WHITEPAPER

Transforming MOSA Systems: The Benefits of Deploying a Data-Centric Architecture

AUTHOR

Andre Odermatt
Principal Application Engineer, RTI

OVERVIEW

A Modular Open Systems Approach (MOSA) is the method recommended by the U.S. Department of Defense (DoD) for the implementation of open systems. MOSA uses a decoupled system architecture that allows system components to be incrementally added, removed or replaced throughout the lifecycle of a system platform, providing opportunities for enhanced competition and innovation.

The DoD recognizes the importance of data and considers data as a strategic resource. A data-centric architecture is uniquely able to accomplish this goal because it not only allows systems to be decoupled and independent from each other, but also focuses the systems on the data itself. This approach supports both MOSA and DoD data strategies. A data-centric architecture makes data the common element for systems to act on and consume, while also producing data that other systems can use.

A data-centric architecture can be achieved by using open standards which are at the core of MOSA. Current DoD standards, including the Future Airborne Capability Environment™ (FACE) Technical Standard, already embrace a data-centric architecture by emphasizing the importance of data. For example, the FACE™ approach encourages the use of a data model and requires FACE Units of Conformance (UoCs) to have a well-defined data model that can provide the structure to communicate through the Transport Services Segment (TSS) of the FACE architecture.

A data-centric architecture is an excellent use case for a Publish-Subscribe architecture, based on the open Data Distribution Service (DDS™) standard, which also emphasizes the importance of data. Individual systems declare the intent to subscribe to data or advertise the data they have to offer. The Publish-Subscribe protocol then matches publishers and subscribers with each other and makes sure the data gets to the right place at the right time. New participants can join at runtime without the need for code modifications, enabling rapid updates and technology refreshes using the existing data architecture.

DATA CENTRICITY

Most software developers and system architects take an application-focused view on defining systems. They start off with defining the applications which are needed in a system (e.g., a GUI, processing, etc.). The next step is defining what the interfaces should be, and what is needed by each application. The interfaces keep evolving as new functionality is added and in most cases requires additional information. However, what really matters is the data, not the application. The application is simply a way to display the data of interest. The same data is as important if it is displayed in a different application. For example, you can look at revenue numbers in Excel, Word, Notepad, Vi, etc. The way the application displays the data may make it easier for the user to understand; however, the data has the key information and not the application. The information is in the data.

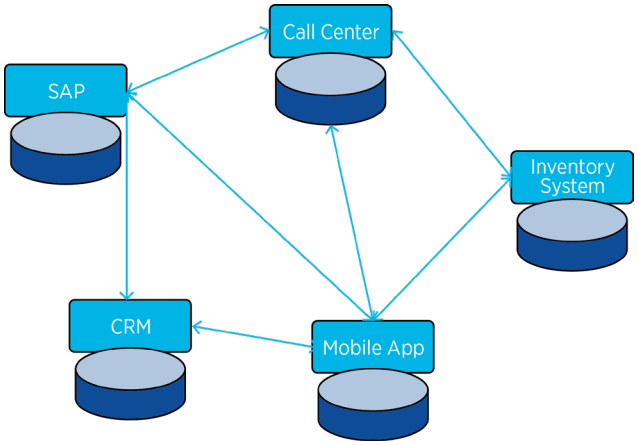

Figure 1: Application-centric approach

Data centricity puts data at the center of the architecture. Data is the primary asset of the system.

Realizing the importance of data, the FACE approach requires a data model which provides a standard method for defining the data format and meaning for information-sharing between software components (e.g., FACE Unit of Portability (UoP)). The FACE Data Architecture is a key component of the FACE Technical Standard and is essential to designing systems and developing interoperable UoPs.

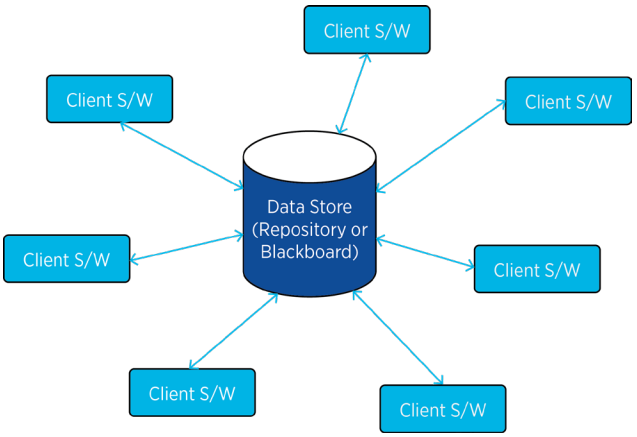

Figure 2: Data-centric architecture

The DoD is rapidly transitioning to a data-centric enterprise. Data centricity enables decision dominance at the speed of war and supports DoD efforts such as Joint All-Domain Command and Control (JADC2) that integrates sensors to command and control (C2) to effector systems across different operational commands, from the tactical edge to the cloud. This aligns exactly with the DoD Data Strategy announced in September 2020 that directs DoD leaders to evolve all DoD assets into data-centric assets that treat data as a weapon system.

In a data-centric system, the design starts with the data model. What data do I need, and how will it be represented? Different components can access shared data and communicate through shared data. Access rules can be put in place so only allowed and authenticated users can access the data. Since the data scheme is defined and all applications act on the same data, integration of new components is minimal and in most cases no data transformation is required. An example of a data-centric architecture is a database architecture. A common database schema is created and accessed through a defined protocol. Another advantage of this approach is that there are no dependencies between applications as the applications do not directly communicate with each other. All communication is done through accessing (reading or writing) data.

For a real-time system, it is not practical to exchange data by writing and reading data into a central database. In addition to the latency, the database server would be a single point of failure. To facilitate communication in a real-time system, the DDS standard, managed by the Object Management Group® (OMG®), was developed. DDS not only supports a data-centric architecture; it also uses a simple Publish-Subscribe paradigm. As we established earlier, a data-centric approach focuses on data and has a data model. DDS requires a data model and is aware of the data being exchanged. In addition, the Publish-Subscribe paradigm with the discovery feature allows new applications to subscribe or publish any of the data defined in the data model.

The benefit of the Publish-Subscribe system is that the application does not need to know where to send the information or from where to request the information. To illustrate this, we will look at a classic message system which is similar to email. As a participant, I must email a status report every week. I need to know whom to send it to. I need to list all the recipients. If there is a new person who is also interested in the weekly status email, that person needs to let me know so I can add the address to the future status emails.

Publish-Subscribe works more like social media where I post my status and whoever follows me receives the update. If a new person is interested in the status, that person can follow (subscribe) to my status and automatically get the updates. I do not need to know how many followers I have. All I do is post (publish) my status update. DDS has another advantage. Since it is data-centric, I can apply filters on any of the data elements. Instead of getting a notification of every new status report, I now get updates only when a filtering criteria is met. Instead of getting every status report, I only get the status reports which have relevant information. The filter criteria can be changed at any point if the data of interest changes.

THE DATA DISTRIBUTION SERVICES (DDS) STANDARD

Let us have a closer look at DDS. The DDS standard is a set of standards which defined the application API as well as the Real-Time Publish-Subscribe (RTPS) wire protocol that enables a transport-agnostic, open standards-based capability to rapidly connect applications from a wide range of programming languages, operating systems and computing platforms. DDS offers the ability to separate transport from the application logic. The DDS standard currently includes the following:

- High-performance, scalable, secure, and data-centric Publish-Subscribe abstraction
- Completely decentralized architecture with dynamic discovery service that automatically establishes communication between matching peers
- Rich Quality of Service (QoS) characteristics for control over every aspect of data distribution, such as data availability, resource usage, reliability and timing
- Interoperable data sharing, platform-independent extensible data modeling, encoding and representation
- Recent extensions for Remote Procedure Calls (RPC), security, resource-constrained devices, web integration and OPC UA integration

DDS provides data sharing that is controlled by QoS policies. Applications communicate by publishing and subscribing to DDS Topics identified by their Topic name. Subscriptions can specify time, content filters, and receive a subset of the data being published on the Topic. Different DDS domains are completely independent of each other, as they do not share any data communication. In the case of UDP, this is achieved by using separate ports for each domain. There is no data-sharing across DDS domains. Figure 3 is an example of DDS communication using a single domain.

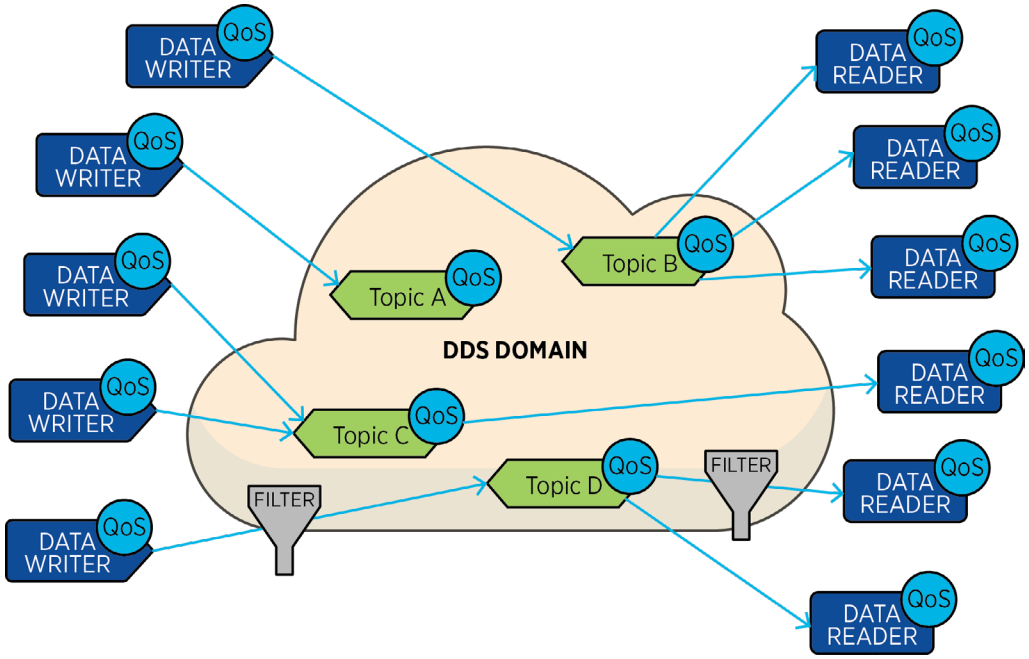

Figure 3: Data-centric Publish-Subscribe

DDS participants can be on the same machine or across a network. In either case, the application uses the same DDS API for communications. Because there is no need to know or configure IP addresses or consider the differences in machine architectures, the process of adding additional communication participants on any operating system or hardware platform becomes an almost trivial task.

MODULAR OPEN SYSTEMS APPROACH (MOSA)

MOSA is a technical and business strategy for designing affordable and adaptable systems. The Five Principles of MOSA are:

1. Establish an Enabling Environment
2. Employ Modular Design
3. Designate Key Interfaces
4. Use Open Standards
5. Certify Conformance

HOW DDS AND DATA CENTRICITY SUPPORT MOSA

The first principle of establishing an enabling environment is really a business practice which supports the program manager in deploying MOSA throughout the acquisition lifecycle.

MOSA requires a system architecture that allows several major system components at the appropriate level to be incrementally added, removed or replaced throughout the lifecycle of a major system platform, to afford opportunities for enhanced competition and innovation. Using a data-centric approach addresses principles two and three. A data-centric architecture inherently enables a module design as all components access a common set of data which also serves as the interface description. Creating a data model or using a standard data model is the foundation of a successful data-centric MOSA system. This can be a separate domain-specific data model or can be part of a specification (e.g., UMAA, ROS).

Using open standards is another key principle. With open standards, application developers are not locked into a specific technology or vendor. The market decides which technologies are the most viable. Open standards establish protocols and building blocks that can help make applications more functional and interoperable which streamlines product development. Many of the open standards have multiple revisions and have improved from lessons learned over time. Using open standards is much more cost- and time-effective than new development. DDS is an example of an open standard which fits perfectly in many of the MOSA applications. The DDS standard has an active and vibrant community continuously working to extend its applicability. The DDS standard currently includes the following:

- Core specification (DDS, RTPS, DDS-XTypes™, DDS-Security™)
- Language mappings
- APIs: C, C++, Java, and others
- Gateways (XRCE, Web, OPC)

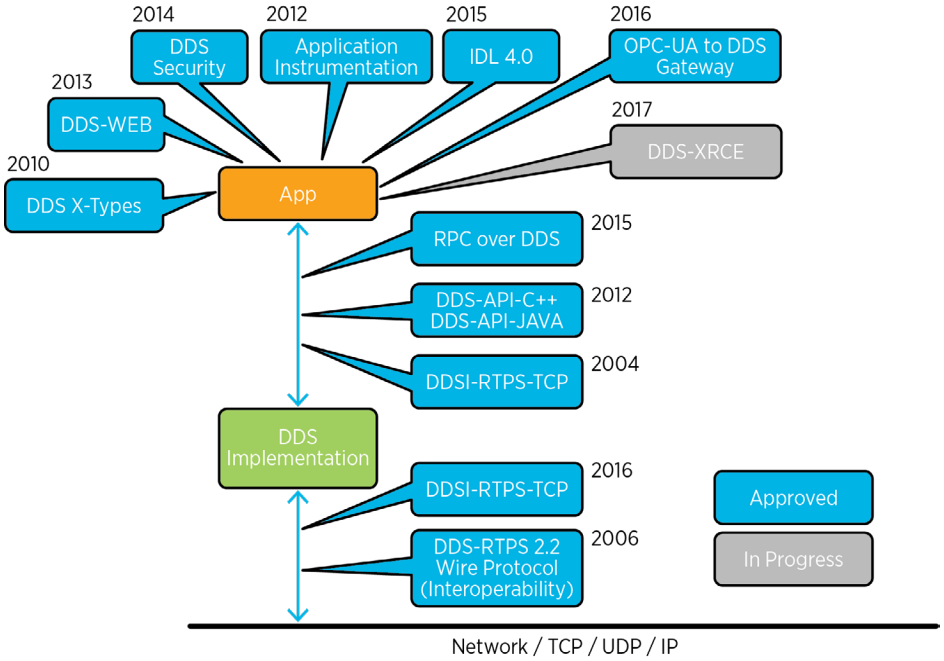

Figure 4: Timeline of individual DDS specifications

DDS is also used in many other standards including FACE, UMAA, ROS, GVA, and others. Figure 5 shows some of the standards where DDS is used.

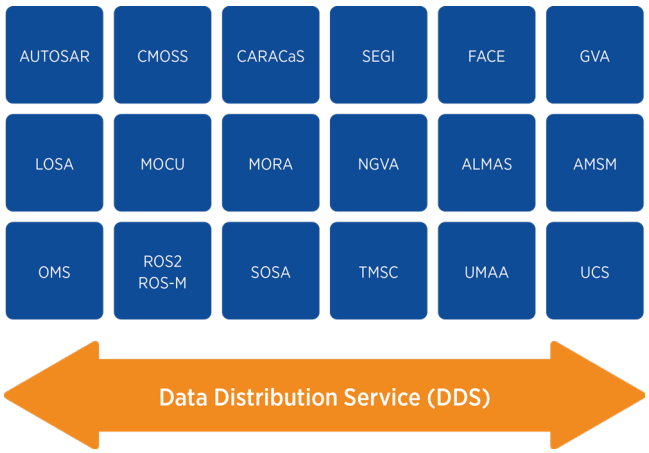

Figure 5: DDS in other Standards

DDS USE CASES

DDS and RTPS packets are used for mission-critical applications, connecting embedded systems as a reliable communication layer between sensors, controllers and actuators. DDS runs many of the most complex systems in the world. DDS powers the largest energy plants in North America, connects perception to control in autonomous vehicles, coordinates combat management in defense systems, drives a new generation of medical robotics, controls hyperloop and flying cars, provides 24/7 medical intelligence for hospital patients and emergency victims, and more.

At the TSOA-ID 2022 conference, RTI demonstrated interoperability between different native DDS applications (e.g., Unity® and Epic Games® Unreal Engine®) and standards using DDS (e.g., FACE, ROS). The demonstration consists of two drone simulators: one using the Unity gaming engine, and the other using the Epic Unreal Engine gaming engine. These drone flight simulators publish location, video and lidar using the Sensor Open Systems Architecture (SOSA™) Interaction Binding and publish data through a SOSA-aligned plugin. The lidar data is displayed using the RVIZ, a ROS 2 application. Location is fed into different Cockpit Display Systems (CDS). Figure 6 shows the architecture of the demonstration. The integration of all those components was made possible by using a common data model. New components can easily be added, and variations of the demo have been demonstrated at different trade shows.

CONCLUSION

MOSA recognized that to reduce cost, time to market, and to develop better, more flexible solutions, it must embrace open architectures. The era of black box and vendor-locked technologies has ended, and the industry must embrace open standards in all types of software ranging from commercial off-the-shelf to open source. MOSA is not just an innovative idea,

it is a legal requirement — specifically Title 10 U.S. Code 2446a. (b), Sec 805 of DoD Acquisition Regulations — to achieve the development of affordable, adaptable systems.

It has been a few years since the DoD Data Strategy was created, and the DoD continuously makes strides toward its goal of transforming into a data-centric agency. To support that transition, new systems need to embrace a data-centric strategy.

Open standards such as DDS will become an essential part in supporting the DoD's transition to a data-centric world. A data-centric open standard such as DDS enables the fundamental step for designing, developing and deploying next generation, distributed mission-critical systems and supports the MOSA principles in achieving: 1) Lower lifecycle costs in deployment and maintenance; 2) Faster time to market; and 3) Improved reliability in deployment.

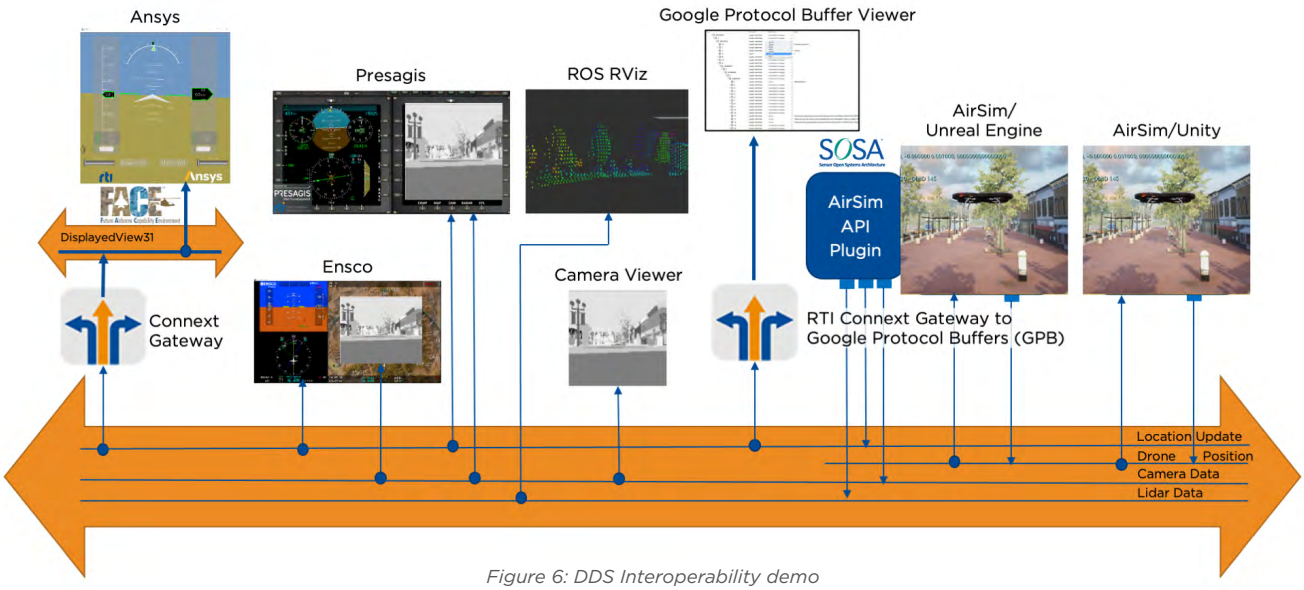

Figure 6: DDS Interoperability demo

ABOUT RTI

Real-Time Innovations (RTI) is the largest software framework company for autonomous systems. RTI Connext® is the world's leading architecture for developing intelligent distributed systems. Uniquely, Connext shares data directly, connecting AI algorithms to real-time networks of devices to build autonomous systems.

RTI is the best in the world at ensuring our customers' success in deploying production systems. With over 1,800 designs, RTI software runs over 250 autonomous vehicle programs, controls the largest power plants in North America, coordinates combat management on U.S. Navy ships, drives a new generation of medical robotics, enables flying cars, and provides 24/7 intelligence for hospital and emergency medicine. RTI runs a smarter world.

RTI is the leading vendor of products compliant with the Object Management Group® (OMG®) Data Distribution Service (DDS™) standard. RTI is privately held and headquartered in Sunnyvale, California with regional offices in Colorado, Spain and Singapore.

Download a free 30-day trial of the latest, fully-functional Connext software today: www.rti.com/downloads.

RTI, Real-Time Innovations and the phrase "Your systems. Working as one," are registered trademarks or trademarks of Real-Time Innovations, Inc.
All other trademarks used in this document are the property of their respective owners. ©2022 RTI. All rights reserved. 50053 V1 1122

Your systems.
Working as one.

CORPORATE HEADQUARTERS

232 E. Java Drive, Sunnyvale, CA 94089
Telephone: +1 (408) 990-7400
Fax: +1 (408) 990-7402
info@rti.com

rti.com
company/rti
rti_software
connextpodcast
rtisoftware
rti_software