2012 NDIA GROUND VEHICLE SYSTEMS ENGINEERING AND TECHNOLOGY SYMPOSIUM
VEHICLE ELECTRONICS AND ARCHITECTURE (VEA) MINI-SYMPOSIUM
AUGUST 14-16, MICHIGAN

OPEN MANAGEMENT GROUP DATA DISTRIBUTION SERVICE (OMG-DDS) AS A DATA TRANSPORT FOR VEHICULAR INTEGRATION FOR C4ISR/EW INTEROPERABILITY (VICTORY) SERVICES

Leonard Elliott
Vehicle Electronics and Architecture
TARDEC
Warren, MI

Nikia Williams
Vehicle Electronics and Architecture
TARDEC
Warren, MI

Venu Siddapureddy
Vehicle Electronics and Architecture
TARDEC
Warren, MI

ABSTRACT

A key objective of the Vehicular Integration for C4ISR/EW Interoperability (VICTORY) Architecture is to use open standards to increase the portability of C4ISR/EW systems and enhance interoperability within military ground vehicles. When possible these technologies are adopted by VICTORY and when existing specifications are inadequate, best-practices are used to develop the necessary adaptations. Many Commercial Off-The-Shelf (COTS) publish/subscribe messaging solutions are available and the Open Management Group (OMG) Data-Distribution Service (DDS) is one such technology that provides open interfaces, open data formats, and open protocols. This paper will discuss the current VICTORY messaging approach and the benefits and disadvantages of using OMG-DDS as a data transport for VICTORY services.

INTRODUCTION

The Vehicular Integration for C4ISR/EW Interoperability (VICTORY) project is an initiative by the U.S. Army to improve upon current military ground vehicle electronics architecture. One of the key objectives for VICTORY is to maximize C4ISR/EW portability by identifying open standards to define 1) open interfaces, 2) open data formats, and 3) open protocols [1]. The VICTORY standards are developed by a government-industry standards body using an adopt-adapt-author methodology [2]. The currently defined scope of VICTORY does not support real-time or safety-critical applications.

The VICTORY technical approach provides a data-bus centric design, shareable hardware components, and shared software services [2]. The data-bus centric design is provided via shared software services (VICTORY services), and the VICTORY Data-bus (VDB), which has Ethernet as a core component. It is important to emphasize that VICTORY is specified for intra-vehicle communication where high network availability and control of network topology by vehicle designers may be assumed.

VICTORY services have clearly defined management and data interfaces. The management interfaces allow VICTORY services to be monitored and controlled over the VDB via the Simple Object Access Protocol (SOAP) and remote procedure calls (RPC). Many of the data interfaces supply data to consumers using customized VICTORY Data Messages (VDM) and a simple form of publish-subscribe (pub/sub) messaging that leverages UDP multicast. Many commercial off-the-shelf (COTS) pub/sub messaging technologies are readily available for use in architectures such as VICTORY, but no apparent evaluation of these technologies has been conducted by the VICTORY workgroup. The choice of using customized message format (i.e. VDM) and messaging behavior for VICTORY service data interfaces seems to be an inconsistency in the VICTORY adopt-adapt-author methodology.

In this paper, we will briefly discuss pub/sub messaging and several commonly used COTS messaging technologies. We will then discuss the VICTORY approach to pub/sub data dissemination on the VDB. Finally we will discuss an experiment which involved modifying a reference implementation of the VICTORY core services developed by the U.S. Army Tank and Automotive Research Development and Engineering Center (TARDEC). The experiment involved modifying the reference implementation of the VICTORY 1.0 Position Service by replacing the VDM data-interface with an OMG-DDS interface and capturing notes regarding the effort required and OMG-DDS interoperability. We will then compare the VDM and OMG-DDS approaches.

PUBLISH-SUBSCRIBE OVERVIEW

Pub/sub communication is a messaging pattern that has the benefit of providing loose-coupling and scalability by allowing senders to push messages to receivers without explicitly having to maintain information about receivers or their connection states. Three COTS messaging systems that support this integration pattern are described below.

Java Message Service (JMS)

JMS is a Java application programming interface (API) specification that debuted in 2001 and is typically implemented in a centralized or brokered architecture. JMS provides messages, channels, queues, and other constructs for supporting many loosely-coupled integration patterns. JMS is typically implemented using a centralized or brokered architecture. While JMS provides a standard Java API, there is no standard wire protocol: therefore different JMS messaging systems do not interoperate [3].

Advanced Message Queuing Protocol (AMQP)

AMQP is an application layer protocol which debuted in 2003 and is now an open standard governed by the Organization for the Advancement of Structured Information Standards (OASIS). AMQP is typically implemented using a multi-brokered architecture and provides an on-the-wire format for supporting messaging systems. AMQP does not define a standard API [4].

Open Management Group Data-Distribution Service (OMG-DDS)

OMG-DDS is an open standard that debuted in 2003 and supports real-time pub/sub communication. OMG-DDS is generally implemented in a peer-to-peer architecture (i.e. middleware functions are spawned within the user-application). OMG-DDS consists of two standards managed under OMG, the Data Centric Publish-Subscribe (DCPS) API standard and the Real-Time Publish Subscribe (RTPS) protocol standard.

OMG-DDS will be highlighted because it:

- Is an open standard that is supported by multiple vendors and programming languages.
- Provides mechanisms for supporting real-time and high-performance communication.
- Is supported by both a standard API (DCPS) and standard wire protocol (RTPS) allowing designers to switch vendors with minimal code changes and interoperate with systems developed using different OMG-DDS implementations.
- Is usually implemented using a peer-to-peer architecture, meaning that there are no brokers to act as bottle necks or single points of failure.

The focus of VICTORY is interoperability which is emphasized throughout this paper. OMG-DDS vendors often provide useful and performance-enhancing extensions to the OMG-DDS standard and some of these are later integrated into the standard. An evaluation using only standard OMG-DDS features was desired so special care was taken to avoid these specializations.

VICTORY DATA MESSAGING

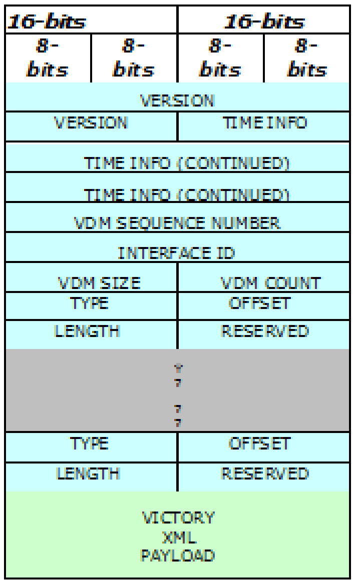
VICTORY specifies a customized messaging solution that allows VICTORY services to support pub/sub communication through their data interfaces. To disseminate VDMs, VICTORY services utilize the standard Internet Protocol version 4 (IPv4) User Datagram Protocol (UDP) multicast and the IPv4 Internet Group Management Protocol (IGMP) to manage subscribers. Information about publishers is available to subscribers dynamically using Zero Configuration Networking and SOAP RPCs to the service's management interface. Alternatively, information can be made available to subscribers a priori via a VICTORY Configuration Language Instance Document (VCLID) which provides a static description of the system. Using the VDM approach, services may access the management interfaces via SOAP RPC to:

- Increase/decrease the VDM publishing frequency.
- Turn on/off VDM publishing.

- Change the multicast address for publishing.

VICTORY also supports the basic notion of Quality of Service (QoS) by specifying that network components support IPv4 Differentiated Service Code Point (DSCP), a 6-bit field in the IP header that indicates IP packet traffic priority.

VDMs consist of a binary formatted header with XML payloads. Figure 1 illustrates the layout of the VDM format with binary encoded information in blue and XML formatted data in light green (see [2] for a more detailed description).



16-bits		16-bits	
8-bits	8-bits	8-bits	8-bits
VERSION			
VERSION		TIME INFO	
TIME INFO (CONTINUED)			
TIME INFO (CONTINUED)			
VDM SEQUENCE NUMBER			
INTERFACE ID			
VDM SIZE		VDM COUNT	
TYPE		OFFSET	
LENGTH		RESERVED	
⋮			
TYPE		OFFSET	
LENGTH		RESERVED	
VICTORY XML PAYLOAD			

Figure 1: VICTORY Data Message Layout

The VDM structure supports many of the standard messaging concepts including sequence numbers, format indicators, identifiers, and timing information [3]. A notable absence from the standard list of message attributes includes the message expiration indicator, but given the scope of VICTORY, and that a high-availability network on the platform is assumed, the expiration mechanism may be unnecessary. Alternatively the IPv4 mechanism known as time to live (TTL), which governs how long datagrams remain in the network, may serve as a rudimentary message expiration date (although receivers will be unaware that the expired message was ever sent).

Implementations of the VICTORY core services on Red Hat Enterprise Linux (RHEL) have been developed at TARDEC and have undergone thorough testing. These services have been successfully ported to x86_64, i386, and ARM Cortex-A8 architectures. All tests conducted so far suggest that the VDM approach provides simple messaging that can easily be implemented using standard POSIX libraries (i.e. sys/sockets.h) on a variety of operating systems (OS) and processor architectures.

INSERTION OF OMG-DDS INTO VICTORY SERVICES

As mentioned in the Publish-Subscribe Overview section, OMG-DDS appears to be a promising candidate for meeting the integration requirements of those applications within the military ground vehicle that require more complex pub/sub communication (including those which fall into the real-time scope). OMG-DDS incorporates many features including an extensive QoS, service discovery, and automatic negotiation of QoS.

One of the central concepts in OMG-DDS is the QoS policy which provides a rich set of features that can be changed within the application and sometimes without recompiling (via vendor-specific extensions in XML configurations). There are many parameters in the standard QoS that are available for tuning publisher and subscriber behavior. A few key QoS settings include:

- Deadline: The maximum time between data samples.
- Durability: Previously published data can be stored and sent to late joining subscribers.
- Lifespan: Specifies how long data sent by user application is considered valid.
- Liveliness: Allows readers to detect when matching writers are no longer available.
- Ownership: Specifies ownership of a Topic by a specific Data Writer.
- Reliability: Allows samples lost by the network to be recovered by the middleware.
- Time Base Filter: Allows readers to specify a minimum separation time for received samples.

In the following section we provide an overview of the software architecture of the VICTORY core services developed by the U.S. Army at TARDEC. We will then discuss the steps taken to convert the VDM interfaces used by the VICTORY 1.0 Position Service and the steps taken to replace this service's VDM data-interface with several vendor implementations of OMG-DDS. The versions of OMG-DDS used were Real-Time Innovations Inc. (RTI) Connext DDS 4.5f and PrismTech's OpenSplice 6.1.1.

VICTORY 1.0 Core Service Software Architecture

The reference implementation of VICTORY 1.0 core services developed at TARDEC includes the Position, Orientation, and Direction-of-Travel services. As their names indicate, these services connect to sensors using various physical connections, translate the proprietary/controlled data formats to VICTORY formatting, and publish the corresponding data on the VDB. The VICTORY services developed at TARDEC use standard POSIX libraries, C++, and Genivia gSOAP, a tool that provides C/C++ bindings and support for SOAP, XML Services Definition (XSD) and Web Service Definition Language (WSDL). These services have three essential processes (each implemented in a POSIX thread):

1. Connect to a sensor, interpret data, and update data structures.
2. Read data structures, build and transmit VDMs.
3. Listen, accept, and serve SOAP management calls and read or update data structures accordingly.

Figure 2 shows a high-level diagram of the VICTORY Position Service and its interaction with the VDB. The service connects to a Defense Advanced GPS Receiver (DAGR) using RS-232 and publishes a VDM with the position message. An example mapping application is shown subscribing to the Position VDM, but in our experiment the VDM subscriber simply reads, parses, and displays the received data.

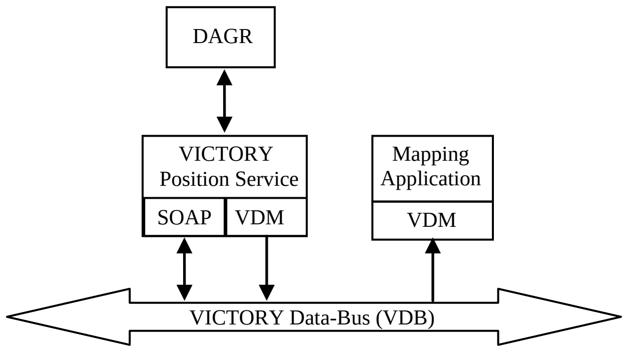


Figure 2: VICTORY Service Data Flow

Porting VDM Interfaces to OMG-DDS Interfaces

VICTORY defines all application interfaces, messages, and data types inside of the WSDL and XSD files and these files are a key product of each VICTORY version. OMG-DDS utilizes another OMG standard known as Interface Definition Language (IDL) to achieve a programming language and platform independent data description. Once these IDLs are defined, vendor-supplied tools are used to generate language specific bindings (in our case C++). Figure 3 shows a generalized excerpt from the VICTORY 1.0 schema that describes *absolutePosition_t* versus the corresponding IDL type *AbsPos_t* that we defined as part of our experiment (shown in Figure 4).

```
<xsd:simpleType name="latitudeBounds_t">
   <xsd:restriction base="xsd:double">
      <xsd:minInclusive value="-90"/>
      <xsd:maxInclusive value="90"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="longitudeBounds_t">
   <xsd:restriction base="xsd:double">
      <xsd:maxInclusive value="180"/>
      <xsd:minExclusive value="-180"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="absolutePosition_t">
   <xsd:sequence>
      <xsd:element name="latitude"
      type="vmt:latitudeBounds_t"/>
      <xsd:element name="longitude"
      type="vmt:longitudeBounds_t"/>
      <xsd:element name="altitude"
      type="xsd:double"/>
      <xsd:element name="mgrs" type="xsd:string"
      minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

Figure 3: Generalized VICTORY Absolute Position Type (XSD)

```
#ifndef VICTORYTYPESDDS_IDL
#define VICTORYTYPESDDS_IDL
module VICTORY
{
   module Types
   {
      struct doubleMeasurement_t
      {
          double value;
          double uncertainty;
          boolean estimated;
          boolean valid;
      }; //@top-level false

      //
      // Military Grid Reference System
      //
      struct MgrsPos_t
      {
         string<10> gridZoneDesignator;
         long easting;
         long northing;
      }; //@top-level false

      struct AbsPos_t
      {
         doubleMeasurement_t latitude;
         doubleMeasurement_t longitude;
         doubleMeasurement_t altitude;
         boolean hasMgrs;
         MgrsPos_t mgrs;
      };#pragma keylist AbsPos_t
  }; //Types
}; //VICTORY
#endif
```

Figure 4: Generalized VICTORY Absolute Position Type (IDL)

The mapping between XML and IDL constructs is not one-to-one so some modifications were made in order to port the functionality of the XSD type to IDL. In particular VICTORY type definitions make extensive use of XSD optional elements and this concept does not exist within IDL. This functionality can be implemented in IDL using presence indicator flags. Some OMG-DDS vendors provide tools for converting XSDs to IDL, but at the time of this experiment, those tools did not support all elements used within the VICTORY XSDs (i.e. auto-generation of IDLs failed).

Lines of code in Figure 4 that contain the # and //@ tokens declare vendor-specific IDL pre-processor directives. Some of the data definition concepts described in the DCPS API appear to be missing from the IDL standard and have vendor-specific support through the use of IDL pre-processor directives. For example, OMG-DDS allows user-defined types to be keyed on data members. This means that different data values with the same key value represent successive values for the same instance, while different data values with different key values represent different instances [4]. For example, to add a keyed index field *id* to A*bsPos_t*, for RTI Connext DDS the directive *//@key* must be placed next to the declaration of *id*, and for PrismTech OpenSplice the directive *#pragma keylist AbsPos_t id* must be placed after the declaration of *AbsPos_t*. The difficulty these issues posed during our experiment was insignificant.

Replacing the VDM Interface with OMG-DDS Publisher

After IDLs were created and the C++ bindings were generated, the VDM interfaces in the VICTORY Position Service and subscriber application were replaced with DDS publishers and subscribers. The steps taken to replace the VDM publisher with a DDS publisher were:

1. Declare and initialize DDS Domain Participant, Topic, QoS, Publisher, and Data Writer.
2. Map the VICTORY Position Service internal C++ data structures using the assignment operator and the bindings created from the IDL files.
3. Call the DDS Data Writer *write()* method to publish the data on the VDB.

The steps for replacing the VDM subscriber with a DDS subscriber were:

1. Declare and initialize DDS Domain Participant, Topic, QoS, Subscriber, and Data Reader.
2. Map the VICTORY Position Service DDS data to internal C++ data structures.
3. Call the DDS Data Reader *take()* method to retrieve the received samples.

Figure 5 shows the resulting data flow for the modified VICTORY services. OMG-DDS was implemented in our VICTORY Position Service using ~100 lines of C++ code.

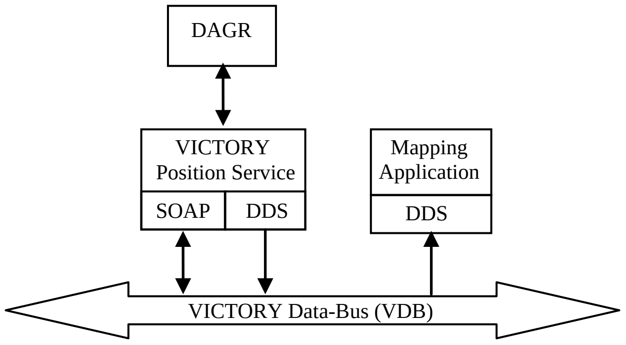


Figure 5: Modified VICTORY Service Data Flow

This preliminary experiment used OMG-DDS to replace the basic VDM behavior so default QoS was sufficient. The QoS settings and other attributes used during this experiment are outlined in Table 1.

QoS & Attributes	**Writer**	**Reader**
Deadline	Infinite	Infinite
Domain ID	0	0
Durability	Volatile	Volatile
Latency Budget	0 sec	0 sec
Liveliness duration	Infinite	Infinite
Liveliness kind	Automatic	Automatic
Ownership	Shared	Shared
Reliability	Reliable	Best Effort
Topic	Position	Position
Type	AbsPos_t	AbsPos_t

Table 1: Effective QoS and Other Attributes

OMG-DDS DCPS API Standard

A goal of this experiment was to determine how many code changes were required to support both RTI and PrismTech OMG-DDS implementations. Both implementations were integrated into the CMake build tool used for building TARDEC's VICTORY reference implementation. Vendor specializations related to the DCPS API only occurred in two places during this experiment and these specializations related to how type-support was implemented and the way that general purpose DataWriters were cast to specialized DataWriters using the *narrow()* method. Figure 6 and 7 show excerpt code listings that illustrate the minor changes that were incorporated to support both RTI and PrismTech OMG-DDS implementations.

```
#ifdef RTI
ddsTypeName =
   VICTORY::Types::AbsPos_tTypeSupport::
   get_type_name();
ddsRetCode =
   VICTORY::Types::AbsPos_tTypeSupport::
   register_type(
   ddsDomParticipant, ddsTypeName );
#endif

#ifdef PRISMTECH
VICTORY::Types::AbsPos_tTypeSupport absPosTs;
ddsTypeName = absPosTs->get_type_name();
ddsRetCode = absPosTs->register_type(
   ddsDomParticipant, ddsTypeName );
#endif
```

Figure 6: Vendor Specific Syntax for Type-Support Constructs (C++)

```
#ifdef RTI
ddsSpecificWriter =
   VICTORY::Types::AbsPos_tDataWriter::narrow(
   ddsDataWriter );
#endif

#ifdef PRISMTECH
ddsSpecificWriter = VICTORY::Types::
   AbsPos_tDataWriter_narrow( ddsDataWriter );
#endif
```

Figure 7: Vendor Specific Syntax for DataWriter Casting (C++)

OMG-DDS RTPS Interoperability Standard

Another objective for this experiment was to determine how much effort was required to achieve interoperability between different vendor implementations of OMG-DDS. Periodically, OMG-DDS vendors conduct interoperability demonstrations and the reports indicate that getting different OMG-DDS implementations to work together is possible, they do not indicate how much work was required to achieve interoperability.

We tested every combination of RTI and PrismTech publishers and subscribers for the modified VICTORY Position Service. This turned out to be a simple task because different vendor publishers and subscribers residing on separate hosts communicated directly out-of-the-box. For publishers and subscribers communicating on the same host we encountered only one difficulty. RTI provides a shared memory transport option, which provides high-performance communication between RTI publishers and subscribers on the same computer (shared memory is one of the fastest forms of inter-process communication). This is a default setting and needed to be disabled before PrismTech and RTI publishers and subscribers could communicate on the same host. The following line of C++ code was added to disable the shared-memory transport in both the publisher and subscriber.

```
#ifdef RTI
ddsDomPartQos.transport_builtin.mask =
      DDS_TRANSPORTBUILTIN_UDPv4;
#endif
```

VDM AND OMG-DDS APPROACHES COMPARED

The previous sections examined both VDM and OMG-DDS and both approaches appear to be well-suited for satisfying the requirements of integration problems that are likely to fall under the scope of VICTORY as it is currently defined. Table 2 summarizes a number of considerations for comparing the two approaches.

	VDM	**OMG-DDS**
COTS Available	No	Yes
Lines of Code	1500	100
Licensing Issues	No	Yes
Open Standard	Yes	Yes
Standard API	No	Yes
Standard Wire Protocol	Yes	Yes
Unbrokered Architecture	Yes	Yes

Table 2: Aspects of VDM and OMG-DDS Compared

The VDM approach is relatively simple and developers will likely be familiar with all of the tools and concepts needed to implement it. Vehicle programs should not be hampered by licensing issues during the acquisition process because the tools required for this approach are standard for most platforms and programming languages, and this could be made even easier if the VICTORY community develops a set of libraries. The VDM approach is limited by the behavioral model that it was designed to support, and while it is simple, the software development effort required to support it is not insignificant.

OMG-DDS is a powerful and versatile tool that is capable of supporting more complex pub/sub behavior. OMG-DDS is straightforward to use in a simple case but when more advanced features are used it can become very complicated (e.g. user manuals and API references are ~1000 pages). OMG-DDS fundamentals can be learned in several days, and then it is very easy to implement VDM-like functionality. The number of lines of code an application developer needs to write is approximately 1/15 of that required for a VDM implementation (using C++). The licensing of OMG-DDS could pose a problem for vehicle programs because pricing varies based on tools, developer seats, OS, CPU architecture, technical support, etc. The standard API and wire protocols should provide integrators with a large selection of implementations and should help them avoid vendor lock-in.

SUMMARY

The VDM and OMG-DDS messaging approaches both appear to be suitable approaches for satisfying the integration requirements which are likely to occur within the current scope of the VICTORY architecture. It is important to note that that neither approach eliminates the multitude of difficult design decisions that integrators will still need to address. While the VDM approach may not be robust or flexible enough to satisfy some of the more demanding messaging behavior required of control and/or real-time applications, it is still an acceptable choice given the scope of VICTORY. The simplicity and lack of licensing issues surrounding the VDM approach may allow vehicle programs to deliver VICTORY capability in a relatively short time frame. The VDB provides a high-performance infrastructure, capable of hosting more demanding distributed applications and once the basic capability is realized, the VICTORY community should consider leveraging the power of OMG-DDS as an incremental capability or as part of a VICTORY real-time extension.

REFERENCES

[1] VICTORY Standards Support Office (VSSO), VICTORY Architecture - Version A, VSSO, 2011, http://www.victory-standards.org/

[2] VSSO, VICTORY Specifications – Version 1.0, VSSO, 2011. http://www.victory-standards.org/

[3] Hohpe, G. and Woolf, B., Enterprise Integration Patterns, Addison-Wesley, 2003.

[4] Vinoski, S. *Advanced Message Queuing Protocol*, IEEE Internet Computing, November 2006

[5] OMG, Data Distribution Service for Real-Time Systems Version 1.2, The Open Management Group, 2007, http://www.omg.org/spec/DDS/1.2/PDF/