Integrating Agents into Data-Centric Naval Combat Management Systems

Cihat Eryiğit
Istanbul Technical University
Computer Engineering Department
Maslak 34469 Istanbul, Turkey
ceryigit@armerk.tsk.mil.tr

Şima Uyar
Istanbul Technical University
Computer Engineering Department
Maslak 34469 Istanbul, Turkey
etaner@itu.edu.tr

Abstract

The number of systems and software developed by using agent oriented technologies has increased dramatically within the last decade. In parallel with this increase, lots of agent based tools, techniques, methods and platforms have been proposed. Some of the agent platforms were designed for generic purposes, while the others were tailored for specific needs. Although there are many existing platforms, we still need novel agent platforms supporting the new promising application domains. Naval combat management systems is one of these domains due to its complex and dynamic nature. In the Information Age, intelligent systems become an inevitable necessity for combat management systems while the network-centric warfare supersedes the conventional platform-centric concepts. In this paper, we will discuss how agent technology can be integrated with data-centric naval combat management systems. The proposed solutions are based on the Data Distribution Service (DDS) infrastructure. DDS is an Object Management Group (OMG) middleware standard using the publish-subscribe paradigm and quality of service parameters for data-centric asynchronous communication.

1. Introduction

The main purpose of warships is accomplishing the assigned tasks by eliminating threats and attacks during the mission. The naval warfare tasks generally are classified according to threat environment. These are Anti Surface Warfare (ASUW), Anti Submarine Warfare (ASW), Anti Air Warfare (AAW) and Electronic Warfare (EW). Intelligence, surveillance, and reconnaissance activities are additional important duties of warships. Modern naval platforms have the following systems: Sensors, Weapons, Fire Control Systems, Guided Missile and Torpedo Launchers, External and Internal Communication Systems, Tactical Data Links, Navigation Systems, Aircraft and Helicopter Support Systems.

The responsibility of naval combat management systems is to command and control all warfare systems listed above and give decision support to operators and commanders. Naval combat management systems are becoming more and more complex due to advances in weapon and sensor technologies. On the other hand, conventional platform centric military concepts, tactics, doctrines and strategies are being replaced by network centric ones [1, 5]. Intelligent agent systems are the best candidates for dealing with this complexity. Studies in [2, 3, 4, 6] can be shown as examples of combat management system applications developed using agent technologies.

Data Distribution Service (DDS) is the first open international middleware standard directly addressing publish-subscribe communications for real-time and embedded systems [8]. Its first version was published in 2004 by Object Management Group (OMG). DDS middleware is widely used as the communication infrastructure in existing defense systems. One of these systems is naval combat management systems. Additionally, US Navy's Open Architecture Computing Environment (OACE) recommends the DDS and CORBA for the next generation defense systems. In this work, we will discuss how agent technology can be integrated with data-centric naval combat management systems. The proposed solutions are based on the Data Distribution Service (DDS) infrastructure.

In the following section, details and functionality of Data Distribution Service will be explained. Later, we will discuss why DDS is appropriate as an agent platform. In section 4, two different integration methods will be described. The final section will conclude the paper and present the future work.

2. Data Distribution Service (DDS)

DDS [8] is a new specification for Data-Centric Publish-Subscribe (DCPS) communication mechanism. It aims to facilitate the communication between distributed applications (data senders and receivers) by providing a software API which can be used regardless of the system's network topology.

DDS introduces a virtual global data space where applications on the same or different nodes can share information by simply reading and writing data-objects described as topics. Fig. 1 shows the relation between the components of DDS; domains, domain participants, topics, publishers, data writers, subscribers, and data readers.

Domains: A DDS domain represents a separate global data space. All the information in a domain is only accessible by the applications registered to that domain. Although the figure comprises only a single domain, DDS can support multiple domains in order to provide data isolation.

Domain Participants: The applications participating into a DDS domain are represented as domain participants. As can be seen from the figure, a node may contain one or more domain participants.

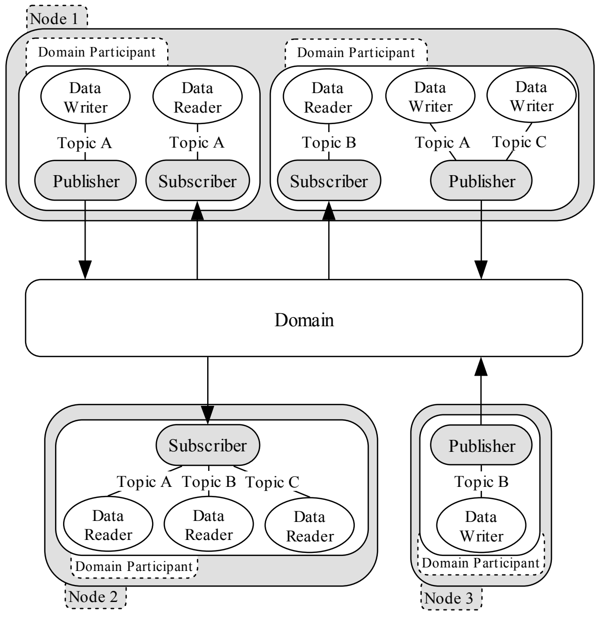

Fig. 1. DDS Components.

Topics: DDS topics define the data structures used in the communication. They consist of two parts: topic name and topic type where the later is the description of the data to be published or subscribed. Within the topic type, one or more data elements can be defined as "key" which can be used to sort or query the data.

Publishers: Publishers are the objects responsible for the data distribution. They may publish different types of data, e.g. the publisher of Node 1 2nd domain participant which publishes both Topic A and Topic C.

Data Writers: They are the objects which set the values of the data objects of a given type and pass it to the publisher for sending. Unlike publishers, they can only work with a single data type.

Subscribers: Subscribers are the objects responsible for the reception of the data for which it is subscribed and making it available for the receiving application. They may receive different types of data, e.g. the subscriber of Node 2 in Figure 2 which receives Topic A, Topic B and Topic C.

Data Readers: Data readers are the objects attached to subscribers in order to access the received data. They allow the application to declare the topic to which it is subscribed. The application can access the received data both synchronously and asynchronously by using the Listener Callback Routine and Wait-Sets. In the former method, DDS runs immediately a listener callback routine when the subscribed data is received. In the later one, the application waits until one or more condition is met before accessing the data. As a third method, it is also possible for the application to query the data reader in order to check if the data is available.

3. DDS As An Agent Platform

Agent technologies are suitable for problem domains which have the following characteristics according to [9];

– There is a dynamic number of components and new components to be introduced at any time.
– An external control of the entities comprising the system is not possible or not wanted, i.e. the system components have to be autonomous and self-dependent.
– The coordination within the system takes place by using complex communication relationships.

When we examine the architectural benefits of the Data Distribution Service and the characteristics specified above, we see that these items overlap almost completely. DDS design supports the agent technology natively. In the following subsection, this support will be explained explicitly.

3.1. Environment

"An agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives" [10] is one of the well-known agent definitions. Although there is no common agent definition, most of them emphasize an environment which agents interact with and effect. DDS's global data space is ideal for representing agent environment by using transient or persistent durability QoS parameters. If durability parameter is defined as transient, environmental data is preserved during runtime. On the other side, persistent data can be used to initialize the environment on startup.

3.2. Autonomy and Dynamism

Asynchronous communication based on the publish-subscribe paradigm of DDS, supports the agent autonomy. Agents do not have to rely on the existence of the other agents to act or setup communication.

Agent modules can be created dynamically at any time and any node without using any specific order.

3.3. Knowledge Base and Fault Recovery

Agents need to keep a knowledge base and internal data structure for reasoning and programming respectively. DDS data space can be used for both purposes. When the DDS data space is used with the transient durability parameter, agents get fault tolerance capability. When an agent crashes, it can recover its previous state from DDS after restart.

3.4. Mobility

As stated previously, DDS creates a global data space spanning all nodes in the specified domain. Agents can be moved easily between nodes without transferring the agent's state explicitly by using Durability QoS parameter.

3.5. Environment Isolation

It is possible to create different data spaces separated from each other by the help of DDS domains. Agents belonging to different groups or executing different tasks can communicate in their isolated domains. It is also feasible to allow coordinator agents to communicate within more than one domain. In addition, this capability can be used for debugging and test purposes.

3.6. Monitoring and Discovery

Liveliness QoS parameter can be used to monitor states of agents and discover the new agents joining the domain. There is no need to build application level heartbeat and liveliness check mechanisms.

3.8. Reliability and Priority

Transportation of agent messages can be regulated with respect to their importance and priority. An agent can be sure that its message will be delivered to all recipients if it is sent with the reliable QoS parameter. There is no need to build application level acknowledgement mechanisms. Additionally, agent messages can be prioritized by using transport priority QoS parameter.

3.9. Portability and Scalability

Abstract communication Application Programming Interface (API) of DDS isolates the agents from host operating system and hardware. This isolation increases the portability of agent applications. On the other hand, resource limit QoS parameter can be used for limited platforms to achieve scalability.

4. Integration

It is necessary to integrate agent technologies with DDS infrastructure for using agents in data-centric naval combat management systems. However, we need the agent platforms, development environments, test and debugging tools to develop agent-based systems for this domain. It is not possible to use the existing agent platforms directly because there is no agent platform which uses DDS as a communication infrastructure. There are two alternatives for integration of agents and data-centric naval combat management systems. One of solutions is DDS adaptation of an existing agent platform. The other one is to expand DDS infrastructure to support agent-based systems.

4.1. Adapting An Existing Agent Platform

The Jadex agent platform (www.informatik.uni-hamburg.de/projects/jadex) is a good candidate to integrate agents with DDS infrastructure. Jadex is an agent platform based on the BDI reasoning model [7]. It supports both agent platform categories mentioned above: middleware platform and reasoning platform. Jadex does not introduce any special agent programming languages; it uses common languages like XML and Java. This allows Jadex developers to use existing development environments and tools easily. One of the distinguishing features of the Jadex agent platform is that the reasoning layer of Jadex was designed to operate with different kinds of middleware platforms. Current version (0.96) of Jadex supports three middleware platforms: Jadex Standalone Platform, JADE and DIET (experimental). It is possible to use existing agent middleware platforms with Jadex by implementing specified adaptation interface. By the help of this capability, Jadex can be integrated with DDS infrastructure.

4.2. DDS-based Agent Platform

In this section, we will present the design of a new prototype agent platform developed by using the Data Distribution Service components and concepts. As explained above, DDS is very suitable as an agent platform but agent specific infrastructure services should be implemented. FIPA agent management model can be used as a reference for this purpose.

The basic DDS agent platform should be comprised of the following items:

- Agent Manager service
- DDSAgent base class
- AgentControl Topic
- RegistryEntry Topic
- DfEntry Topic
- AclMessage Topic.

Agent Manager service and RegistryEntry Topic are used to fulfill the similar functionality of FIPA Agent Management System. DfEntry Topic is defined for the same functionality as the FIPA Directory Facilitator. Message transportation is provided by DDS infrastructure.

Agent Manager creates and registers DDSAgents and controls their execution via invoke, suspend, resume and terminate commands. AgentControl Topic is used to control the execution of agents. Agent Manager writes the AgentControl topic by filling in the id of the agent to be controlled and the desired command. When a DDSAgent gets the AgentControl topic, it acts according to the specified command. DDSAgents read only AgentControl topics addressed to themselves. Registration of agents is also made by the Agent Manager after the creation of the agent. Agent Manager does not hold internal registration information. RegistryEntry topic is specified for this purpose. Durability QoS parameter of RegistryEntry topic is set to transient and thus all instances of RegistryEntry are kept by DDS. Unique agent identification numbers are created and assigned by the Agent Manager on creation. It is also possible to monitor existing agents and their states by the help of the Agent Manager.

DDSAgent class is the base class for all user defined agent classes. DDSAgent implements a simple task-based model and contains three major functionalities; initialization, execution of periodic tasks and execution of conditional tasks. In initialization, Data Readers and Data Writers are created for topics we are interested in. After initialization, periodic and conditional task are executed in a loop. DDS Conditions and Wait-Set functionality are used for implementation of conditional tasks.

Directory facilitator provides yellow page services to agents. It is not necessary to develop a separate service for the directory facilitator. It is sufficient to define the DfEntry topic and set the durability parameter to transient. DfEntry topic keeps the services, protocols, ontologies and languages supported by the specified agent. Creation, deletion, modification and search operations can be made by DDS write and read functions.

In data-centric naval combat management systems, all sensor data and command orders are distributed over DDS infrastructure. Agents should be integrated into the existing systems by using one of the aforementioned approaches, so that they can obtain the combat system data, process them and fulfill the required combat functionality.

5. Conclusions and Future Work

Each problem domain has its own unique characteristics and requirements. It is critical to choose right technologies when building systems. We showed that the agent technology is very suitable for naval combat management systems and current studies indicate that it will be widely used in the near future. We also showed that architecture, concepts and semantics of the Data Distribution Service natively supports the requirements of agent-based systems. We proposed two approaches for integration of agent technology and data centric naval combat management systems. One of the solutions offers building a new flexible and expandable agent platform on top of DDS, but it requires a large amount of development effort. On the other hand, adapting an existing agent platform can be useful for rapid integration with limited capability.

In future work, we will compare the proposed solutions by making experimental case studies and examine the advantages and disadvantages of the both approaches in more detail.

6. References

[1] Albert, D.S., Garstka, J.J., Stein F.P. 2000. Network Centric Warfare: Developing and Leveraging Information Superiority. CCRP Publication Series, Second Edition.

[2] Allsopp, D., Beautement, P., Kiton M., Bradshaw, J. M., Suri, N., Tate A., Burstein, M. 2003. The Coalition Agents Experiment: Network-Enabled Coalition Operations. Journal of Defense Science Special Issue on Network-enabled Capabilities 8(3) (September 2003), 130-141.

[3] Beaumont, P. 2004. Multi-Platform Coordination and Resource Management in Command and Control. M.Sc. Thesis. University of Laval, Quebec, Canada.

[4] Computer Technology Associates – CTA. 2005. Agents for Network-Centric Warfare and Time Critical Targets, Case Study.

[5] Force Transformation Director, Office of the Secretary of Defense. 2003. Network Centric Warfare: Creating A Decisive Warfighting Advantage. DoD, USA.

[6] Jou S-H., Kao. S-J. 2002. Agent-based infrastructure and an application to internet information gathering. Journal of Knowledge and Information Systems, 4(1) (January 2002), 80–95.

[7] Lars Braubach, Alexander Pokahr, Winfried Lamersdorf. Jadex: A BDI Agent System Combining Middleware and Reasoning, Chapter of Software Agent-Based Applications, Platforms and Development Kits, Birkhäuser Book, Editors: R. Unland, M. Calisti, M. Klusch.

[8] Object Management Group - OMG. 2004. Data Distribution Service for Real-Time Systems Specification, Version 1.0, December 2004. Object Management Group Specification.

[9] Weis, G. 2001. Agent Orientation in Software Engineering. Knowledge Engineering Review 16(4) (December 2001), 349-373.

[10] Wooldridge, M. 2002. Introduction to Multiagent Systems. John Wiley & Sons, Chichester, England.