17th ICCRTS

“Operationalizing C2 Agility”

Title of Paper:

Towards operational agility using service oriented integration of prototype and legacy systems

Topics:
Primary:
Topic 7: Architectures, Technologies, and Tools
Alternatives:
Topic 8: Networks and Networking, Topic 5: Experimentation, Metrics, and Analysis

Authors:
Frank T. Johnsen, Trude H. Bloebaum, Ketil Lund, and Espen Skjervold
Norwegian Defence Research Establishment (FFI) P.O. Box 25 2027 Kjeller, Norway

Point of Contact:

Frank T. Johnsen

Norwegian Defence Research Establishment (FFI)

P.O. Box 25, 2027 Kjeller, Norway

frank-trethan.johnsen@ffi.no

Towards operational agility using service oriented integration of prototype and legacy systems

Abstract

NATO has identified Web services standards as a key enabler for interoperability between the different military systems of the various NATO nations. Interoperability is required on several levels: Network hardware compatibility (NATO has workgroups that are addressing this), network protocol compatibility (IP has been chosen by NATO), middleware compatibility, and so on. In this paper, we address Web services technology, that is, the middleware aspect of interoperability.

An operational military network is complex, consisting of heterogeneous networks. Some networks have infrastructure and high capacity, others are mobile ad hoc networks (MANETs) consisting of wireless links with varying throughput, error-rates, and mobility (so-called disadvantaged grids). Agile deployments can be achieved by leveraging the implementation, flexibility and interoperability that Web services can provide. This paper presents an experiment where prototype solutions and legacy systems were service-enabled and interconnected by employing a combination of established standards and bespoke solutions.

1. Introduction

Web services technology is becoming increasingly popular. Businesses use the technology not only as a middleware within their own corporate networks, but also across the Internet as a means of providing business-to-business services. The technology, being based on standards, makes it a simple and cost effective way to build loosely coupled systems. Since Web services are so successful in civil applications, it makes sense to attempt to employ this technology for military applications as well.

The NATO Network Enabled Capability (NNEC) Feasibility Study [23] presents a discussion of technology, focusing on the needs of future interoperable military communications. The information infrastructure has to allow for communication across system and national boundaries while at the same time taking legacy systems into account. The study identifies the Service Oriented Architecture (SOA) concept and Web services technology as the key enablers for NNEC.

The World Wide Web Consortium (W3C) definition of Web services [25]:

> A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

As pointed out by Alberts [16], *agility is the critical capability that organizations need to meet the challenges of complexity and uncertainty*. Because Web services represent such big advantages with respect to agility, interoperability, reuse and reduction of cost and development time, it is our goal to make the use of Web services as pervasive as possible within our national military networks.

Web services support both traditional request/response communication, as well as publish/subscribe - a well-known pattern for event-driven, asynchronous communication. The pattern is particularly well suited in situations where information is produced at irregular intervals, and has been identified as one of the core enterprise services (CES) by the NATO CES working group [24]. Simply speaking, publish/subscribe means that you will only receive the information that you have subscribed to. This concept utilizes a combination of push and pull. As opposed to a general ”push” mechanism there are benefits in that you may select (subscribe) to the information sent to you. Using pure “pull” mechanisms introduces a compromise between wasted requests (when no new information is available) and liveness/latency requirements. In this paper we focus on service-enabling legacy systems, offering their data to the information infrastructure using publish/subscribe. We present an experiment in which we connected non-SOA systems together in a SOA infrastructure. The goal of this technical experiment was to show how Web services can be used not only to build a flexible service infrastructure, but also as a means of including legacy systems into this infrastructure without having to change the systems themselves. This was accomplished by building a publish/subscribe infrastructure in which a service enabled client could subscribe to information from legacy systems.

An important benefit of performing such an experiment, in addition to it providing a proof-of-concept implementation, is that it illustrates the operational benefits that can be achieved through the use of a flexible and agile information infrastructure.

The remainder of this paper is organized as follows: In Section 2, we introduce the different components (i.e., the relevant standards and bespoke solutions) of our experiment. Section 3 presents the operational systems involved in the experiment, and describes how they were integrated. We focus on the lessons learned, so that others can benefit from our experiences. Section 4 concludes the paper.

2. Components

When performing experiments based on Web services, there are two main types of services involved:

- Core services, which provide infrastructure components which in turn will be used by other services. These are services such as service discovery, mediation services and publish/subscribe services.
- Functional services, which are the services that provide the information needed by user applications.

In our experiment, we implement a subset of the core services specified by the CES working group, in order to support a specific set of functional services. The functional services were an NFFI service and an incident reporting service. In this section we present the components we used in order to support these services.

2.1 Wrapping of operational systems

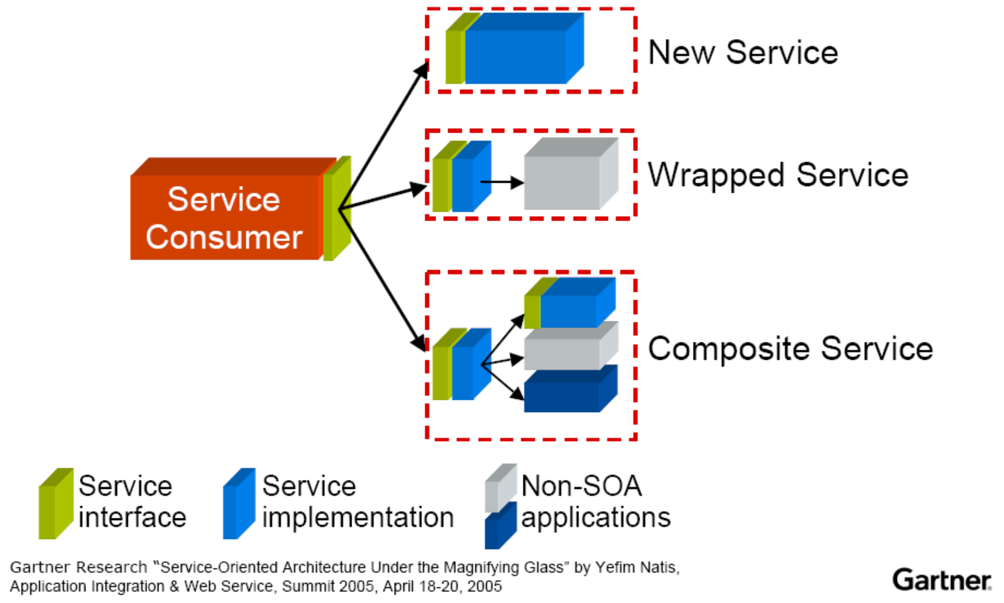

New Service
Service Consumer
Wrapped Service
Composite Service
Service interface
Service implementation
Non-SOA applications
Gartner Research "Service-Oriented Architecture Under the Magnifying Glass" by Yefim Natis, Application Integration & Web Service, Summit 2005, April 18-20, 2005
Gartner

Figure 2.1 Creating services

Services in a SOA can be realized in three different ways, as shown in Figure 2.1:

- They can be implemented as services from the start. This is what is normally done when implementing new systems.
- Existing (non-SOA) systems can be "wrapped", in the sense that a piece of software is placed in front of the system, presenting services to the outside world, while communicating "natively" with the wrapped system.
- Existing services and systems can be combined into new, aggregated services, by using software that coordinates access to the individual systems and services, and presents the aggregate as a new service.

None of the Norwegian operational systems used in the experiment offered the information we required through a service enabled interface, and we therefore used the wrapper approach on these systems. Our SOA viewer (see Section 2.3) has the ability to offer the information it receives from other services as a new service, thus representing a type of composite service.

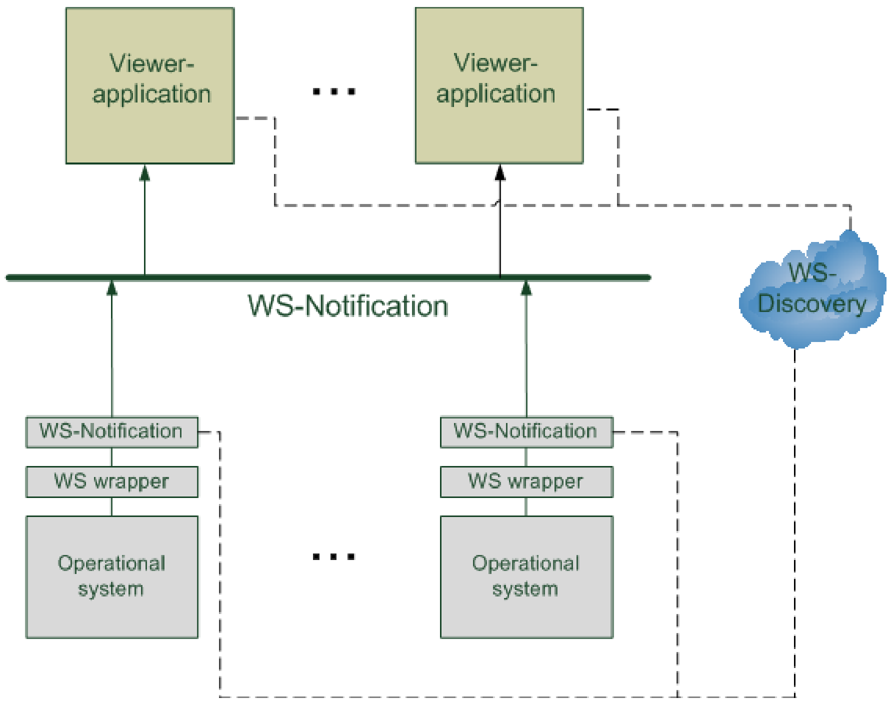


Figure 2.2	Wrapping systems

The general principle behind wrapping the services is shown in Figure 2.2. We leverage existing standards where applicable to facilitate interoperability, such as WS-Notification and WS-Discovery. These standards are introduced below. Each wrapped operational system delivers data (notifications) using WS-Notification, which is then responsible for delivering the notifications to all subscribers. In the experiment, only the SOA viewers acted as subscribers (in addition to being able to act as a service in itself, as described above). In addition, all services announced their presence using WS-Discovery, such that clients (in our case the viewer applications) could dynamically discover and invoke these.

2.2 Web services standards employed

2.2.1 WS-Notification

WS-Notification [3] is an OASIS standard defining publish/subscribe for Web services. There are three parts in the specification:

- WS-BaseNotification [4] which defines standard message exchanges that allow one service to subscribe and unsubscribe to another, and to receive notification messages from that service.

- WS-BrokeredNotification [5] which defines the interface for notification intermediaries. A notification broker is an intermediary that decouples the publishers of notification messages from the consumers of those messages; among other things, this allows publication of messages through a chain of proxies.

- WS-Topics [6] defines topic-based filtering using an XML model to organize and categorize classes of events into “Topics”. It enables users of WS-BaseNotification or WS-BrokeredNotification to specify the types of events in which they are interested.

In summary, the specifications standardize the syntax and semantics of the message exchanges that establish and manage subscriptions, as well as the message exchanges that distribute information to subscribers. For further discussion of publish/subscribe in general and WS-Notification in particular in a military context, see [2].

In the experiment we used a freely available implementation of WS-Notification called WSMG [10], which is provided as open source by the University of Indiana. The framework implements the WS-BaseNotification standard, among other things. The NC3A used Apache's ServiceMix ESB on their side, which also implements WS-Notification.

2.2.2 WS-Discovery

Discovery of Web services can be performed either by utilizing a service registry, or a decentralized non-registry based solution. There are three standards addressing Web services discovery, two registries and one non-registry solution. The registries, UDDI and ebXML, suffer from liveness and availability problems in dynamic environments, as we described in [19]. The third standard for Web services discovery is WS-Discovery [22]. It is better suited to dynamic networks than the registries due to a decentralized discovery mechanism, thus removing the single point of failure that a centralized registry constitutes.

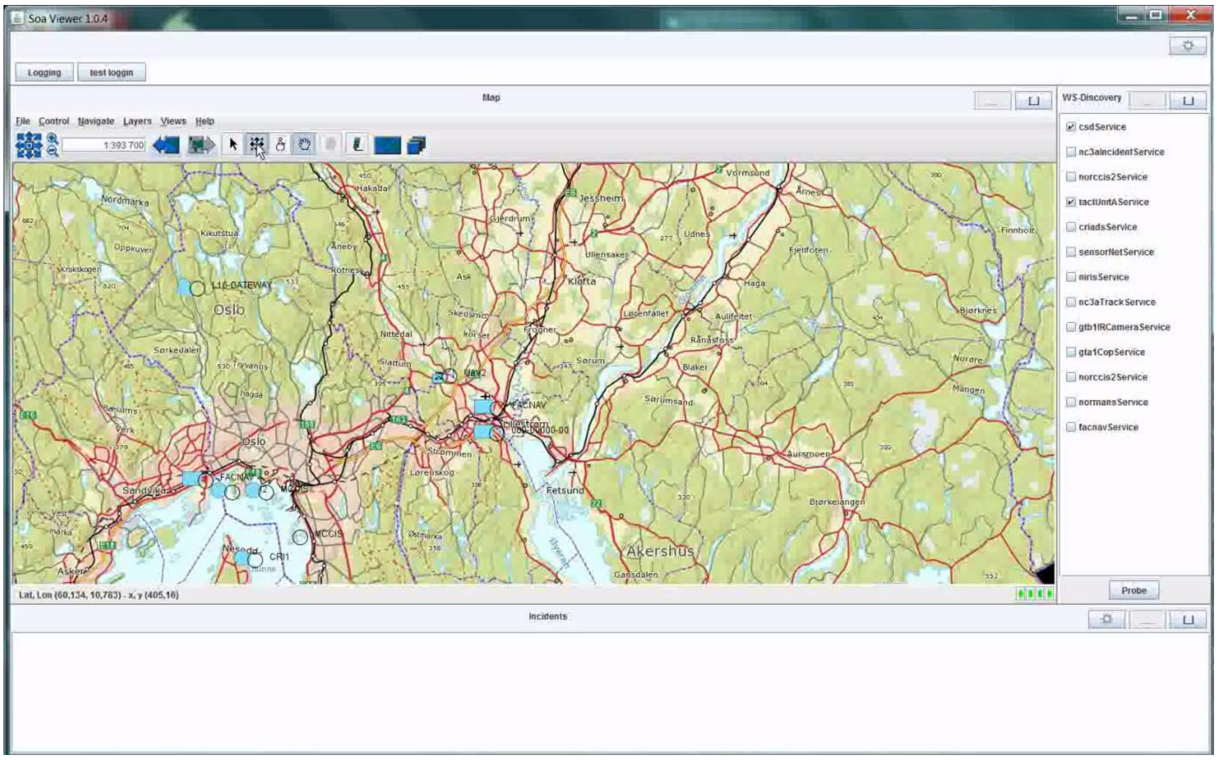
Previously, we have attempted to use WS-Discovery in a disadvantaged grid, and found that it unsuitable as it generated large traffic volumes and flooded the modem buffers [12]. However, by reducing the overhead of WS-Discovery using compression, it may be better suited for use in military networks as well. The recent W3C standard for efficient XML interchange (EXI) can potentially make WS-Discovery suitable for both civil and military networks. We have combined an open source implementation of WS-Discovery with an open source implementation of EXI, as described and evaluated in [13]. In the experiment we used this EXI-enabled version of WS-Discovery. The library was integrated into the SOA viewer, and used there to provide the user with an overview of available services. See the following section on the SOA viewer for further details.

2.3 SOA viewer

The SOA viewer was developed for use as an information integration and visualization core during the experiment. It was not intended to compete with existing visualization systems in any way, but rather to illustrate how existing services can be utilized by new software, in the same manner as we have used Google earth in previous experiments, e.g., [19].

The SOA viewer provides functionality for consuming, visualizing, aggregating and re-publishing services using established standards. The user is able to subscribe to different types of services, and the SOA viewer receives push-based notifications whenever the services offer updated information. WS-Discovery was chosen for publishing and discovering services, and the SOA viewer was integrated with the EXI-enabled library presented above. WS-Notification was chosen as the publish/subscribe protocol, because it is the publish/subscribe standard of choice in the NATO CES [24]. The SOA viewer's graphical user interface (GUI) lists all available sources found by using WS-Discovery, and when checked by the user, the services are subscribed to by sending a subscription message to the WS-Notification subscription manager.

Based on the mapping specified in the SOA viewer's configuration, the GUI will change to accommodate the current selection of chosen services, by hiding or showing different panels capable of visualizing different types of information. Specifically, geographic information-based services are presented in a map, instant messaging services are presented in a chat-panel, and incident reports are presented in a panel offering drill-down capability. By modifying the GUI based on the user's selections, the user is able to concentrate on relevant information only. This reduces information overhead, and facilitates more rapid and agile decision making.


Soa Viewer 1.0.4
Logging
test loggin
Map
File Control Navigate Layers Views Help
1:393 700
WS-Discovery
Probe
Incidents
Lat, Lon (60,134, 10,783) - x, y (405,18)

- [✓] csdService
- [ ] nc3aIncidentService
- [ ] norccis2Service
- [✓] tactUnitAService
- [ ] criadsService
- [ ] sensorNetService
- [ ] nirisService
- [ ] nc3aTrackService
- [ ] gtb1IRCameraService
- [ ] gta1CopService
- [ ] norccis2Service
- [ ] normansService
- [ ] facnavService

Figure 2.3 SOA viewer screenshot

In order to present geographic information-based services in a map, the SOA viewer was integrated with OpenMap, a Java-based open source toolkit for showing map data. The toolkit fetched maps from a Norwegian map provider, “Statens Kartverk”, using Web Map services (WMS). Thus, the SOA viewer was able to display detailed maps over Norway in multiple scales. The XMPP protocol was chosen for instant messaging in our experiments, as this is the protocol chosen for use in the Collaboration CES [24]. The SOA viewer offers multi-user chat by integrating with Ignite Realtime’s Smack API, communicating with XMPP servers such as Openfire and Prosody. The SOA viewer’s information flow is illustrated in Figure 2.4.

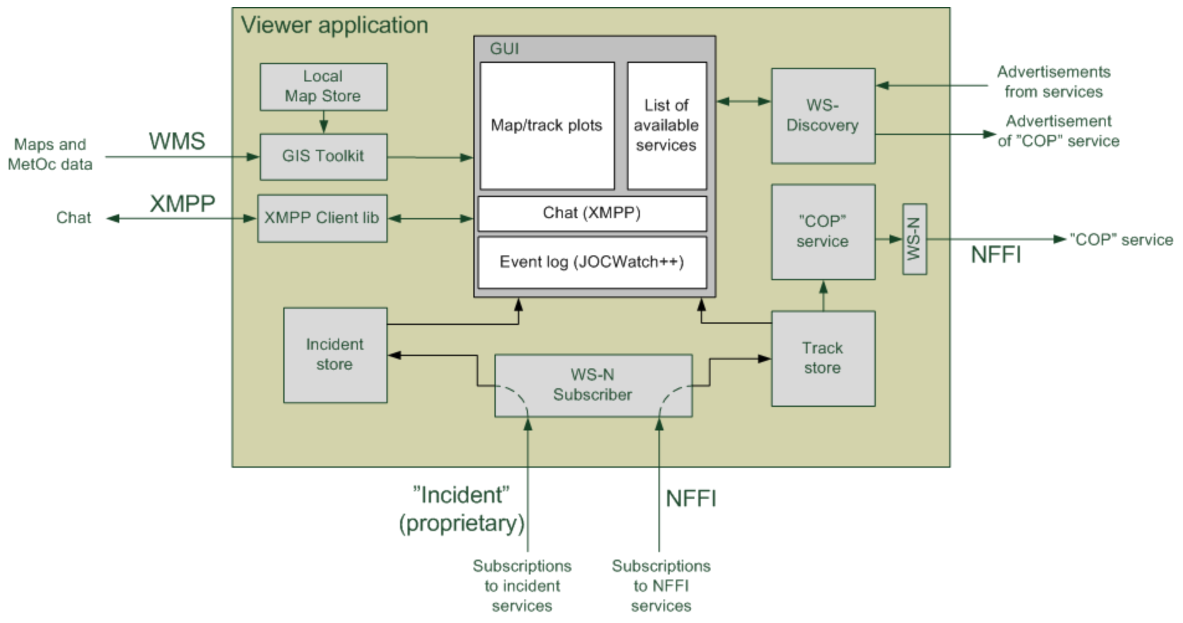


Figure 2.4 SOA viewer information flow

Upon receiving incident reports, the SOA viewer presents the entries in a list, and when clicked on, a popup dialog displays detailed information about the event, together with an image of the incident if provided.

In addition to consuming services, the SOA viewer provides functionality for aggregating information and re-publishing it as its own services. In the experiment, the SOA viewer was configured to aggregate all geographic information based entities (Tracks), and re-publish them as a COP available as a separate publish/subscribe service. The SOA viewer would publish the COP whenever the contents of its internal track store changed, allowing other systems to subscribe to the COP topic and to receive notifications.

2.4 Bespoke solutions

2.4.1 DSProxy

In a previous survey of techniques for adapting Web services to disadvantaged grids, we have identified proxy servers [17] as one of the most important components. Through the use of proxies one can utilize unmodified Web services at the client and server machines, and only the intermediate nodes in the network need use the proxy software, thus enabling increased deployment agility. This reduces complexity in the development of applications and servers, and thereby also costs. We have addressed many of the issues presented in [17] in our in-house developed Delay and disruption tolerant SOAP Proxy (DSProxy).

The proxy aims to tackle the challenges associated with utilizing Web services in tactical environments and disadvantaged grids. In practice, the DSProxy is a middleware component enabling delay and disruption tolerant Web services for heterogeneous networks. It is designed to work with existing COTS Web services clients and services. All DSProxy components share the same core functionality, but can be configured with different sets of plug-ins. For further details, please see [18].

2.4.2 JBridge

The JBridge is a helper application that was developed in-house. We used it for publish/subscribe-enabling request/response Web services. A JBridge instance polls a request/response Web service at pre-defined intervals, and each time compares the response with the previous one (see Figure 2.5). If the response has changed, it is sent to the WSMG broker as a new notification (it is also possible to configure the JBridge to send a notification every time it polls, regardless of whether the response has changed or not).

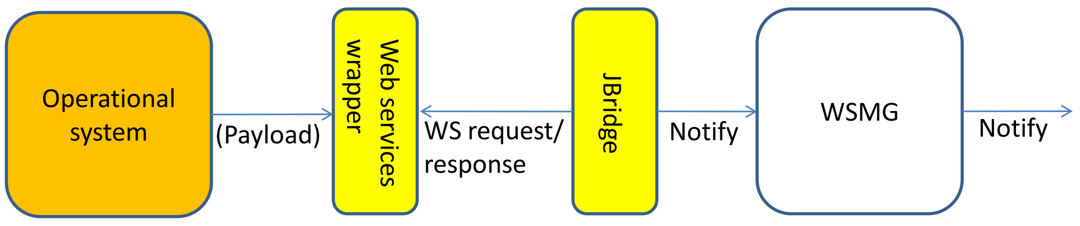


Figure 2.5 Using JBridge to Publish/subscribe-enable request/response Web services

2.5 Communications equipment

2.5.1 WM600

WM600 provides seamless extension of the wired LAN as to provide connectivity between C4ISR applications even when on the move. The WM600 operates in the 225 to 400 MHz UHF band to provide a good balance between range in non-line of sight terrain, data rate and RF power output. The radio can provide different network capacities based on selected RF bandwidth and coding. The radio operates in a pure data radio mode or in a specific combined voice and data mode. Data rates up to 2.5 Mbps are provided [26]. In this experiment we used the WM600 as a pure data radio.

2.5.2 MRR

The multi-role radio (MRR) is a VHF radio which supports both data and voice communication and is designed to work with a variety of applications. It has high ECCM capability, with both direct sequence spread spectrum and frequency hopping. MRR has a very long range with multi-path handling. It offers up to 64 kbps (w/ forward error correction), synchronous and asynchronous communication [27].

3. Experiment

3.1 Experimental systems involved

3.1.1 Wireless sensor networks

Wireless Sensor Networks are expected to provide greatly enhanced situational awareness for war fighters in the battlefield. Sensors distributed throughout the battlefield are, however, of limited value unless the sensors are reliable during the entire operation and the information produced is made available in a timely manner. We have focused on these issues by enabling wireless sensor networks as a capability using Web services. The sensors are capable of reporting data (e.g., motion, temperature, etc.) via a Web service interface, and can also be configured via such a service. For further details, see [1].

In the experiment, we used a subset of the functionality described above. There, our setup was only triggered by motion, and we had no subscriptions for the temperature or camera capabilities. Since the experiment was built around two information formats – incidents and blue force tracks – the sensor network output was presented in the infrastructure as an “FFI incident”, i.e., coded according to an incident schema we had agreed upon for this experiment (see Appendix A in [20]).

3.1.2 NORMANS

FFI develops concept, requirements and technology supporting the future network enabled soldier. An overview of the Norwegian Modular Network Soldier (NORMANS) is given in [7], and the C2I system is presented in [8].

Previously, we have Web services-enabled NORMANS at CWID 2007. Since then the NORMANS protocol has been revised. We created a new wrapper for NORMANS in the SOA viewer in a similar manner as that used at CWID 2007 (see [9] for details of the experiment).

3.2 Operational systems involved

3.2.1 Criads

Criads is a Norwegian operational system for coastal radar tracks. It can export several formats, among them OTH gold [21]. There is, however, currently no support for exporting NATO Friendly Force Information (NFFI) [14] tracks. In our experiment we chose NFFI as the common format for our infrastructure, and needed to provide all output coded in this format. OTH gold is a text format for, e.g., tracking. Messages can be issued whenever there is new information available, and each track has a unique identifier. The current message can reference previous identifiers (i.e., update or delete them) so a track store is necessary to keep an overview of the current tracks. To convert OTH gold to NFFI we first created a simplified OTH gold track store, with support for those parts of the specification used by Criads. This track store connected to a TCP socket in Criads providing an OTH gold feed. The track store used a mediation service to translate OTH gold to NFFI, and exposed the NFFI track as an NFFI SIP3 request/response service. This service was then publish/subscribe-enabled by a connection from the JBridge. The JBridge would invoke the NFFI track service at regular intervals, and publish the response using WS-Notification via WSMG. This approach is best used when you want to disseminate tracking information at regular intervals, and already have an existing request/response NFFI service. The approach is illustrated in Figure 3.1 below.

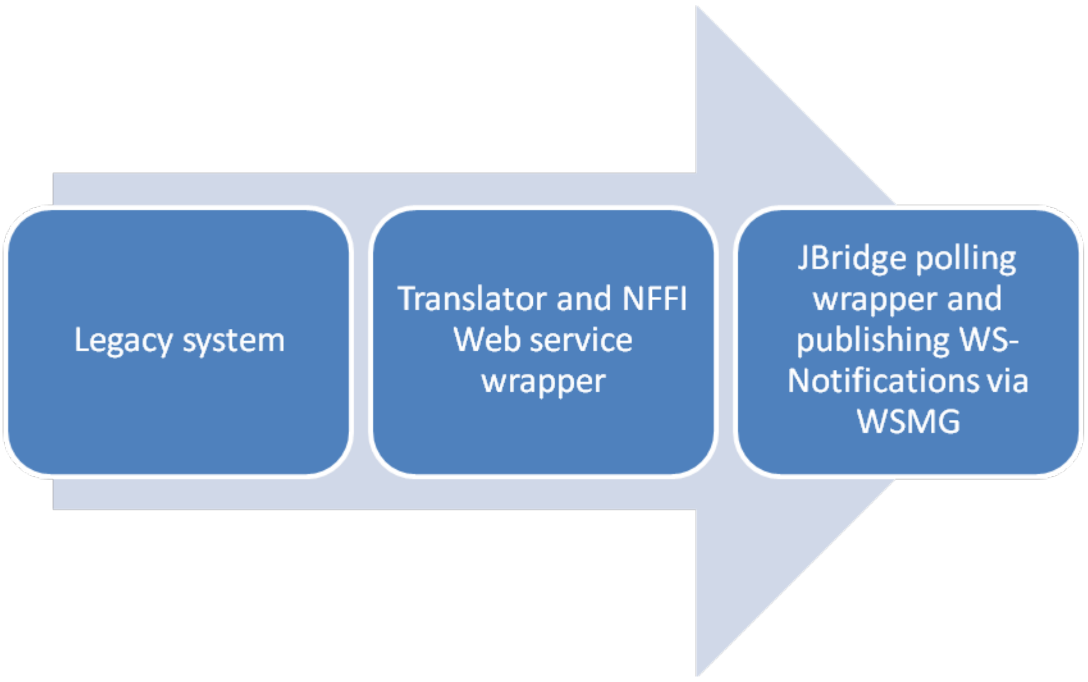

Legacy system
Translator and NFFI Web service wrapper
JBridge polling wrapper and publishing WS-Notifications via WSMG

Figure 3.1 Web service-enabling a legacy system (e.g., Criads) by means of a Web services wrapper and WS-Notification framework. (Information flow from left to right, consumers not shown)

3.2.2 NORCCIS II

NORCCIS II is a Norwegian C2 system which can export its tracks using several formats, NFFI included. However, the NFFI support in NORCCIS II follows an older specification using NFFI over TCP, and not the Web service interface from the current NFFI SIP3. We solved this by creating an NFFI TCP client that we connected to the configured NORCCIS II export socket. The tracks pushed over this socket were NFFI 1.3 formatted, i.e., using the same schema as SIP3 does. Thus, creating a Web service wrapper was a straightforward task. This means that even if both NORCCIS II and Criads are fundamentally different systems, the Web services offered from the wrapper used the exact same interface. Thus, the JBridge was used in this case as well, in the same manner as described above for Criads. Being able to reuse functionality in this manner, due to the use of common interfaces, is one of the main strengths of the SOA approach. This, in turn, leads to increased operational agility.

3.2.3 NIRIS

NIRIS is a middleware integration core intended to provide a recognized air picture. It can export its track store in different formats. The version we got in our lab was unable to export NFFI, but it could export NVG. NVG is a flexible XML-based format which can describe incidents, positioning information, etc. When used for tracking, NVG contains a lot of information, which cannot be expressed by NFFI. In our NVG to NFFI conversion module we extracted the data that could be mapped to the mandatory parts of NFFI. We used the same NFFI Web service as for Criads and NORCCIS II, but in this case populating it with data converted from the NVG interface in NIRIS.

In our experiment, we used NIRIS to provide data from the Maritime Command and Control Information System (MCCIS) and the Tactical Data Link (TDL). MCCIS has been developed to contribute a high quality recognized maritime picture. It can provide a near-real-time Common Operational Picture (COP). The TDL adheres to the Link 16 message standard, and was developed to meet the information exchange requirements of all tactical units, supporting the exchange of surveillance data, electronic warfare data, mission tasking, weapons assignments and control data. In the execution of our experiment, these systems were represented by simulated input to NIRIS.

3.2.4 CSD

We had an installation of the coalition shared database (CSD). The CSD originated in the multinational project Multi-sensor Aerospace-Ground Joint Intelligence Surveillance and Reconnaissance Interoperability Coalition (MAJIIC). The CSD's purpose is to collect various types of intelligence information in a searchable database. In our experiment it acted as a source for surveillance images. The CSD was wrapped in a Web service that could provide incidents with images. The images needed to play the scenario were loaded into the CSD database by an operator at the appropriate time in the timeline, and thus made available for publishing over WS-Notification via a Web services wrapper.

3.3 Experiment information infrastructure

The scenario for the experiment was built around an expeditionary operation to a conflict with an international coalition involved to protect civilians and initiate a peace process. In this paper we focus on the technology aspects of the experiment; employing Web services as the information infrastructure. For further details on the scenario and experiment execution, please see [15].

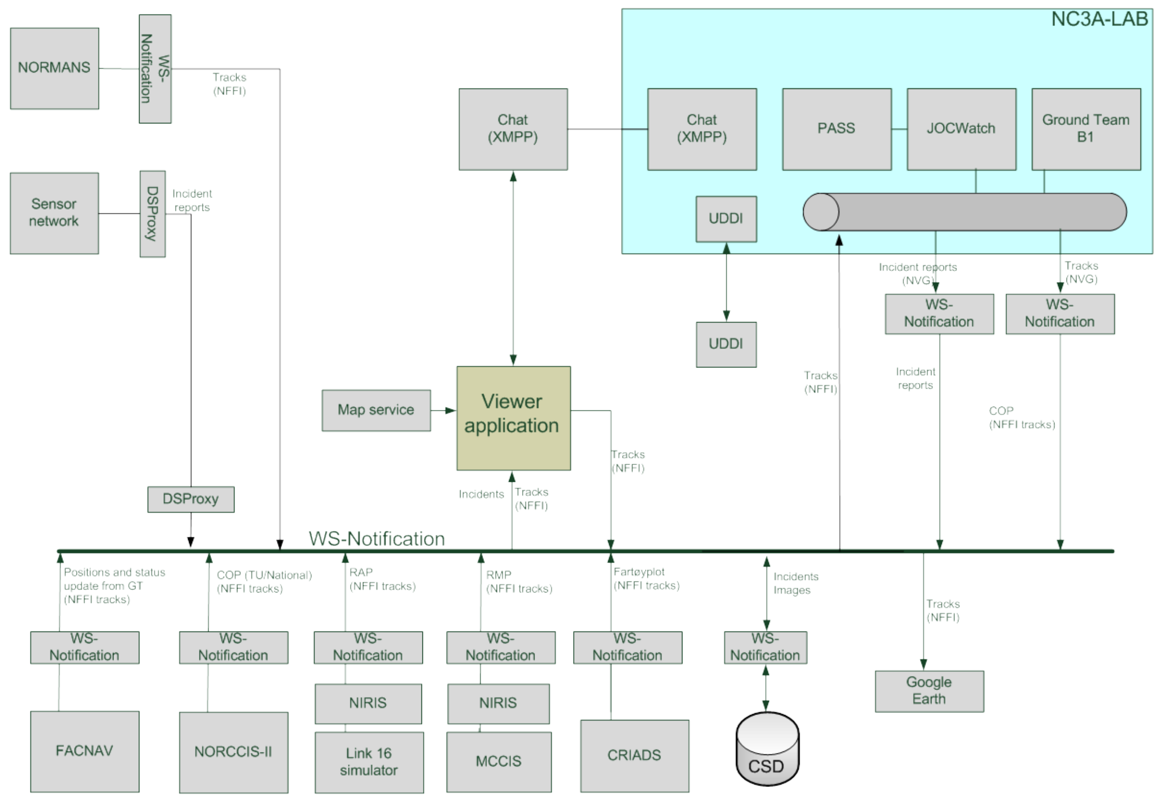


Figure 3.2 Information flow between systems

Figure 3.2 shows the information flow between the different systems. As described earlier, each operational system was wrapped with a Web service, delivering notifications to a WS-Notification service. In addition, we had XMPP-based chat between all units.

We also used Google Earth as an alternative viewer, in order to demonstrate the capabilities of COTS software. Having a second viewer allowed us to show how easy it can be to replace a component in a SOA infrastructure, as long as the components implement the same interfaces.

3.4 Lessons learned

The complete setup was a complex and heterogeneous system, with a large number of interconnected autonomous parts.. Thus, it was a challenging task getting everything up and running. The main cause of this, however, was the choices we made with respect to infrastructure software, as discussed below.

3.4.1 WS-Notification infrastructure (WS-Messenger)

We chose to use a freely available implementation called WS-Messenger (WSMG), which is provided as open source by the University of Indiana. When used in the experiment, however, WSMG displayed a number of shortcomings:

- It added proprietary tags in the notification messages, meaning that the notification receiver had to be tailored for use with WSMG.
- No support for XML in the payload, which meant that the payload had to be Base64 encoded.
- Finally, if WSMG was unable to deliver a notification to a subscriber (for instance because the subscriber had crashed), it would cease submitting notifications to all subscribers of that topic. Instead, it would start buffering the notification, pending the return of the missing subscriber. This behavior was a major cause of delays in the experiments, since one subscriber crashing or losing connection would mean that everything had to be set up again from the start.

Consequently, we will advise against using WSMG for WS-Notification, even in a test environment if you need to disseminate XML payloads. It should also be noted that WSMG is not a commercial product, but rather an academic project. This means that support and further development of the software is uncertain.

3.4.2 Wrapper software

The principle of wrapping existing software with a Web service front-end works very well. The complexity of the wrapper is dependent on the type of connection it has to the actual operational system. Several of the systems offered an outgoing TCP connection that the wrapper was listening to. The messages received on the TCP socket were then transformed to a self contained XML document, and made available on the request/response interface.

Finally, in the process of transforming the NII into a SOA, it is important to think beyond just Web services enabling existing systems. By turning systems into services, they become available to a much broader community than the existing, often stove-piped, systems are. This fact should be taken advantage of by analyzing business processes and required data flows in order to establish which interfaces are required and where. For instance, in some cases it may be advantageous to break existing monolithic systems into smaller, individual modules.

3.4.3 JBridge

The JBridge was only intended as a helper component for use in the experiment. It was introduced because we wanted to avoid having to use WSMG libraries in the wrapper for enabling publisher functionality. Instead of building the publisher functionality into each wrapper individually, we chose to make one generic component that could be used in front of all Web services enabled operational systems. However, the JBridge brought increased complexity into the experiment, and it did represent an additional source of possible errors.

When Web services-enabling operational systems in the future, we therefore recommend that the publisher functionality is integrated into the Web services wrapper. Furthermore, when building new systems that support Web services, it is also important that the publisher functionality is included. In cases where integration of publisher functionality is not possible, the poller function (i.e., the JBridge equivalent) should be located on the same machine as the actual Web service, in order to avoid network traffic.

3.4.4 Service discovery

Our chosen mechanism, WS-Discovery, is a decentralized service discovery mechanism intended for use in local area networks, relying on UDP multicast for disseminating information. Consequently, WS-Discovery can only be used in networks that support multicast, which is not necessarily the case for all types of military networks. It is also a relatively “chatty” mechanism, which means that it should not be used on networks with very limited bandwidth. As described in Section 2.2.2, some measures can be introduced (e.g., compression) in order to reduce the network traffic.

The experiment shows that WS-Discovery works fine over medium bandwidth radios like the Kongsberg WM600, while it is not advisable, even with compression, on radios with very low bandwidth, such as the MRR. WS-Discovery works very well when it comes to liveness, i.e., the announced services reflect quite well which services are actually available.

It should also be noted that WS-Discovery is only intended for run-time use. This means that a query for a service will only return the endpoint address of that service, not the WSDL. Consequently, it is only possible to invoke services that you already have a client for, i.e., that you have previously downloaded a WSDL for the service, generated the client stub and integrated this with the client software. This is not a problem, however, since the required WSDLs are normally obtained during design time, either by using a registry and repository, or by downloading them directly from the service provider.

A more general problem concerning publish/subscribe and service discovery, which also affects WS-Discovery, is that all service discovery mechanisms (as the name implies) focus on discovery of services. The problem is that seen from a WS-Discovery point of view, all publish/subscribe (WS-Notification) services are equal. They all provide the same service (which is to deliver notifications), but they can provide notifications on different topics. Consequently, the information about which topics each service published on had to be distributed beforehand.

It is clear that when using publish/subscribe for information dissemination, it is necessary to focus both on how to search for and discover available topics, and also the semantics of topics, i.e., how to structure the description of the information in the notifications into topics and subtopics.

3.4.5 DSProxy

In the experiment, we used DSProxy on the MRR connection between the sensor network and the main network. The MRR represents a considerable challenge when using Web services, since the available bandwidth is very low and communication delays are high.

The DSProxy, being intended for low bandwidth networks, worked very well, and we successfully disseminated information from the sensor network across the MRR link.

3.4.6 Format translation

In the experiment we had to handle a number of different formats coming from the different operational systems. For tracks, these included OTH gold, NVG and proprietary formats, and all were translated into NFFI, which we used internally. The translation code was placed within the wrapper where needed.

A challenge when integrating information from different formats is loss of information. For instance, when translating from NVG to NFFI, relatively much information is lost. If, at a later stage, it is necessary to translate back from NFFI to NVG, a lot of the original information has been lost. There are also examples of the opposite problem; when translating from OTH gold to NFFI, it is necessary to add information, since OTH gold is a much simpler format.

In general, when introducing SOA, which data formats to use and how to handle them is a matter that needs careful consideration. Closely related to this, it is very important to be aware of how data flows between systems, in order to avoid circular data flows. That is, data that originates in one system flows to another, and then eventually (possibly via further systems) flow back into the originating system. This may lead to false confirmation of data, i.e., where users of system A believes data has been confirmed by system B, while system B have actually just returned data it has received from system A. Thus, managing data origin and avoiding information loops is very important.

4. Conclusion

The experiment was executed at FFI in June 2011 in cooperation with the NC3A. Our goal was to give a practical demonstration of some of the benefits of using a SOA. We started with a number of existing military operational systems and service-enabled these by using so-called wrappers. All these systems were then connected in a common information infrastructure, offering their information through services.

From a practical point of view, we had two main challenges during the experiment execution:

- Setting up and using several large and complex operational systems was a non-trivial task
- Our choice of WS-Notification implementation, WSMG, caused a lot of problems.

As a demonstration of the benefits of SOA, however, we consider the effort a success. We proved that service-enabling of existing National operational systems is a manageable task. However, in the spirit of NNEC, new systems should preferably be designed as service oriented from the start.

We also showed that, once the systems are service enabled and deployed, a whole new flexibility is gained with respect to information dissemination and being able to dynamically configure and reconfigure an information infrastructure. Service integration is truly a step in the right direction towards increased operational agility for C2 systems.

5. References

[1] "Integrating Wireless Sensor Networks in the NATO Network Enabled Capability using Web Services", J. Flathagen and F.T. Johnsen, IEEE MILCOM 2011

[2] "Information exchange in heterogeneous military networks", K. Lund, T. Hafsøe, F.T. Johnsen, E. Skjervold, A. Eggen, L. Schenkels, J. Busch, FFI-report 2009/02289

[3] OASIS WS-Notification (2006) TC http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn

[4] WS-BaseNotification 1.3 OASIS Standard, approved October 1st 2006 http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf

[5] WS-BrokeredNotification 1.3 OASIS Standard, approved October 1st 2006 http://docs.oasis-open.org/wsn/wsn-ws_brokered_notification-1.3-spec-os.pdf

[6] WS-Topics 1.3 OASIS Standard, approved October 1st 2006 http://docs.oasis-open.org/wsn/wsn-ws_topics-1.3-spec-os.pdf

[7] "Norwegian Modular Network Soldier (NORMANS)", R. Lausund, S. Martini", FFI Facts, November 2006

[8] "NORMANS KKI" (in Norwegian), J. Flathagen and L. Olsen, FFI Facts, November 2006

[9] "Experiment report: 'SOA – Cross Domain and Disadvantaged Grids' – NATO CWID 2007", R. Haakseth et al., FFI-report 2007/02301

[10] WS-Messenger (WSMG) home page http://www.extreme.indiana.edu/xgws/messenger/

[11] "SOAP-over-UDP version 1.1 OASIS standard 1 July 2009" R. Jeyaraman (ed.) http://docs.oasis-open.org/ws-dd/soapoverudp/1.1/os/wsdd-soapoverudp-1.1-spec-os.pdf

[12] "Using Web Services and XML Security to Increase Agility in an Operational Experiment featuring Cooperative ESM Operations", Trude Hafsøe, Frank T. Johnsen, Nils A. Nordbotten, Espen Skjervold, 14th International Command and Control Research and Technology Symposium (ICCRTS), Washington DC, USA, June 2009.

[13] "Adapting WS-Discovery for use in tactical networks", Frank T. Johnsen and Trude Hafsøe, 16th International Command and Control Research and Technology Symposium (ICCRTS), Quebec City, Canada, June 2011.

[14] "NFFI Service Interoperability Profile 3 (SIP3) Technical Specifications (Version 1.1.5)", Vincenzo de Sortis, Working Paper EPW002625-WP05, NC3A, The Hague, March 2009

[15] "Experiment Report: SOA-pilot ", R. Rasmussen and B.J. Hansen, FFI Report 2011/02407

[16] "Agility, focus, and convergence: The future of command and control", David S. Alberts, International C2 Journal, Vol 1, No 1, 2007

[17] "Adapting Web Services for Limited Bandwidth Tactical Networks", T. Hafsøe, F. T. Johnsen, K. Lund, and A. Eggen, 12th International Command and Control Research and Technology Symposium (ICCRTS), Newport, RI, USA, June 2007.

[18] "Robust Web services in heterogeneous military networks", Ketil Lund, Espen Skjervold, Frank T. Johnsen, Trude Hafsøe, and Anders Eggen, IEEE Communications Magazine, October 2010.

[19] "Experiments with Web services at Combined Endeavor", Frank T. Johnsen and Trude Hafsøe, 15th International Command and Control Research and Technology Symposium (ICCRTS), Los Angeles, CA, USA, June 2010.

[20] “SOA-pilot 2011: Information infrastructure”, Ketil Lund, Frank T. Johnsen, Trude H. Bloebaum, Espen Skjervold, FFI report 2011/02235.

[21] “Operational specification for over-the-horizon targeting gold”, Revision D, OS-OTG (Rev D), Published under the direction of CNO (N62) by Navy center for tactical systems interoperability, 1 September 2000

[22] “Web services dynamic discovery (ws-discovery) version 1.1 OASIS standard 1 July 2009”, V. Modi, and D. Kemp (eds.), http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.pdf

[23] “NATO network enabled capability feasibility study.”, P. Bartolomasi, T. Buckman, A. Campbell, J. Grainger, J. Mahaffey, R. Marchand, O. Kruidhof, C. Shawcross, and K. Veum, Version 2.0, October 2005.

[24] “Core Enterprise Services Standards Recommendations: The Service Oriented Architecture (SOA) Baseline Profile Version 1.7”, NATO Consultation, Command and Control Board (C3B), 11 November 2011

[25] “Web services glossary W3C working group note 11 February 2004”, Hugo Haas and Allen Brown (eds.), http://www.w3.org/TR/

[26] ”WM600 datasheet”, Kongsberg, http://www.kongsberg.com/en/kds/products/defencecommunications/~/media/KDS/Files/Products/Defence%20Communication/wm600_datasheet_rev_rc_small.ashx

[27] ”MRR datasheet”, Kongsberg, http://www.kongsberg.com/en/KDS/Products/~/media/KDS/Files/Products/Defence%20Communication/MRR%20500%20brosjyre%20RD.ashx