THE INTERNATIONAL

C2 JOURNAL

VOLUME 4, NUMBER 1, 2010

SPECIAL ISSUE

Agility and Interoperability for 21st Century Command and Control

GUEST EDITOR

Sandeep Mulgund
MITRE

Semantic Service Discovery for Interoperability in Tactical Military Networks

Frank T. Johnsen
Marianne Rustad
Trude Hafsoe
Anders Eggen
Tommy Gagnes

CCRP
Focus
& Convergence
for Complex Endeavors

THE INTERNATIONAL C2 JOURNAL

About the Journal

The International C2 Journal was created in 2006 at the urging of an international group of command and control professionals including individuals from academia, industry, government, and the military. The Command and Control Research Program (CCRP, of the U.S. Office of the Assistant Secretary of Defense for Networks and Information Integration, or OASD-NII) responded to this need by bringing together interested professionals to shape the purpose and guide the execution of such a journal. Today, the Journal is overseen by an Editorial Board comprising representatives from many nations.

Opinions, conclusions, and recommendations expressed or implied within are solely those of the authors. They do not necessarily represent the views of the Department of Defense, or any other U.S. Government agency.

For more information

Visit us online at: www.dodccrp.org
Contact our staff at: publications@dodccrp.org

Semantic Service Discovery for Interoperability in Tactical Military Networks

Frank T. Johnsen, Marianne Rustad, Trude Hafsoe, and Anders Eggen (Norwegian Defence Research Establishment, NO)

Tommy Gagnes (Telenor ASA, NO)

Abstract

Interoperability, both inter- and intra-nation, is a main concern when attempting to fully realize NATO Network Enabled Capabilities (NNEC). The NNEC vision implies an information infrastructure that supports prioritized access to information, services, and resources from the strategic level, down to the tactical level where communication resources usually are scarce. Web services technology has been identified as a key enabling technology in the NNEC feasibility study. Using this technology all the capabilities in a network can be exposed as services that can be discovered and used across heterogeneous networks. In this paper we discuss service discovery solutions for Web services and their limitations when considering their use in the network-centric battlefield. We focus on both the technological aspects that are needed to accommodate the disruptive nature of military tactical networks, and in particular the semantic aspects that need to be utilized in order to achieve system interoperability. We suggest the use of semantic technology to address the interoperability challenges related to service description and selection. However, the existing distribution mechanisms lack some key features that make them subject to research. A standardization process involving this technology is needed within NATO.

Introduction

The NATO organization is focusing on NATO Network Enabled Capabilities (NNEC) as a means to more efficiently utilizing available resources both within and across system- and national borders. The aim of NNEC is to increase mission effectiveness by networking military entities. Mission effectiveness does however greatly depend on the participants' ability to interact with each other in an efficient manner.

Having a common overview of a situation, and a common understanding of the task at hand, are key factors when it comes to efficient communication between people. There are of course many other factors that affect the interaction between people, but having the ability to share information in an efficient manner allows for creating a shared situational awareness that can reduce some of the challenges involved. Achieving technical interoperability, which is the interoperability between communication systems and application such as C2 systems, will require a common information infrastructure for NNEC.

Today, this interoperability between national systems usually takes place at the brigade level or higher. An information infrastructure designed to support NNEC operations must however fulfill all the communication requirements of the member nations' forces, including horizontal information exchange between systems at lower levels.

A common information infrastructure for NNEC will most likely consist of a federation of networks at different operational levels that will be required to work together. Not only will these systems have different network characteristics, such as data-rates, mobility, broadcast and error probability, but they will also be owned and managed by the individual coalition partners. The systems used within the different nations might not be directly compatible with each other, so achieving interoperability between the systems will require the use of interface points between them.

In order to achieve this cross-system interoperability, one needs to abstract away from the implementation details of the individual systems, and provide a loosely coupled framework that allows for communication through clearly defined interfaces. This allows the individual nations to choose and manage their own systems independently, while still allowing for an easy integration of systems.

Service Oriented Architecture (SOA) is an architectural paradigm that is based on this principle of loosely connected systems, which are integrated using standardized protocols through predefined interfaces. Web services, the most commonly used technology for implementation of the SOA principle, have been identified as a key enabling technology for the NNEC information infrastructure. Web services are standard based, but not all aspects of Web service implementation are covered by these standards, and the existing standards often exist in several versions, have optional elements or are lacking in detail. In order to ensure compatibility between the individual national systems, it is vital that all coalition partners utilize the standards in the same manner. This requires the development of interoperability profiles, which must be agreed upon through for instance NATO standardization.

One example of such work is an ongoing NATO work group that defines a set of *Core Enterprise Services*, which may be used alone or as building blocks for more advanced services. One such Core Enterprise Service is *service discovery*, which is the process of identifying a service. This is an important part of any SOA, but it is particularly challenging in dynamic environments such as military tactical systems. Our previous research has shown that it is feasible to utilize Web services in military networks (Lund et al. 2007, 47-53), provided you know their location. However, when the location is not known in advance, it is important to have service discovery solutions that are interoperable in order to be able to find and utilize services at different operational levels across heterogeneous networks. In this paper we provide a survey of the means to discover services. The differences between the systems used by coalition partners imply that

it will not be possible to use a *one size fits all* solution for service discovery. Different solutions will have to be used for different types of networks in order to best utilize the network characteristics (Johnsen and Hafsøe 2009).

As mentioned above, discovering available services is particularly challenging in networks that are not only dynamic, but that consist of a collection of systems maintained by different entities. This makes governance issues such as service management an important factor. In our solution, we have chosen to focus on the technical aspects of achieving interoperable service discovery. Adding management functionality to our prototype is left for future work.

In order for a system to use resources available in other systems, finding services and determining how to use them must be done in a manner that requires as little manual configuration as possible. One issue with Web services standards is the inability to describe what the service is *capable* of doing. Semantic Web Services (SWS) (McIlraith et al. 2001, 46-53) extend the service descriptions of current Web services technology with rich, explicit semantics to improve service discovery, selection and invocation significantly. SWS are believed to facilitate run-time, capability-based discovery of services. This is essential for solving the problem of run-time service discovery in tactical networks while improving interoperability.

Interoperability Challenges

There are a number of interoperability issues that must be addressed in order to achieve service discovery across systems, which in turn allows for the implementation of cross domain Web services:

- Services must be described in a standardized manner, using a description format that is rich enough to ensure that end systems know enough about the service to perform automated interaction with services belonging to other systems. The current

Web service standard for service descriptions, the Web Services Definition Language (WSDL) (Curbera et al. 2002, 86-93), offers only syntactical descriptions.

- A method for distribution of the service descriptions is needed. It must be possible to distribute service descriptions across networks at different operational levels, and across system- and national boundaries. Current Web service discovery solutions, which are designed for Internet usage, are not sufficient for the network-centric battlefield.

- Provided we have rich service descriptions that we are able to distribute in an efficient manner, there is still a major challenge in ensuring that the understanding of these descriptions is common among nations. There is a need for both human and machine readable semantics to ensure that this is achievable.

- Due to the complexity of a coalition network, it is highly unlikely that one will be able to find one mechanism for distribution of service descriptions that will work in all systems. This means that there will be a number of different service description distribution mechanisms that need to co-exist. These mechanisms must be made interoperable through the use of interoperability points, most likely in the form of gateways that are able to translate between the mechanisms.

Service Descriptions

A recurring issue in service-oriented systems has been that service descriptions are mostly based on syntactical descriptions rather than semantic meaning. This means that it is impossible to create a fully dynamic SOA, because one has no means of understanding the semantics, that is, what to expect from a service discovery and invo-

cation at run-time. SWS add an additional layer of explicitly defined semantics to the service description, which adds expressiveness and possible semantic interoperability in discovery.

Consider, for instance, a price-comparing agent querying different providers for the price of an item. Given that the providers expose methods like *getPrice*, or *getOffer*, the agent has no way of knowing e.g. the currency, whether the prices include sales tax or not. This is called the semantics of the invocation. Naturally, this price example is fairly trivial, but it does illustrate some of the benefits of semantic technology, which can be used as a means of ensuring a conceptual joint understanding which supports interoperability. Other aspects of the services also need to be described, for instance, are operations idempotent? Do service operations need to be invoked in a certain sequence? Are there any other operation pre-conditions?

Service Description Expressiveness

In Figure 1 the expressiveness of the standard Web services description is shown in relation to the expressiveness added by the use of ontologies in SWS. Expressiveness is divided in *standard Web services syntax (WS-syntax)* and *Ontology*. The WS-syntax enables syntactical interoperability, but relies on human understanding of the semantic elements. In the knowledge representation world, an *ontology* (Gruber 1993, 199-220) is a formal, explicitly described vocabulary of the entities in a given domain, including their properties, relations, and possibly rules that further model the domain. An ontological description is more expressive as it includes both the interoperability aspect of the WS-syntax but also enables machine processable semantics. In general you see that accumulating service description elements increases the expressiveness of the description.

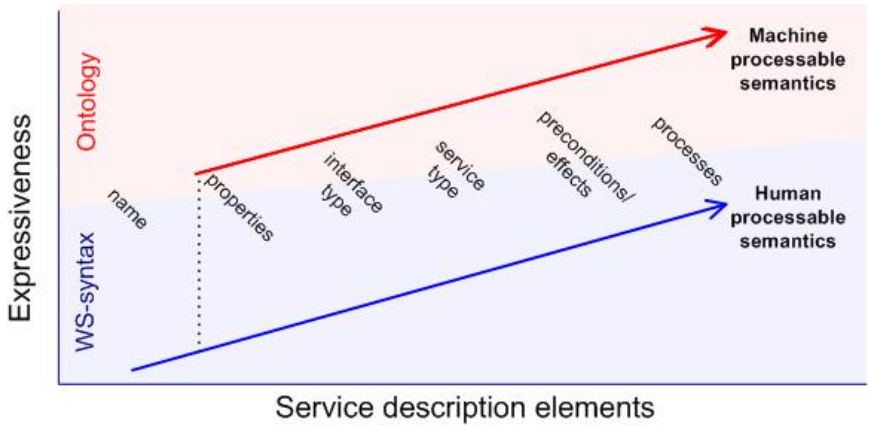

Figure 1. Service description expressivity

An important thing to note is that making service selection based on name, type and attributes without explicit semantic description is dependent on *a priori* knowledge of the invocation semantics, or protocol, of the service. A service name does not tell the computer anything about the capabilities of a service, which is important to know in the cases where we want to select and invoke services of an unknown type in run-time. Ideally, a client would just need to have an idea of the goal or task to accomplish, and a proper service would be found.

Service Descriptions for Web Services

There are several ways in which a service can be described. The simplest way to discover a service is probably to use a *service name*, but this could be ambiguous because the client cannot be sure that it can communicate with all services with a given name. The use of a service name also leads to tighter coupling between service and client, since a client may ignore available services it could have used only because they have the wrong name.

Therefore, a more interesting property to search for is the *interface type*, or signatures, of the service. This basically describes the messages that a service can receive and return. In some cases, interface types can be grouped into a *service type*, which is a collection of interface types. In many cases, service clients are designed with certain interface types in mind. As such, it is feasible to limit search results to services of this interface type. Still, both service and interface type could be ambiguous properties for selecting services. Using a namespace, for instance, could help reduce this ambiguity.

An important aspect of the service-oriented computing idea is the reuse of services as building blocks in new systems. By composing services, one can create new functionality from combining existing services. This is often called service composition or orchestration, and can be done by using a workflow execution engine.

For orchestration of services, that is, managing the execution order of and information flow to and from a set of services, Business Process Execution Language for Web services (BPEL4WS, or WS-BPEL) (OASIS 2007) has been specified. BPEL4WS aims to make it possible to model workflow and business processes, and execute the model as a series of coordinated Web service invocations. BPEL4WS is based on design-time pre-composition, whereas SWS facilitate this in a *dynamic* (e.g., at run-time) fashion.

Service Descriptions for Semantic Web Services

Services described ontologically can enable additional interoperability on top of syntactical interoperability achieved by Web services standards through ontologies as well as more precise and automated discovery, selection and invocation of services. One important aspect of using ontologies is that services can be *polymorphic*, enabling service selection according to a class hierarchy. Also, to reduce ambiguity, and to allow for automated interoperability between service and consumer, semantic service descriptions are needed.

The model for semantic Web service discovery initially is that users have goals or needs that must be resolved. A formalized goal description is used in the service selection part, and this selection is often called matchmaking. The service descriptions serve as the basis for service selection, which is either done at the client, or by some middle agent, e.g. a *broker*. When a service has been selected, it can be invoked. This is usually done by means of Web services standards like WSDL and SOAP (Curbera et al. 2002, 86-93).

SWS are an important part of the Semantic Web vision (Berners-Lee et al. 2001), where agents are expected to take on tasks on behalf of their users. To help realize the Semantic Web, the W3C has recommended the Web Ontology Language (OWL) as an ontology representation language for the World Wide Web. An interesting property of OWL is that it is possible to import ontologies by referencing other ontologies on the Web and aligning concepts in the different ontologies. For example, the concept *position* equals *location* and allows a client to search for position and get a location in return. Thus, a set of distributed ontologies can serve as building blocks for new ontologies, allowing for independence, scalability and interoperability. Ontologies have a strong formal foundation in logics, and as such, it is possible to perform automated reasoning about them and to infer new information.

There are several efforts in the area of SWS research. In this paper we focus only on the OWL-S (see McIlraith and Martin 2003, 90-93) and (Martin et al. 2004, 26-42) service ontology for SWS, which is one of the most prominent solutions. For an overview of existing SWS research see (Hansen et al. 2007, 1-56). OWL-S is modeled in OWL, and can be very useful for reasoning about services. Reasoning can be done on the type of service available as well as its operational signature, preconditions, effect and service processes. For instance, if a service covers some geographical area, it is important that the client can use such properties for selecting an appropriate service. As a simple example of semantics, the operation "getTemperature" returns an integer. This does not define the returning integer as

degrees in Fahrenheit or Celsius. In the WSDL the output is defined as an integer, but in OWL-S the parameter would be defined as an output and as the appropriate degree. Implementing Web services without explicit semantics, one could define operations such as "getTempC" and "getTempF" which would return the temperature in Celsius and Fahrenheit respectively. In that case we would be using the name of the function to describe implicit semantics. This is sufficient for human understanding, but it is insufficient for computers. You need to express the semantics in an explicit, standardized way for a computer to be able to perform reasoning.

If a service has several operations, there may be a certain order in the invocations, or some sort of control flow must be followed. This is called the *process* of the service. A process can also include invoking other, external services to accomplish its goals. A description of such a service process may also be an important part of service description. This is especially relevant when composing new services from existing ones. As a part of OWL-S, processes are described for enabling reasoning on the control flow.

In OWL-S it is possible to specify that to be guaranteed a certain *effect*, a set of *preconditions* must be satisfied before invoking the service. This may or may not be specified explicitly in the description of the service.

SWS allow composition of services in run-time as well as to use mediators to fulfill clients' needs. Service composition is done to create new services by combining other services. If there is a mismatch of interface, protocol, data or semantics between the service and what is expected from the client, it could still be possible to invoke a service. This may improve the value for the client significantly, and can also provide aggregated functionality that would otherwise not be found. Such composition could be done either by the client

itself or by a broker. For example, a service provides NFFI[1] track data but your system requires information structured as JC3IEDM.[2] Without service composition, you would be unable to use the NFFI service. However, discovering a service capable of translating NFFI to JC3IEDM will allow you to get the data you require through a composition of the NFFI service and the translation service.

Storing Service Descriptions

The service descriptions need to be stored in a network accessible framework which allows clients to find and download them. Thus, we are in need of a mechanism which can be used to store semantic service descriptions. This is our main goal, and there are many factors at play in the choice given the dynamic battlefield environment. All in all, the solution should:

- Be able to discover services in a LAN and/or WAN,
- Be able to discover registries in a LAN and/or WAN,
- Mirror the network state, i.e. a query should return only active services and/or registries,
- Be robust,
- Support rich, semantic service descriptions; and
- Ideally have no DNS and WWW dependency.

1. NATO Friendly Force Information (NFFI) is a format being used in Afghanistan to track friendly forces. It supports information exchange via several modes, one such mode being XML formatted data suitable for Web services.
2. Joint Consultation, Command, and Control Information Exchange Data Model (J3CIEDM) is a model that enables the interoperability of systems required to share C2 information. JC3IEDM is managed by the Multilateral Interoperability Programme (MIP).

For Web services there are two competing standards for registries, UDDI and ebXML, as well as a standard for a LAN discovery mechanism, WS-Discovery. We will look at each of these solutions in turn, paying special attention to the requirements mentioned above.

UDDI

Universal Description, Discovery and Integration (UDDI) (Curbera et al. 2002, 86-93) is the most frequently used registry for Web services. Basically, UDDI allows service providers to register their services and service consumers to discover these services both at design-time and run-time.

In principle, UDDI is centralized, but mechanisms for *federating* several registries have also been specified in newer versions of the specification. Having multiple registries or letting a registry consist of several nodes that replicate data increases robustness. In UDDI, replication between registry nodes must be configured manually. It is also possible to let several separate UDDI registries exist independently of each other, but information will not be replicated unless a custom scheme is designed. Additionally, a hierarchical model may be used, using a *root* registry and *affiliate* registries. In this case, a root registry must be chosen, and affiliate registries may be defined as child registries of the root registry. This must be done to avoid duplicate identifiers, or keys. A replication scheme for intra-registry replication between nodes is defined, which allows for fault-tolerance. The replication topology must be configured.

UDDI supports rich service descriptions, and one can find services by name, type, binding, and according to a taxonomy.[3] UDDI has third part support for OWL-S based discovery (matchmaking). UDDI provides a flexible model in that specific service types can be registered with the registry and referenced by service instances that implement

3. A taxonomy is a hierarchical classification.

the service type. This is called a technical model, or *tModel*, in the UDDI information model. A tModel can include pointers to further description of a service, such as a WSDL description and bindings. Since UDDI is designed to be general, OASIS describes ways to map WSDL documents and also BPEL4WS abstract processes to the tModel fields of the UDDI registry. Especially the former facilitates a number of interesting queries, where search can be based on WSDL portTypes and bindings, that is, the signature of the service. This is very important for run-time selection of services.

The UDDI registry supports reconfiguration as long as services do not go down unexpectedly. If so, advertisements will be in the registry forever because there is no liveness information in the current versions of UDDI.

UDDI does not include a repository mechanism. It can only hold URIs pointing to content which is stored elsewhere.

ebXML

Another effort in the Web services world is electronic business XML (ebXML) (Patil and Newcomer 2003, 74-82). It is a collection of specifications for conducting business-to-business integration over the Web. It allows registering services in a similar way as UDDI according to its own registry specification.

The ebXML registry also defines inter-registry interaction, or cooperation between registries, a so-called *federation* of registries. Note that an ebXML federation is different from that of a UDDI federation: ebXML supports a non-hierarchical multi-registry topology. Here each registry has the same role, and registries may join or leave a federation at any time. This allows flexible deployment. Federated queries are supported, enabling query forwarding to other registries without the need to replicate data first. It is also possible to cache registry data locally, allowing for a loose form of replication.

One should note that the ebXML registry is meant to support both discovery and business collaboration, as opposed to UDDI, which mainly targets discovery.

The ebXML registry information model is similar to that of UDDI but somewhat more flexible. Business capabilities such as processes and services can be published in the registry. It is possible to use taxonomies to classify the registered items. Support for rich service description in ebXML is currently very similar to that of UDDI. Both support finding services by name, type, binding and according to a taxonomy. However, ebXML supports more advanced queries. Also, ebXML supports OWL-Lite[4] semantics.

Just like UDDI, ebXML has issues with liveness, in that it supports reconfiguration as long as services do not go down unexpectedly. Unlike UDDI, the ebXML registry can store vocabularies like XML Schema or ontologies since it also specifies a repository for such items.

WS-Discovery

WS-Dynamic Discovery, or WS-Discovery for short, is a standard for Web services discovery in ad hoc networks (Modi and Kemp 2009). It is similar to UPnP (Richard 2002, 221-346) in that it is based on local-scoped multicast, but SOAP over UDP is the advertisement transport protocol used. Query messages are called probe messages. Services on the network evaluate probes, and respond if they can match them. To ease the burden on the network, WS-Discovery specifies a discovery proxy that can be used instead of multicast (e.g., if a registry such as UDDI is present, it should be used instead). This means that clients and services can run in two modes, dependent on whether there is a discovery proxy available or not.

4. OWL-Lite is the least expressive variation of OWL.

WS-Discovery is suited for service discovery in a LAN only, since it is based on local scoped multicast. However, if a registry is present, then WS-Discovery enables you to find that registry in your LAN. That registry can in turn allow you to find services in the WAN. WS-Discovery, when operating without a discovery proxy (i.e., registry), is fully decentralized. However, when a discovery proxy is present, it should be used instead, and then the robustness of the registry solution is important. As none of the registry standards take liveness into account, WS-Discovery will only accurately reflect the service network in its decentralized mode.

With WS-Discovery, service matching is based mainly on the WSDL port type supported by the service and administrative scope. The port type is described by a namespace URI, and some scope limitation can be done through a simple filter. WS-Discovery does not support discovery based on semantic descriptions. It has no repository mechanism, and information (i.e., WSDL includes and XML schema includes) referenced in discovered descriptions need to be fetched from somewhere else.

Technology Overview

In summary, we can say that several key properties are missing when deploying today's standards for Web service discovery in dynamic environments. See Table 1 below for a summary.

Table 1. Capabilities of current Web services discovery mechanisms

	Registry Discovery on LAN/WAN	Service Discovery on LAN/WAN	Query response mirrors network state	Resilient towards partial network failure	Descriptions	No DNS and WWW dependency	Semantic Descriptions
WS-Dynamic Discovery	LAN	LAN	Yes, but not on discovery proxy	OK	Namespace ++	NO	NO
UDDI	N/A	WAN	NO	Data replication between registries	UDDI information model	NO	Third party support
ebXML	N/A	WAN	NO	Registry federation	ebXML information model	Repository mechanism	Based on OWL-Lite

No single solution covers the service discovery requirements of the different network types that take part in a complex combined endeavor. Thus, we need to research solutions which can accommodate the network-centric battlefield, while at the same time ensuring interoperability with other systems.

Current prototype

We have implemented an experimental service discovery solution which is tailored for tactical networks (Johnsen 2009, 1-8). Our prototype, shown in Figure 2 below, uses our special purpose MANET service discovery solution. Using this mechanism each node periodically distributes a list of its service to other nodes. These announcements include the geographical position of the node and a list of the services this node offers. Both standard Web service data and semantic data are contained in each announcement. The solution is fully decentralized, and employs hashing and compression techniques to reduce data-rate requirements. Further, it addresses the issue of liveness by leveraging the previously mentioned periodic advertisements and local cache timeouts. Each node maintains a local cache of services for which it has received a service announcement. This list is purged periodically, thus ensuring that stale service announcements are removed.

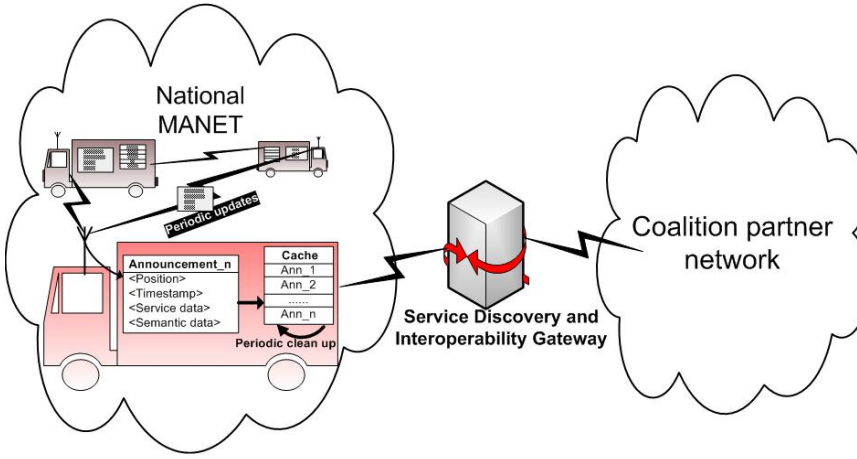

Figure 2. Decentralized discovery prototype

This mechanism is intended for use in MANETs only, and exchange of service announcements between this mechanism and other existing mechanisms, such as WS-Discovery, is handled through the use of service discovery gateways. We have implemented a gateway that provides cross-network distribution of service announcements. The gateway transparently translates between different service discovery protocols. It solves the liveness problem by explicitly de-registering services that disappear from the MANET in the registries it connects to. A service that appears in the MANET will, in a similar fashion, be explicitly published in the registry by our gateway. The gateway can also take services published in a registry and re-publish them in the MANET. In this case, the services published into the MANET will accurately reflect the state of the registry. The gateway can take a subset of the services in a registry for re-publication in the MANET. Which services to expose are a matter of policy, and need to be configured in the gateway. Our experimental discovery solution and interoperability gateway have been successfully demonstrated through experiments at Combined Endeavor 2009 in cooperation with the NC3A (Johnsen et al., 2009). There we used an ebXML registry in the fixed network, and our experimental discovery solution in the MANET, see Figure 3.

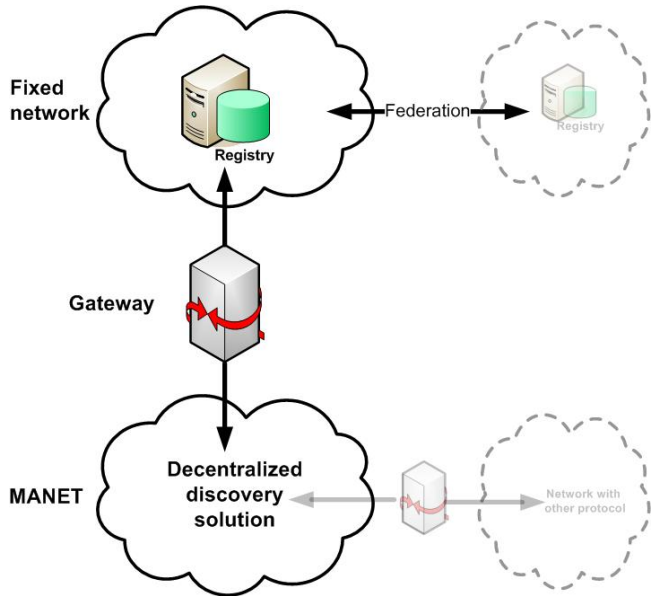

Figure 3. Combined Endeavor experiment setup

Following this exercise, we have extended the prototype with the semantic capabilities presented in this paper. We have tested the prototype in our lab in small scale experiments functioning as a proof-of-concept. In the following section we present an example which illustrates what can be achieved by using the prototype.

Semantic Web services interoperability example

We now introduce an example that serves to illustrate a few of the possibilities of Semantic Web services using our prototype. Consider a coalition force, where many participants from different nations need to cooperate to work as efficiently as possible. Their systems were not initially designed to work together, but a common vocabulary for information exchange has been standardized. This vocabulary, however, has been extended individually by the different nations, so

that specialized vocabularies now exist, but with a common top-level ontology. The nations each have their own networks, but interoperability points exist between them for information exchange.

Ideally, all participants working in the same geographical area should be able to share information with each other. Also, information sharing with people higher up in the hierarchy would be necessary. Figure 4 shows the area of operations (shaded green) and the coverage areas of four sensors. The services exposing sensor information are described semantically, their coverage areas being represented as a function of coverage radius r and current position x and y. Here r is fixed, but the position may change if the sensor is fitted on a mobile platform.

In an operation, there is a need to get data covering the area of interest. Sensor service A covers the entire area, and also a lot more than is needed. Sensor service B and C each cover parts of the area, whereas sensor service D is not usable in this case since it does not cover the area of interest. Also, sensor service B, C, and D use a proprietary format (x,y) instead of the well known (latitude, longitude). Our client software can only use data of the latter type. Thus, this makes sensor service A the "best" service to use. However, in a dynamic environment the service may become unavailable. Without SWS you would not be able to receive any usable data. However, due to the common vocabulary mentioned above, you are able to use a *semantic broker* which, after a service discovery process, discovers not only the sensor services B, C and D, but also a conversion service that can translate between (x,y) and (lat, long). The semantic broker performs an orchestration of the chosen services (i.e., conversion service and sensor service B and C), and the client finally receives the data it needs.

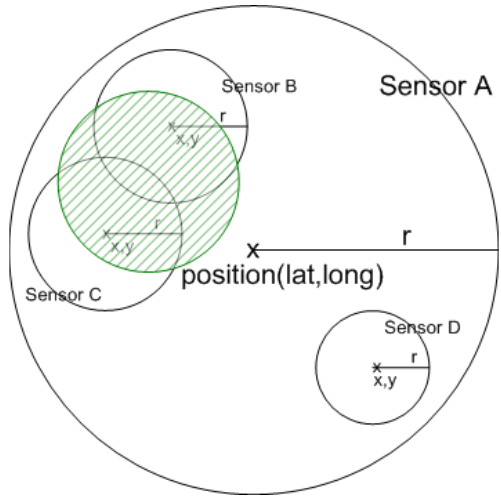

Figure 4. Coalition force area of interest (shaded green), along with four sensor services (A-D) and their positions (x,y) and coverage area (r)

Figure 3 illustrates the needed OWL-S concepts for explicitly defining our example service. Here the service is described in terms of a *profile*, which is used to advertise the service. It defines the input the service requires, and the output it provides. The *process* describes how to interact with the service, and in the case of an orchestrated service, how the orchestration is composed of individual services process descriptions. "Process1" is the first process in the orchestration and "Process2→n" define the rest of the processes in the orchestration. The *grounding* provides information about service specific details such as where the service is located. Typically, the information known about the Web service's interface (i.e., the WSDL) is a part of the grounding. For the sake of this example we have chosen to show a simple sequence construction though others exist.

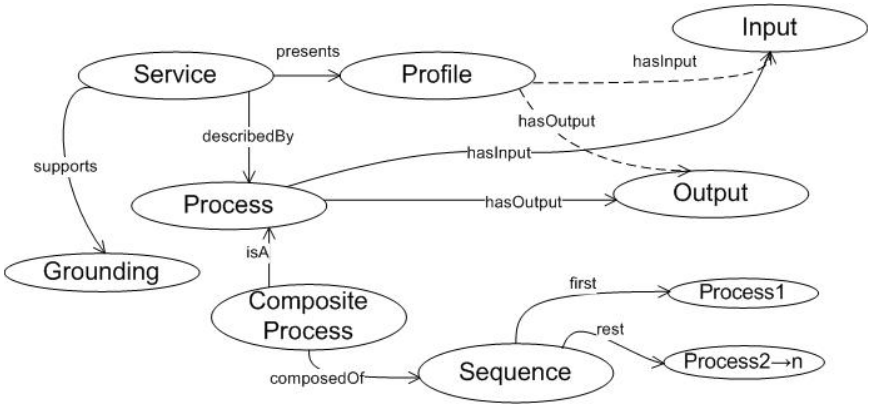

Figure 5. A part of the OWL-S service ontology

Figure 6 shows the service instances of the translator service and the sensor service orchestrated using the sequence construct. The orchestrated service uses the property "first" to indicate that the translator service is to be run first, then the sensor service can be invoked and the client gets the wanted result.

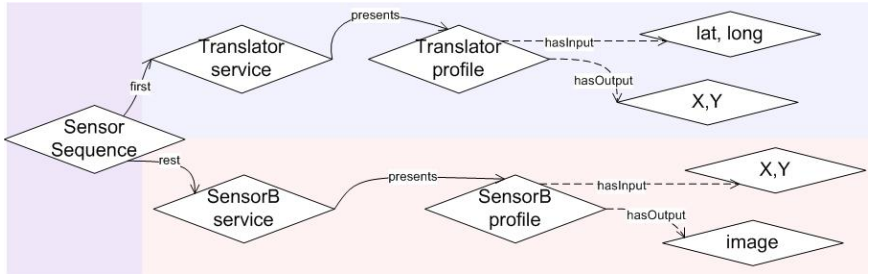

Figure 6. Orchestration of translation service and sensor service

Conclusion and future work

In this paper we have identified interoperability challenges in military networks. NATO has identified service discovery as a Core Enterprise Service to improve interoperability. By using SWS we also enable interoperability on a conceptual level improving on-the-fly discovery, orchestration and invocation. This is the foundation for building an interoperable and agile system. We have implemented a prototype as a proof-of-concept, and shown that the solutions outlined in this paper are feasible to implement. In the future, one could investigate interoperability on a larger scale, for example by leveraging the techniques described in (Mittal et al. 2009). Also, governance issues such as service management, policies, quality of service, etc. must be addressed.

Currently there is no standardized solution that ensures the best possible interoperability situation for dynamic environments. We are participating in the NATO CES WG and RTO IST-090, where we are working towards NATO standardization of the outlined solutions. In order to ensure compatibility between the individual national systems, it is vital to use standards and NATO developed interoperability profiles. We are planning further experiments in that context to fully demonstrate the inherent power of semantic technology.

References

Berners-Lee, T., J. Hendler, and O. Lassila. 2001. The Semantic Web. *Scientific American* May:29-37. <http://www.scientificamerican.com/article.cfm?id=the-semantic-web>

Curbera, F., M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana. 2002. Unraveling the Web services Web, An Introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing* 6(2):86-93. <http://www.cc.gatech.edu/classes/AY2009/cs6210rkr_fall/papers/00991449.pdf>

Gruber, T. R. 1993. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition* 5(2):199-220.

Hansen, B.J., T. Gagnes, R. Rasmussen, M. Rustad, and G. Sletten. 2007. Semantic Technologies. *Norwegian Defence Research Establishment (FFI), FFI-Report 2007/02461*.<http://rapporter.ffi.no/rapporter/2007/02461.pdf>

Johnsen, F.T. and T. Hafsøe. 2009. Web Services and Service Discovery in Military Networks. Proceedings of the *International Command and Control Research and Technology Symposium (ICCRTS)*, Washington, DC. <http://www.dodccrp.org/events/14th_iccrts_2009/papers/010.pdf>

Johnsen, F.T. 2009. An NFFI-based Web Service Discovery Mechanism for Tactical Networks. *Military Communications and Information Systems Conference (MCC 2009)*, Prague, Czech Republic.

Johnsen, F.T., J. Flathagen, T. Hafsøe, M. Skjegstad, and N. Kol. 2009. Interoperable Service Discovery: Experiments at Combined Endeavor 2009 *Norwegian Defence Research Establishment (FFI), FFI-Report 2009/01934*.

Lund, K., A. Eggen, D. Hadzic, T. Hafsøe, and F. T. Johnsen. 2007. Using Web Services to Realize Service Oriented Architecture in Military Communication Networks. *IEEE Communications Magazine* October, 45(10):47-53.

Martin, D. L., M. Paoalucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara. 2004. Bringing Semantics to Web Services: The OWL-S Approach. Proceedings of the *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC)*, San Diego, California. <http://www.cs.cmu.edu/~softagents//papers/OWL-S-SWSWPC2004-final.pdf>

McIlraith, S. A., T. C. Son, and H. Zeng. 2001. Semantic Web Services. *IEEE Intelligent Systems* 16(2):46-53.

McIlraith, S. A. and D. L. Martin. 2003. Bringing Semantics to Web Services. *IEEE Intelligent Systems* 18(1):90-93.

Mittal, S., B. P. Zeigler, and J. L. Risco-Martin. 2009. Implementation of a Formal Standard for Interoperability in M&S/Systems of Systems Integration with DEVS/SOA. *The International C2 Journal* 3(1). <http://www.dodccrp.org/files/IC2J_v3n1_01_Mittal.pdf>

Modi, V. and D. Kemp, eds. 2009. Web Services Dynamic Discovery (WS-Discovery) Version 1.1. *Organization for the Advancement of Structured Information Standards (OASIS), OASIS Standard.* <http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.pdf>

Organization for the Advancement of Structured Information Standards (OASIS). 2007. Web Services Business Process Execution Language Version 2.0. *OASIS Standard.* <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>

Patil, S. and E. Newcomer. 2003. ebXML and Web Services. *IEEE Internet Computing* 7(3):74-82.

Richard, Golden G. III, 2002. *Service and Device Discovery: Protocols and Programming*. McGraw-Hill Professional.