

**Web-oriented Architecture**  
**– Network-based Defence development made easier**

Mikael Kirkeby Fidjeland and Bård K. Reitan

Norwegian Defence Research Establishment (FFI)

October 23th, 2009

FFI-rapport 2009/01784

1084

P: ISBN 978-82-464-1658-8

E: ISBN 978-82-464-1659-5

## Keywords

Tjenesteorientert arkitektur

Brukerdeltakelse

Nettverksbasert forsvar

INI

Informasjonsdeling

## Approved by

Hilde Hafnor

Project manager

Vidar S. Andersen

Director of Research

## English summary

Information is power. Network-based Defence is about leveraging this power to achieve more effective operations. New technologies are necessary, but so are organisational changes.

*Web-oriented Architecture* (WOA) is an emerging alternative to traditional Service-oriented Architecture (SOA). This approach is based on the principles and technologies of the World Wide Web in order to take advantage of the properties and strengths of scope, scalability, interlinking, extendibility and ease of use. The main abstraction in WOA is *resources* or business objects such as units, positions and observations. These resources are uniquely and globally identified using Unified Resource Identifiers (URIs), and resources may be represented in different formats depending on different requirements. The use of hypermedia ensures that resources can contain links to other resources, creating a web of data.

In addition to machine-to-machine integration of information, WOA focuses on human interaction and social aspects as well. Web 2.0 denotes the changes in utilization of the World Wide Web that includes user participation, user generated content, bottom-up processes and inventive, new and open services. Web 2.0 services and applications are generally based on WOA principles, making Web-orientation an architecture that promotes collaboration and cooperation.

For the Norwegian Armed Forces' future information infrastructure (INI), extensive user involvement appears to be an important criterion for success. The challenges are to allow the users to participate, design the parts for participation and promote participation.

## Sammendrag

Informasjon er makt. Nettverksbasert Forsvar handler om å utnytte denne makten for å oppnå mer effektfulle operasjoner. Ny teknologi er nødvendig, men ikke tilstrekkelig. Organisatoriske endringer er også en forutsetning for å lykkes.

*Web-orientert Arkitektur (WOA)* er et oppdukkende alternativ til tradisjonell tjenesteorientert arkitektur (SOA). Denne tilnærmingen er basert på prinsipper og teknologier fra verdensveven (the World Wide Web) for å utnytte egenskaper og styrker som omfang, skalering, sammenlenking, utvidbarhet og brukervennlighet. Hovedfokuset i WOA ligger på *ressurser* eller forretningsobjekter som for eksempel, enheter, posisjoner og observasjoner. Disse ressursene identifiseres unikt og globalt ved hjelp av URIs (Uniform Resource Identifier). Ressurser kan representeres i ulike formater etter behov. Bruk av hypermedia gjør at ressurser kan inneholde lenker til andre ressurser og på den måten skape ett nett av data.

I tillegg til maskin-til-maskin integrasjon fokuserer WOA på menneskelig interaksjon og sosiale aspekter. Web 2.0 betegner forandring i bruk av verdensveven til å inkludere brukerdeltagelse, brukergenerert innhold, nedefra-og-opp prosesser og kreative, nye og åpne tjenester. Web 2.0 tjenester er generelt basert på WOA prinsipper, noe som gjør WOA til en arkitektur som fremmer samhandling og samarbeid.

For Forsvarets fremtidige informasjons-infrastruktur (INI) virker gjennomgående og aktiv brukerdeltagelse å være en betingelse for suksess. utfordringene ligger i å tillate, legge tjenestene til rette for og fremme slik deltagelse.

# Contents

<b>1</b>	<b>Introduction</b>	<b>7</b>
1.1	Goal	8
1.2	Motivation	8
1.3	Methods used and Structure of Report	9
<b>2</b>	<b>Background and Related Research</b>	<b>10</b>
2.1	Network-based Defence (NBD)	10
2.2	Service-oriented Architecture (SOA)	12
2.2.1	SOAP, WSDL and UDDI	13
<b>3</b>	<b>Web-oriented Architecture (WOA)</b>	<b>14</b>
3.1	Representational State Transfer (REST)	15
3.1.1	Resources and Uniform Resource Identifiers (URIs)	15
3.1.2	Hypertext Transfer Protocol (HTTP)	16
3.1.3	The REST architectural style	18
3.2	Web Orientation	20
3.2.1	Networked systems and the OSI Reference Model	20
3.2.2	The Application layer and REST manipulation	23
3.3	Web 2.0	24
3.3.1	Web 2.0 - The user centric view	25
3.3.2	Web 2.0 and Enterprise 2.0	25
3.3.3	Mashups	26
3.4	The Semantic Web – further work	27
<b>4</b>	<b>Military context RESTful services - Examples</b>	<b>28</b>
4.1	Blueforce Positions	28
4.1.1	Background and motivation	29
4.1.2	Service interface	30
4.2	MGRS Translator	31
4.2.1	Background and motivation	31
4.2.2	Service interface	32
4.3	Common Environment Interpretation - Observations	33
4.3.1	Background and motivation	33
4.3.2	Service interface	34

<b>5</b>	<b>Discussion</b>	<b>36</b>
5.1	WOA in the Information Infrastructure	36
5.2	SOAP versus REST	36
5.2.1	Control	37
5.2.2	Social aspect	38
5.2.3	Focus of Abstraction	38
5.2.4	Integration	39
5.2.5	Presentation	40
5.2.6	Security	42
<b>6</b>	<b>Challenges, consequences and opportunities for the Norwegian Armed Forces</b>	<b>42</b>
6.1	WOA or SOA	42
6.2	Web based	44
6.3	Data-driven	44
6.4	User participation	45
6.5	Security	46
6.6	No release cycles	46
<b>7</b>	<b>Conclusion</b>	<b>47</b>
	<b>References</b>	<b>48</b>
	<b>Abbreviations</b>	<b>52</b>
	<b>Appendix A Service interfaces</b>	<b>53</b>
A.1	BlueForce Positions	53
A.1.1	HTML/Web application interfaces	53
A.1.2	REST API	53
A.2	MGRS Translator	55
A.2.1	HTML/Web application interfaces	55
A.2.2	REST API	55
A.3	Collective Environment Interpretation	56
A.3.1	HTML/Web application interfaces	56
A.3.2	REST API	56

# 1 Introduction

*“The Web is intended to be an Internet-scale distributed hypermedia system, which means considerably more than just geographical dispersion. The Internet is about interconnecting information network across organizational boundaries”.*

(Fielding, 2000)

As Network-Based Defence is realized and the Information Infrastructure (INI) evolves, more information will be available in the network. The nodes in the network utilizing this information are people, sensors, weapon systems, information systems and so on. In order to achieve the potential benefits of Network-Based Defence, the information needs to be seamlessly integrated between these nodes.

Service-oriented Architecture (SOA) is a methodology for system integration, where processes and functionality are modelled as loosely coupled services. Service orientation is mostly implemented with Web services using a message exchange protocol. Such Web services offer *operations* or *methods* that clients may invoke. This approach to Service orientation is often encumbered with bureaucratic entities such as middleware, registries and governance frameworks. The time and cost associated with adding and integrating new services may be accepted in some instances, but the flexibility and agility may be hampered.

*Web-oriented Architecture* (WOA) is an emerging alternative to the traditional Web services. The fundamental properties are taken from the World Wide Web itself, the motivation being to leverage strengths such as scope, scalability, interlinking, extendibility and ease of use. The principles are formalized in a doctoral dissertation of Roy Fielding introducing the architectural style of *Representational State Transfer*. The basic elements in this model are *resources*. Documents, services, pictures, maps, pieces of information, collections of other resources are examples of resources.

Resources are identified and located by *URIs* – Uniform Resource Identifiers – the same way that web pages are identified and located using the address field in web browsers. By using URIs, there is a uniform, unique and global way of identifying and referring to resources. The cognitive step from using URIs for identifying and locating web pages to using URIs for identifying and locating information resources should be small. A resource may have many representations – some intended for machines and some intended for humans. Further, a resource may contain other URIs – that is references to other resources – and thereby *linking* and integrating information. A

resource may be as finely or coarsely granulated as needed, meaning that both small and detailed services as well as large, complex and composite services may be modelled as resources.

## 1.1 Goal

This report presents the concept of Web-oriented Architecture to the Norwegian Armed Forces. The goal is to show some positive effects made possible by being Web-oriented and how these relate to the concept of Network Based Defence. We will discuss the differences and similarities between this alternate approach and traditional Service-oriented Architecture, and how they may complement each other in an Information Infrastructure.

We will argue that a Web-oriented Architecture facilitates a network of *information resources*, that it is *decentralized* and that it eases *information sharing* and *collaboration*.

## 1.2 Motivation

Rapid and effective information sharing are principal properties of a Network-based Defence. In order to move power towards the edge of the network, information must be available whenever and wherever it is needed. The consumers of information will be both software and people, thus the information should also be available in a form that people may understand.

Social and organizational as well as technical factors are important when transforming to a Network-based Defence. Transformation requires organizational and cultural changes. Most initiatives in the realm of service orientation are focused on machine to machine integration and social aspects are thus lacking in traditional SOA. A Web-oriented approach allows for representations of resources directed towards human consumption as well and we will also show the social characteristics of WOA. Most new web applications offered by the likes of Google, Yahoo and Amazon use Web-oriented interfaces. It is easy to build dynamic and user-friendly interfaces on top of such resources. The direct connection between resources and their URIs should make it easier to conceptualize where and what kind of information being available. The use of URIs makes it possible to easily bookmark and share resources, further enabling collaboration and cooperation. The decentralized control and organic growth in a Web-oriented Architecture are properties enabling an agile organisation.



### 1.3 Methods used and Structure of Report

This report begins with an analytical approach discussing the theoretical background of Service-oriented Architecture and Representational State Transfer. Their respective related technologies will be examined. We continue with an exploratory method showing examples of Web-oriented services and resources, before discussing differences between Web-orientation and traditional Service-orientation.

We begin this report in chapter 2 by describing related research about Network-based Defence and the technologies used in a Service-oriented Architecture. In chapter 3 we introduce Web-orientation – the technologies as well as the emerging user contribution and participation seen in the contemporary Web. We continue by describing examples of information resources and services relevant for the Norwegian Armed Forces in chapter 4. Web-orientation, both inherent properties and its similarities and differences to traditional approaches, will be discussed in chapter 5. In chapter 6 we describe and discuss practical and specific consequences and challenges for the Norwegian Armed Forces with respect to implementing a Web-oriented architecture, before we conclude this report in chapter 7.



Figure 1.1: Word cloud for this report, generated by <http://wordle.net>.

Figure 1.1 above shows a word cloud generated by the content of this report, giving some indication of the important terms and concepts covered in the following chapters.

## 2 Background and Related Research

This chapter gives a short introduction to Network-based Defence in the context of information and information systems. Some knowledge about the concept of Network-based defence is assumed. We will also describe the protocols and principles behind a Service-oriented Architecture.

### 2.1 Network-based Defence (NBD)

Network-based Defence is much about leveraging the power of information. In the end, the goal is more efficient and more effective operations. Recent advances in information technology are necessary prerequisites for achieving this, but organisational changes are also needed. *Network-centric Warfare* (NCW) is the term used in the United States, whereas within NATO the term *NATO Network Enabled Capability* (NNEC) is used. More background information may be found in (Alberts, Gartstka and Stein, 2001) and (Alberts and Hayes, 2003).

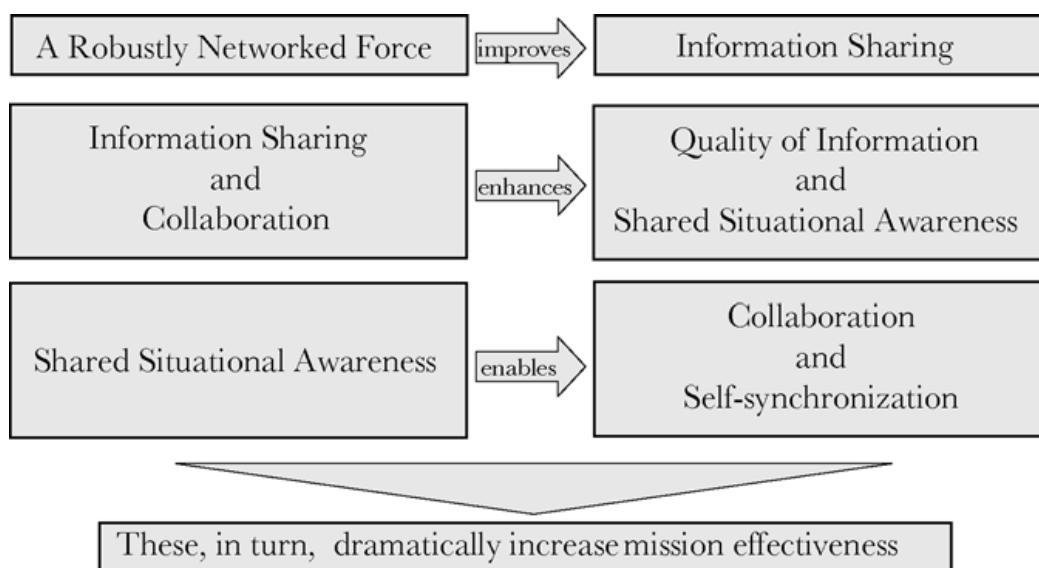


Figure 2.1: *The Tenets of NCW* (Alberts and Hayes, 2003).

Figure 2.1 shows the basic tenets of the underlying theory behind NCW and Network-based Defence (NBD). *Robustly networked* refers to both technological and organisational aspects of the force. A robustly networked force improves information sharing and collaboration, which again enhances quality of information and shared situational awareness. The last tenet regarding self-synchronization is the most diffuse (van Bezooijen et al., 2006) and controversial, as it implies a bottom-up approach to

control. However, the conclusion of increased mission effectiveness still stands and is the reason behind the recent focus on NBD. Note that organizational changes and challenges apply to all steps in this model. Valaker and Fidjeland (2008) discuss how instant messaging may be used in a military organisation and how this form of communication differ from other media such as radio or email. Other nations have experienced that the real-time aspect of instant messaging compress planning processes and thus increases the operational tempo, whereas the ability to monitor several chat rooms simultaneously increases situational awareness. However, introduction of new technologies must be accompanied with official endorsement and organisational awareness, which often is the most challenging part when implementing parts of a NBD.

In a study of NBD and NCW literature, Reitan and Pålhaugen (2004) argue that NBD is not a coherent and unifying concept, but rather a collection of hypotheses. They list six subjects covering the most important aspects of NBD:

- Network organization
- Decentralization
- Centralization
- Common situational awareness
- Common intent
- Geographical independence

A networked organisation allows for flexible and optimal use of resources. New technology allows organisations to be both centralized and decentralized, depending on the challenges at hand. A decentralized organisation would allow self-synchronization and intention-based command, the goal being faster decisions and the ability to take advantage of “windows of opportunities”. Centralization may in some instances increase the ability for a central command to coordinate forces at a tactical level. Common situational awareness and common intent increases the tempo of coordinated operations and increases the quality of information. Finally, geographic distribution of resources makes a more robust and available force.

Hedenstad et al. (2008) recommend a number of measures to implement in a Norwegian NBD. Among these is the use of pull instead of push for certain information use cases. Subscription based services are mentioned, and one way of doing this is by providing information in RSS feeds. By nature, services in a Web-oriented Architecture are pull-based. By using pull, information is requested by whoever needs it whenever it is needed. When using push someone has to decide up front who is going to receive a piece of information. This aspect is also addressed in Alberts and Hayes (2003):

*“[...] information related capabilities are all enabled by the post and smart pull approach inherent to a robustly networked environment.”*

As the number of nodes in the network increases, deciding who gets what using push gets ever more difficult. Smart pull allows operators, officers and soldiers decide when and what information they need. Another aspect mentioned in Hedenstad et al. (2008) is the need for various information systems to handle various standardized message formats. Information in a Web-oriented Architecture may be available in several different formats. Information is modelled as resources, and the resource may be represented in a format agreed upon by the server and the client from a limited selection. See more on this in section 3.2.2.

## **2.2 Service-oriented Architecture (SOA)**

Service-orientation is a method for system integration in a distributed environment. Functionality is modelled around *services* with clear boundaries between service providers and service consumers. Loose coupling is a key property of Service-orientation, meaning that the provider and consumer make no assumptions of the other parts inner workings. The service *interfaces*, defining the interaction, are separate from the implementations. The interaction and interface are standard based such that different clients may use the same service regardless of underlying operating systems, programming languages and so on. The services should be flexible and reusable, and the modularity coarse-grained such that each service offer some kind of business value. One of the selling points of Service-oriented Architecture (SOA) is that the development of services should be business driven, not technology driven. It is supposed to lessen the gap between the IT part and business part of an organisation. As the set of flexible, loosely coupled services gets larger, the business or organisation becomes more agile – i.e. having the opportunity to quickly alter existing business processes or creating new ones.

As SOA is not primarily about technology but rather about design and architecture, there are several ways to implement a SOA. The most common way, with the most support from software vendors and tools, is to use the triumvirate of the SOAP, WSDL and UDDI technologies. These are described in subsection 2.2.1 below. Often when talking about of SOA, people refer to this set of technologies. For the rest of this report we refer to this as *traditional SOA* or *SOA+SOAP*. Another approach is to use message-oriented middleware. Using this, service requests are put on message queues and then asynchronously processed by a service. This is also known as an event-driven architecture and examples of frameworks for implementation are WebSphere MQ from IBM and JMS (Java Messaging System) from Sun Microsystems. The emerging Web-oriented Architecture, discussed in chapter 3, can be seen as a sub-style of SOA, having other constraints, using different technologies and less strict standards.

In order to better compare the technological differences between traditional SOA and WOA we will go into some detail of the three protocols used in traditional SOA before moving on to discussing WOA. For more information on SOA, please see (Johnsen et al., 2008; Lund et al., 2007) or other publications from the FFI project P1086 Secure and Pervasive SOA.

### 2.2.1 SOAP, WSDL and UDDI

SOAP, WSDL and UDDI are the three most important standards used in traditional SOA. UDDI (Universal Description, Discovery and Integration) is a specification for service registration and discovery. Service providers register their services in an UDDI registry. This registry acts as a service broker, which service consumers may use to find services. The broker may also be used to bind clients to service implementations dynamically at runtime.

Services are described by using WSDL (Web Service Definition Language). A WSDL document formally describes a Web Service's types (format of input and output messages), interface (methods or operations), bindings to a protocol (e.g. to SOAP) and fault handling. The WSDL definition of a service is part of the entry in an UDDI registry, along with other metadata describing the service. A WSDL document is also available at the service endpoint itself, meaning that it is possible to integrate without a UDDI registry if you know the location of the service.

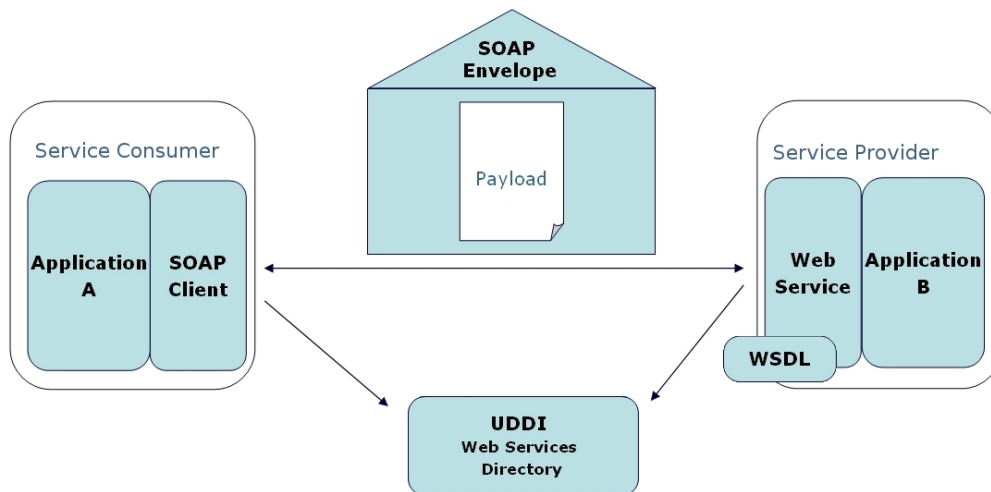


Figure 2.2: Web Services using SOAP, WSDL and UDDI (Mooney, 2007).

SOAP is the protocol defining the interaction between the client and the server, i.e. the service consumer and the service provider. The protocol defines how to exchange XML-based messages. Both the service request from the clients and response from the servers are SOAP messages. This is a XML message containing a SOAP envelope,

which again contains optional headers, a required body (the payload of the message) and optional fault elements.

Figure 2.2 shows how these three standards work together. A service provider has implemented a Web Service. This service is defined with a WSDL document and is registered in an UDDI service directory. The service consumer uses the UDDI registry to find an appropriate service. The messages between the client and server are XML-based and wrapped in a SOAP envelope. Even though HTTP is most often used as the transport protocol, SOAP is not bound to a specific underlying protocol and others such as SMTP may be used.

In addition to the three standards mentioned above, there exists a myriad of other standards associated with traditional SOA and Web Services (Bustamante, 2005; McKendrick, 2006). This set is often referred to as the *WS-\* standards* and includes specifications such as WS-Addressing for transport-neutral message routing and WS-Security. The growing number of standards has led major software vendors such as IBM, Microsoft, Sun, Oracle and SAP to create the WS-I organisation to oversee the interoperability of their respective implementations. WS-I publishes profiles such as WS-I Basic Profile 1.0/1.1/1.2 that provide guidance for interoperability using core specifications like the above mentioned SOAP, WSDL and UDDI.

### 3 Web-oriented Architecture (WOA)

As we have seen, SOA is about architecture and design – not the underlying technology. The traditional SOAP, WSDL and UDDI approach to SOA have been criticized in the blogosphere and developer community for being overly complex, slow to implement and not yielding the promised return of investment (Hinchcliffe, 2008a; Hinchcliffe, 2008b; Smith, 2008).

While major software vendors have extended their tool support for the *WS-\** protocols, software developers have started to embrace a simpler, more organic approach - *Web-oriented Architecture (WOA)*. WOA takes inspiration from the Web itself, first of all the properties of distribution and scalability, along with the well-known underlying technologies such as HTTP and client/server concepts such as resources and URIs (see sections below).

By *the Web* we understand the system more formally known as the World Wide Web. The Web was begun in the early 1990s as a hypertext system at CERN (Berners-Lee and Cailliau, 1990). It has since grown to become the most common use of the Internet, which is the actual global network of computers. Other uses of the Internet are email, file transfer, internet telephone, instant messaging and media streaming. The Web

consists of interlinked hypertext documents – i.e. homepages or web pages – viewed in a web browser. Today the Web offers more sophisticated services such as online banking, shopping and other applications. The usage has in many cases been transformed to include user contribution and collaboration, and at the same time re-labelled to Web 2.0. This will be further discussed in section 3.3 below.

WOA extends the concept of interlinked document addressed by URIs to more general information resources identified and addressed by URIs. The underlying principles are formalized as *Representational State Transfer* (REST) in a doctoral thesis by Roy Fielding (Fielding, 2000).

### 3.1 Representational State Transfer (REST)

Roy Fielding is one of the principal authors behind HTTP (Fielding et al., 1999) and has also been involved in work with the HTML and URI specifications. Both the HTTP protocol and URIs are principal specifications underlying both the Web in general and REST in particular. The next two subsections describe these in further detail, before we return to discussing REST.

#### 3.1.1 Resources and Uniform Resource Identifiers (URIs)

In the context of the Web, a *resource* is anything available and identifiable on the Web. Early on, this was typically static web pages or documents. Today various services such as online banking systems, web applications, and search engines and so on are examples of resources. A resource is identified by an URI, a Uniform Resource Identifier (Masinter et al., 2005; Mealling and Denenberg, 2002). An URI is simply a string following some scheme, and URIs are thus not limited to be used on the Web alone. Addresses such as `http://www.ffi.no` follow one such scheme; email addresses such as `mailto:mikael.fidjeland@ffi.no` another. The first one identifies a web page and also where to retrieve it. The second identifies an electronic mailbox. An URI does not necessarily imply *how* to retrieve a resource, nor that it is retrievable at all. Generally, anything that may be identified by an URI is a resource. A book may be identified by its ISBN number using an URI such as `urn:isbn:0-395-36341-1` and a physical person by an URI like `urn:no:mil:ffi:employee:mkf`. Even abstract concepts may be identified, for example using `urn:no:mil:nbf` to refer to the concept of Network-based Defence. Note that email is not part of the Web, neither are books or abstract concepts.

In a Web-oriented Architecture, URIs are used to identify web resources or information pieces. These *will* be available over a network, and it is this kind of resources we will talk about in the remaining chapters and sections of this report. URIs identify resources *globally* and *uniquely* – both internet wide (truly global scope) and within an

organization or private network. Given an URI there is no ambiguity regarding what concept or resource is referred to. Jacobs and Walsh (2004) identifies many benefits of URIs, stating among other things that “*global naming leads to global network effects*”.

### 3.1.2 Hypertext Transfer Protocol (HTTP)

Hypertext Transfer Protocol (HTTP) is an application layer client/server protocol for distributed systems (Fielding et al., 1999). The OSI seven layered model – including the application layer – will be discussed in subsection 3.2.1 below. HTTP manipulates resources identified by URIs – retrieving, updating, deleting and so on. The protocol defines eight verbs or methods for manipulating resources. The four most important, GET, PUT, POST and DELETE, are described in Table 3.1. The four others are OPTIONS, HEAD, TRACE and CONNECT.

Method	Semantics	
GET	Retrieve resource identified by the request URI.	Safe and idempotent
PUT	Update or create the resource identified by the request URI with the data enclosed in the request message body.	Idempotent
POST	Process the data enclosed in the request message body.	
DELETE	Delete the resource identified by the request URI.	Idempotent

*Table 3.1: The four main HTTP request methods.*

HTTP is a stateless protocol, with the GET, PUT and DELETE methods directly manipulating resources through their URIs (by retrieving, updating/creating or deleting them). The difference between PUT and POST may seem subtle. Using PUT, the client must know the URI of the resource and upload a representation of that resource. A subsequent GET of the same URI should return the uploaded resource. POST may be used when creating a new resource and the URI is not known (for example when adding a new row to a database) or for adding input to some service (such as search parameters or metadata).

The GET method is safe, meaning that the client is not responsible for any side effects on the server. GET, PUT and DELETE are idempotent, meaning that multiple requests have the same effect as a single request. For example, deleting a resource twice still results in it being deleted and updating a resource three times with the same message body results in the same resource stored on the server.



The client may modify the request by adding request headers. Authorization headers may be used to authenticate the client by supplying credential, for example user name and password. The client may also specify the media type of the response (or representation of resource, see section 3.1.3 below). Media types may be binary image formats, HTML documents, XML messages, and so on.

The server replies to a request by sending a mandatory status code and reason phrase, optional headers and an optional message body. The three digit status code and accompanying reason phrase describe the result of the request; received (1\*\* codes), successful (2\*\* codes), client side error (4\*\* codes) or server side error (5\*\* codes). Table 3.2 shows some common response codes and reason phrases with an explanation.

<b>HTTP status code</b>	<b>Explanation</b>
200 OK	The request has succeeded.
201 Created	The request has been fulfilled and resulted in a new resource being created.
303 Other	The response to the request can be found under a different URI and should be retrieved using a GET method on that resource.
400 Bad Request	The request could not be understood by the server due to malformed syntax.
401 Unauthorized	The request requires user authentication.
403 Forbidden	The server understood the request, but is refusing to fulfil it. Authorization will not help and the request should not be repeated.
404 Not Found	The server has not found anything matching the request URI.
405 Method Not Allowed	The HTTP method in the request is not allow for the request URI.
500 Internal Server Error	The server encountered an unexpected condition which prevented it from fulfilling the request.
503 Service Unavailable	The server is currently unable to handle the request due to a temporary overloading or maintenance of the server.

*Table 3.2: Common HTTP response status codes. Excerpt from (Fielding et al., 1999).*

GET requests will typically result in a message body being sent with the response, whereas the server will respond with no message bodies to PUT and DELETE requests. POST requests could result in either, depending on what the service at that URI is supposed to do. As stated, HTTP is a stateless protocol for distributed information systems. Most people use it when browsing the Web. The web browser is the HTTP client, sending a GET request when you type an URI into the address field or click on a link. The web server responds with the requested web page (the “200 OK” response code is hidden from the user) or a “404 Not Found” error if the requested

page do not exist. The REST architectural style, discussed below, generalizes the principles of the Web into a style of software architecture for distributed hypermedia systems.

### 3.1.3 The REST architectural style

REST, Representational State Transfer, is presented by Fielding (2000) as an architectural style for distributed hypermedia systems. It is discussed within a framework for understanding software architectures through architectural styles and classification of these styles. One of the aims of Fielding's work was to formalize the architecture behind the emerging Web. A number of requirements for applications in the Web domain are listed in (Fielding, 2000):

- **Low Entry-barrier.** Since participation on the Web is voluntary, the barrier to create and structure information must be low in order to enable sufficient adoption. Hypermedia is both simple and general regarding content creation, management and reading. It also allows for unlimited structuring by linking content.
- **Extensibility.** Requirements change over time, thus the system must be able to handle these changes.
- **Distributed Hypermedia.** Distributed hypermedia is designed for large-grain data transfer. By hypermedia we understand embedding application control (links to other resources clicked on by users or followed by software clients) within the presentation of information, and by distributed we understand that information is stored at various remote locations.
- **Internet-scale.** By internet-scale distributed system we understand more than just geographical dispersed. It also includes anarchic scalability – the entire system is not under control of one entity. A consequence of this is that clients cannot maintain knowledge of all servers and also that servers cannot retain knowledge of state across requests.

Architectural styles are described by constraints that apply to them. Starting with an empty *Null Style*, the REST style is derived by adding constraints. This set of constraints must meet the requirements listed above, in addition to minimizing latency and network communication and maximizing the independence and scalability of component implementations. The most important of these constraints are:

- **Client-server.** This constraint applies to the principle of separation of concerns, e.g. user interface for the client and data storage for the server. Some of the benefits are portability of user interface, scalability by simplifying server components and that components may evolve independently.

- **Stateless.** When communication between clients and servers are stateless, each request contains all necessary information. This support the properties of visibility (monitoring systems do not need to look beyond a single request), reliability and scalability
- **Uniform interface.** This is a central feature to REST. All services implemented using the REST style will have interfaces defined by URIs and the HTTP methods. Resources available through such a service will be identified by URIs. Representations of these resources may be manipulated using the HTTP methods (mainly GET, PUT, POST and DELETE). Also, there may be links to other resources embedded in the information.

Figure 3.1 below shows the interaction between the client and server and the manipulation of *representations* of resources. The client is typically a web browser, but may also be other software clients communicating with the server using HTTP.

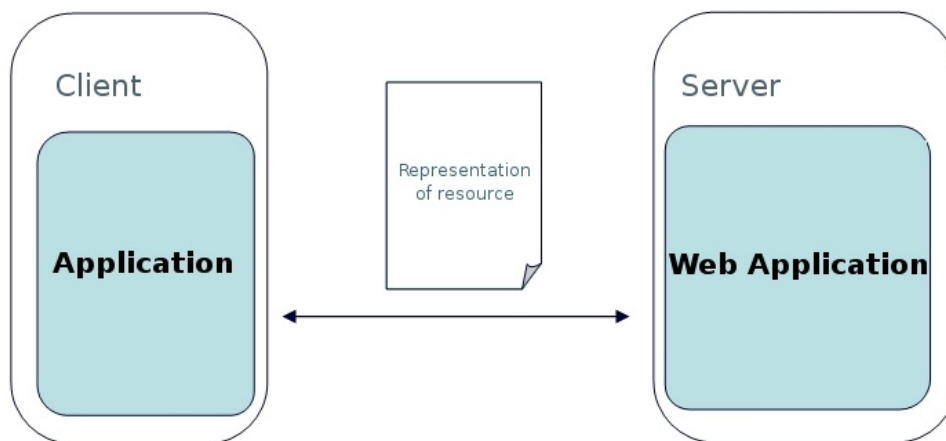


Figure 3.1: Representation of a resource communicated between client and server.

A resource may have several different representations. By representation we understand a serialization or binary representation of the underlying, more abstract resource. A representation may be a document of a given format, a file, an image or message entity. These representations are manipulated using the four HTTP methods GET (retrieving), PUT (updating or creating), POST (creating or adding data) or DELETE. In chapter 4 we will see examples of resources, possible representations and manipulations.

REST is an architectural style that incorporates lessons learned and experiences from the success of the World Wide Web. The Web can be seen as an information system *implementing* the REST style whereas organisations may implement their internal information systems using REST. Services implemented using REST principles are often referred to as *RESTful*.

## 3.2 Web Orientation

WOA uses the stack of well known technologies used on the Web: TCP/IP, HTTP, HTML, JSON, XML and so on. What are modelled are information *resources* and methods are restricted to the four HTTP verbs. Further, a resource may have several representations – some for human consumption and some for software client consumption. A client-server system such as the Web is an instance of the more general class of networked or distributed systems. When discussing or implementing distributed systems it is often referred to the OSI seven layer model. This model is well known and has been discussed thoroughly the last thirty years in various settings. We bring it up here again to show how the Web protocol stack fits into this model.

### 3.2.1 Networked systems and the OSI Reference Model

Open Systems Interconnection (OSI) is a standardization effort for networking. One result of this initiative is the OSI reference model (Zimmermann, 1980), also known as the seven layer model. This is an abstract design for distributed systems and computer network protocols. Functionalities and responsibilities needed for communication over computer networks are divided into the seven layers shown in Figure 3.2.

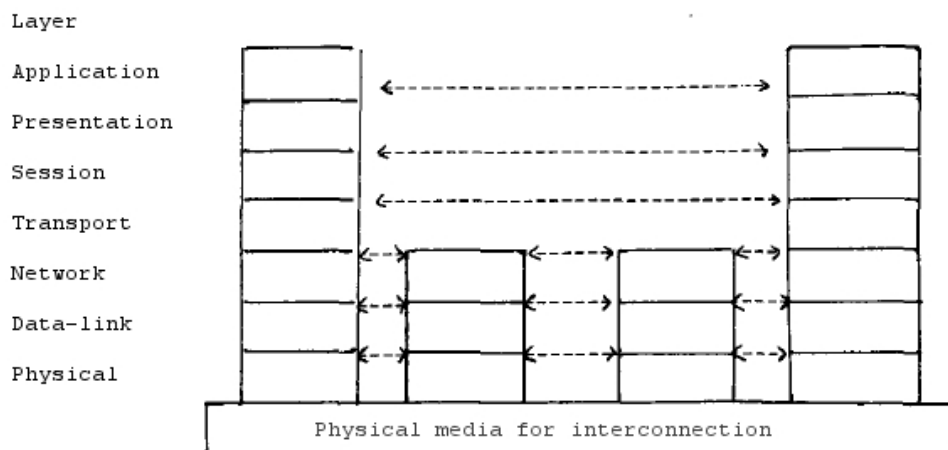


Figure 3.2: The seven layers OSI architecture. From (Zimmermann, 1980)

Each layer uses services from the layer beneath it and offers services for layers above it. Functionality in lower levels is hidden. For example, the network layer is responsible for routing through the network whereas the upper layers only see the communication with the end point. Figure 3.2 shows two applications at two different machines (the column at the far left and the column at the far right) with network traffic being routed through two intermediate routers (the two middle columns). The application at one machine only sees the application at the other. All nitty-gritty details of physical connections, networking, routing, error correction and so on are dealt with at lower layers.

The functionality offered at the different layers will typically be implemented as network protocols. The OSI model is an abstract model or a reference model and does only describe the functionality and do not offer any implementation specifications. The Internet stack of network protocols is an actual implementation of the layered model, even though it only has four layers. Table 3.3 shows these layers and the corresponding OSI layers. The lowest layer in the TCP/IP model is the Link layer which works in the context of the local network and network hardware. The actual TCP/IP protocols belong to the Network and Transport layer respectively and are thus independent from the underlying hardware (which explains how TCP/IP has been implemented on top of most existing networking hardware).

OSI Model	TCP/IP Model	Protocol	Data unit	Security
<i>Client application/end system</i>			Resource	Authorization
Application	Application	HTTP	Representation of resource	Authentication
Presentation				
Session				
Transport	Transport	TCP	Packet	TSL/SSL encryption
Network	Internet	IP	Frame	IPsec
Data-link	Link	Network hardware	Bit	
Physical				

Table 3.3: The Web Architecture and the underlying protocols.

The three top layers in the OSI are implemented as one layer in the TCP/IP model – the Application layer. Protocols in this layer are used by applications for network communication. For example, HTTP is used by web browser, web applications and services in a Web-oriented Architecture. Other protocols at the application layer are FTP (file transfer) and SMTP (email). Note that the applications such as web browsers or web servers are *using* the protocols at the application layer; they are not part of this layer themselves.

There are different security protocols available at the various layers. IPsec is a low-level protocol that authenticates and encrypts individual IP packets. TSL and SSL offer security and data integrity through cryptography. The HTTP protocol itself has built-in schemes for authentication, e.g. a username and password combination. Authorization has to be done in the server-side end system. Authentication is proving who you are; authorization defines what you are allowed to do.

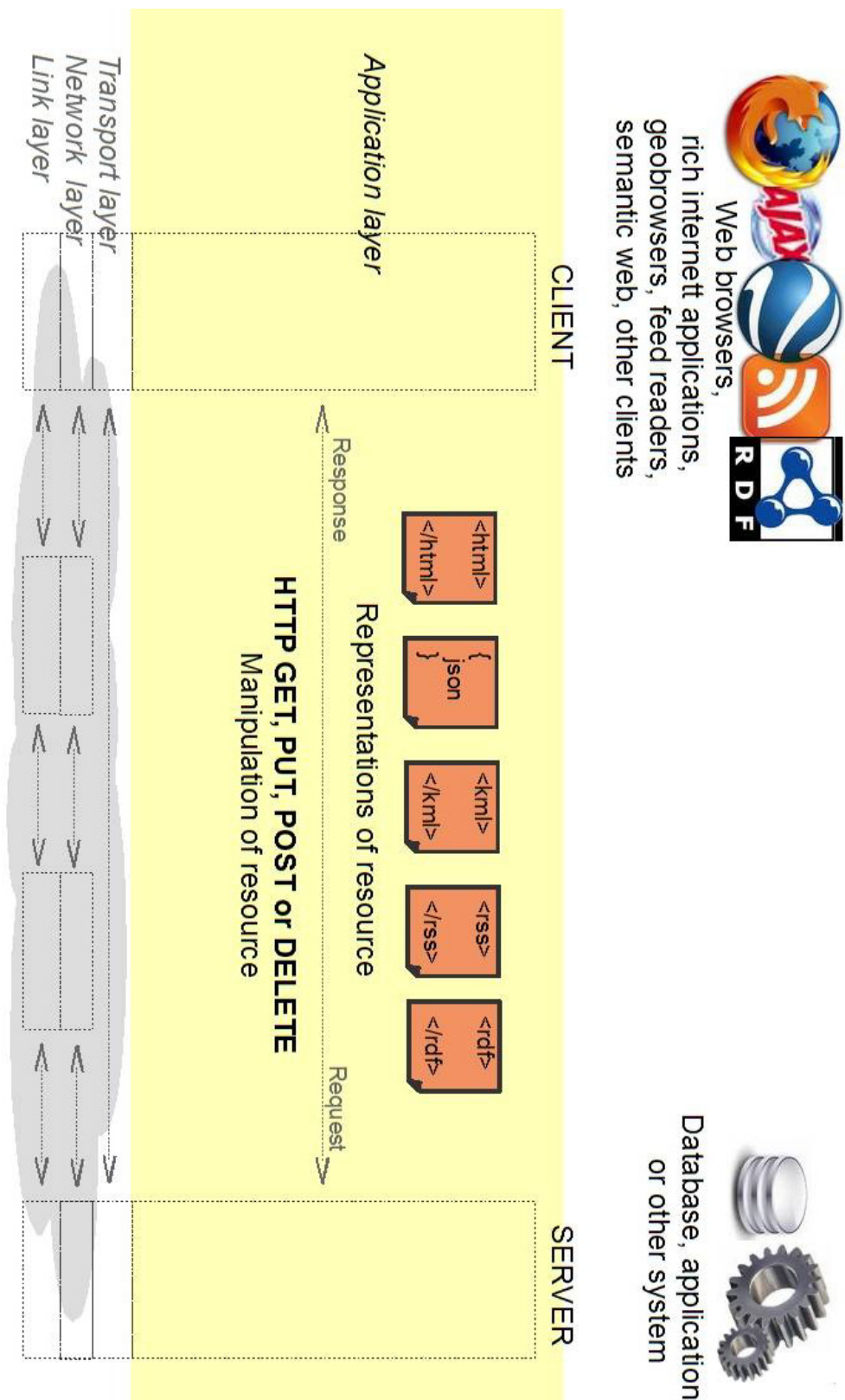


Figure 3.3: The main focus is the application layer where the representations of resources are exchanged and manipulated. The layers refer to the TCP/IP model.

### 3.2.2 The Application layer and REST manipulation

The OSI model describes a general model for distributed networking. The concept of client-server only applies to specific protocols in the Application layer and it is of course this layer that is the main focus of REST and WOA type services. The actual services use the HTTP protocol whereas the other protocols are hidden. We also see that the level of abstraction increases for each layer, from single bits through frames and packets towards representations of resources in the application layer. The resource itself – the final abstraction – is represented and manipulated in the client application or end system.

Figure 3.3 above shows the interaction between a client (left hand side) and a server (right hand side). The client uses the HTTP protocol to make a GET, PUT, POST or DELETE request for a resource identified by its URI. Depending on the type of request, a representation of the resource is sent over the network.

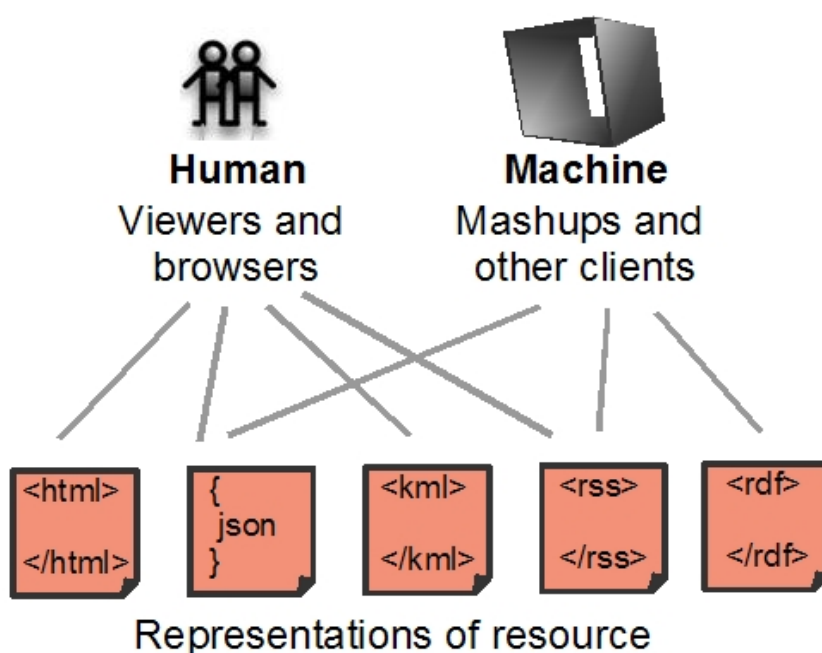


Figure 3.4: Resources may be consumed by both human end users and other software systems.

A resource is typically represented in one or more document format. If there is more than one representation available, the client and server have to agree on which one to select. This may be done by *content negotiation* as defined by the HTTP specification. Common document formats for resource representation are

- **HTML** – the language of the Web, used to create web pages.
- **XML** – defines syntax to create other languages. RSS for web feeds and KML for map layers are examples of XML based languages.
- **JSON** – simple message format for machine to machine communication, often used by rich internet applications.
- **RDF** – a metadata data model, used to describe resources by adding semantics.

The inherent ability to represent an information resource in more than one format is one of the strength of the Web-oriented Architecture. Some document formats are aimed at human consumption, whereas others are meant for machine-to-machine communication (see Figure 3.4).

### 3.3 Web 2.0

Web 2.0 is a term that is used to describe the changes seen in utilization of the World Wide Web. Web 2.0 does not describe a new version of the web or a particular new technology as the term may suggest. Rather, as said, it expresses changes in the utilization of the Web and promotes some principles that survived the bursting of the dot-com bubble. By now it is very much a buzz word lacking a clear definition, but that does not mean it is useless.

Tim O'Reilly has been active in promoting the term Web 2.0. According to O'Reilly (O'Reilly, 2005) some principles for the Web 2.0 applications are:

- The web as a platform
- Harnessing Collective Intelligence
- Data is the Next Intel Inside
- End of the Software Release Cycle
- Lightweight Programming Models
- Software Above the Level of a Single Device
- Rich User Experiences

O'Reilly's principles are about design patterns and business models. These are high level principles. Nevertheless, these principles embrace WOA, and WOA is one obvious part of a realization of these principles. It is also worth noticing that O'Reilly's descriptions of Web 2.0 are quite design oriented. Lately, the use of the Web 2.0 term seems to focus less on the technical, architectural and design issues and puts an even greater emphasis on the social aspects of the Web.

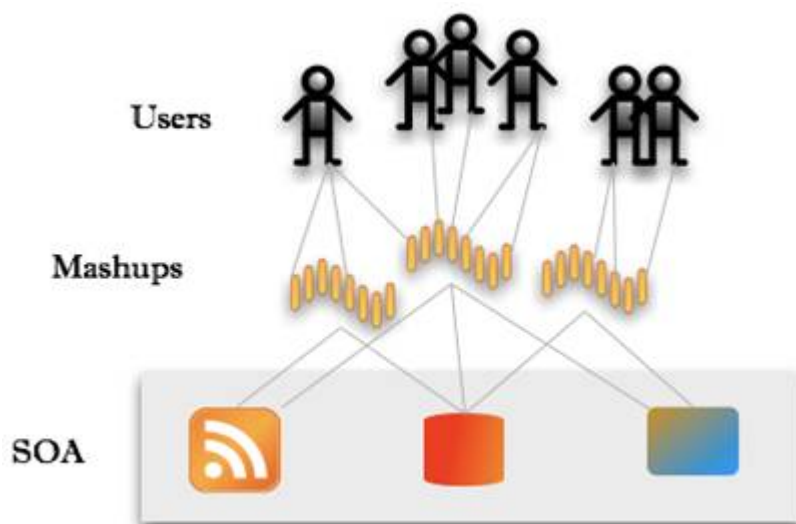




Enterprise 2.0 and Web 2.0: In Enterprise 2.0 management plays an important role, while in Web 2.0 management is absent – there is no "the management" for Web applications. The presence of management yields possibilities to promote the technology, and likewise, to constraint the use of the technology. With WOA being so closely related to Web 2.0, it is also likely that this difference (management vs. no management) is important when considering WOA in an enterprise environment versus WOA on the Web.

### 3.3.3 Mashups

A mashup is a service or resource that uses several other services or resources, usually from different providers, to assemble a new service. The combination of resources is usually novel and the mashup provide a rapid, ad-hoc, solution to a user's needs. Mashups is something different from orchestration of services in a SOA. The concept of mashups fits well to the idea of loosely coupled resources in a WOA.



*Figure 3.6: Mashups provide a new layer between users and the services and resources (Warner, 2008)*

Mashups may provide a layer between users and the existing resources or services (see Figure 3.6). This layer renders possible user adoptions like combining multiple resources or services into new resources and user configuration or user adjustments. All this is possible without touching the underlying services or resources. While the bureaucracy related to "traditional SOA" becomes more and more obvious, WOA and mashups are much about avoiding such bureaucracy, and further to provide new functionality on top of already available services and resources.

Light solutions – as WOA and mashups represent – may be an important supplement to centrally provided IT solutions. To many user-defined, low-demand, and/or non-critical applications, centralized solutions may be too ambitious and non-efficient use of resources. Using mashups may be an approach to provide user-defined functionality at a correct level of ambition. Providing for mashups may also be a way to stimulate bottom-up processes in the organization which may bring innovation.

Reitan (2009) discuss the use of mashups for the Norwegian armed forces, and argue that mashups can provide for innovation related to the use of information technology. Further, the use of mashups may provide for a multitude of functionality and a better match between what functionality is offered and what is demanded.

### **3.4 The Semantic Web – further work**

Information resources and URIs are basic elements behind the ideas described in this report. Links between resources create a web of data, but the meaning or semantics of these links must be interpreted by humans. *The Semantic Web* (Berners-Lee, Hendler & Lassila, 2001) is an evolution of the World Wide Web, adding semantics in a machine readable form. The Semantic Web consists of a set of increasingly powerful technologies, from Resource Description Framework (RDF) defining a graph data model describing resources, to the Web Ontology Language (OWL) which is a knowledge representation language. URIs are used in the Semantic Web to identify resources, but here resources may as well be abstract concepts, ontology classes and so on. Within the context of a Web-oriented Architecture, RDF can be used to add metadata about information resources as well as creating a web of data suitable for automated browsing by software agents. OWL would allow for even more complex data manipulation, connecting information resources to defined ontologies and inferring new knowledge through reasoning over a set of rules.

This combination Web principles, user participation, service orientation and the Semantic Web is also the focus of a EU funded research program called Soa4All (Dominuge et al., 2008; Di Nitto et al., 2008). Figure 3.7 shows the important concepts of this project. The aim is to realize “[...] *a world where billions of parties are exposing and consuming services via advanced Web technology: the main objective of the project is to provide a comprehensive framework that integrates complementary and evolutionary technical advances*” (<http://soa4all.eu>).

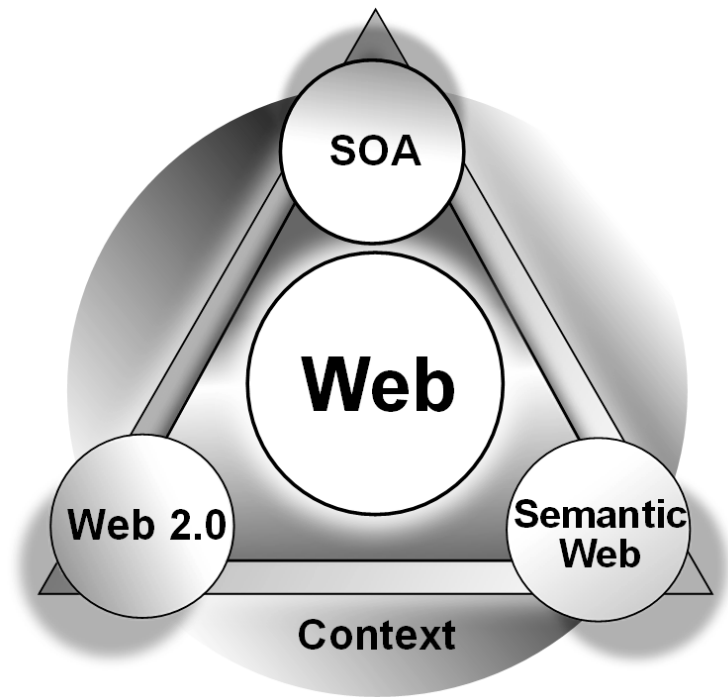


Figure 3.7: Cornerstones of SOA4All (Dominuge et al., 2008).

We will not go into further details about the underlying technologies or possible use cases for the Semantic Web combined with WOA+REST in a military setting in this report, but these ideas and the implications and consequences for the architecture of the INI should be further explored.

## 4 Military context RESTful services - Examples

In this chapter we will present three examples of applications implemented in a RESTful way within a Web-oriented Architecture. We will use these to illustrate issues discussed in the next chapter as well as the properties of WOA described in the previous one. Hopefully these examples will show both the technological aspects – resources, URIs, HTTP methods and representations – as well as the socio-technical aspects such as Web 2.0 and user participation. The services discussed here are implemented as demonstrators in project SINETT’s experimental information infrastructure – the SINI.

### 4.1 Blueforce Positions

This first example application handles positions of friendly forces. Of course, blue forces tracking systems already exist, but what we want to show is how such systems may be implemented within a WOA and what benefits this would bring.

#### 4.1.1 Background and motivation

Real-time reporting and sharing of own geographic position is one of the basic measures identified by Hedenstad et al. (2008). Sharing of positions was also identified as a service likely to show network effects. Consequently, the utility of such a service will in general increase exponentially with every new entity reporting its position to the service. The argument was then that the full potential is reached only when everybody is able to share their position. And thus, extra effort should be put in to make sure that everybody may do so.

Existing blue force tracking systems are mostly vertical silos, meaning that there is not a particular focus on sharing information or integrating with other systems or information sources. The Blueforce Positions service is data centric and a prominent goal is that everybody should be able to share their position. Existing systems most often require the reporting entities to carry specialized terminals. Giving everybody a specialized terminal is not necessary the best strategy. With the Blueforce Positions service no specialized equipment is absolutely required. However, specialized equipment may increase quality of the reported information.

The WOA+REST approach is well suited to provide for the flexibility that is needed to allow for multiple and tailored ways to report to the service, and continuously exploring ways to exploit the data in the service. The Blueforce Positions service identifies and locates representations of positions as URIs. For a given unit, say “bn1-2a”, a JSON representation of the position is identified by and located at `http://position.mil.no/bn1-2a.json`

We will talk more about security in the next chapter, but it is worth mentioning that when we talk about sharing a position in a WOA, we mean sharing with those that are *authenticated* to access that information. Some information may be available to all; some information may have access restrictions. The openness lies in the use of standard message formats such as GeoJSON and standard protocols such as HTTP in a distributed environment. This ensures that information is available when indented and that integration between different systems is made easy.

Another measure discussed by Hedenstad et al. (2008) is *battle space history* and a service handling different sorts of battle space information. By attaching a timestamp to all registered positions, past positions of units are made searchable. A battle space history service might then use the Blueforce Position as one of its inputs.

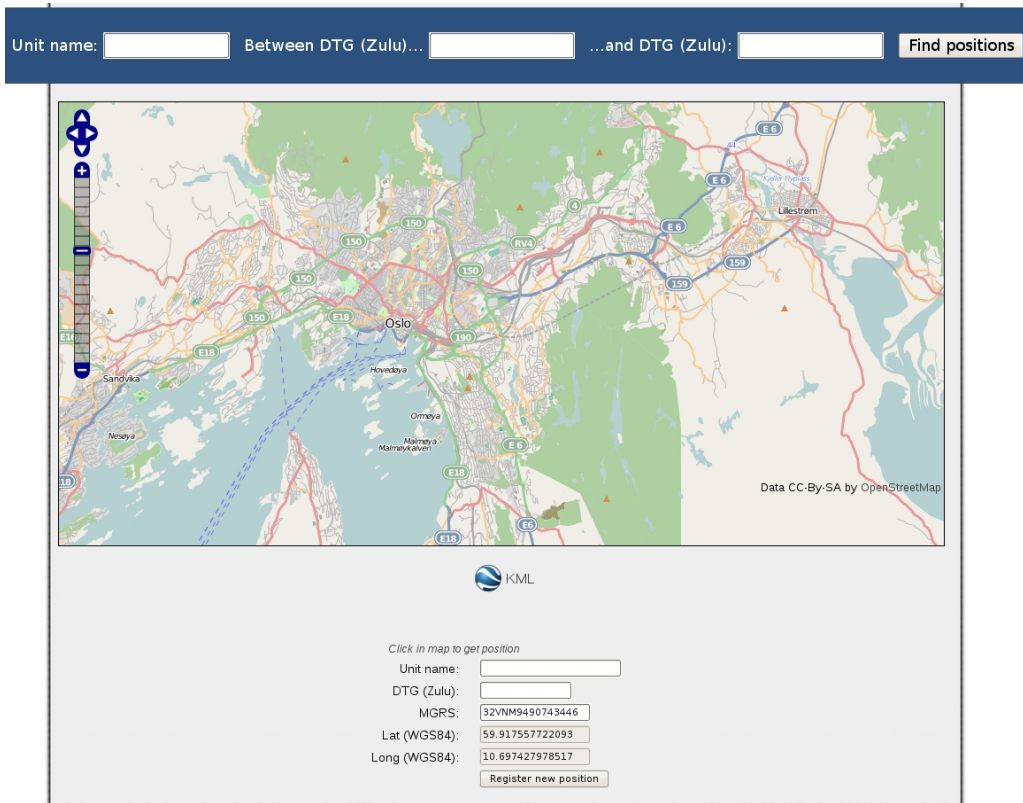


Figure 4.1: Graphical user interface in web browser.

#### 4.1.2 Service interface

The Blueforce Position service is implemented as a web application. For human end users there is a HTML graphical user interface shown in Figure 4.1 above. There is a search form for searching in all registered positions by unit name and/or dates. Further there is a map showing the result of the search and links to other representations of the search result, e.g. KML map layer format. At the bottom there is another form that may be used to report own position. Latitude and longitude values may be retrieved by clicking on the map. For other end users such as other systems or other clients (e.g. hand-held mobile clients), there are a RESTful interface to manipulate positions using the HTTP method GET to retrieve or PUT to update. The full interface including the REST API is defined in Appendix A.1.

For example would a GET request of the URI `http://position.mil.no/bn1-2a.json` return a JSON representation of the latest position that specific unit such as

```
{
  "type": "Point",
  "coordinates": [11.047, 59.975],
  "mgrs_pos": "http://mgrs.mil.no/59.975,11.047/",
  "timestamp": "2009-07-03T12:03:53Z"
}
```

PUTting a similar JSON message would result in the unit's position being updated. This message follows the specification of GeoJSON (Butler et al., 2008) with additional fields for MGRS position and timestamp.

The field for the MGRS position shows another feature of WOA; *interlinked data*. Similar to hypertext, a resource may include links to other resources. In this example there is a link to another service, the MGRS Translator described next.

## 4.2 MGRS Translator

This is a service translating between MGRS notation and WGS84<sup>1</sup> longitude/latitude decimal degrees for positioning. The service is a simple wrapper around a C library from the National Geospatial-Intelligence Agency<sup>2</sup>. This library may also be used to compile translators running separately on hand-held devices or used internally in other geospatial applications.

### 4.2.1 Background and motivation

Valaker et al. (2009) analyses challenges related to translations and verification of information when working in a cross cultural environment such as within a Network-based Defence where units from the army and navy are working together and sharing information. The navy uses latitude and longitude to define a position whereas the army still uses the military grid reference system (MGRS). When these two branches need to exchange positions, they have to agree on one of the formats or translate from one to another (either automatically or manually).

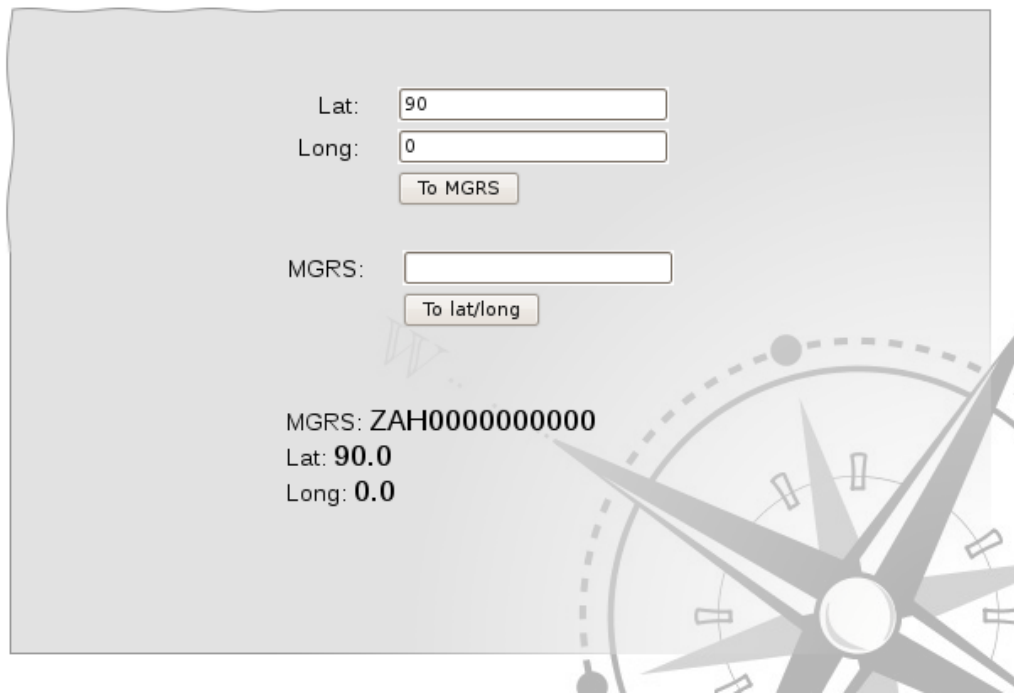
The MGRS Translator is a service translating between these two notations (MGRS and latitude/longitude). As for Blueforce Position service described above, there is a web based user interface (see Figure 4.2 below) displaying forms for entering positions in either MGRS or latitude/longitude.

---

<sup>1</sup> World Geodetic System revision 84, a standard for use in cartography, geodesy and navigation

<sup>2</sup> <http://earth-info.nga.mil/GandG/geotrans/index.html>

## MGRS - Lat/long Translator



Lat:

Long:

MGRS:

MGRS: ZAH0000000000  
Lat: 90.0  
Long: 0.0

Figure 4.2: Web form for translating positions.

### 4.2.2 Service interface

Other interfaces offer translation functionality for resources with the position details encoded into the URI. For latitude 59.917 and longitude 10.764 the corresponding MGRS is translated in a resource identified by `http://mgrs.mil.no/59.917,10.764/`. Retrieving this resource returns a GeoJSON representation of the position:

```
{  
  "type": "Point",  
  "coordinates": [10.764, 59.917],  
  "mgrs": "32VNM9863043482"  
}
```

A similar response is returned when translating the other way, say using the URI `http://mgrs.mil.no/32VNM9863043482/`. The full interface is defined in Appendix A.2



### 4.3 Common Environment Interpretation - Observations

The Common Environment Interpretation (CEI) is a service that at its basis is much like a pan-organizational, detailed battle log. The CEI is an organizational wide database for observations and reflections. It is based on the hypothesis that everybody may be a sensor and also inhabit an analytical capacity. Albeit, each contribution may be minor, the aggregation of many small observations and reflections may yield insight that is hard to reach in any other way. The CEI gives each and everybody somewhere to report or discuss abnormalities as a means to reach a common understanding of the environment.

The CEI makes it possible to share observations within the organization, across organizational borders and over time (e.g. between deployments). The service and its motivation are similar to the American Tactical Ground Reporting System (TIGR)<sup>3</sup>, but take the community building and social aspects of the service a step further.

The CEI aims to encourage a collective interpretation of the environment. The service allows for corrections to observations and comments to observations. These are activities normally associated with analysis, but with the CEI possibly a joint effort between the soldiers in the field and the intelligence specialists.

#### 4.3.1 Background and motivation

Everyone within an operation may make observations, make reflections and make their own hypothesis for what is the actual situation. Traditionally, minor incidents, concerns, and the individual soldier's personal analysis may not be reported. If such minor issues are reported, they are likely to follow the chain of command and being subject to its necessary filtering. The spreading of minor information entities is limited, and the probability that the information reaches the neighbours in the hierarchy, for which the information may be useful is close to zero. Also, such information is likely to be lost with time. Apparently this is an organizational problem. Nevertheless, the underlying challenge is the overwhelming efforts it will take to collect, manage, distribute and expose the right people to such information. The CEI addresses this challenge. The CEI is built on the assumptions that:

- Everyone is a sensor.
- Everyone may contribute useful reflections.
- Historical observations and reflections may also have value.
- Interaction between people concerning observations and interpretation of these will promote common understanding and generate new knowledge.

---

<sup>3</sup>See [http://www.darpa.mil/ipto/programs/assist/assist\\_tigr.asp](http://www.darpa.mil/ipto/programs/assist/assist_tigr.asp) for more information on the TIGR.

Further, the CEI will:

- make it easy to report observations and reflections, and ensure that no effort is necessary to receive and record these, such that minor incidents and concerns are more likely to be reported.
- make it easy to correct and comment on observations and reflections, such that new issues are brought to the table, and the analytic capacity of the collective is exploited.
- organize observations, ensure easy retrieval, and allow for personal configuration, such that all information is available in a manner where it is still possible to work with, and make sense of such information.

#### 4.3.2 Service interface

Given the nature and intention of this service, the main interface is the web application used by the users. The start page is shown in figure 4.3 below, and contains a simple registration form, a map, links to other formats such as RSS, RDF and KML and lists of the latest observations and a tag cloud showing the most used tags for all the observations. There is a more advanced form (see figure 4.4) available as well – this one giving advanced users the ability to attach images and classify the observation according to the APP-6A/MIL-STD-2525B hierarchy.

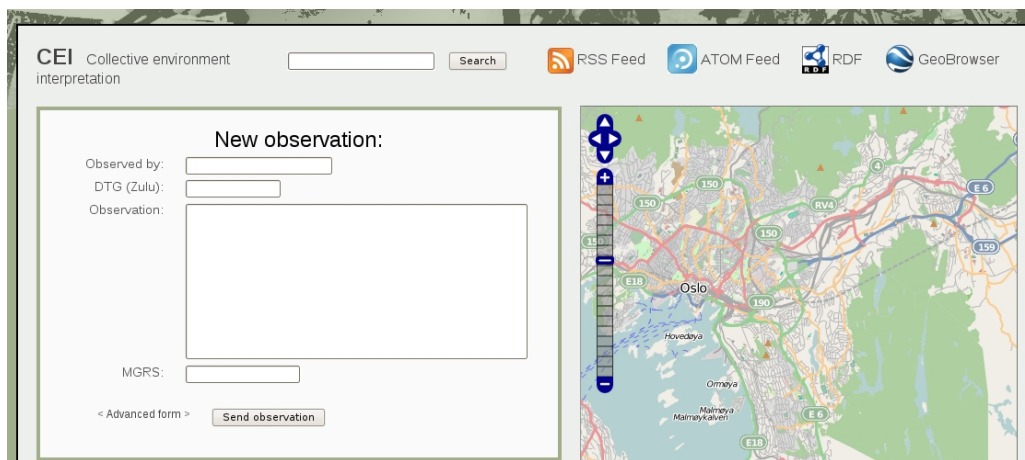


Figure 4.3: Web application start page with registration form.

After an observation is registered, users may comment on it, tagging it further and attaching more images. Both content and tags are searchable and RSS feeds based on tags are available such that users may subscribe to certain observations. As the tags themselves is defined by the users, exactly what categories one might subscribe to is both extendible and could change over time as requirements change. The full interface is defined in Appendix A.3.

### New observation:

Observed by:

DTG (Zulu):

Observation:

MGRS:

Lat (WGS84):

Long (WGS84):

DTG slutt (Zulu):

Tags:

Attach image:

APP-6A:

Affiliation:

Battle dimension:

Ground track: Unit:

Function ID, click to select and drill down:

Status:

Size modifier:

Type modifier:

Country code:

Order of battle:

Figure 4.4: Advanced registration form with APP-6A/MIL-STD-2525B classification.

## 5 Discussion

In this chapter we will discuss some of the distinct characteristics of a Web-oriented Architecture. We will also look into further details of the differences between traditional SOA and WOA.

### 5.1 WOA in the Information Infrastructure

WOA is based around a client/server architecture and a post-pull approach to information management. This together with the shared focus on both human interactions with information systems as well as machine-machine integration makes WOA an architecture that promotes collaboration and cooperation. As stated before, post-pull allows for a more manageable information sharing regime. The use of URIs to identify and locate information resources also facilitates this, as URIs may be linked to, bookmarked, shared, emailed, discussed and so on. By utilizing concepts from the Web, adoption could be faster and understanding of the underlying technologies and principles more intuitive.

We have seen that RESTful services are distributed and decentralized. Another property of URIs is that they are globally unique. By globally unique we understand that there is no ambiguity regarding what resource the URI refer to. This uniqueness holds within an organisation such as a branch of the military, between organisations (or military branches) and globally between international coalition partners.

Another important concept from the Web is the *interlinking* of information resources. This is also an effect of using URIs, as a resource may easily contain links in the form of URIs. These links create a web of data. Generally, it is the links between nodes in the network that creates added value to the overall system. This is often referred to as network effects or Metcalf's law (Shapiro & Varian, 1999). We will discuss this network of information resources further when we compare how integration is done with traditional SOA and WOA below.

### 5.2 SOAP versus REST

A direct comparison between SOAP and REST does not make sense as REST is an architectural style whereas SOAP is a messaging protocol. As WOA is a subset of SOA it is also difficult to compare these, so what we will do in this section is to compare the traditional approach to SOA using SOAP with the approach to WOA using REST principles. We call these two approaches *WOA+REST* and *SOA+SOAP*.

There are both technical and functional differences. For many tasks, the technologies may even be interchangeable. Therefore, we need to take a look at some of the other factors that make these technologies differ. E.g. the design principles and business models associated with these technologies are different. The environments of which they have evolved within are quite different. As a result, they are not simply two different technologies or architectural different approaches, but rather two different suites of technology with norms and associated principles. Table 5.1 outlines some of these differences. In the following subsections we will discuss each of these in further details.

	<b>WOA+REST</b>	<b>SOA+SOAP</b>
Control	Decentralized	Centralized
Social aspect	Most Web 2.0 applications implemented using WOA principles	Absent
Abstraction	Resources	Methods
Integration	Hypermedia, web of data. Many and open ended integration points	Business processes. Few and well-defined integration points
Presentation	Interfaces for both humans and machines, unrestricted data representation	Only machine interfaces, only XML as data representation
Security	Application layer authentication, transport layer encryption. Host-to-host security.	WS-* protocols for security. Process-to-process security possible.

*Table 5.1: WOA+REST versus SOA+SOAP*

### 5.2.1 Control

In an enterprise, solutions are usually developed with the assumptions that there is one single authority in charge – someone is in control. As long as these assumptions hold, it is possible to force standardization and minimize heterogeneity. Further, intrusive solutions and strongly dependent services are still possible. SOA+SOAP is primarily applied where this assumption of "one architect" holds. For long this has been a reasonable assumption, but as systems become larger, includes entities outside the organization, the control span of this single authority is challenged (CTSB, 2000). Nevertheless, it is mostly in such environments that SOA+SOAP has evolved and has been deployed; with a centralized and bureaucratic orchestration to achieve interoperability.

The environment from where WOA+REST has evolved is very different. Most important, on the Web there is no single authority - no one is "in control" of the overall system. Heterogeneity is a given, and homogeneity is Utopia. Interoperability is reached by open, published interfaces and by adaptation, rather than coordination and making deals. Intrusiveness and coordination are not viable options. Being first, being large, being influential, being available or simply being useful, are all important factors for adaptation. "Being in control" is not an option in this environment since no one is actually in control.

Rosado and Castelo (2008) use the term "*shadow IT*" for systems built without central corporate approval. These systems are built because centrally controlled IT architectures are not deployed fast enough to solve real problems at the edge of the organization. Conflicts often arise between central IT departments (who want to centralize) and the sub-departments deploying these edge applications. Rosado and Castelo argue that shadow IT can in fact drive innovation and effectiveness, and that shadow and central IT may both be leveraged to achieve an effective, agile IT environment.

### 5.2.2 Social aspect

SOA+SOAP being an enterprise invention, its primary focus has been on transactions and implementing well defined business processes. Efficiency and streamlined processes are key aspects. Highly structured information is necessary and users are seldom invited to influence this. There is normally little room for personal adjustments or adaptations as focus is on completing a task or a transaction.

WOA+REST, with roots in the Web 2.0 principles, is much more focused on interaction than completing transactions. These solutions are more about facilitating something, without the technology imposing on users how work should be done. McAfee (2006) is writing: "*The technologies of Enterprise 2.0 are trying not to impose preconceived notions about how work should be categorized or structured. Instead, they're building tools that let these aspects emerge*". To let these aspects emerge, interaction and social aspects are important.

### 5.2.3 Focus of Abstraction

One of the most prominent differences between WOA+REST and SOA+SOAP is the focus of abstraction. By this we mean what aspect that is the main focus when modelling some domain. For SOA+SOAP it is *remote methods, verbs* or *action*. For WOA+REST it is *information resources, nouns* or *business objects*.

Positions, observations, units, events – these are all concepts that may be more natural to model as resources. The MGRS service from the previous chapter actually do an

action of *translating* between formats, meaning that it could be more naturally modelled as a Web Service method rather than a REST resource. Still, the MGRS example illustrates how easy it is to make such services available to both human end users as a web application and other software systems as a RESTful service.

#### 5.2.4 Integration

The different focus of abstraction has implications for how integration of information is done. Web Service orchestration is a term used to describe the act of creating business processes by composing various Web Services. Each step in such a process is an action, and some of them invoke Web Services on remote servers.

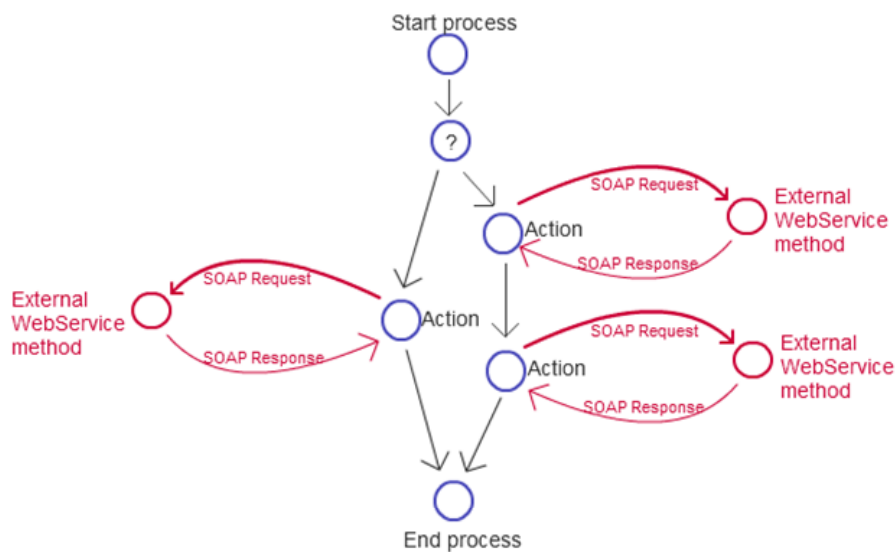


Figure 5.1: External methods are steps in a business process.

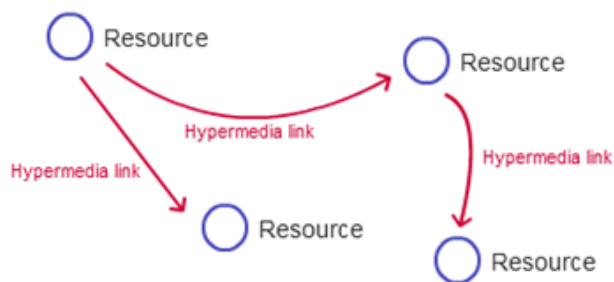


Figure 5.2: Resources contain links to other resources creating an interlinked web of data.

Figure 5.1 above shows a generic example of a business process. A business process is a structured, repeatable activity that is essential to a business or an organization and

that produces a specific service or product. There are standard languages such as BPEL for describing and implementing business processes. As these processes perform some action they are usually themselves available as traditional SOA Web Services.

Within a WOA, information is integrated through a web of data as shown in figure 5.2. The hypermedia structure where information resources contain links to other resources creates this web. Similar to how we browse the web by clicking on links, software clients may browse the web of data by following the URIs.

### 5.2.5 Presentation

We saw in the previous section 3.2.2 that RESTful services may offer different representations or formats of the same resource. SOAP on the other hand lives strictly in the land of XML. Table 5.2 below shows other technical differences between WOA+REST and SOA+SOAP.

	<b>WOA+REST</b>	<b>SOA+SOAP</b>
Protocol	HTTP	HTTP, SMTP, JMS, (limited, but extensible)
Interface	Uniform, only GET, PUT, POST and DELETE	Unrestricted, different contract for each service.
Presentation	Interfaces for both humans and machines, unrestricted data representation	Only machine interfaces, only XML as data representation

*Table 5.2: Technical differences.*

First of all, WOA+REST is tightly bound to the HTTP application layer protocol. All interfaces of RESTful services are uniform by the use of the HTTP methods to manipulate resources and URIs to identify and locate them. With traditional SOA, one is free to select other protocols such as SMTP or JMS, even though HTTP is by far the most popular.

Traditional SOA view these protocols as the *transport* for SOAP. From the OSI and TCP/IP models discussed above in section 3.2.1 we have that HTTP, SMTP and JMS belong to the *application layer*, and that TCP (or UDP) belong to the transport layer. Actually, if using the OSI model, the SOAP protocol is the application using the application layer protocols. We see that SOA+SOAP adds at least another layer, and that it views the whole Internet stack (HTTP and TCP/IP) as the transport. Many of the additional WS-\* standards add yet more complexity. WOA is more in line with the concept of the OSI model, as it is based on Web technologies and protocols.



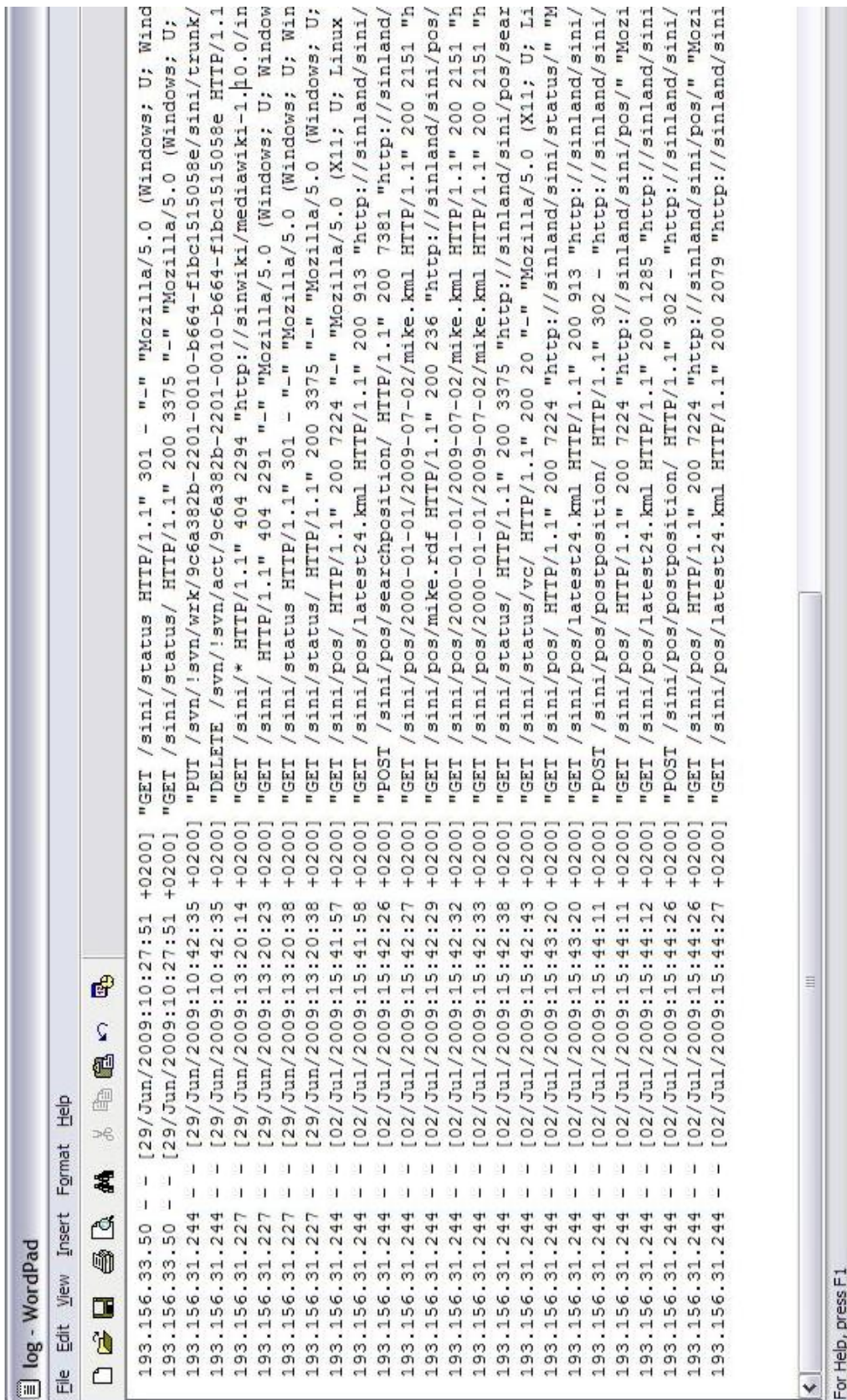


Figure 5.3: Extract from web server access log.

## 5.2.6 Security

One reason for using HTTP with SOAP is that HTTP slips unrestricted through most firewalls. HTTP was thus the most pragmatic choice for easing adoption and implementation of Web Services using SOAP. This, of course, is contrary to the purpose of firewalls. Monitoring SOAP traffic going over HTTP requires the firewall to first recognize the body as XML, then that the XML actually is SOAP. It would still be difficult to figure out exactly what that SOAP message is supposed to do, as there is no uniform interface. In other words, other security measures are needed for SOA+SOAP.

Considering WOA+REST on the other hand, interpreting the HTTP traffic is straightforward. The URIs and HTTP methods leave a clear audit trail in the web server access log (Mooney, 2007). Figure 5.3 above shows excerpts from such a log. The fields of the log are roughly client IP address, timestamp, HTTP request method, URI, HTTP version, HTTP response code, size of response body, and details about the client (web browsers in these instances). There exists a multitude of analyzing tools for such logs.

SOA+SOAP web services often use security mechanisms from HTTP (authentication) and TCP (TSL or SSL encryption). There exist other security protocols for SOAP such as WS-Security that offers application to application security and allows for more targeted measures.

## 6 Challenges, consequences and opportunities for the Norwegian Armed Forces

In this chapter we will be more specific about possible consequences of a Web-oriented approach when implementing parts of the INI.

### 6.1 WOA or SOA

With respect to traditional SOA, WOA offers an alternate approach to service orientation. These two approaches are similar, but not strictly overlapping. Are your design process bottom-up or top-down? Are you modelling business processes or information resources? Hinchcliffe (2008c) views WOA and traditional SOA as having different, but overlapping focus (see figure 6.1). Further, Hinchcliffe (2008a) argues that “[...] *most of the top-down activities that SOA initiatives have been putting in place, such as governance and cross-functional business architecture alignment, are just as appropriate -- if not more so -- when it comes to making WOA successful.*”

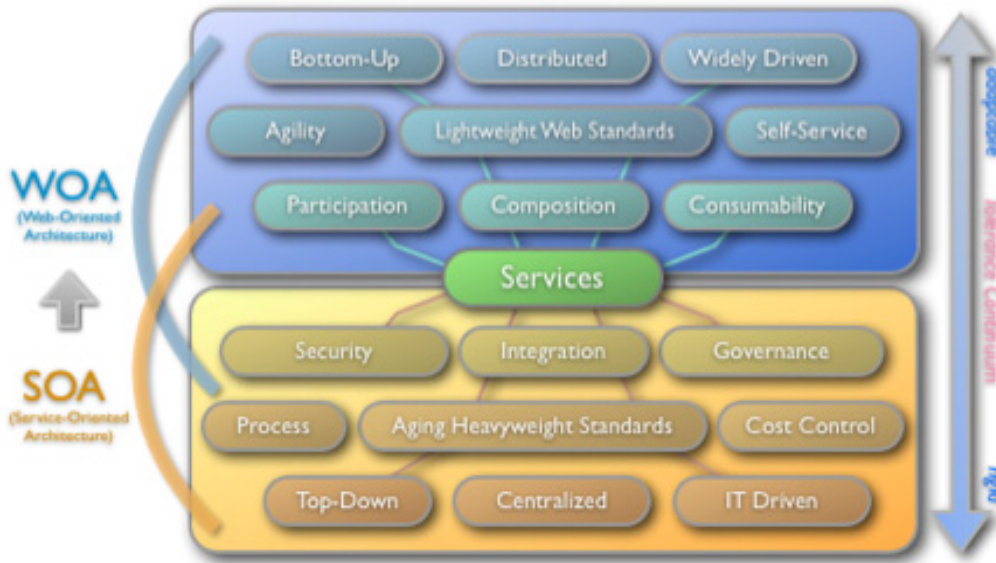


Figure 6.1: WOA versus SOA: Not a competition; an evolution of focus (Hinchcliffe, 2008c)

As we see, WOA+REST and SOA+SOAP can be used complementary. For Norwegian Armed Forces this may mean using WOA+REST for “edge applications” – that is agile services at the edge of the organisation and possibly for prototyping new inventive services. In the work of Hedenstad et al. (2008), several new services were suggested. A WOA+REST architecture for these services, at least initially, would make sense since they would be new services with functionality for which the organization has no experience. Establishing thorough requirements for such services is close to impossible: A way of working is not set and procedures should be left to evolve together with the formation of the new services. The advantages of a WOA+REST architecture could be exploited for rapid prototyping and continuous development and innovation together with personnel in the field. SOA+SOAP, on the other hand, may be exploited for established or centrally governed services. Further, it is also possible to include RESTful services into a SOAP, UDDI and WSDL infrastructure using WSDL version 2.0 (Mandel, 2008).

As the SOA+SOAP approach is often taken as given for new projects, we suggest that The Norwegian Armed Forces try to better understand the type of problem one is developing for, and from there do an analysis of what a WOA+REST approach would mean for that particular problem and especially the co-evolution of services and organization.

## 6.2 Web based

Services implemented using WOA and REST principles need to be run on web servers. The Norwegian Armed Forces needs the infrastructure and competence to operate and maintain both the servers and other infrastructure elements such as DNS name servers. On the client side, a modern web browser is required. In practice this means Internet Explorer version 7 or newer, or any of the more recent releases of Opera, Firefox or Safari web browsers.

A big advantage of the web based client-server architecture is that rolling out new services or applications do not require any install on the client. All that is needed is to promote and advertise the new service (and its location on the network), and the application will run in the standard web browsers on the client. This also means that the services will be available on any machine connected to the network.

RESTful services, however, do offer more than purely being applications running over a network. The HTTP methods offer an API to manipulate resources (or more specifically, *representations* of resources). For the INI of the Norwegian Armed Forces, this implies that clients (including, but not restricted to web browsers) will get access to data and information that at present is hidden in databases and vertical silos. There are HTTP methods for both reading (GET) and writing (PUT/POST) information. The Norwegian Armed Forces can start with introducing APIs for reading information resources over the Web as a first step and then add functionality for writing and updating those resources at a later stage.

## 6.3 Data-driven

Traditional SOA was supposed to be business-driven, meaning that processes and methods should be designed from a business perspective, but has turned out to be rather technological heavy. WOA is data-driven, focusing on business objects as information resources and how these link to each other. New services may be constructed by combining other resources or services using mashups. A presentation by ACT (Allied Command Transformation) at a recent RTO (Research and Technology Organization) workshop<sup>4</sup> describes mashups as “*[the] area with the greatest potential for cost effective value add to SACT across each of the four application areas*”. The Norwegian Armed Forces should identify the important business objects that exists and which are interesting resources to be shared and integrated in a NBD. We have from previous chapters that resources are nouns such as capabilities, capacities, observations, positions, operation plans, intelligence analysis and so on.

---

<sup>4</sup> MSG-074: Exploiting Commercial Games and Technology for Military Use - 7th Workshop

Focusing on business objects and making data available as information resources is consistent with the data strategy presented in (NATO NEC, 2005). The post-pull nature of WOA, the web of data and the possibility of creating mashups are all properties supporting the “*post before processing*” approach envisioned in (NATO NEC, 2005).

Resources must be identified on the network using URIs, so a common naming policy for URIs may be useful. Further, to ease integration, resources should be represented by standard data format types where such exists. HTML is given for web applications, but for other representations there are many choices. JSON and XML are by themselves not standard data format types, but rather languages to create such types. Some examples of de facto standards that have emerged are: Atom, RSS and GeoRSS for information feeds; KML and GML for displaying information in geobrowsers; GeoJSON for various geographic data structures; and RDF for metadata and relationships between resources.

#### **6.4 User participation**

Extensive user involvement appears to be a key feature of a successful future INI. In chapter 3.3 some opportunities associated with the term Web 2.0 and Enterprise 2.0 were discussed. The ability to exploit such opportunities within the INI will depend on the ability to include the INI's users to contribute content and to participate in the formation and the evolution of the INI. In short, the challenges are to (1) allow the users to participate, (2) design the INI and its parts for participation and (3) ask for and promote participation.

The first is related to what degree bottom-up processes is encouraged and left to evolve in the organization, such to provide for innovation and to learn from experience. This is much a management issue, and for example the ability and opportunity of a leader to monitor instead of using strict control is central.

The second challenge is where WOA+REST may play an important role. This challenge is again two parted. At first it is the design and architecture of the individual services. O'Reilly (2004) talks about “Architecture of participation” to describe the nature of systems designed for user contribution. The openness, flexibility, personalization are all factors that promote user contribution and is likely to make it easy to contribute. Secondly, the architecture of the INI gives the premises for integration of services and for further development. This is where WOA excels. A WOA with its light services is an excellent basis for further developments; new functionality build on top of existing services or integrated with these. Mashups are, as mentioned in chapter 3.3.3, a way for users to come close to the development of new functionality, or in some cases to put together the functionality themselves.

The third challenge is to ask for and promote participation. This is again a management issue. Many of these services are likely to show network effects – they provide marginal increasingly (think: exponentially) more value with every new person or entity that participate. Or for an INI: The INI is likely to be more valuable the more services are available. This is important to remember when designing services or when deciding for an architecture. Services like the Blueforce Positions (4.1) and the Common Environment Interpretation (4.3) are likely to need assistance to get rolling and become valuable services. Such assistance of infant services is a well known necessity of this type of services and often addressed as bootstrapping. The presence of management yields possibilities to promote the services and give incentives for use and participation. The management's actions related to such services are very likely to decide whether a service will succeed or fail.

## **6.5 Security**

As users contribute more with generating and maintaining content, users may also be given more responsibility with respect to trusting and assessing the reliability of both the services themselves as well as the information they offer.

Security has not been thoroughly discussed in this report. Even though we encourage a creative and bottom-up approach to designing new services, the Norwegian Armed Forces should define common security policies and authentication and authorisation mechanisms. These should cover both SOA+SOAP as well as WOA+REST type services. Identity management systems are commercially available, and single sign-on solutions for distributed services have been implemented for the Norwegian eGovernment solution MinSide.no.

## **6.6 No release cycles**

In a complex world requirements change over time. Sometimes requirements are not clear before the services are actually in use. WOA+REST allows for fast and easy development of new services as well as fast and easy adjustments of existing services. The development of new services will be more resilient as you may discover failures early and often rather than late and catastrophic.

Online services tend to have different life cycles than the traditional software applications. There are no clear cut release cycles, rather the services are in a state of continuous development – they live “forever”. This also have consequences for the IT operations organization as there are less clear boundaries between development and operations of the services. These two activities will often be done in parallel, by the same people, and often with active user participation during the development.

The obvious consequence for the Norwegian Armed Forces is that one does not purchase software products, but rather keep teams to operate and continuously develop services. Consequently, IT costs are moved from investments to operations – A possible obstacle within the current regime and its measure of merits.

## 7 Conclusion

The Web is an actual implementation of WOA. The underlying technologies and principles have been proven and tested in mankind's most extensive information system. By many means WOA and REST are embodiments of best practices gained after years of experience with regard to scalability, availability and ease of both use and implementation.

Information sharing and collaboration are basic tenets in a Network-based Defence. With WOA this is ensured by a post-pull, decentralized architecture consisting of a network of information resources. We have seen that WOA is in accordance with NBD, NNEC and NCW theory by being data-driven and information oriented. The focus on information resources and data rather than on processes gives organizations and users the ability to be agile – the technology does not impose predefined, and hard to change, rules and restrictions on business processes.

REST and WOA may be used complimentary to traditional approaches to service-orientation. WOA+REST may be more appropriate for “edge applications” where rapid prototyping and deployment is essential, where requirements are uncertain or change often, or where user participation gives additional value. User collaboration is further facilitated with WOA by the focus on human end users, as well as the more bottom-up approach of Web/Enterprise 2.0 towards control and user participation. WOA and REST are technologies enabling and enhancing collaboration as well as supporting new innovative applications. We have shown examples of military use cases such as the Common Environment Interpretation service where a RESTful implementation combined with Web 2.0 principles let users participate in order to add value, as well as defining interfaces for integrating with other systems. It is Network-based Defence development made easier.

## References

- Alberts, D.S., Garstka, J.J., & Stein, F.P. (2000). *Network Centric Warfare. Developing and Leveraging Information Superiority*. 2<sup>nd</sup> ed. CCRP Publication Series, Washington D.C.
- Alberts, D.S., & Hayes, R.E. (2003). *Power to the Edge: Command and Control in the Information Age*. CCRP Publication Series, Washington D.C.
- Benjamins, R., Davies, J., Dorner, E., Domingue, J., Fensel, D., Lopez, O., Volz, R., Wahler, A., & Zaremba, M. (2007). Service Web 3.0. *STI Innsbruck Technical report*. <http://www.sti-innsbruck.at/results/browse/technical-reports/details/?uid=35>
- Berners-Lee, T., & Cailliau, R. (1990). World Wide Web: Proposal for a HyperText Project. <http://www.w3.org/Proposal.html>
- Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American Magazine, May 2001*.  
<http://www.scientificamerican.com/article.cfm?id=the-semantic-web>
- van Bezooijen, B.J.A., Essens, P.J.M.D., & Vogelaar, A.L.W. (2006). Military Self-synchronization: An exploration of the Concept. *Proceedings of the 11th ICCRTS*.
- Bughin, J., Manyika, J., & Miller, A. (2008). McKinsey Global Survey - Building the Web 2.0 Enterprise. *McKinsey Quarterly*. July 2008.
- Bustamante, M.L. (2007). Making sense of all these Crazy Web Service Standards. *InfoQ*. <http://www.infoq.com/articles/ws-standards-wcf-bustamante>
- Butler, H., Daly, M., Doyle, A., Gillies, S., Schaub, & T., Schmidt, C. (2008). The GeoJSON Format Specification. <http://geojson.org/geojson-spec.html>
- Computer Science and Telecommunications Board, CSTB (200). Making IT Better: Expanding Information Technology Research to Meet Society's Needs. The National Academy Press, Washington, D.C.  
[http://books.nap.edu/openbook.php?record\\_id=9829](http://books.nap.edu/openbook.php?record_id=9829)



- Di Nitto, E., Gonzales-Cabero, R., Hamerling, C., Kopecký, J., Shafiq, O., Vitvar, T., Ripa, G., Xu, L., & Zuccalà, M. (2008). Design Principles for a Service Web. *SOA4All.EU, Fundamental and Integration Activities, Service Web Architecture*. <http://www.soa4all.eu/resources.html?func=startdown&id=25>
- Domingue, J., Fensel, D., & Gonzales-Cabero, R. (2008). SOA4All, Enabling SOA Revolution on a World Wide Scale. *SOA4All.EU*
- Fielding, R. (2000). Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine.
- Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T (1999). Hypertext Transfer Protocol – HTTP/1.1. RFC2616 <http://tools.ietf.org/html/rfc2616>
- Hafnor, H. (2006). INI som Nettsentrisk virksomhetsomgivelse – Bruk av ”Enterprise Metadata” og ”Communities of Interest” (COIs). Kjeller: FFI-rapport 2006/00862.
- Hedenstad, O.E., Bentstuen, O.I., Bjordal, H., Leere, A.B., Reitan, B.K., Schjeldrup, T.E., & Sendstad, O.J. (2008). (U) Prosjekt 1092 NbF Implementasjonsplan – anbefalte tiltak. Kjeller: FFI-rapport 2008/01350.
- Hinchcliffe, D. (2008a). The SOA world begins considering Web-Oriented Architecture in earnest. *Dion Hinchcliffe’s Blog – Musings and Ruminations on Building Great Systems*. <http://hinchcliffe.org/archive/2008/09/08/16676.aspx>
- Hinchcliffe, D. (2008b). What Is SOA? It’s the Future of Service-Oriented Architecture. *Dion Hinchcliffe’s Blog – Musings and Ruminations on Building Great Systems*. <http://hinchcliffe.org/archive/2008/02/27/16617.aspx>
- Hinchcliffe, D. (2008c). The WOA story emerges as better outcomes sought for SOA. *Dion Hinchcliffe’s Blog – Musings and Ruminations on Building Great Systems*. <http://blogs.zdnet.com/Hinchcliffe/?p=213>
- Jacobs, I. and Walsh, N. (2004). Architecture of the world wide web, volume one. <http://www.w3.org/TR/webarch/>.
- Johnsen, F.T., Flathagen, J., Gagnes, T., Haakseth, R., Hafsvøe, T., Halvorsen, J., Nordbotten, N.A., & Skjegstad, M. (2008). Web services and service discovery. Kjeller: FFI-rapport 2008/01064.

- Lund K., Eggen, A., Hadzic, D., Hafsøe, T., Johnsen, F.T. (2007). Using Web Services to realize Service Oriented Architecture in military communication networks, published in IEEE Communications Magazine. Kjeller: FFI-rapport 2007/02341.
- Mandel , L. (2008). Describe REST Web services with WSDL 2.0. *IBM developerWorks*. <http://www.ibm.com/developerworks/webservices/library/ws-restwsdl/>
- Masinter, L., Fielding, R., and Berners-Lee, T. (2005). Uniform resource identifier (uri): Generic syntax. <http://www.gbiv.com/protocols/uri/rfc/rfc3986.html>.
- McAfee, A. (2006). Enterprise 2.0: The Dawn of Emergent Collaboration. *Sloan Management Review*, Reprint 47306; Spring 2006, Vol. 47, No. 3, pp. 21-28
- McKendrick, J. (2006). Web services standards: 60 and counting. *A ZDNet blog*. <http://blogs.zdnet.com/service-oriented/?p=553>
- Mealling, M. (Ed), Denenberg, R. (Ed) (2002). Uniform Resource Identifiers (URIs), URLs, and Uniform Resource Names (URNs): Clarifications and Recommendations. *W3C URI Interest Group*. <http://www.rfc-editor.org/rfc/rfc3305.txt>
- Mooney, R. (2007). Security and Web Services. *XTech 2007*, “The Ubiquitous Web”, Paris France. <http://2007.xtech.org/public/schedule/detail/37>
- Muehlen, M., Nickerson, J.V., & Swenson, K.D. (2005). Developing web services choreography standards--the case of REST vs. SOAP. *Decision Support Systems Volume 40, Issue 1, Web services and process management, July 2005*, 9-29.
- NATO NEC (2005). NNEC Data Strategy. NATO/EAPC, Version 1.1, EAPC(AC/322-SC/5)N(2005)0018.
- O'Reilly, Tim (2004). The Architecture of Participation. *O'Reilly Media*. [http://www.oreillynet.com/pub/a/oreilly/tim/articles/architecture\\_of\\_participation.html](http://www.oreillynet.com/pub/a/oreilly/tim/articles/architecture_of_participation.html)
- O'Reilly, Tim (2005). What Is Web 2.0 – Design Patterns and Business Models for the Next Generation of Software. *O'Reilly Media*. <http://oreilly.com/web2/archive/what-is-web-20.html>
- Pautasso, C., Zimmermann, O., & Leymann, F. (2008). RESTful Web Services vs. “Big” Web Services: Making the Right Architectural Decision. *Proceedings of the 17<sup>th</sup> International World Wide Web Conference*, ACM Press, Beijing China.

- Reitan, B.K. (2009). Mashups – noe for Forsvaret?. Kjeller: FFI-notat 2009/00001.
- Reitan, B.K., & Pålhaugen, L. (2004). Forventningene til Nettverksbasert Forsvar – 6 Tema. Kjeller: FFI-rapport 2004/04004.
- Reitan, B.K., & Hafnor, H. (2007). Sosiale teknologier for samhandling og nettverking: Fra publisering til deltagelse og sosial interaksjon. Kjeller: FFI-rapport 2007/02600.
- Rip, P. (2005). Enterprise Web 2.0. *EarlyStageVC*.  
[http://earlystagevc.typepad.com/earlystagevc/2005/10/enterprise\\_web\\_.html](http://earlystagevc.typepad.com/earlystagevc/2005/10/enterprise_web_.html)
- Rosado, P., & Castelo, R. (2008): Shadow IT: Edge Applications in a Service-Oriented Enterprise. *SOA Magazine Issue XVIII: May 2008*.  
<http://www.soamag.com/I18/0508-3.asp>
- Shapiro, C., & Varian, H.R. (1999). *Information Rules*. Harvard Business Press
- Smith, R. (2008). A Simpler Approach to SOA. *InformationWeek, August 11 2008 Issue*.
- Valaker, S., Danielsen, T., & Fidjeland, M.K. (2009). Oversettelse, verifikasjon og prioritering av informasjon. Kjeller: FFi-rapport 2009/00362.
- Valaker, S., & Fidjeland, M.K. (2008). (U) Chat and chatbot in use by Panserbataljonen during Exercise Vårminck 2008. Kjeller: FFi-notat 2008/01833.
- Vinoski, S. (2007). REST Eye for the SOA Guy. *IEEE Internet Computing, vol. 11, no. 1, pp. 82-84, Jan./Feb. 2007, doi:10.1109/MIC.2007.22*
- Warner, C., & Crupi, J. (2008). Enterprise Mashups Part II: Why SOA Architects Should Care. *The SOA Magazine, Issue XXI, August 2008*.  
<http://www.soamag.com/I21/0808-1.asp>
- Zimmermann, H. (1980). OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications, Vol. com-28, No.4, April 1980*.

## Abbreviations

API – Application Programming Interface/Application Protocol Interface  
ACT – Allied Command Transformation  
BPEL – Business Process Execution Language  
HTML – Hypertext Markup Language  
HTTP – Hypertext Transfer Protocol  
INI – Information Infrastructure  
IP – Internet Protocol  
IPsec – Internet Protocol Security  
IT – Information Technology  
JMS – Java Message Service  
JSON – JavaScript Object Notation  
MGRS – Military Grid Reference System  
NBD – Network-based Defence  
NCW – Network-centric Warfare  
OSI – Open Systems Interconnection  
OWL – Web Ontology Language  
RDF – Resource Description Framework  
REST – Representational State Transfer  
RTO – Research and Technology Organization  
SOA – Service-oriented Architecture  
SOAP – a messaging protocol (actually not an acronym)  
SSL – Secure Sockets Layer  
TCP – Transmission Control Protocol  
TLS – Transport Layer Security  
UDDI – Universal Description Discovery and Integration  
URI - Uniform Resource Identifier  
WGS84 – World Geodetic System of 1984  
WOA – Web-oriented Architecture  
WSDL – Web Service Definition Language  
WWW – the World Wide Web  
WS-I – Web Services Interoperability Organisation  
XML – Extensible Markup Language

## Appendix A Service interfaces

Note: All URIs shown below are examples of how services and resources could be identified and located in a military context. No services (at least not any of those listed below) are available at these locations p.t.

### A.1 BlueForce Positions

#### A.1.1 HTML/Web application interfaces

<b>URI</b>	<b><code>http://position.mil.no/</code></b>	
<b>Methods</b>	GET	Return .html page showing search form, map of current positions, and a form for registering own position
<b>URI</b>	<b><code>http://position.mil.no/postposition/</code></b>	
<b>Methods</b>	POST	Handling registration of own position. POST variables: unit_name - string lat - latitude, decimal degrees WGS84 long - longitude, decimal degrees WGS84
<b>URI</b>	<b><code>http://position.mil.no/searchposition/</code></b>	
<b>Methods</b>	POST	Searching for registered positions, showing result in map in html page and link to KML and RDF resources. POST variables: unit_name - string after - date on format YYYY-MM-DD before - date on format YYYY-MM-DD

#### A.1.2 REST API

<b>URI</b>	<b><code>http://position.mil.no/latest24.kml</code></b>	
<b>Methods</b>	GET	Retrieve KML representation of all unit's positions and tracks latest 24 hours.
<b>URI</b>	<b><code>http://position.mil.no/&lt;unit name&gt;.kml</code></b>	
<b>Methods</b>	GET	Retrieve KML representation of <unit name>'s position.
Example: GET <code>http://position.mil.no/bn1-2a.kml</code> returns		

```
<kml xmlns:rdf="http://earth.google.com/kml/2.2">
  <Document>
    <Placemark>
      <name>bn1-2a.kml </name>
      <description>2009-06-24T12:39:40</description>
      <Point>
        <coordinates>11.047,59.975</coordinates>
      </Point>
    </Placemark>
  </Document>
</kml>
```

<b>URI</b>	<b>http://position.mil.no/&lt;unit name&gt;.rdf</b>	
<b>Methods</b>	GET	Retrieve RDF representation of <unit name>'s position using the W3C Basic Geo vocabulary.

Example:

```
GET http://position.mil.no/bn1-2a.rdf
```

returns

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
  xmlns:mgrs="http://mgrs.mil.no/">
  <rdf:Description about="http://position.mil.no/bn1-2a">
    <geo:lat>59.975</geo:lat>
    <geo:long>11.047</geo:long>
    <mgrs:pos rdf:resource="http://mgrs.mil.no/59.975,11.047/" />
  </rdf:Description>
</rdf:RDF>
```

<b>URI</b>	<b>http://position.mil.no/&lt;unit name&gt;.json</b>	
<b>Methods</b>	GET	Retrieve GeoJSON representation of <unit name>'s position.
	PUT	Store position for <unit name> from GeoJSON representation

Example:

```
GET http://position.mil.no/bn1-2a.json
```

returns

```
{
  "type": "Point",
  "coordinates": [11.047, 59.975],
  "mgrs_pos": "http://mgrs.mil.no/59.975,11.047/"
}
```

Note that the coordinates array in the GeoJSON object lists longitude before latitude

## A.2 MGRS Translator

### A.2.1 HTML/Web application interfaces

<b>URI</b>	<b><code>http://mgrs.mil.no/</code></b>	
<b>Methods</b>	GET	Return .html page showing search form for translations between MGRS and latitude/longitude decimal degrees.

### A.2.2 REST API

<b>URI</b>	<b><code>http://mgrs.mil.no/&lt;mgrs string&gt;/</code></b>	
<b>Methods</b>	GET	Retrieve GeoJSON representation with lat/long point and MGRS for given <code>&lt;mgrs string&gt;</code> .
Example: GET <code>http://mgrs.mil.no/32VNM9863043482/</code> returns <pre>{   "type": "Point",   "coordinates": [10.763992, 59.917003],   "mgrs": "32VNM9863043482" }</pre>		
Note that the coordinates array in the GeoJSON object lists longitude before latitude.		
<b>URI</b>	<b><code>http://mgrs.mil.no/&lt;lat&gt;,&lt;long&gt;/</code></b>	
<b>Methods</b>	GET	Retrieve GeoJSON representation with lat/long point and MGRS for given <code>&lt;lat&gt;</code> and <code>&lt;long&gt;</code> values.
Example: GET <code>http://mgrs.mil.no/59.917,10.764/</code> returns <pre>{   "type": "Point",   "coordinates": [10.764, 59.917],   "mgrs": "32VNM9863043482" }</pre>		
Note that the coordinates array in the GeoJSON object lists longitude before latitude.		

## A.3 Collective Environment Interpretation

### A.3.1 HTML/Web application interfaces

<b>URI</b>	<b><code>http://cei.mil.no/</code></b>	
<b>Methods</b>	GET	Return .html page showing start page with registration form and map showing latest observations.

### A.3.2 REST API

<b>URI</b>	<b><code>http://cei.mil.no/register/</code></b>	
<b>Methods</b>	POST	Register new observation sending observation encoded as JSON in request body.
<b>URI</b>	<b><code>http://cei.mil.no/&lt;tag&gt;.rss</code></b>	
<b>Methods</b>	GET	Retrieve GeoRSS representation with lat/long point and description for all observations tagged with <tag>.
Example: GET <code>http://cei.mil.no/sof.rss</code> returns a RSS feed of all observations tagged with 'SOF'. The feed is set to reload every 10 <sup>th</sup> minute.		
<b>URI</b>	<b><code>http://cei.mil.no/latest24.rss</code></b>	
<b>Methods</b>	GET	Retrieve GeoRSS representation with lat/long point and description for all observations the latest 24 hours.
Example: GET <code>http://cei.mil.no/latest24.rss</code> returns a RSS feed of all observations registered the latest 24hours. The feed is set to reload every 10 <sup>th</sup> minute.		
<b>URI</b>	<b><code>http://cei.mil.no/latest24.kml</code></b>	
<b>Methods</b>	GET	Retrieve KML representation with description for all observations the latest 24 hours.
Example: GET <code>http://cei.mil.no/latest24.kml</code> returns a KML document to open in geobrowsers such as Google Earth.		