A Comparison of Simulation and Operational Architectures

José-Ramón Martínez-Salio
Jose-Maria Lopez-Rodriguez
NADS
Madrid, Spain
jrmartinez@nads.es
jmlopez@nads.es

Dan Gregory
Thales
Palaiseau, France
dan.gregory@thalesgroup.com

Angelo Corsaro
PrismTech
Marcoussis, France
angelo.corsaro@prismtech.com

Keywords:
IEEE HLA, OMG DDS, Real Time Simulation Middleware, Open Architectures, QoS, Interoperability, Network Centric systems, Fault Tolerance

ABSTRACT: *Architectural approaches for enabling communication among operational systems have evolved over the past few years. Their evolution has been independent of recent developments in the architectures applied to simulation systems. For example, the Object Modeling Group's Data Distribution Service (DDS) standard has emerged as a potential alternative to complement or even to overlap with current simulation standards and architectures. While DDS has many apparent similarities with simulation standards, there are also many important differences. As stated in studies like Live Virtual Constructive Architecture Roadmap (LVCAR) sponsored by the US DoD M&S Coordination Office (MSCO), current simulation architectures need to evolve to meet new challenges; for example in interoperability and security. Technology developed by, and lessons learned by, the operational architecture community may be able to complement evolving simulation approaches and vice versa. In this paper, we compare the functionality of operational standards and architectures, such as DDS, with simulation standards and architectures, such as HLA. Particular attention is paid towards fault tolerance, quality of service, security, and data-model inheritance.*

1. Introduction

Architectural approaches for enabling communication among operational systems (operational architectures), and architectures applied to simulation systems (simulation architectures) have evolved independently.

Operational and simulation architectures have some similarities and some differences; a comparison of their functionality may help identify areas where simulation architectures can be improved.

1.1 A Background to Simulation Architectures

In the mid-eighties a US Government funded program known as SIMNET pioneered distributed simulation. Then, in the early nineties, the Institute of Electrical and Electronics Engineers (IEEE) standard, Distributed Interactive Simulation (DIS), was developed closely following the SIMNET protocol. DIS was a NATO standard up to 2010, which demonstrates its success. Today it is still one of the most used standards, mainly in virtual simulation.

In the 1990s a new IEEE standard, the High Level Architecture (HLA) was developed as a successor for DIS and other standards such as ALSP used in constructive simulation. HLA was designed as the first universal standard for live, virtual and constructive (LVC) simulation, providing services for any type of simulator). Nowadays it is the most widely used standard in simulation and is following a five-year evolution cycle (the latest release of the standard, called HLA-evolved, was delivered in 2010).

Although not main stream, but still relevant are the Test and Training Enabling Architecture (TENA) and the Common Training Instrumentation Architecture (CTIA). These are US Government developed middleware and protocol-based solutions that were designed to solve some of HLA's issues especially with respect to the quality of data distribution and network management. TENA and CTIA have improved upon some HLA capabilities but have not become international standards and are mainly based on the use of Government Off-The-Shelf (GOTS) provided by US Government.

The above history is summarized in Figure 1.

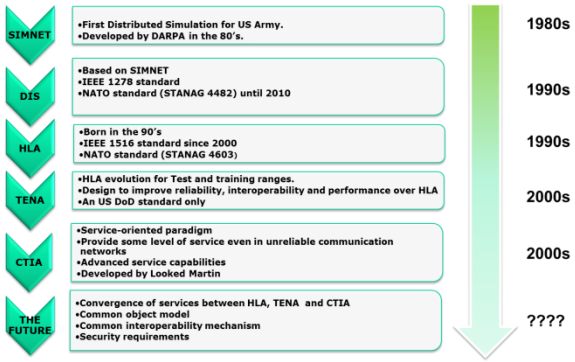

Figure 1. A Brief History of Simulation Architectures

There is an overwhelming use of HLA and DIS in simulation. Together they represent more than the 70% of simulators, as shown in Figure 2. Due to its simplicity DIS is used in new simulation developments, and the standard is still undergoing active development.

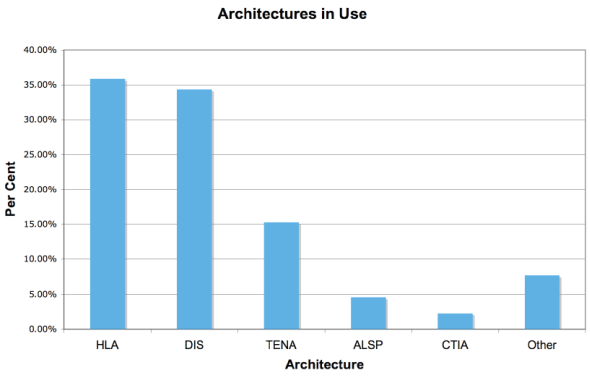

Figure 2. Usage of Simulation Architectures (From [1])

HLA was developed with the idea of a universal standard trying to cover all the possible scenarios in simulation. By introducing HLA services it covered the most common scenarios in simulation known at the time. The HLA approach has been a great success over the last fifteen years.

But with time, new and more challenging simulation goals became mainstream. For example, at first it was sufficient to have one simulator working alone; then the necessity of connecting several together arose. This was followed by demanding pressures of connecting simulators distributed geographically over heterogeneous networks with high fidelity graphics and real-time data flows. Security then became an important consideration, particularly when networks started to extend across national boundaries.

Summarizing the results stated in [1] and other studies such as the "First WAVE" Collaborative Simulation [2], we can consider a range of new challenges for simulation architectures:

Interoperability:

- Over different networks in a system of systems scenario.
- Heterogeneous architectures.
- Among multiple simulation standards or multiple implementations of the same standard.
- Data synchronization of alternative sources.

Performance:

- Maintaining performance while increasing simulation scale.
- Prioritization of data with low latency requirements.

Security:

- Confidentiality.
- Integrity.
- Authenticity.

Full plug & play capabilities:

- Dynamic simulation elements.
- Negligible discovery time.

Fault tolerance and data distribution:

- Avoiding that a single node can compromise the simulation.
- Bottlenecks become critical when scalability increases.

Trying to comply with these emerging needs using current simulation architectures is a hard task.

1.2 A Background to Operational Architectures

Operational architectures have leveraged in the past few years from a very rich middleware ecosystem. In brief we could classify the middleware technologies as:

- Message Oriented Middleware (MoM)
- Distributed Object Middleware (DOM)
- Data Centric / Big-Data Middleware (DCM)

The main differences between the classes of middleware listed above are firstly the abstraction provided, and secondly the interaction patterns supported.

MoMs are built around the abstraction of a *message*, where the message is composed by a header, often some properties and a body – which is typically treated as an un-interpreted sequence of bytes. The Java Message Service and IBM MQ-Series are examples of MoM.

DOMs provide the foundational abstraction of a distributed object or service over which operations can be invoked. Object Management Group (OMG) Common Object Request Broker Architecture (CORBA) and Java Remote Method Invocation (RMI) are examples of DOM.

Finally DCMs provide the foundational abstraction of a more-or-less sophisticated relational data model. Distributed databases and the OMG Data Distribution Service for Real-Time Systems (DDS) are examples of DCM.

Most of modern middleware technologies and standards support multiple *interaction paradigms*, such as, Publish/Subscribe, Request/Reply, Message Queue, etc. However some of these paradigms are more efficiently implemented in some technologies than in others. Finally, most modern middleware technologies provide both synchronous as well as asynchronous and non-blocking application programming interfaces (APIs).

Simulation technologies have in some instances taken advantage of middleware approaches commonly used in operational systems. This is the case for TENA which uses CORBA at its foundation.

OMG's DDS specifies a data-centric publish and subscribe model similar to HLA's. Differently from HLA, DDS also specifies the wire protocol, thus ensuring native interoperability among implementations from different vendors.

OMG's DDS is widely used and deployed on a large class of operational systems, ranging from network centric systems of systems, to supervisory control and data acquisition (SCADA), medical, aerospace, etc. A DDS-based system was recently deployed to connect a 40,000-point SCADA system controlling 30 generators and the transmission switchyard in the Grand Coulee Dam. DDS is also used in the European Union for air traffic control systems, connecting the most critical components of the CoFlight flight data processor (FDP). It is used in naval combat management systems: deployed in eighteen navies on 100+ ships ranging from small frigates to aircraft carriers. It is also used as an ultra-low latency infrastructure in high-frequency auto-trading. Furthermore, DDS is used in some training and simulation systems where high levels of scalability are required.

Finally, DDS is increasingly used in high performance large-scale simulations. Thus making it an increasingly relevant technology for simulation architectures.

Simulation architectures are developed specifically for simulation and are not generally employed outside of their domain. There are cases when operational architectures are applied to simulation, such as for CORBA and DDS.

2. Comparison of HLA and DDS

HLA and DDS were chosen for an in-depth comparison as they are representative and popular architectures in their respective communities, and share a number of similarities in their approach.

[3] described the differences between HLA and DDS in 2006. Since then HLA has experienced a major revision and the DDS standard has significantly evolved.

The above work concluded that DDS is suitable for a large subset of the class of problems targeted by HLA. This paper expands on the comparison and brings it up to date, grouped by each of the major functions of HLA and then by major DDS functions for which there are no equivalents in HLA.

Reference was made to [4] and [5], as well as [6].

2.1 Federation Management

In HLA a federate (simulation application) joins a federation (simulation execution). In DDS a domain participant is the entry point to a domain. There may be many domains on a network, and as for federations, there is no data flow across domains.

Save and restore functions are not present in DDS, although a durability quality of service (QoS) parameter can be set to ensure that data is written on to transient or persistent stores (such as the disk) and available following a major application failure or a restart.

2.2 Declaration Management

Simulation data exchange models (SDEMs) describe the data types to be exchanged among simulation applications. HLA calls these Federation Object Models (FOMs) which are described in Extensible Markup Language (XML) according to the Object Model Template (OMT) specification. FOM modules may be used to extend the classes contained within referenced FOMs.

DDS provides the concept of Topic, defined as the combination of a unique name, a type and a QoS describing its non-functional properties. Topic types can mark a collection of their attributes as the key. Each unique key value then identifies a Topic Instance.

Topic types can be described in either IDL, annotated java classes, XML, XML Schema Definition Language (XSD) and Unified Modeling Language (UML). Topic types can be created or extended at runtime using a structural type system that, in a sense, works like C++ templates. There is no need to declare sub typing, the structural properties of a type are what matters. DDS Topics are defined within a domain and distributed by the DDS middleware implementation and can be discovered dynamically. Consistency of multiple definitions of the same type is ensured by the middleware, which throws an exception if a conflicting definition is created – where conflicting in this case means not structurally compatible. Introspection may be used to access attributes that were not known before runtime.

DDS topic types can be annotated to control the allowed level of extensibility/mutability. Several annotations are also applicable at an attribute level allowing control of optionality, etc..

A data writer can publish data for multiple instances of a given Topic. A data reader subscribes to a Topic and receives samples for all the instances that are currently defined in the system. The topic types do not have to be identical, but must be on a sub-type relationship. This is similar in HLA evolved where one federate can subscribe to a class and receive updates or interactions from subclasses albeit without support for polymorphism.

2.3 Object Management

The model for object and interaction management is rather different between DDS and HLA.

In DDS, data readers and data writers have an associated cache. For data writers this cache contains a subset of the data sent. For data readers the cache contains (a subset of) the data received. The size and retention policy of the cache can be controlled via QoS policies. Samples within the reader cache can be inspected and left on the cache by a "read" operation, or they can be consumed via a "take" operation.

In HLA, there is a distinction between objects, which are created or deleted, and have attributes which may be updated; and interactions, which are singleton messages containing parameters.

DDS's equivalent of HLA object handling is to use keys to uniquely identify instances. These instances are effectively put into separate FIFO buffers within a data reader's cache. The default cache policy in DDS is to have a buffer length of one (per instance) and that it should contain the latest received update for each instance. If a "read" operation is performed, the instance remains in the cache and is available to be consulted at a later time.

It is possible to extend the cache length if historical information is needed. Unlike in HLA, incremental updates are not handled by DDS. This has to be implemented at a user level, using optional attributes.

The DDS equivalent of an HLA interaction would be to set the cache depth to keep all, and to "take" objects from the cache.

Additionally in DDS, queries can be made on the contents of the cache. These are specified as a Structured Query Language (SQL) WHERE clause, and return an array of matching instances.

A number of lifecycle notifications can be set up in DDS including the creation, disposal and update of an instance. Instances can be identified as having been read, or not; being alive, which implies that writers exist; not alive, indicating that writers do not exist; or disposed, which means that the writer has explicitly disposed of the instance. These are useful for handling data life-cycle and faults.

Encodings are handled by DDS and endianness is automatically handled by the data reader. Thus no conversions are made if both reader and writer systems have the same endianness, therefore saving computing

cycles. HLA evolved provides encoding helpers to perform encoding operations.

2.4 Ownership Management

As stated by [3], negotiated ownership handover is not facilitated by DDS. Ownership services, in operational architectures, are more oriented towards fault tolerance. QoS policies are used to specify ownership strength. However content based filtering can be combined with ownership strength to trigger ownership handover when predefined conditions are met.

2.5 Time Management

Time management is a function that is mainly oriented to simulation applications. Operational systems generally use time management to handle system clock updates. However the management of time is key to simulation applications. HLA includes a complex mechanism for handling discrete event time advancement. This is useful for constructive simulation, where federates can advance up to the point at which a state change is anticipated.

DDS adds time stamps to each object. These can be set automatically, or specified explicitly through the API. This function could enable the implementation of a discrete event time model; however its orchestration would have to be implemented by the user, or by a service built over DDS.

Additionally, in DDS, objects can be ordered in the cache according to the source timestamp or the destination timestamp. This is equivalent to HLA's time stamp ordered or receive ordered policies. Destination order is the default in DDS as it assumes that member applications' clocks are not synchronized.

2.6 Data Distribution Management

HLA has the concept of dimensions, ranges, region specifications, region templates, region realizations, attribute designators, and region designators. These enable data to be published only when there is a match between the region specified by a subscriber and that specified by a publisher. A description of the precise mechanism is beyond the scope of this work.

DDS uses the concept of partitions. Partitions, identified by strings, can be used to organize data within a domain. Publishers and subscribers can join partitions using Portable Operating System Interface (POSIX) compliant regular expressions. The subscriber will only receive data from publishers that have a matching partition string. The subscriber's string can be a regular expression, facilitating subscription to multiple partitions. This is a simpler approach to HLA's, however it is less flexible, although more dynamic, as partitions can be specified at runtime.

Additionally DDS can filter objects based on their content, by specifying a SQL WHERE clause. Filtering differs from querying, as described above, as it prevents objects from being needlessly transmitted across the network.

2.7 Quality of Service

DDS provides a richer set of QoS policies than HLA. These policies, of which there are 22 in total, can be set through the API, or can be fetched from a QoS provider in an XML format.

When publishers and subscribers are matched, their respective QoS parameters are taken into consideration. For instance a best effort publisher will not be matched with a reliable subscriber; however a reliable publisher will be matched with a best effort subscriber.

The following categories of QoS policies are available in DDS:

- temporal and importance characteristics;
- data availability;
- data delivery;
- fault detection;
- replication.

Temporal and importance characteristics are often of interest to the simulation community, where low latencies are needed between closely coupled systems. A latency budget can be specified to enable the middleware to optimize factors such as batch publishing in order to meet the budget. A deadline can be set to indicate the minimum publishing rate and the maximum period over which a subscriber is prepared to wait. A time-based filter can be used to specify the requested maximum frequency of objects a subscriber wishes to handle. This is similar to HLA's smart update rate. Transport priority may also be set to enable the middleware to prioritize the objects sent with the highest priority value. These policies can help balance latency with throughput and enable fine control and supervision of characteristics that may be important to a simulation.

Data availability policies cover durability, lifespan, history and ownership. Durability can be set to volatile, meaning that objects are sent to running subscribers; transient local, in which the publisher resends objects to late joining subscribers; transient, in which other peers transmit objects to late joining subscribers even if the

publisher is no longer running; and durable, in which all peers can be restarted and the object will still be sent to a late joining subscriber. Lifespan is the maximum duration for which an object is valid.

Data delivery policies define partitions, and destination ordering, as described above, and additionally, presentation, which enables coherent sets of objects to be grouped and presented together, or for objects to be presented as they arrive. Reliability can be set to either reliable or best effort, as for HLA. However DDS also allows a duration to be specified that determines for how long it will keep unacknowledged objects for retransmission.

Fault detection covers the liveliness of instances, and provides a means for fail-over applications to take over if primary applications do not assert their liveliness.

The replication parameters control ownership as discussed above.

2.8 Security

Interoperable security is in the process of being added to the DDS specification [7]. This will enable either topics or individual attributes to be encrypted. DDS will provide built-in key management and encryption algorithms, which can be replaced with third party components as required.

2.9 Link Compatibility

Dynamic link compatibility is available in HLA. DDS only supports this feature for its Java API, as the C++ language bindings depend on templates.

2.10 Discovery Time

DDS provides a discovery mechanism to enable applications to identify other DDS applications on the network. HLA has no such mechanism.

2.11 Web Services

HLA provides a Simple Object Access Protocol (SOAP) interface for web services. DDS is in the process of standardizing web-enabled DDS [8], which may provide access to DDS through web services including Representational State Transfer (REST), and SOAP, Resource Description Framework Site Summary (RSS), "Atom", and Extensible Messaging and Presence Protocol (XMPP).

2.12 Wire Protocol

In 2006 an interoperability protocol called DDS Interoperability (DDSI) was added [9]. DDSI has been implemented by all major DDS middleware vendors since its specification was published.

2.13 Wide Area Network Support

OMG is standardizing features to support the connectivity of applications separated by wide area networks (WANs). These will include the ability to specify the use of Transmission Control Protocol (TCP) among applications separated by networks that do not support User Datagram Protocol (UDP). Similarly alternative protocols will be used over networks that do not support UDP multicast. Also firewall and network address translation (NAT) gateway traversal will be addressed, probably using the standardized set of methods known as Session Traversal Utilities for NAT (STUN).

2.14 Unreliable Networking Support

Support is provided in DDS for communication over unreliable networks. Depending upon the reliability and liveliness QoS parameters, a publisher will wait for a connection to the subscriber to become available. Disruptions in network connectivity of a short duration may be transparent to the application.

2.15 Scalability and Performance

The HLA and DDS specifications do not specify scalability or performance constraints. They both have functionality aimed at optimizing scalability and performance. In good quality, full dedicated networks both can be considered to be equivalent in performance. HLA uses TCP for reliable communication and UDP for "best effort" and DDS, using UDP for both, can at least match the results. On the other hand DDS has an advantage over heterogeneous networks and in disconnected, intermittent, limited networks (DILs). Scalability in DDS is better than for HLA implementations that use a client-server model. A distributed architecture and discovery functionality allows scalability improvements.

3. Conclusions

There are differences and similarities between operational and simulation architectures. Operational architectures are focused on the distribution of data; simulation architectures add functionality that is specific to simulation such as time management, yet also have data distribution needs. The functionality of

operational data distribution middleware such as DDS exceeds the data-distribution functionality of simulation architectures such as HLA, as shown in Figure 3.

There are functions present in operational architectures that are not supported by current simulation architectures. Some of these, such as temporal QoS, prioritization and support for unreliable networks may offer benefits to the simulation community.

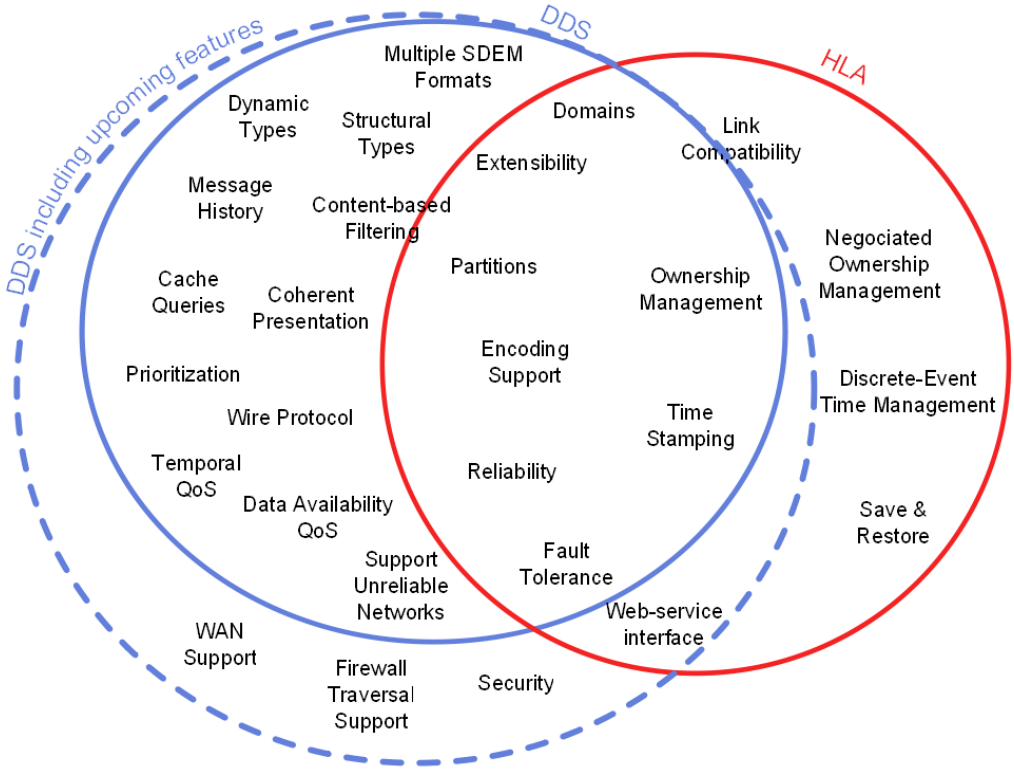

Figure 3. Functionality Supported by DDS and HLA.

4. References

1. **Henninger, Amy and et.al.** *LVCAR Final Report.* s.l. : Institute for Defence Analyses, 2008.

2**NATO Science and Technology Organization**Mission Training via Distributed Simulation and First WAVE: Final Reporthttp://www.rto.nato.int/pubs/rdp.asp?RDP=RTO-TR-SAS-034RTO-TR-SAS-034

3. **Joshi, Rajive and Castellote, Gerardo-Pardo.** A Comparison and Mapping of Data Distribution Service and High-Level Architecture. *RTI.* [Online] 2006. http://www.google.com/url?sa=t&rct=j&q=%22A+Comparison+and+Mapping+of+Data+Distribution+Service+and+High-Level+Architecture%22&source=web&cd=1&ved=0CE4QFjAA&url=http%3A%2F%2Fwww.rti.com%2Fwhitepapers%2FComparison_and_Mapping_of_DDS_and_HLA.pdf&ei=pMbhT_bA.

4**IEEE Computer Society**IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Federate Interface SpecificationIEEE Computer SocietyIEEE 1516.1-2010

5. **Object Management Group.** Data Distribution Service for Real-time Systems. [Online] 1.2, 2007. http://www.omg.org/spec/DDS/1.2/.

6. *HLA Evolved – A Summary of Major Technical Improvements.* **Moller, Bjorn, et al.** s.l. : Simulation Interoperability Standards Organization, 2008. Simulation Interoperability Workshop Fall 2008. 08F-SIW-064.

7. **Object Management Group.** DDS Security RFP. *OMG.* [Online] 2010. http://portals.omg.org/dds/sites/default/files/dds_security_rfp_mars_10-12-37.pdf.

8. —. Request for Proposal Web-Enabled DDS. [Online] 2009. http://portals.omg.org/dds/sites/default/files/webenabled_dds_rfp_mars-09-09-19.pdf.

9. —. DDS Interoperability Protocol (DDS-RTPS) v2.1. [Online] January 2009. http://www.omg.org/spec/DDS-RTPS/2.1/.

10. **Corsaro, Angelo.** The use of DDS for Air-Traffic Management: The COFLIGHT project. *OMG.* [Online] 2007. http://www.omgwiki.org/dds/content/document/use-dds-air-traffic-management-coflight-project.

Author Biographies

JOSÉ-RAMÓN MARTÍNEZ-SALIO has an MSc in Industrial Engineering (specialized in robotics and electronics) by the Polytechnic University of Madrid, Spain. He is Technical Presales engineer director of NADS. Until 2005 he was focused in the development area of SimWare product and was responsible of management different projects in the simulation area. He has 15 years of experience in Simulation projects, DDS and high technology projects, as principal engineer and project manager.

JOSE-MARIA LOPEZ-RODRIGUEZ has a MSc in Industrial Engineering (specialized in robotics and electronics) by the Polytechnic University of Madrid, Spain. He is Vice-president for Business Development and cofounder of NADS (Nextel Aerospace, Defence & Security). Until 2006 he was managing the development of NADS's R&D projects, mainly related to the development of simulation middlewares and framework compliant with HLA and DDS. He has about 15 years of experience in Simulation, C2 and Virtual Reality projects, as principal engineer, project manager and business development manager. He has more than 10 papers in international congresses, mainly related to Simulation and Virtual Reality issues.

DAN GREGORY is a Thales Expert in modelling & simulation. He works for Thales's technical directorate, chairing the Modelling & Simulation Network of Excellence that coordinates M&S technologies and capabilities across the group. Dan also chairs the Network Centric Operations Industry Consortium's Modeling & Simulation team. Dan has extensive experience of using simulation for operational analysis having worked for the UK Ministry of Defence and QinetiQ before joining Thales.

ANGELO CORSARO is OpenSplice DDS Chief Technology Officer (CTO) at PrismTech. As CTO, Angelo directs the technology strategy, planning, evolution, and evangelism. Angelo leads the strategic standardization at the Object Management Group (OMG), where he co-chairs the Data Distribution Service (DDS) Special Interest Group. Angelo is a widely known and cited expert in the field of real-time and distributed systems, middleware, and software patterns, has authored several international standards and enjoys over 10+ years of experience in technology management and design of high performance mission- and business critical distributed systems.