaches the Web it has usually already been processed and presented to fit some particular purpose or need, and can usually not be understood or reused outside that context.

The Semantic Web, in contrast, is about making the raw unprocessed data part of the very fabric of the Web itself. The Web, so the argument goes, ought primarily to be about data, and only secondarily about presentation and layout. Therefore, the Semantic Web emphasizes machine-readable self-descriptive data as a key concept.

Data on the Semantic Web is supposed to be self-descriptive in the sense of being encoded and wrapped in a description that is sufficiently rich to make the interpretation of the data software-independent. It is data that encodes its own interpretation—*meaningful data*—and it is designed precisely to transcend application barriers and software borders.

Condensed into a single sentence, therefore, the Semantic Web is about supporting intelligent processing of information across software boundaries and data sources based on the intended interpretation of the data, as given by the encoding of the data itself.

To that end the W3C has published a suite of mark-up languages of increasing degrees of complexity and expressiveness. At the lower end of the spectrum we find *The Resource Description Framework*, henceforth RDF, which is a basic model for encoding relational data in a format suitable for HTTP-based network environments. At the other extreme we have the highly expressive *Web Ontology Language*, which is a language for formally describing the semantics of data encoded in RDF. We shall describe each in turn below.

3.3 RDF

RDF is language for making statements about resources in the form of subject-predicate-object expressions.[1] These expressions are known as triples in RDF terminology. The subject denotes the resource/topic, and the predicate denotes a property of the resource that relates it to the object (which is another resource).

RDF is an abstract model implemented in several concrete file formats, and so the particular way in which a triple is encoded varies from format to format. Common to all, however, is the naming scheme, which is a fundamental feature of the RDF data model. More specifically, RDF uses Uniform Resource Identifiers (URIs) as constants (i.e. names of entities): a fundamental prerequisite for the Semantic Web is the ability to state facts and assertions *unambiguously* in a web-wide scope. The Semantic Web is designed to heed the so-called AAA principle—*Anyone can say Anything about Any topic*—so there is a very real and pressing need for a supply of identifiers that refer uniquely, no matter where on the web they are used.

RDF solves this by using the Web's addressing scheme itself as a basis for naming. URIs by design

[1]The use of 'resource' has historical roots and is not very clear terminology. Briefly put a resource can be any object or topic, abstract or particular, that one can point to or talk about, e.g. a concept, a person or a web page. In other words, 'resource' is synonymous with 'entity'.

provide a foundation for a data-sharing infrastructure because they all exists within a universal namespace that comprises the web as such. Thus, information expressed in RDF does not run the risk of context-dependent semantic interference or obfuscation.

3.4 OWL

The Web Ontology Language (OWL) is a Semantic Web language designed to represent complex knowledge about *classes* of things, and relationships between classes of things. Whereas RDF deals with concrete objects of a given domain, OWL deals with the relationship between its general terms.

OWL is a computational logic-based language such that knowledge expressed in OWL can be reasoned with by computer programs either to verify the consistency of that knowledge or to make implicit knowledge explicit. An OWL *ontology* encodes the meaning of a general term by specifying the relationship this term bears to other terms in the domain, e.g. (where general terms are underlined):

- *Every <u>project</u> has at least one <u>participant</u>.*
- *<u>Projects</u> are always <u>external-</u> or <u>internal projects</u>.*
- *The <u>superior</u> of my <u>superior</u> is also my <u>superior</u>.*

When an ontology is superimposed onto an RDF data set, the result is a logical theory. The general statements in the ontology act as axioms that allow new facts to be derived from the explicitly stated RDF data. In less clear language one may say that the ontology axioms give the *meaning* of the general terms, which in turn dictates the interpretation of the data by defining the inferences that apply to it.

3.5 The SPARQL Query Language and Protocol

Querying the Semantic Web requires a language that recognizes RDF as the fundamental syntax, whereas exchanging data requires a standardized way to store and request RDF data. The SPARQL specification is designed to cater for both needs.

The SPARQL specification was made a standard by the RDF Data Access Working Group of the World Wide Web Consortium in 2008. It defines both a query language *and* a protocol for data exchange. The query language is a syntactically-SQL-like language for querying RDF data sets via pattern matching. The language's features include basic conjunctive patterns, value filters, optional patterns, and pattern disjunction.

The SPARQL protocol, on the other, hand is a method for remote invocation of SPARQL queries. It specifies a simple interface that can be supported via HTTP or SOAP that a client can use to issue SPARQL queries against a so-called SPARQL endpoint, which is a conformant SPARQL protocol service.

At its simplest a SPARQL endpoint is a URI to which queries can be sent, and which returns answers to the queries as a response. A SPARQL endpoint may be thought of as a web-oriented database designed to support queries and data exchange in a networked, HTTP-based environment. The query interface is completely generic, that is, it does not, unlike the typical SOA service presuppose any particular use of the data as given by a custom made web-service interface.

3.6 Paradigmatic Use Case: Data Integration

Ontology-based data integration is concerned with unifying data that overlap in content, interpretation and relevance, but which resides in different repositories or databases. One of the principal functions of an ontology is to mitigate the semantic heterogeneity of the sources involved in order to enhance interoperability.

Semantic heterogeneity involves a mismatch between concepts and their interpretations, and is due to one of three things: semantically equivalent concepts, semantically unrelated concepts, and/or semantically related concepts where the relation is not made explicit. In the first case, two sources use different terms to refer the same concept, i.e. to synonyms. This is the case if, say, the same concept is modeled differently by different systems, for example if the concept vehicle is modeled as a "unit with a propulsion engine" in one source and as a "self-propelled unit" in another. In the second case, the same term is used by different systems to denote completely different concepts, e.g. homonyms like "bow" as a weapon vs. "bow" as the front of a ship. In the third case semantic heterogeneity may stem from a situation where, say, one source contains records of scientific staff and another of administrative staff, whilst one cannot interact with both types of record simultaneously simply as records of employees.

An ontology can reduce the effects of semantic heterogeneity in at least three ways: (1) the vocabulary provided by the ontology serves as a stable and unified query interface to the databases, (2) the ontology enables translation of all the relevant information sources into a common frame of reference, and (3) the ontology supports consistent management and recognition of inconsistent data.

3.7 Compatible Architectural Styles

Although the Semantic Web is not primarily an architectural concept, its emphasis on rich and semantically self-descriptive data is highly compatible with certain paradigms for publishing data on the web that have emerged in recent years. These paradigms do add architectural traits to the existing web within which the principles behind data-orientation and semantic technologies live particularly comfortably. We describe two of the more well-known examples below, namely Linked Data and REST.

Linked Data

Linked data describes a method of publishing RDF as an interlinked part of the web fabric itself. It builds upon RDF, and thus on standard Web technologies such as HTTP and URIs, but rather than just storing RDF in SPARQL endpoints, it embeds the data items in the web itself by making URIs interlinked and clickable. That is, Linked Data describes a recommended best practice for exposing, distributing and connecting data items by extending the hypertext architecture of the web to a *hyper* *data* architecture (Bizer et al. 2009).

Linked Data is usually summarized with reference to the following four maxims:

1. Use URIs to identify things.
2. Use HTTP URIs so that these things can be referred to and looked up ("dereferenced") by people and user agents.
3. Provide useful information about the thing when its URI is dereferenced, using standard formats such as RDF/XML.
4. Include links to other, related URIs in the exposed data to improve discovery of other related information on the Web.

Tim Berners-Lee restated the linked data principles as the following "extremely simple" rules:

- All kinds of things, conceptual or not, may have names that start with HTTP.
- When URIs are resolvable, I get important information back. I will get back some data in a standard format which is data that somebody might like to know about that thing, about that event.
- What I get back has got relationships, whenever it expresses a relationship then the other thing that it's related to is given one of those names that starts with HTTP.

Linked Data enables information from different sources to be connected and queried. By providing links (in terms of RDF triples) applications may exploit the extra knowledge from other data sets when developing an application; by virtue of integrating facts from several data sets, the application may provide a much better user experience.

REST

Representational State Transfer is a pattern of resource operations that has emerged as a de-facto standard for service design. Whereas the traditional SOAP-based approach to web-services uses full-blown remote objects with remote method invocation and encapsulated functionality, REST deals only with data structures and the transfer of their state. REST's simplicity, along with its natural fit over HTTP, has contributed to its status as a method of choice for web applications to expose their data (Battle & Benson 2008).

At the core of REST based design is a set of state transfer operations universal to any data storage and retrieval system. These operations, as commonly interpreted on the web, are referred to by the acronym CRUD, for “Create, Read, Update, Delete.” The REST community has adopted an informal mapping of CRUD operations onto the commands provided by the HTTP protocol: POST, GET, PUT, and DELETE, respectively. These commands identify the particular CRUD operation being requested of the resource identified by the URL endpoint (ibid.).

The REST design methodology integrates well with the resource paradigm of the Semantic Web. The Semantic Web uses URIs as resource identifiers, so the URL-based identifiers of REST fit naturally into its scheme. The Semantic Web, like REST, also deals strictly with assertions describing objects and their state; no parallel exists for SOAP-like remote method invocation. Finally, all common operations on the Semantic Web with the exception of query— data fetch, insertion, and deletion—are the fundamental operations in a REST-based system. It follows that REST-based web sites are an ideal carrier of semantic data.

3.8 General Features of Semantic Web technologies

RDF, SPARQL and OWL, make Semantic Web technologies ideally suited for data integration in HTTP-based environments: RDF encodes information using globally unique URIs, making it possible to express factual knowledge unambiguously in a network-wide scope. SPARQL endpoints are directly accessible over HTTP and do not require connection objects in a programming language, whereas SPARQL queries do not make any assumptions about the physical location of data.

The combination of RDF, SPARQL and OWL therefore brings to the web some of the same traits that characterize traditional knowledge-based systems. These include:

Declarativeness: Information model ling is based on a declarative description language that offers a level of abstraction in which data is managed according to its *conceptual content*, rather than, say its manner of storage or its manner of computation.

Basis in logic: Standard techniques from logic offer precise measures of important meta-properties of a data repository, for instance its consistency and or/the correctness and completeness of the reasoning procedures.

Reasoning as processing paradigm: The analysis of data is typically fueled by automated reasoning, which enables the interrelations between facts to be explored with depth and penetration, yielding considerable analytical power.

Data-orientation: Emphasis is placed on providing rich and semantically self-descriptive data that can be interpreted and processed intelligently by *any* application capable of decoding the format in question.

Qualitative information: The Semantic Web deals strictly with assertions describing objects and their state. This makes it particularly suitable for representing qualitative information.

4 Semantic Technologies vs. SOA; some Differences

The most conspicuous difference between the SOA conception and that of the Semantic Web is perhaps the fact that the Semantic Web is not primarily an information infrastructure or a system architecture concept. It is rather a particular methodology or technique for encoding, integrating and analyzing large sets of data. Contrasting it with key notions in the SOA conception, we have, among others, the following dissimilarities:

Background: SOA comes from software engineering, understood as the application of a systematic, disciplined, quantifiable approach to the design, development, operation, and maintenance of software. The Semantic Web comes from the sub-discipline of artificial intelligence concerned with knowledge representation and reasoning.

Orientation: SOA is service-oriented and procedural in the sense that it focuses on operations, and their input and corresponding output, and the pipe-lining of operations into processes. The Semantic Web, in contrast, is data-oriented and declarative and deals strictly with assertions describing objects and their state.

Aim: SOA denotes a set of abstract principles for reasoning about *behavior and interaction* of operations, their preconditions and post-conditions, and how to combine them into workflows, in order to achieve a particular end. The Semantic Web, in contrast, represents a concrete take on the problem of reasoning about qualitative data, such as vehicles, types of vehicles, their parts and their capabilities. Furthermore, the Semantic Web supports analysis, integrating and processing such data on the Web according to its intended interpretation.

In the next Section, we show how the different capabilities of SOA and Semantic technologies make them potential partners in building an infrastructure in line with the NNEC concept, benefiting from the strengths of the two concepts.

5 Opportunities

Based on the key characteristics of SOA and ST, and the discussion of their conceptual differences, we now present some potential synergetic appliances of the two paradigms in combination. We describe some example applications to show relevance and feasibility.

5.1 Semantic Web-services

Web-service standards, like WSDL and SOAP, only specify the syntax, and not the semantics, of service operations and data types. Thus, web-services depends on a human-in-the-loop to interpret the semantic interoperability of web-services and their clients. Semantic web-services use ontologies as the underlying data model to implement semantically enhanced mechanisms for service discovery, composition, and execution, enabling automated reasoning about web-services and web-service compositions, and thus better adaptability towards dynamic environments.

5.1.1 Related Work

Research in semantic web-services has been focused on semantic QoS models and ontologies, and corresponding policy based frameworks. As an example, (Ben Mabrouk et al. 2009) describes an ontology based quality model based on OASIS' Web Service Quality Model (WSQM). Another example is (Chaari et al. 2008), where a QoS-based ontology and WS-Policy-based framework for WS publication and selection is presented. Finally, (Hafsøe et al. 2010) describes an ontology-based approach to QoS-aware service discovery and orchestration of web-services in a MANET environment. Some ontologies for semantic web-services has been submitted for W3C recommendation, including OWL-S (W3C 2004*a*) and Web Service Modeling Ontology (WSMO) (W3C 2005).

There seems to be a lack of implemented systems and demonstrations available. As mentioned in Section 2.7, most web-service based systems in civilian context are statically and manually administered. Then, semantically enhanced descriptions can be replaced by descriptions more naturally read and processed by the human user. However, as argued by (Hafsøe et al. 2010), semantically enhanced web-services could be useful in military tactical networks, where resources are scarce and rapidly changing. The approach in (Hafsøe et al. 2010) is briefly described in the use-case below.

5.1.2 Case: Semantically Enabled QoS Aware Service Discovery and Orchestration for MANETs

(Hafsøe et al. 2010) describes an ontology-based approach to QoS-aware service discovery and orchestration of web-services in a military tactical MANET environment. The nodes in a Mobile Ad Hoc Network (MANET) are likely to be heterogeneous, and have different capabilities, both with regard to connectivity and end system resources. The discovery mechanism presented in (Hafsøe et al. 2010), called Service Advertisements in MANETs (SAM), enable clients to find the services that best match their capabilities by extending the service discovery mechanism with QoS attributes.

(Hafsøe et al. 2010) describes how service providers and consumers may specify both service types and QoS offers and requirements using an OWL-S based service ontology called OWL-S LiQ. Client requests are matched with available services in two steps; First functional aspects, that is the service inputs and outputs, are matched to find alternative services that provide the required functionality, then QoS parameters are matched to select the best alternative. If a service matching the client requirements can not be found, the matching algorithm attempts to apply service orchestration, to find a composition of services matching the requirements.

5.2 SOA and ST as Complementary Information Systems Technologies

The SOA and Semantic Web paradigms are very different conceptually, and are not naturally considered competing frameworks. On the contrary, larger information infrastructure may benefit from utilizing both, but in different functional roles.

In such an infrastructure, one would typically delegate responsibility for secure and reliable data transportation, -processing and -dissemination to a SOA layer—utilizing standards such as WS-Security, WS-ReliableMessaging, and WS-Policy—whereas one would employ semantic technologies to integrate, analyze and interpret the data that flows between the SOA components. In the example below, we describe an application where such a delegation is applied in a military context, to support operative decision making.

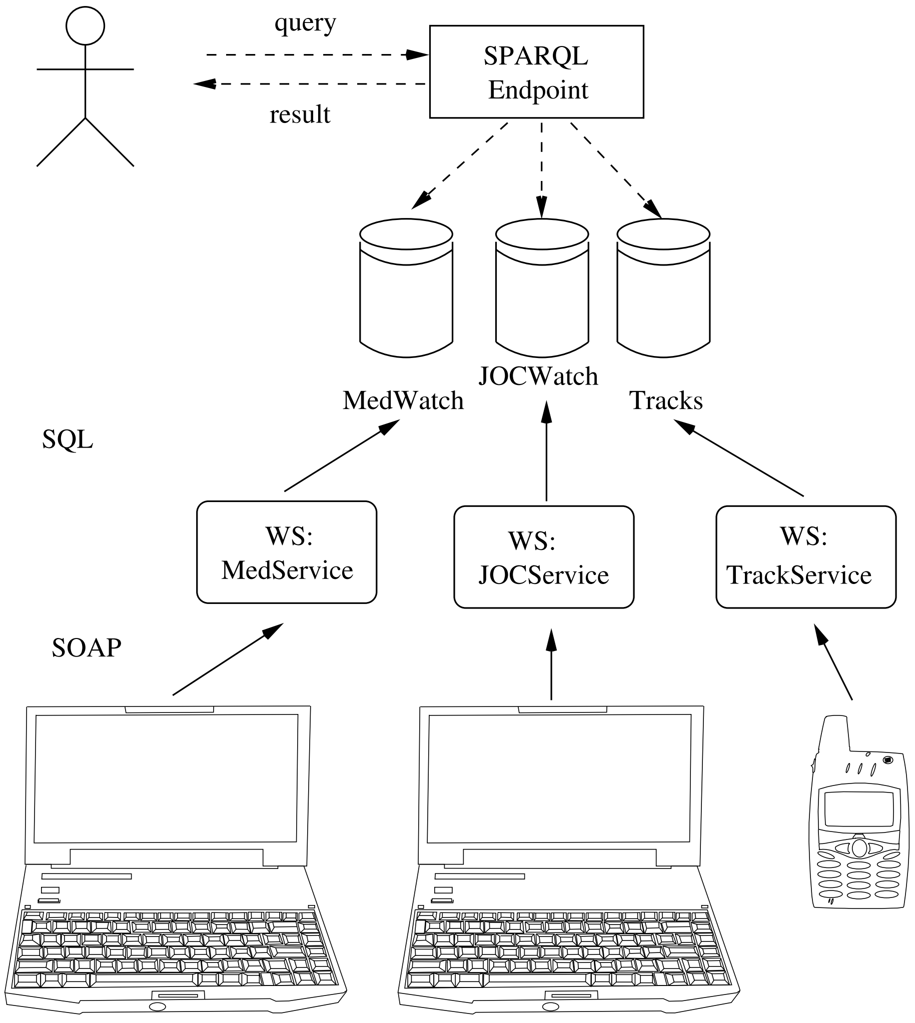

Figure 5.1 SOA and ST as complementary technologies

5.2.1 Case: From Sensor Data to Operative Decision Making

In 5.1 is illustrated a scenario where a semantic database is combined with services to collect and Analyse sensor data from operative units. A military analyst is monitoring planned medical evacuation flight missions, and needs to be alerted when missions are threatened by enemy activity. The analyst would want access to a Blue Force tracking system to obtain information about friendly units within a certain distance from the site of events, in order to summon those with the capabilities to support the evacuation mission.

Suppose there are three operational information systems involved in the task: A) JOCWatch, an event log containing information on incidents of relevance to the command B) MedWatch, a system for medical mission tracking designed to support the planning, logging and monitoring of medical evacuation missions, and C) Track Source, a unit tracking service providing timestamped geopositional information regarding friendly units in the field.

These system, which are systems in actual use, have links between them: MedWatch missions are (potentially) related to JOCWatch events through a shared incident. JOCWatch events are typed according to category e.g. as a SAFIRE event, which is an event that involved a hostile surface-to-air attack, whereas units in the Track Source are typed according to capability, e.g. as Artillery.

Note that, whereas missions and events are linked by an actual foreign key in the MedWatch database, event types and capabilities are linked only conceptually. The conceptual link may consist in e.g. the fact that a unit of type Artillery is adequately equipped to counter a hostile SAFIRE event, although this is not explicitly stated anywhere.

Semantic Web-aspects of the Case:

The above mentioned links, whether conceptual or not, are typical examples of the kind of static qualitative relationships that ontologies are designed to capture. An ontology for the case at hand would define the relevant concepts needed to express the analyst's information needs. For instance an ontology could define a concept ThreatenedMission as a MedWatch missions that is related to a ThreateningIncident, a concept ThreateningIncident as a JOCWatch incident that is related to a ThreateningEvent, and a ThreateningEvent as a JOCWatch event that is both a MilitaryOperation (from the JOCWatch ontology) and a HostileEvent. The ontology would also define significant relationships, for instance a relation canCounter to correlate unit types with event types.

Given such an ontology, the analyst will be able to express his or her information need in the form of a query phrased in terms of the above mentioned abstract and presumably familiar concepts: *Find all medical evacuation missions and friendly units such that a) the mission can be classified as being threatened; and b) that the friendly unit can counter the specific type of threat that the enemy poses.* A semantic application would typically apply a reasoner to compile this query into simpler ones that are executable directly against the sources. The analyst would not need to now which sources were involved, or even that there is more than one, but would simply interact with the system through the abstract interface offered by the ontology.

SOA Aspects of the Case:

Parts of the data in operational information systems such as JOCWatch and MedWatch will come from live reporting of incidents in the field, conceivably from gps-sensors, handheld mobile devices and radar. While some data elements can be read directly from the device, such as location data can be read from the gps sensor, other elements has to be typed in by the user, such as the description of an incident. However, the data types and formats are known in advance. Thus, web-services could be defined, specifying input messages for carrying MedWatch, JOCWatch, or Track data.

The system can be implemented as follows: A web-service client runs on the user device, collecting sensor data from the device and input data from the user. The client compiles SOAP messages and transmits them to the server-side web-service end-points. The web-services receive SOAP messages, performs necessary message processing, and insert information into the JOCWatch, MedWatch, and Track source databases.

5.3 Semantic Data Integration with Service Access

While a SPARQL endpoint offers a generic query interface to the client, some situations may require more restricted data access. By wrapping a semantic query interface as a service, we may benefit from combining the data analysis and reasoning capabilities provided by Semantic Technologies, with SOA features:

- Web-services exert *control over data access* through the service contract, which limits the information access to specific and predefined queries, and data abstraction, hiding data sources and schemas.
- Web-services enable *composition* through the specification of workflows, where the execution of SPARQL queries may occur in certain processing steps. That is, service composition may involve *i)* piping the result of a SPARQL query into a service, or *ii)* piping the output of a service into a SPARQL query.
- Application of *SOA standards* such as WS-Security, WS-ReliableMessaging, and WS-Policy provides service level security and QoS over data access.

Restricted data access with services, fix the allowed queries at service design time, and thus limits the choices available to the user, as compared with direct access to a SPARQL endpoint. When the query interface and the source schema is protected by a service layer, the user can no longer decide which information he wants to query, nor specify new queries on demand. Thus, the cost of adding access control with web-services is reduced usability of the semantic technology.

5.3.1 Case: Information Dissemination with Services and Information Assurance

In the scenario described in Section 5.2.1, several different information consumers, both at the tactical and strategic level, would be interested in results from semantic queries to the MedWatch, JOCWatch, and Track source databases. As an example, operatives in the tactical domain would be interested in information about planned medical missions close to their own location, while analysts on the strategical level would be interested in statistics and trends on injuries and diseases in all the areas where soldiers are deployed. Furthermore, parts of the information could be shared with allied forces.

Requirements

To ensure that information is efficiently and securely shared with receivers, the data dissemination mechanism should satisfy the following requirements:

- *Control over data access*: To limit data access of different categories of users, and thus the potential for information aggregation, users should not have direct access to the data source. Rather, they should have access to predefined queries which provide users with exactly the information they need, and not more.
- *Automation of processing*: To make the processes of extracting and distributing information

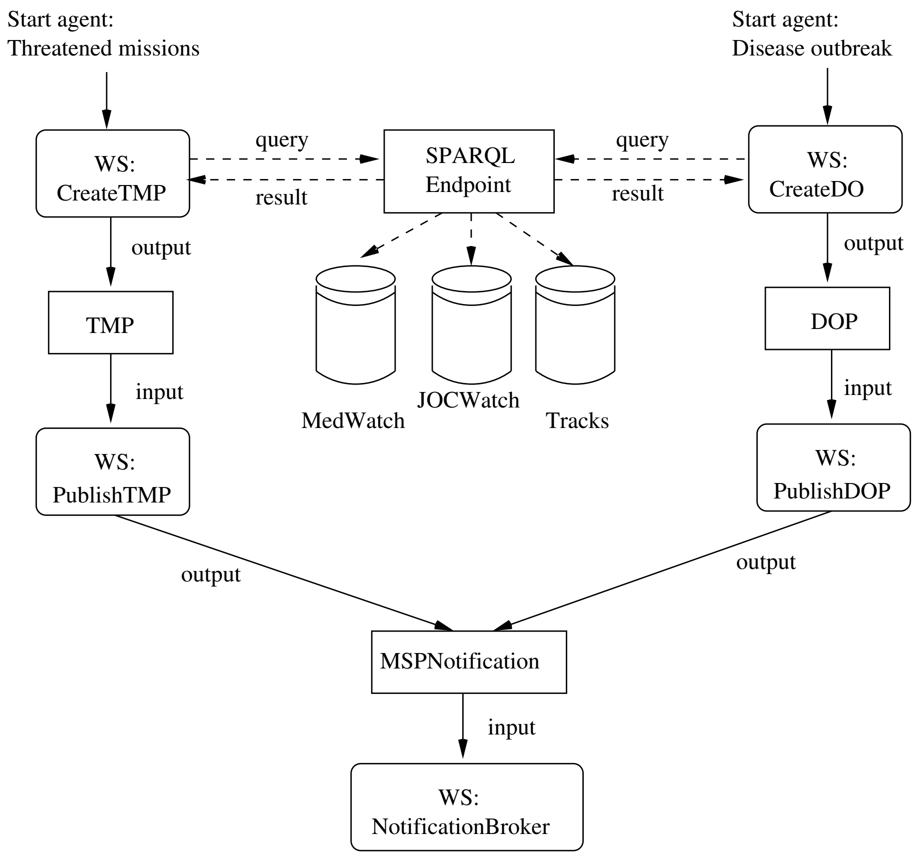

Figure 5.2 SPARQL-WS workflow

efficient and user friendly, these processes should be automated and as self-managed as possible.

- *Information assurance*: Data which is processed, stored, or transmitted must be protected by security mechanisms, ensuring data confidentiality, integrity, availability, authenticated access, and non-repudiation. As security mechanisms need to provide role-based, object-level protection, lower-level security does not suffice.

Application of SOA Mechanisms:

Web-service technologies provide mechanisms that satisfy these requirements. Therefore, results from SPARQL queries, as described in the scenario, can be distributed using web-service technologies. Either a pull or a push approach can be applied; With a pull approach, SPARQL queries are wrapped and made available as a public web-service endpoint to which any web-service client can connect. With a push approach, the same wrapped query is regularly invoked, and the result is published using a web-service based notification service. Both alternatives can be protected using the WS-Security framework, controlled by security policies written in WS-Policy.

Figure 5.2 illustrates how the push approach could be implemented using web-service workflows. Two agents repeatedly execute workflows that query a SPARQL endpoint and distribute interesting results using web-service technologies. The *Threatened mission agent* extracts information about planned missions and threats as described in Section 5.2.1, and issues a warning when a threatened mission is identified. The *Disease outbreak agent* detects patterns in illness reports, and tries to predict disease outbreaks based on them. When there are signs of an outbreak, an alarm is triggered.

The services CreateThreatenedMissionPicture (CreateTMP) and CreateDiseaseOutbreakPicture (CreateDOP) submit queries to the three data sources (JOCWatch, MedWatch and the Track source). The outputs from CreateTMP and CreateDOP, ThreatenedMissionPicture (TMP) and DiseaseOutbreakPicture (DOP), are submitted to the next two services in the workflows, PublishTMP and PublishDOP. These services transforms the TMP and the DOP into notifications, conforming to the WS-Notification standard, and publish the notifications to a web-service broker. The broker distributes the notifications to any client that has subscribed to TMPs or DOPs.

WS-Notification can be combined with WS-Security to *i*) authenticate notification publishers and subscribers, and *ii*) guarantee integrity and confidentiality of notifications.

6 Conclusions

In this document, we have discussed the role of SOA and Semantic Technologies in the perspective of NNEC. With NNEC, the NII must facilitate interaction between technical solutions deployed by allied forces.

Roughly stated, the SOA toolbox contributes to the interaction between services, remotely accessed and composed by means of standardized interfaces, whereas Semantic Technologies add the capability to reason over data (i.e. to analyze/synthesize information) across domains and systems. Hence, we claim that SOA and Semantic Technologies are orthogonal technologies which are conceptually very different, and therefore typically suitable for different things. The table below summarizes some important points of contrast:

Service-oriented Architecture	**Semantic Technologies**
service-oriented	data-oriented
procedural	declarative
describes workflows and processes	describes static relationships in qualitative data
methodology for structuring interaction	technique for integrating and analyzing data sets

One may benefit from combining SOA and semantics in the same information infrastructure, given that each paradigm is relegated to its appropriate functional role; SOA as a principled approach to building workflows from predefined, composable, and interoperable services, and ST as a provider of information integration, analysis, and reasoning. By sketching two plausible example applications, we have identified some potential synergies that may result from combining the two:

- One possibility is to use semantics for data representation, integration and analysis, whilst data harvesting and secure transportation is delegated to SOA components.

- Alternatively, one may use SOA technologies to disseminate knowledge gained from semantic integration and reasoning through more controlled service interfaces, with the potential of adding SOA-mechanisms providing role-based and object-level security.

- Finally, web-services may use ontologies to implement semantically enhanced web-service descriptions, enabling automated reasoning about web-services and web-service compositions based on inference.

Of course, these alternatives are not mutually exclusive.

Generally, with regard to information collection, processing, and dissemination, SOA and semantic technologies represent a span between the great flexibility and data usability of the Semantic Web, where information is available to the user as raw data, and the stricter regime of SOA where specific elements of information is made available through discrete services. Combining them gives the opportunity to make a trade-off between flexibility and control which may be a interesting alternative in a military context.

References

Battle, R. & Benson, E. (2008), ‘Bridging the semantic web and web 2.0 with representational state transfer (rest)’, *Web Semant.* **6**(1), 61–69.
`http://dx.doi.org/10.1016/j.websem.2007.11.002`

Ben Mabrouk, N., Georgantas, N. & Issarny, V. (2009), A Semantic End-to-End QoS Model for Dynamic Service Oriented Environments, *in* ‘ICSE Workshop on Principles of Engineering Service Oriented Systems - PESOS 2009’, IEEE Computer Society, Vancouver, Canada.
`http://hal.inria.fr/inria-00468220`

Bizer, C., Heath, T. & Berners-Lee, T. (2009), ‘Linked data - the story so far’, *Int. J. Semantic Web Inf. Syst.* **5**(3), 1–22.

Booth, M., Buckman, T., Busch, J., Caplan, B., Christiansen, B., van Engelshoven, R., Eckstein, K., Hallingstad, G., Halmai, T., Howland, P., Rodriguez-Herola, V., Kallgren, D., Onganer, S., Porta, R., Shawcross, C., Szczucki, P. & Veum, K. (2005), Nato network enabled capability feasibility study vii version 2.0, Technical report, NC3A.

Chaari, S., Badr, Y. & Biennier, F. (2008), Enhancing web service selection by qos-based ontology and ws-policy, *in* ‘Proceedings of the 2008 ACM symposium on Applied computing’, SAC ’08, ACM, New York, NY, USA, pp. 2426–2431.
`http://doi.acm.org/10.1145/1363686.1364260`

Erl, T. (2005), *Service-Oriented Architecture: Concepts, Technology, and Design*, Prentice Hall PTR, Upper Saddle River, NJ, USA.

Fielding, R. T. (2000), REST: Architectural Styles and the Design of Network-based Software Architectures, Doctoral dissertation, University of California, Irvine. `http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm`

Hafsøe, T., Johnsen, F. T. & Rustad, M. (2010), Semantically enabled qos aware service discovery and orchestration for manets, *in* 'Proceedings of the 15th International Command and Control Research and Technology Symposium'.

Manes, A. T. (2007), Enterprise Service Bus: A Definition, Technical report, The Burton Group.

OASIS (2006*a*), 'Reference model for service oriented architecture 1.0', https://www.oasis-open.org/committees/soa-rm/.

OASIS (2006*b*), 'Web services notification', https://www.oasis-open.org/committees/wsn.

OASIS (2006*c*), 'Web services security: Soap message security 1.1 (ws-security 2004)', http://docs.oasis-open.org/wss/v1.1/.

OASIS (2007), 'Web services business process execution language version 2.0', http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html.

OASIS (2009), 'Web services dynamic discovery (ws-discovery)', http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.html.

Rodriguez, A. (2008), 'Restful web services: The basics', http://www.ibm.com/developerworks/webservices/library/ws-restful/.

W3C (2000), 'Simple object access protocol (soap) 1.1', http://www.w3.org/TR/soap/.

W3C (2001), 'Web services description language (wsdl) 1.1', http://www.w3.org/TR/wsdl.

W3C (2004*a*), 'OWL-S: Semantic Markup for Web Services', http://www.w3.org/Submission/OWL-S.

W3C (2004*b*), 'Web services glossary', http://www.w3.org/TR/ws-gloss.

W3C (2005), 'Web Service Modeling Ontology (WSMO)', http://www.w3.org/Submission/WSMO.

W3C (2007), 'Web services policy 1.5 - framework', http://www.w3.org/TR/ws-policy/.