Achieving battlespace awareness in network-centric warfare by integrating web and agent technologies

John M. McCormick*, Peter M. Gerken, Kevin P. Barry, Brian Sidharta
Lockheed Martin Advanced Technology Laboratories
3 Executive Campus, Cherry Hill, NJ, USA, 08002

ABSTRACT
Transformation of military information systems to a network-centric paradigm will remove traditional barriers to interoperability and enable dynamic access to information and analysis resources. The technical challenges of accomplishing network-centric warfare (NCW) require the engineering of agile distributed software components imbued with the ability to operate autonomously on behalf of human individuals, while maintaining system level integrity, security, and performance efficiency on a grand scale.

In this paper, we will describe how agents provide a critical technology enabler for applying emerging commercial technologies, such as web services, into network-centric warfare problems. The objective of our research is developing and sharing battlespace awareness and understanding. Our agent information service manages information collection and dissemination/publishing activities on behalf of fusion services in an autonomous, yet controllable fashion. Agents improve the scalability and reliability at the system of systems level through dynamic selection and exploitation of web services based upon needs and capabilities.

Keywords: Agents, Network-Centric Warfare, Battlespace Awareness, Web Services, Distributed Computing

1. INTRODUCTION
Efforts to transform military information systems to a network-centric paradigm suffer the same fundamental challenge faced by the business enterprise; the lack of a consistent architectural framework where applications can be rapidly developed, integrated, and reused. More importantly, we need an architectural framework that allows the assembly of components and services for the rapid and dynamic delivery of solutions [1]. Service-oriented architecture (SOA) approaches have gained popularity in the commercial sector for integrating disparate business enterprise applications, and represent a viable approach to network-centric warfare (NCW) applications. However, NCW applications require a level of agility far beyond the business enterprise. In this paper we present our approach to accomplishing this through integrating web services with an agent framework to build and maintain battlespace awareness. Our research has shown that agent frameworks fit naturally into SOAs to achieve requisite capabilities of NCW applications. We start by providing background information on the fundamental concepts, and details of our challenge problem for evaluating the approach. We then detail the technical approach employed to the challenge problem, focusing on the SOA aspects of the architecture. We conclude with lessons learned from the development of several software prototypes.

2. BACKGROUND
Legacy information systems have been architected for solving tightly defined needs within the military. Each has been typically addressed to solve that need without much thought for interoperability. Thus, systems of today are hampered by a lack of dynamic interoperability. Recent military research has been directed to leverage commercial web technology developments as a leap ahead capability for departing from these “stovepiped” information systems and to address interoperability. Many of our technical approach decisions are driven by this context of applying web technology to the military, necessitating a brief discussion of NCW principles and identification of critical differences between the NCW environment and the commercial enterprise. Armed with this context, an overview of service-oriented architectures and agent frameworks lays the foundation for the technical approach.

Network Centric Warfare is an emerging concept that will revolutionize the way military organizations operate. The concept is centered upon enabling full information exchange across organizations of varying roles and capabilities. This

*jmccormi@atl.lmco.com, phone 1 856 792-9709, fax 1 856 792-9948, atl.lmco.com
ubiquitous access to information is the key technical enabler for the goal of NCW: Information Superiority [2] across the Force. This is a broad term for holding an advantage over an adversary in terms of information and knowledge. This concept is deeply technical, but it also involves the ability to exploit the information within context of each user. The DoD has defined three domains that are central to NCW [2]:

- Physical Domain, the secure connectivity and interoperability of the Force,
- Information Domain, the ability to access, share and analyze information,
- Cognitive Domain, high quality awareness, and shared understanding of the battlespace.

The foundation is being achieved with respect to the Physical Domain. Typical bottlenecks of processing and bandwidth are being alleviated by emerging technology. Upon achieving robust connectivity relevant data must be gleaned for each user to augment that user/team’s knowledge. This is the interplay between the information layer and the Cognitive Layer. By achieving excellence in these domains the network centric military can achieve and maintain valuable battlefield awareness to achieve self synchronizing, responsive organizations with an increase in speed of command, lethality and survivability [2].

A key tenet of NCW is enabling edge users and systems, those closest to the fight, with unprecedented amounts of quality information. NCW allows these users to engage in a Smart Pull [3] to access and exploit information. The concept of Smart Pull decouples the information collection assets from the user by allowing users to manage their own information needs. Users have the capability to asynchronously subscribe to information feeds that are directly relevant to their mission context, without the limitation of traditional network management. This presents unique technical challenges since the tactical military operates in a harsh physical and computing environment. Although processing, storage and bandwidth capabilities are increasing, the edge warfighter will remain the most constrained due to the unique requirements of the battle. Furthermore, increases in technology capabilities must be designed to assist the warfighter without addition of tasks that would distract from the fight.

The edge user will greatly benefit by the shift to NCW automation assistance for information acquisition, management of resources (bandwidth and processing), sense making, and collaboration with manned and unmanned teammates. Commercial technology can provide the catalyst to achieving this, but we must proceed with forethought. Service oriented technologies such as browser based computing, XML/SOAP based messaging, and web service networks must be carefully examined and explored for feasibility in the military context.

Our research focuses on Service Oriented Architecture (SOA) attributes of NCW by developing novel approaches that prudently apply commercial information technology to the needs of the warfighter. We have built information systems that provide shared awareness and understanding of battlefield data, enabling Smart Pull of information with automation assistance for the warfighter. Our research shows that the careful application of commercial technology enables transformational capabilities when the unique challenges of the military are key constraints in our solutions. Clearly achieving NCW will be an iterative trade off between technology and tactics, but as information technology and Network Centric Tactics trend toward convergence the military’s edge users will benefit.

### 2.1 Service-Oriented Architectures

As business information technology has moved to distributed computing approaches, several models have been employed. Client-server and n-tier models are two common examples. A Service Oriented Architecture (SOA) is an architectural realization of a model designed to solve problems with the following requirements:

- Dynamic discovery of the software component's capabilities,
- Separation between the description of the software's capabilities and its implementation,
- Ability to quickly assemble impromptu processing sequences with minimal coordinated planning efforts, installation procedures, or human intervention.

Within a business environment, a pure architectural definition of a SOA might be something like “an application architecture within which all functions are defined as independent services with well-defined invokable interfaces which can be called in defined sequences to form business processes” [1].

In recent years, web services have risen to prominence within the business enterprise. It is important to note that by themselves, web services do not fully realize a SOA. Web services are a collection of technologies, including XML, SOAP, WSDL, and UDDI, and enable building solutions for specific messaging and application integration problems [1].
The component for defining the sequences of service invocations and providing process control is missing from web services. Each major vendor of web service deployment environments provides some level of capability. Our work in agent-based solutions to intelligent information management proves that agent frameworks fit naturally and synergistically into this role.

2.2 Agent Frameworks

Alan Kay traced the roots of software agents back to the 1950s work of John McCarthy and Oliver Selfridge:

“They had in view a system, that when given a goal, could carry out the details of the appropriate computer operations and could ask for and receive advice, offered in human terms, when it was stuck. An agent would be a ‘soft robot’ living and doing its business within the computer's world.” [4]

Modern agent frameworks provide both a controlled environment, or agent platform, and mechanisms for instantiating agents to operate within that environment. The agent platform represents the physical infrastructure in which agents can be deployed. One mandatory component of an agent platform is an agent management system that exerts supervisory control and access to the agent platform. This includes establishing and enforcing the rules for creation, registration, location, communication, migration, and retirement of the agents. The agents serve as the actors of the system, capable of combining services with internal processing and decision-making to satisfy some objective on behalf of the system or a user [5, 6].

For our research, we used an agent framework developed by Lockheed Martin Advanced Technology Laboratories (LM ATL), the Extensible Mobile Agent Architecture (EMAA). An overview of EMAA is provided in the Technical Approach section. Agent frameworks, such as EMAA, provide a high degree of flexibility in determining how to best utilize services available through a SOA. Architectural requirements of the target application drive the process for making these decisions.

3. ARCHITECTURE REQUIREMENTS

Information collection and dissemination feeds any situational understanding process. Battlespace understanding in a network centric warfare environment is particularly dependent upon the timely collection of digital information. Within our architectural view, multi-level fusion components interoperate to build and maintain the understanding of the battlespace. Information collection processes are driven by the informational requirements of the fusion components. These informational requirements may shift over time due to changes in operational tempo, phases, or commander's direction. Thus a network centric solution must dynamically adapt to these changes without service interruption, or increased burden upon the human user(s).

Our architecture is intended to meet NCW tenets for users across the tactical Army environment, from upper echelons through the lowest echelons. A particular focus is the informational flows to and from the “network edge”, or those users and systems closest to the fight. An Information Collection component for this type of architecture must satisfy the following criteria:

• Decouple data retrieval and service composition details from users.
• Exploit a dynamic array of potential data sources and supporting services.
• Be sensitive to the resource constraints of a tactical computing environment (i.e., bandwidth, cpu and memory).
• Balance the benefits of interoperability with performance requirements.

3.1 Example Use Case Scenario

The following example illustrates how agents integrated with web services meet the dynamic information needs of fusion components by selecting and integrating multiple services. One primary, and very challenging task of the Army intelligence community is enemy course of action (ECOA) development. A fusion system working to aid human users in this activity can organize multiple fusion services, such as track management and aggregation, to build evidence in ECOA hypotheses.

For instance, a user initiates the ECOA service to compute multiple hypotheses of enemy intent. This triggers a series of actions for collecting information feeding the ECOA algorithms. Depending upon the real world scenario, multiple perspectives may be required to accomplish this including, but not limited to: terrain models, weather forecasts, and enemy armament data. These perspectives, and their associated sources, may vary in applicability and availability at runtime. Thus, dynamic assembly of the data would be required to customize the processing for the task at hand.
For our example, the ECOA service uses an aggregation service that groups enemy tracks to gain understanding of the enemy organization. Simultaneously, an ECOA service receiving updated weather effects and terrain data can compute the mobility of the enemy tracks. The addition of armament data enables the ECOA service to include enemy lethality into its assessments. The final product is a prediction of probable ECOAs that can be presented to the human and/or posted for use by other services.

4. TECHNICAL APPROACH

This section describes the architecture developed at LM ATL for managing the interaction between the components, corresponding agents and web services, and finally how LM ATL’s agent technology and web services have been combined for military advantage as in our Example use case.

4.1 Prototype Architecture

Our prototype architecture follows a SOA and NCW principles to provide automation and assistance in building and maintaining battlespace understanding, from upper echelons through the lowest echelons. The information collection behaviors at any echelon are defined by the context of the user. Figure 1 shows the architecture and how its capabilities are tailored to the tactical level of employment. The warfighter needs reliable access to a large variety of data sources and services that can be tailored for multi-level fusion and battlespace understanding.

![Prototype service oriented architecture for battlespace understanding.](image)

There are many types of databases and stand-alone applications available to the warfighter that are useful as separate components. The data sources that a warfighter may use include: an armament database that describes the weapon and sensor ranges of a set of military vehicles; and a terrain database that contains elevation data for the battlefield. A set of applications that a warfighter may use include: a track management application that tracks the position of both blue and red forces; an aggregation component that fuses the individual tracks into a collection of aggregates; a weather component to calculate weather effects on systems; and an enemy course of action tool to determine what the enemy might be doing next. By wrapping the sources in a set of web services, additional insight into the battlespace can be achieved by sharing the data in these components and combining the data with multiple sources.

Our architecture contains four major components: service wrappers, service registry, service composition, agents. In order for a database or an application to be used in a network centric environment, they need to become web serviced enabled so other components can interact with them. The service wrapper for a database contains a set of methods that allows another component to pull specific information that it needs. The service wrapper for an application exposes a list of methods that can be called. The service wrappers register a set of methods with the service registry. The service registry handles all of the bookkeeping necessary to manage and track the web services that are available to the warfighter. Services can register and deregister from the registry as they are needed. The service composition receives requests from other services and tries to match the requests to a set of registered services. The order in which the set of services are invoked is determined by the workflow created be the agent. The agent is created to use the methods registered by the web services in the service registry. More detailed information about these components follows. This architecture combines the web services with software agents to enable dynamic composition to solve complex problems, as in our example scenario.

4.2 Extensible Mobile Agent Architecture

The Extensible Mobile Agent Architecture (EMAA) provides a family of software libraries that enable the creation and deployment of robust and reliable distributed agent systems capable of employing mobility to achieve their functional...
requirements. Developed by the Artificial Intelligence Laboratory of LM ATL, this framework has evolved via over $10M in DARPA and internal funding. EMAA consists of three primary types of components: docks, agents, and services. The relationship of the agents, services, and docks is shown in Figure 2.

![EMAA Architecture Diagram](image)

Figure 2. EMAA architecture diagram.

Conceptually, the dock serves as the operating environment where the agents perform their tasking on a particular host. In order to gain access to its services, an agent must be received and authenticated by the dock. Each dock relies on four basic components in realizing the agent operating environment: the AgentManager, ServiceManager, EventManager, and AgentTransporter.

- The AgentManager provides thread access and scheduling services for the agents at the dock. All agents are registered with the AgentManager upon initialization, or migration to the dock.
- The ServiceManager provides lookup and access to services installed within the dock. The access control can be restricted to allow only authorized agents to access a given service.
- The EventManager provides an asynchronous event-based communication mechanism for use by any component within the dock. Typical uses are for notification when some service or agent has entered an important state.
- The AgentTransporter handles the transmission of agents to and from a dock.

A mobile software agent is an independent thread of processing that can execute tasks at one or more docks across a network with its state preserved. An EMAA agent's code contains the logic to process a workflow of tasks. During migration, EMAA agents are designed to primarily carry data and processing parameters, not complete programs. Services hosted upon a dock provide controllable access to other software components.

EMAA has been designed to provide a network of trusted systems with mobile intelligent agent capabilities, and as such, many features of the architecture directly support agent control and security risk mitigation. The EMAA security architecture builds off the Java 2 security architecture. Security policies can be enforced at a dock based upon digitally signed classes, and secure socket layer transmissions are supported. Ongoing work includes applying digital signatures to mobile agents to allow finer grained permission discrimination at the user or role level.

Further information on EMAA and some of the applications in which it has been applied can be found here [7, 8, 9, 10, 11].

### 4.3 Web Services

The transition to a network centric paradigm will require incorporating existing legacy systems. These are generally “stovepiped” applications and/or databases that do not integrate cleanly with other applications. By wrapping and publishing these systems as web services, the individual applications and databases can be exploited as a collection of services to realize a network centric capability, such as battlespace understanding. A typical web service infrastructure requires the following components: service provider, service registry, and service requestor. Figure 3 shows how these components interact to form a web services solution.

A **service provider** is an owner of a collection of web services. To make these services available, the service provider publishes the service grounding for each service to the service registry. The service grounding describes how to use a service and is typically represented by a Web Services Description Language (WSDL) file [12].

A **service registry** enables discovery and selection of web services by service requestors. In a business enterprise implementation, UDDI is the web services standard for service registration and description [12]. The current UDDI
specification is commercially oriented and doesn’t map well to the military domain. Our approach was to utilize Ontology Web Language – Services (OWL-S) to satisfy our service registry requirements. We discuss the capabilities of OWL-S in more detail in the next section.

A service requestor discovers and invokes web services provided by one or more service providers. After discovering which services match its requirements the service groundings may be retrieved from the service registry. The desired service invocations can then be performed using the designated message format; SOAP/XML is the most common and interoperable [12].

4.4 Integrating Web Services and EMAA

Due to changes in operational tempo, phases, commander's direction, or network unavailability of services, it is not possible to know how the services should be called until runtime. Numerous services may play a role in providing data or calculations necessary for battlespace understanding. Using the service descriptions, the service composition component is capable of selecting multiple services from the service registry and ordering the sequence of service invocations to ensure all parameter dependencies are satisfied. This sequence of service invocations is captured in an agent workflow and assigned to an EMAA agent that processes it, delivering the results to the user.

Figure 4 describes how an agent’s workflow is dynamically determined when a request for information is received. The warfighter sends a request for a set of possible enemy courses of action to the service composition. Based on the service groundings from the service registry, the service composition component determines that ECOA needs the following parameters: a set of aggregates, and sensor/weapon ranges. Likewise, the aggregation service needs track data and terrain data, and the armament service needs to know the weather effects on the tracks. The weather effects are calculated by providing terrain and track data to the weather service. With all these dependencies satisfied, an agent workflow is constructed with the appropriate sequence of service invocations. Based on the combined information provided by the agent, the ECOA service determines that blue forces are at risk of an ambush by red forces.

Solving this complex sequence of dependencies requires more than just method names and parameters provided by a standard WSDL. We need a semantic description of the services so we can infer which services may be useful. We accomplish this by describing the services using OWL-S.
OWL-S was developed to enable users of web services and software automation (i.e., agents) to “discover, invoke, compose and monitor Web resources offering particular services and having particular properties” [13]. A software agent needs to understand a description of the service in order to use it. OWL-S provides a layer of semantic description beyond the existing web services protocols. It is this semantic description the enables the service composition component to reason about service capabilities and needs in order to properly select and assemble service sequences to accomplish complicated tasks realized by the agents.

An OWL-S service description is composed of three elements [13]:
1. Service Profile: describes what the service does, including capabilities and needs. This description is critical for dynamic matchmaking.
2. Service Model: describes how the service works.
3. Service Grounding: describes the mechanics of how to use the service, such as method invocation and parameter requirements. OWL-S uses WSDL as its service grounding specification

Currently our architecture only implements the Service Profile and Service Grounding. We do not have an identifiable requirement or capability enhancement for using the Service Model at this time. Furthermore, the level of modeling detail that could be provided for a military application is not clear.

The OWL-S file captures the methods and associated parameters that the web service provides in a well-defined ontology. When the service composition queries the service registry, it specifies the parameters and results in ontological terms, not base object types. The service registry uses these terms for determining which registered services match. Our OWL-S description includes a parameter indicating whether or not this web service is also an EMAA service. This gives the service composition a choice to invoke the service through SOAP/XML messaging or direct Java call. This attribute was added to improve performance by avoiding expensive SOAP/XML calls when Java methods could be used. This capability required some enhancements to the standard EMAA infrastructure for integration with a web service provider.

EMAA uses a custom configuration file to instantiate and initialize its internal services. Similarly, a web service container, such as Apache Axis, instantiates and configures the web services listed in its own configuration file. Normally, both EMAA and Apache Axis must control the initialization of services. However, in order to allow a service instance to act as both a web service and an EMAA service, an extension to EMAA was created called the GhostServiceManager. Figure 5 shows the initialization process for an exemplar service that can be invoked by either web service protocols or EMAA directly.

Figure 5. Ghost service initialization example.

The GhostServiceManager uses an additional ghost-service list in the EMAA configuration file. On EMAA system startup, however, the services listed in this ghost-service section are not actually installed. Instead, the configuration parameters for each ghost service are stored in memory. When an EMAA-aware web service is instantiated by Axis, that service notifies the GhostServiceManager, which retrieves the appropriate configuration parameters from memory. The GhostServiceManager then establishes access to the service by registering it as an EMAA service, passing the configuration parameters as part of the registration process.
The current implementation of dual access services requires a running EMAA dock to allow for easier configuration. Accessing the service through EMAA is faster because XML-to-Java-object marshalling/unmarshalling is not required. External services or applications can still interact with the component as a web service, but the performance will suffer due to the SOAP/XML protocol.

5. LESSONS LEARNED

The most immediate lesson learned was the steep performance processing price for the interoperability benefits of web services, caused primarily by the XML-to-Java-object marshalling/unmarshalling required for SOAP/XML messaging. It is important to analyze the operational performance of an application to be wrapped as a web service. The design process should seek to find the proper granularity of services to limit the number of web service calls. This service granularity analysis grows in importance as the services approach the tactical edge. Our dual access service approach enabled us to mitigate the messaging costs for web services that we internally developed while still allowing interoperability with external services.

A second lesson learned was a reinforcement of good system interface design. It is far better to have more specialized methods that take simple parameters, than fewer general methods requiring very complex data structures. The smaller objects required for simple parameters reduces the SOAP/XML messaging cost. More importantly, complex data structure parameters pose a significant challenge for resolving the service parameter dependencies within the service composition component.

Elements of this prototype architecture, with services similar to those described above, have been implemented by LM ATL for Army research programs, including the Logistics Command and Control Advanced Technology Demonstration, the Agile Commander Advanced Technology Demonstration, Warrior’s Edge, and the Fusion Based Knowledge for the Objective/Future Force Science and Technology Objective. The merger of web services and agent technology provides a very flexible and dynamic system for assisting the warfighter. In the near term, network centric principles can be realized through the prudent integration of agents and emerging web technologies to create flexible battlespace awareness applications.

REFERENCES

1. K. Channabasavaiah, K. Holley, E. Tuggle, “Migrating to a service-oriented architecture, Part 1”,
13. The OWL Services Coalition, “OWL-S: Semantic Markup for Web Services,”
   [http://www.daml.org/services/owl-s/1.0/owl-s.html](http://www.daml.org/services/owl-s/1.0/owl-s.html)