AUV Workbench: Integrated 3D for Interoperable Mission Rehearsal, Reality and Replay

Jeffrey Weekley, Don Brutzman, Anthony Healey, Duane Davis and Daryl Lee
Naval Postgraduate School, Monterey California

jdweekley@nps.navy.mil brutzman@nps.navy.mil healey@nps.navy.mil

Overview

Autonomous Underwater Vehicles (AUVs) are in operation and under study around the world. As techniques are being developed for practical use, different vehicles and tools are proliferating. Each vehicle requires dedicated tools, broad knowledge and troubleshooting skills to plan and analyze missions. Executing these missions is often fraught with operational challenges as well. Multiple systems or multi-national operations further complicate these maneuvers. Current tools, data archives and equipment do not interoperate and integration of competing systems is difficult already.

Practical strategies are now possible for achieving both control and data interoperability in multi-vehicular and multi-national operations. The use of Extensible Markup Language (XML) and the open architecture described in the Extensible Modeling & Simulation Framework (XMSF) facilitates interoperability and integration into real-world Joint Command & Control, Communications, Computers and Intelligence (C^4I) Systems, models and simulations, and mission planning and analysis tools.

Customized XML languages and related technologies may be used to facilitate interoperability between dissimilar AUVs to extract and integrate mission data into C^4I systems. XML makes data archive maintenance easier, as XML documents can be accessed via an http server, and XML is directly transformable by extensible stylesheets into formats suitable for humans and machines. An example XML-based mission language for the command and control of AUVs is presented. It discusses XML technology and how XML is a viable means of achieving interoperability. An example of a mission profile using existing software is further presented. Integrating XML for AUV mission planning, rehearsal, experimentation and visualization provides compelling and reusable examples.

In particular, example applications for multi-vehicular command and control, mission planning and analysis, preliminary and post mission visualization using web-based Extensible 3D (X3D) Graphics is presented. X3D is the International Standards Organization (ISO) draft specification for 3D interactive graphics for the Web. It greatly extends the power of 3D tools already available via the Virtual Reality Modeling Language (VRML), because it is XML. These tools are free to use, inherently interoperable and scaleable world-wide. Preliminary investigations regarding the binary encoding of X3D and XML are also presented for streaming over noisy bandwidth-constrained networks, such as acoustic modem channels.

XML for Loosely Coupled Data and Interoperability

The key to interoperability is structured data interchange. Currently, each AUV system operates on its own, with a proprietary data format for collected data and no command task language for mission planning sharable among robots. The application of structured data methodologies using the Extensible Markup Language (XML) allows organizations and systems to exchange and process battlespace information cooperatively. [Neushul 2003] Since XML data is self-describing and can be transformed using XML-based tools to fit any particular format, which can be machine or human readable, it is said to be loosely coupled. Data that is tightly coupled with its application (such as the data contained in this document) cannot be read or modified by other programs without great effort. This tight coupling then requires that many applications maintain relationships with many others, a practice that has remained unsustainable across domains and systems. Thus, tightly coupled data reinforces the “stovepipe” approach to systems engineering.

XML Basics

XML and related technologies are extensive and not easily summarized. The World Wide Web Consortium's "XML in 10 Points" is summarized here and gives a good overview. Some of the salient points are summarized here. [*XML in 10 points,* http://www.w3.org/XML/1999/XML-in-10-points]

- **XML is for structuring data** - XML is a simple set of rules that allow you to create tags and attributes which describe the data contained therein. XML avoids common pitfalls in language design: it is extensible, platform-independent, and it supports internationalization and localization. An address book for example, might contain the following:

```
<name>
        <first>John</first>
        <last>Doe</last>
</name>
```

- **XML looks a bit like HTML** - XML uses the tags only to delimit pieces of data, and leaves the interpretation of the data completely to the application that reads it.
- **XML is text, but isn't meant to be read** - Like HTML, XML is not meant to be read, but isn't completely inscrutible, like compiled formats. Furthermore, there is no allowance for missing tags or malformed documents. XML parsers do not repair or interpret XML documents.
- **XML is verbose by design** - Since XML is a text format and it uses tags to delimit the data, XML files are nearly always larger than comparable binary formats. Since many patterns in XML are repeated, it seems suitable for later, binary compression. This is discussed in the section on Cross-Format Schema Protocol (XFSP).
- **XML is a family of technologies** - The XML "Family" is a growing set of modules that offer useful services to accomplish important and frequently demanded tasks. These include: XLink, Xpointe, Cascading Style Sheets, Extensible Style Sheets (XSL), Extensible Stylesheet Language Transformation (XSLT) and Document Object Model (DOM). It is not necessary to be adept at all of these technologies, but they all play in concert to make XML a powerful tool.
- **XML is new, but not that new** - XML is based upon the SGML standard which has been around since the 1980's.
- **XML leads HTML to XHTML** - XHTML is the XML-compliant version of HTML.
- **XML is modular** - Through the namespace mechanism, XML allows you to define a new document format by combining and reusing other formats.
- **XML is the basis for RDF and the Semantic Web** - W3C's Resource Description Framework (RDF) is an XML text format that supports resource description and metadata applications. When the data is self-describing, other technologies can 'crawl' through the data and discover, combine and reformat data in interesting ways.

XML is license-free, platform-independent and well-supported - XML is license free and therefore not susceptible to many of the pitfalls of commercial software. There is an ever-growing, world-wide expertise in its design and implementation.

Figure 1 XML in 10 Points

XML as a Common Ground for AUVs

Since XML provides formatted, self-describing, structured data, which can in turn be validated and translated into other types of data for manipulation and display, it is a good solution to the varied data formats encountered in the world of AUVs. Consider this: if different AUV platforms must interoperate, each with its own set of software and data requirements, N^2 relationships must be maintained. An Application Program Interface (API) is needed for each one of those relationships. APIs are difficult to maintain given the dynamics of technology and competitive market forces. Each time software systems change, each API which was tuned to interface with it must be updated. Clearly, this "stovepipe" approach does not scale well.

XML provides a possible solution to this impediment to interoperability. By being extensible and modular (via namespace), XML for AUVs can provide a common ground. Each system then simply has to map itself into and out from one XML-based format. To support five disparate UAV systems, you then only need to track ten relationships, two each for each system (Input and Output). Furthermore, since AUV mission planning, command and control, and post mission analysis share many commonalities, a AUV Tactical Markup Language would be well-suited as a starting place for all AUVs. It might need little to no modification to work well with existing systems. Future systems could be constructed to “fit”, making interoperability even easier.

According to Tim Berners-Lee’s keynote address at the WWW 2003 Symposium in Budapest, Hungary May 2003, there are three stages to new users adopting XML. The first stage is “what the heck is this stuff, and why is it useful for anything?” Next, the user decides he will use XML, but he doesn’t have to understand or like it. Finally, the user picks up his laptop and tells everyone, yelling, “Look at this! The whole world is here on my laptop!” The purpose was to demonstrate the potential of XML, open standards and web-based technologies in the challenging domain of AUVs.

AUV Tactical Markup Language

The first step in constructing a Common XML-Based Mission and Data Formatting Language using XML is defining a tag set. A tag set is the set of elements and attributes used to describe what is trying to communicated or described. To make a language common and interoperable, tags must come from a central XML registry that allows common access. XML registries are a vital component in the implementation of shared data exchanges. The United States Department of Defense (DoD) XML Registry constitutes guidance in the generation and use of XML among DoD communities of interest and is the authoritative source for registered XML data and metadata components. [See http://diides.ncr.disa.mil/xmlreg/user/index.cfm] Researching the DoD XML Registry resulted in the realization that only some of the necessary tags for a common mission and data formatting language exist.

A preliminary set of XML elements and attributes describing a mission was also developed based on the command language for the Naval Postgraduate School’s (NPS) AUV Aries. “Namespaces are technical mechanisms that allow overlapping XML to be tagged with distinguishing labels.” (From DoD Metadata Registry and Clearinghouse) Some tags were chosen based on already existing tags in the DoD XML Registry. However, many of the necessary tags had not already been defined in other Namespaces. As a result, a proposed namespace specifically for AUVs was developed. In addition to the tagset, a XML Schema Document was developed to validate mission-tasking orders. Finally, as an experimental test, an XSLT template was developed to transform an XML document into a text file to be inputted into the NPS AUV Workbench, a virtual simulation of AUV missions. Using XML and related technologies, generating virtually any type of data file is possible. Follow on work in this area includes the refinement of this language to be able to command all types of AUVs and refinement of the AUV Workbench to allow for input and output of the data in this tactical markup language. [Hawkins and Van Luevan 2003]

The Schema Document for the AUV Tactical Markup Language

The power of XML was greatly enhanced by the development of the Schema. From the W3C, “**XML Schema: Structures** specifies the XML Schema definition language, which offers facilities for describing the structure and constraining the contents of XML 1.0 documents, including those which exploit the XML Namespace facility.” [See *XML Schema Part 1: Structures*, http://www.w3.org/TR/xmlschema-1]

Simply put, a schema is a set of rules that a document follows, which software may need to read before processing and displaying a document. Valid XML differs from well formed XML in its relationship to a schema. Well-formed XML is designed for use without a schema, whereas valid XML explicitly requires it.

Once written, a schema allows the user to check whether an XML document is valid. Valid XML documents employ features that can significantly improve the usability of a document, including: linking mechanisms, entities and attributes. Most XML Web sites are likely to be composed of valid XML documents. Using a schema gives creators the freedom to structure their web sites and data display so that use much greater feature sets than HTML has traditionally allowed. A sample of the Schema for the

AUV Tactical Markup Language appears in Figure 2. The entire proposed Namespace is available by request.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Barbara Van
Leuvan (Naval Postgraduate School) -->
-  <!--This schema describes the AUV mission scripting. Refer to the mission.script.Help file
for a description of the commands. -->

      -  <xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema
elementFormDefault="qualified" attributeFormDefault="unqualified">
-<xs:annotation>
      <xs:documentation>Start of defining data types</xs:documentation>
</xs:annotation>
-  <xs:simpleType name="absoluteHeadingType">
-  <xs:annotation>
      <xs:documentation>Defines valid absolute heading
      values</xs:documentation>
</xs:annotation>
-  <xs:restriction base="xs:decimal">
      <xs:minInclusive value="0" />
      <xs:maxInclusive value="359.9" />
      </xs:restriction>
      </xs:simpleType>
```

Figure 2 Exemplar Schema Document for AUV Tactical Markup Language (Hawkins and Van Luevan, 2003)

The tag, "<xs:restriction base="**xs:decimal**">" gives a clue as to the power of XML. There are many ways to describe a heading. Each one is valid, but mixing them leads to trouble. Having the data defined by this tag disallows non-decimal entries in this case. For instance, giving a compass heading such as "West" would cause an error. While this is a gross example, there are many ways to describe position and heading and confusion over format can require human in the loop review. Effectively, the use of a well-written Schema solves the problem of "Garbage In/Garbage Out".

This schema for the AUV Tactical Markup Language is a working document and can be extended (the X in XML is extensible) such that design considerations, desired data and AUV-specific information can be included by designating additional Namespaces.

Cross-Format Schema Protocol (XFSP)

XML documents are verbose by design. Transmitting these documents by common AUV methods would be cumbersome and probably impossible using acoustic modems in noisy environments. Cross-Format Schema Protocol (XFSP) has been developed as a general approach to binary serialization of XML documents. Elements and attributes are replaced via a tokenization scheme which carefully preserves valid XML document structure. XFSP uses XML schema as the basis for determining key document parameters such as legal elements, attributes and data types. Originally motivated by the flexible definition of networking protocols, binary serialization of XML via XFSP appears suitable both for message streams and document-storage streams. An open-source XFSP software implementation written in Java demonstrates the viability of the XFSP approach for XML document serialization and de-serialization. [See http://www.movesinstitute.org/Theses/Serinthesis.pdf]

As an exemplar, you can compare the file formats in Figure 3. The first column represents X3D and can be regarded as typical for any XML document. [Serin 2003] The second column represents Virtual Reality Modeling Language (file extension .wrl). The third column is binary compressed X3D and the last column represents binary compression of X3D subsequently zipped using GNU zip (gzip). Compression using X3D, XFSP and gzip is significant. [Brutzman et al, 2003]

Serialization of XML

The XSFP algorithm for binary serialization (compression) of XML is quite straight forward. Roughly, the process is:

1. Parse the XML schema document.
2. Create a look-up table for attributes and elements.
3. Assign unique token numbers to the elements and attributes.
4. Walk through the XML document tree and put each element and attribute into the output stream by replacing their tag names with numbers.
5. Preserves the content in binary form as compact XML.

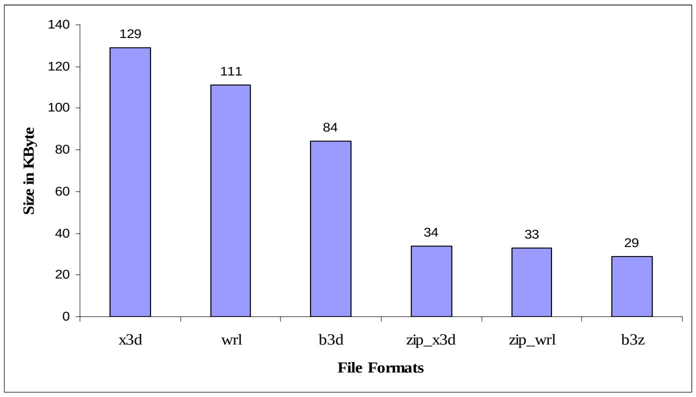


Figure 4 Size Comparison for Files using Binary XML Compression (XFSP)

A simple address book example:

```
<name>
        <first>John</first>
        <last>Doe</last>
</name>
```

Assigning unique numbers to the attributes and elements produces the following table:

Tag Name	Start Tag Number	End Tag Number	Data Type
name	11	20	SFString
first	21	30	SFString
last	31	40	SFString

Figure 5 Example Serialization of XML Tags

Then the serialized version can be generated:

```
<11>
        <21>John<30>
        <31>Doe<40>
<20>
```

While the size is not significantly reduced in this example, repetitive, long tags and even tags in series can be replaced by tokens in this manner. This greatly reduces the overall size of the document.

De-serialization of XML

Since the Schema has been tokenized and both the sender and the receiver have the schema's equivalent tag tokens, the process is easily reversed.

1. Read tag numbers from the stream.
2. Retrieve element or attribute associated with that number.
3. Read the data.
4. Create the elements, attributes and bind data to them.
5. Rebuild the XML tree.

6. Retrieve data from the tree when needed.

The document is then available as a valid, fully realized XML document. It can be processed and translated to fit the needs of the client.

Forward Error Correction (FEC) with Hamming Codes

With the serialization and compression of XML using tokens, it is then possible to add another layer of data assurance known as Forward Error Correction (FEC) using Hamming Codes. [Hamming 1997]

Forward Error Correction (FEC) increases the ability of a receiving station to correct a transmission error, thus increasing the throughput of a data link operating in a noisy environment by avoiding retranmission. The transmitting station must append information to the data in the form of error correction bits, but the increase in frame length may be modest relative to the cost of retransmission or detection.

Application-related communications will be performed as web-service requests. Binary XML compression of messages and data streams will show greater compression performance than is possible with gzip while retaining self-check schema-validation capabilities of XML. NPS further expects to allow optional addition of Hamming-code forward error correction (FEC) to all binary XML messaging, allowing receiver-side error detection and correction without retransmission. When combined with improved XML interoperability, such capabilities can be considered superior to the majority of existing machine-to-machine tactical communication encodings used by fleet systems.

The serialization of XML makes implementing Hamming Codes for AUVs much more feasible. Including Hamming Codes in the transmission of AUV telemetry, Command and Control (C^2) information and even sonar imagery can greatly increase the reliability and reduce retransmission of AUV data.

The AUV Workbench

Another impediment to the effective use of multiple AUVs is the complexity of the mission planning and analysis tools. This complexity is necessary, but each vehicle shares much of the same requirements for mission planning, rehearsal and replay and therefore each of the software tools used for these activities share commonalities. For instance, all mission planning tools must plot waypoints in one form or another. All must have some way for analyzing the data gathered on a mission. For an individual operator, it is impractical to learn all of these tools. For organizations, it may not be feasible to retain a cadre of experts, one for each tool in order to be able to field multiple vehicles. This limits the ability to tactically deploy multiple AUVs.

A common mission planning and analysis tool is proposed as one possible solution. The AUV Workbench is such a tool. (See Figure 8 and Figure 9) It is Java-based, componentized and can be customized. Custom buttons to launch separate applications can be placed on the right side of the window, mission scripts in XML or plain text can be loaded and viewed in 2D and 3D views in the right panel, as well as simplified text in the upper left pane. The graphical views and the mission script are currently linked dynamically, if the text is modified, the change is portrayed in the Mission Viewer. Other functionalities that are inherent in Java's Graphical User Interfaces (GUIs) are included in the workbench: the ability to redraw panels, input/output, tabular design, etc.

AUV Workbench Organization Overview

The AUV workbench consists of four main control threads. These threads communicate with each other either directly or over the network for required interaction. The individual threads are responsible for four distinct functions: mission execution; virtual world dynamics modeling and feedback; mission planning and generation; and mission visualization.

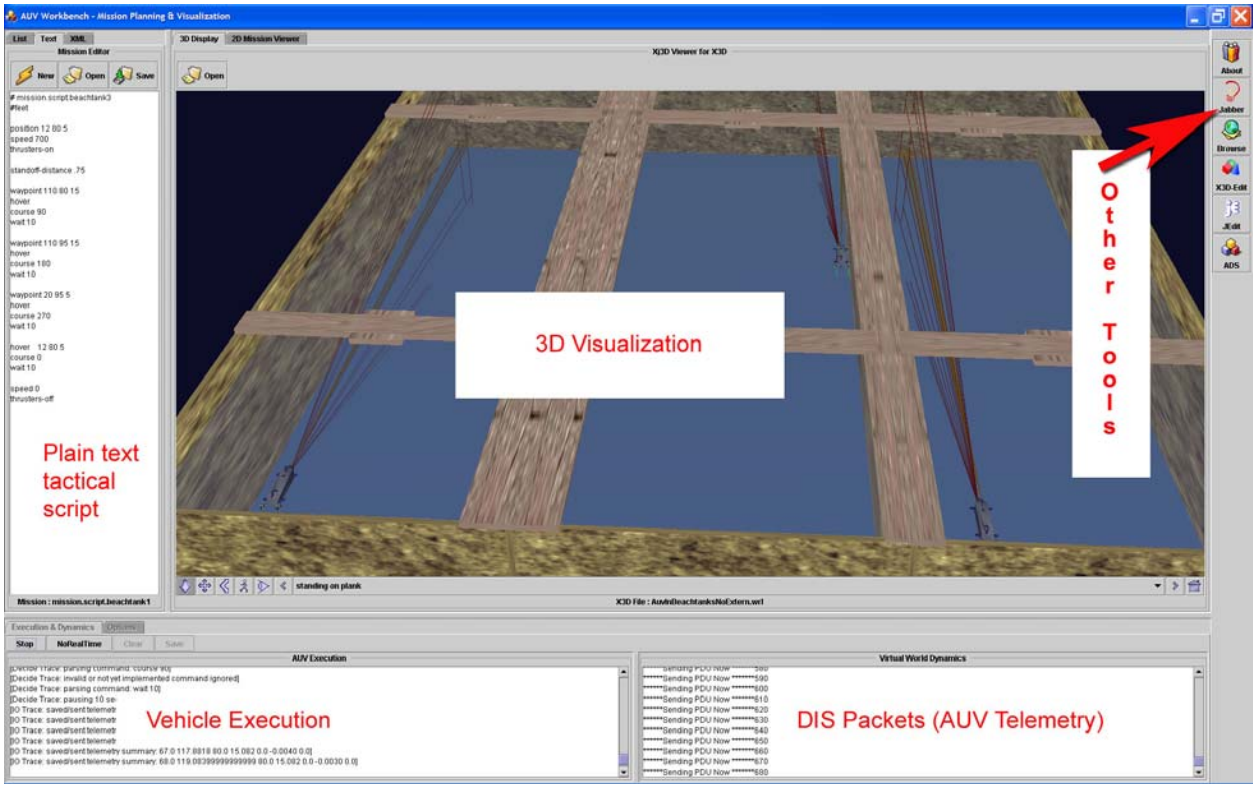
The mission execution thread runs the software from the actual AUV. It is a mirror copy of the code on board the vehicle. Utilization of the actual AUV software facilitates the development of control equations and algorithms, and enables the realistic rehearsal and fine tuning of missions in a benign lab environment prior to attempting their execution in open water. By querying the mathematical model of the virtual world for telemetry data rather than onboard sensors, the AUV software can create for itself the illusion that it is operating in the water and the software will behave accordingly. The AUV Workbench currently contains two versions of the AUV execution software (with the primary differences being implementation programming language and useable command language options). They are compiled C code and Java. While these were both developed to run on vehicles operated by NPS, their behavior can be adapted to other vehicles by adjusting the control constants and by adding, deleting or changing control equations. The vehicle control algorithms can be tested and visualized with various mission scripts, against known hydrodynamics models. Any effort to provide precision control for an AUV requires an accurate estimation of both the vehicles physical and hydrodynamic parameters. Here a vehicle model for controlled steering behaviors was developed and the hydrodynamic parameters were calculated from actual data obtained from operations. [Johnson 2001]

The virtual world dynamics thread implements the AUV hydrodynamics mathematical model. When passed a telemetry string from the AUV execution thread, the model is applied, and then a follow-on telemetry string is generated to pass back to the AUV. Additionally, a Distributed Interactive Simulation (DIS) packet is broadcast over the network to drive the visualization thread of the workbench (as well as any other DIS-enabled visualization application that may be on the network). In addition to hydrodynamics modeling, the dynamics thread contains classes that are utilized to model the vehicle's onboard sensors. Sonar data (or that from any other onboard sensor) can therefore be derived and encapsulated within the telemetry string and DIS packets to allow for the realistic feedback to the AUV execution software, and accurate mission visualization by the human operator. As with the execution software, the hydrodynamics mathematical model and sensor models currently in use were developed to model the vehicles operated by NPS, but can be arbitrarily adapted to other vehicles simply by modifying the control constants.

The visualization portion of the workbench contains a 3D viewer that utilized X3D or VRML models of the AUV and its virtual environment. By reading and interpreting the DIS packets as the come across the network, the viewer automatically animates the vehicle. It provides visual feedback on control settings, sensor effectiveness and utilization.

The final portion of the workbench provides a means of authoring and testing individual missions symbolically using the Graphical User Interface. Missions generated by the workbench take the form of a generic XML-based AUV Script Command Language (the schema currently supports mission scripting, telemetry, sensor archiving and reporting, and inter-vehicle communication). Once a mission has been generated, it is a fairly simple matter to use XSLT to convert it to any of a number of AUV command formats that are capable of running on specific vehicles. To date, transformations have been developed to convert automatically this language to command formats for two vehicles currently or previously operated by the NPS. Additionally, this generic language can be run natively by one of the two versions of execution software packaged with the AUV Workbench.

The vehicle execution code is shown running in the lover left pane of the Workbench (Figure 6). Code sets for various vehicles can be implemented here, so that the vehicle control algorithms can be tested and visualized with various mission scripts, against known hydrodynamics models. Any effort to provide precision control for an AUV requires and accurate estimation of both the vehicles physical and hydrodynamic properties. Here, a vehicle model for controlled steering behaviors was developed and the hydrodynamic parameters were calculated from actual data obtained from operations. [Johnson 2001]



```
# mission script beachtank3
#feet

position 12 80 5
speed 700
thrusters-on

standoff-distance .75

waypoint 110 80 15
hover
course 90
wait 10

waypoint 110 95 15
hover
course 180
wait 10

waypoint 20 95 5
hover
course 270
wait 10

hover   12 80 5
course 0
wait 10

speed 0
thrusters-off
```

```
[Decide Trace: parsing command: course 90]
[Decide Trace: invalid or not yet implemented command ignored]
[Decide Trace: parsing command: wait 10]
[Decide Trace: pausing 10 se
[IO Trace: saved/sent telemetr
[IO Trace: saved/sent telemetr
[IO Trace: saved/sent telemetr
[IO Trace: saved/sent telemetr
[IO Trace: saved/sent telemetry summary: 67.0 117.8818 80.0 15.082 0.0 -0.0040 0.0]
[IO Trace: saved/sent telemetry summary: 68.0 119.08399999999999 80.0 15.082 0.0 -0.0030 0.0]
```

```
*******Sending PDU Now ********580
*******Sending PDU Now ********590
*******Sending PDU Now ********600
*******Sending PDU Now ********610
*******Sending PDU Now ********620
*******Sending PDU Now ********630
*******Sending PDU Now ********640
*******Sending PDU Now ********650
*******Sending PDU Now ********660
*******Sending PDU Now ********670
*******Sending PDU Now ********680
```

Figure 6 The AUV Workbench Graphical User Interface (GUI)

Mission Planning

Many AUVs share commonalities which allow for the development of a common tool for mission planning. Waypoints, speed, heading, depth, and mission duration are some common features of almost every AUV mission. There has been extensive study in what constitutes effective coverage for mine field clearance and it will not be discussed. However, there are not only common features within the mission plan, but there are also commonly found mission plans. "Mowing the Lawn" is a typical mission track and is often repeated regularly, with only minor changes made for local conditions. It has been difficult in the past to plan a mission for one vehicle, and then apply that exact same mission plan to another vehicle to allow for true comparison and experimentation. The missions must be approximated due to the differences in the planning tools. A common tool that "styles" each mission script to its appropriate vehicle would allow for experimentation and validation of archetypal missions across various vehicles.

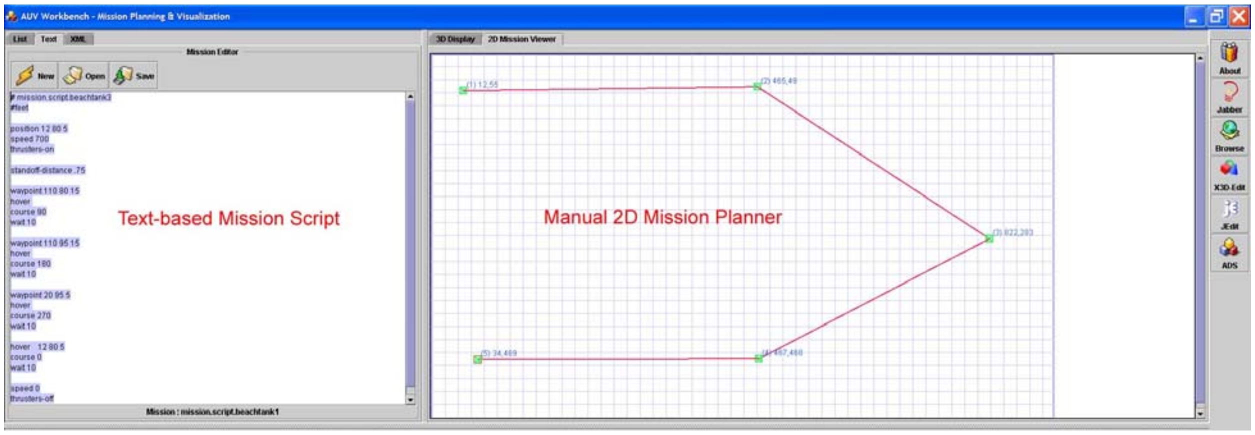
Figure 7 shows a sample mission script, using XML tags, that could be applied to various vehicles for experimentation, without manual translation.

```
<?xml version="1.0" encoding="utf-8"?>
<auv-mission vehicle="aries">
  <mission>
    <position x="12" y="80" depth="5"/>
    <propeller rpm="700"/>
    <thrusters/>
    <standoff range="0.75"/>
    <waypoint x="110" y="80" depth="15"/>
    <hover/>
    <heading value="90"/>
    <wait time="10"/>
    <waypoint x="110" y="95"/>
    <hover/>
    <heading value="180"/>
    <wait absolute="false" time="10"/>
    <waypoint x="20" y="95" depth="5"/>
    <hover/>
    <heading value="270"/>
    <wait time="10"/>
    <hover x="12" y="80" depth="5"/>
    <heading value="0"/>
    <wait time="10"/>
    <propeller rpm="0"/>
    <thrusters on="false"/>
  </mission>
</auv-mission>
```

```
# mission.script.beachtank3, feet

position 12 80 5
speed 700
thrusters-on

standoff-distance .75

waypoint 110 80 15
hover
course 90
wait 10

waypoint 110 95 15
hover
course 180
wait 10

waypoint 20 95 5
hover
course 270
wait 10

hover    12 80 5
course 0
wait 10

speed 0
thrusters-off
```

Figure 7 Sample Mission Planning Script, comparing XML and original plain text format

The AUV Workbench currently supports a simple, text-based mission script, as shown in Figure 8, but it also supports the use of XML-based mission scripts. It can also be extended to include custom applications that can be launched in a separate process by adding buttons to the right of the panels. This becomes a very convenient way for users to bundle applications that they commonly use with AUV mission planning.

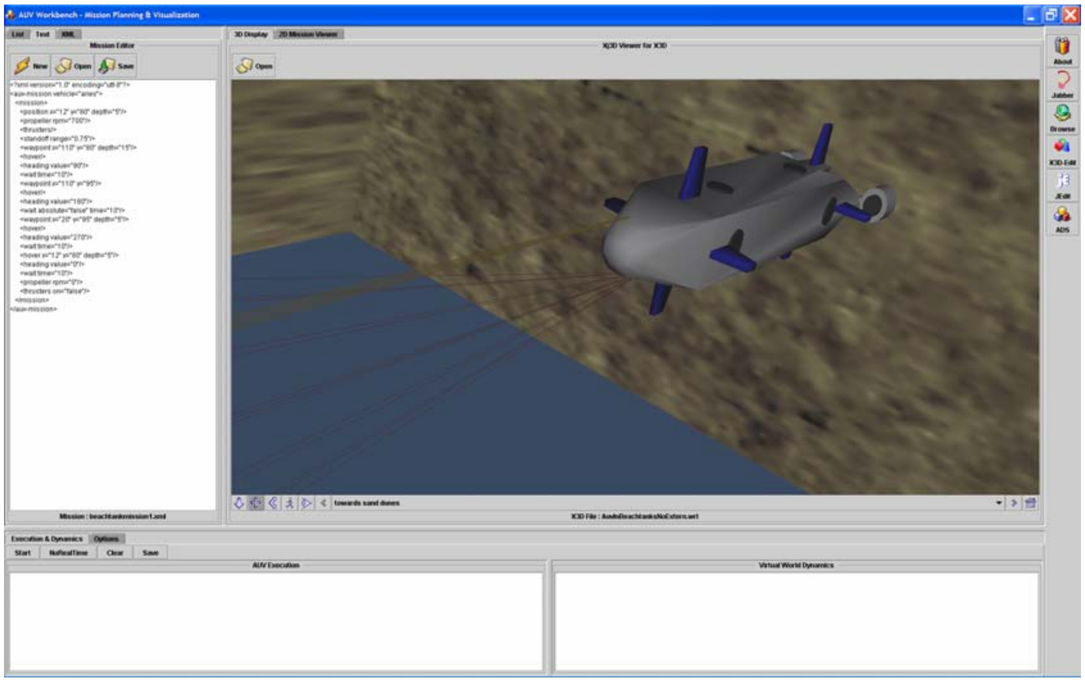


```
# mission.script.beachtank3
#feet

position 12 80 5
speed 700
thrusters-on

standoff-distance .75

waypoint 110 80 15
hover
course 90
wait 10

waypoint 110 95 15
hover
course 180
wait 10

waypoint 20 95 5
hover
course 270
wait 10

hover    12 80 5
course 0
wait 10

speed 0
thrusters-off
```

Figure 8 The AUV Workbench – 2D Mission Planning

Mission Simulation

Experimental AUVs are the result of many years of effort. Other AUVs are expensive. None of them are trivial or insignificant assets. It is important, especially when doing research into new tactics and techniques, to be able to test and simulate. The AUV Workbench is valuable in this regard. A user can test, simulate and visualize missions, new control algorithms and even new vehicles in a controlled environment, safe from the rigors of the underwater environment. It allows systems and software to be thoroughly tested before they are implemented onboard an AUV. The networking capability allows even remote viewing of mission simulations by using the Distributed Interactive Simulation (DIS) protocol. [See Distributed Interactive Simulation Working Group of the Web 3D Consortium http://www.web3d.org/WorkingGroups/vrtp/dis-java-vrml/]



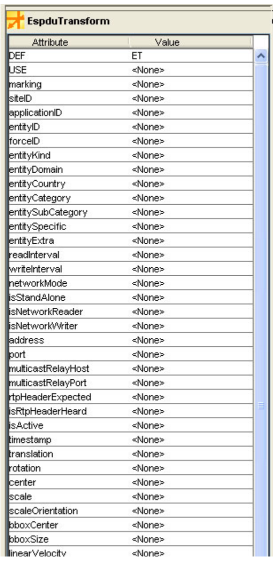
```
<?xml version="1.0" encoding="utf-8"?>
<auv-mission vehicle="aries">
  <mission>
    <position x="12" y="80" depth="5"/>
    <propeller rpm="700"/>
    <thrusters/>
    <standoff range="0.75"/>
    <waypoint x="110" y="80" depth="15"/>
    <hover/>
    <heading value="90"/>
    <wait time="10"/>
    <waypoint x="110" y="95"/>
    <hover/>
    <heading value="180"/>
    <wait absolute="false" time="10"/>
    <waypoint x="20" y="95" depth="5"/>
    <hover/>
    <heading value="270"/>
    <wait time="10"/>
    <hover x="12" y="80" depth="5"/>
    <heading value="0"/>
    <wait time="10"/>
    <propeller rpm="0"/>
    <thrusters on="false"/>
  </mission>
</auv-mission>
```

Figure 9 AUV Workbench - 3D Mission Visualization

Network Support for Distributed Simulations

Because the AUV Workbench uses the Distributed Interactive Simulation (DIS) Protocol to generate network traffic, the missions it generates can be visualized across a local or wide area network. DIS is the Institute of Electrical and Electronics Engineers, Inc., (IEEE) recommended practice for networked simulations. [See http://standards.ieee.org/catalog/olis/compsim.html] DIS is packet-oriented and is a natural match for UDP/Multicast/Broadcast over IP.

From the DIS-Java-VRML Working Group of the Web 3D Consortium, "The primary mission of DIS is to define an infrastructure for linking simulations of various types at multiple locations to create realistic, complex, virtual "worlds" for the simulation of highly interactive activities. This infrastructure brings together systems built for separate purposes, technologies from different eras, products from various vendors, and platforms from various services and permits them to interoperate. DIS exercises are intended to support a mixture of virtual entities (human-in-the-loop simulators), live entities (operational platforms and test and evaluation systems), and constructive entities (wargames and other automated simulations). The DIS infrastructure provides interface standards, communications architectures, management structures, fidelity indices, technical forums, and other elements necessary to transform heterogeneous simulations into unified seamless synthetic environments. These synthetic environments support design and prototyping, education and training, test and evaluation, emergency preparedness and contingency response, and readiness and warfighting. [See http://www.web3d.org/WorkingGroups/vrtp/dis-java-vrml/AnnotatedReferences.html#DIS]


EspduTransform

Attribute	Value
DEF	ET
USE	<None>
marking	<None>
siteID	<None>
applicationID	<None>
entityID	<None>
forceID	<None>
entityKind	<None>
entityDomain	<None>
entityCountry	<None>
entityCategory	<None>
entitySubCategory	<None>
entitySpecific	<None>
entityExtra	<None>
readInterval	<None>
writeInterval	<None>
networkMode	<None>
isStandAlone	<None>
isNetworkReader	<None>
isNetworkWriter	<None>
address	<None>
port	<None>
multicastRelayHost	<None>
multicastRelayPort	<None>
rtpHeaderExpected	<None>
isRtpHeaderHeard	<None>
isActive	<None>
timestamp	<None>
translation	<None>
rotation	<None>
center	<None>
scale	<None>
scaleOrientation	<None>
bboxCenter	<None>
bboxSize	<None>
linearVelocity	<None>

Figure 10 ESPDU Example

The AUV Workbench produces standard data packets for DIS, called Entity State Protocol Data Units (ESPDUs). These ESPDUs contain an entity's position (in x,y,z coordinates), velocity, acceleration, angular velocity, angular acceleration, and orientation. They also contain "articulation parameters", which handles things such as thruster orientation on an AUV. The Aries AUV model was created using X3D-Edit. [See http://www.web3d.org/TaskGroups/x3d/translation/README.X3D-Edit.html]. Figure 10 shows an example of the ESPDU fields found in the ESPDU transform node of X3D . This node accepts network-routed values to drive the model through the virtual world.

Since DIS is an IEEE standard, anyone can buy the standard and write an implementation. Some commercial implementations exists, but a free Java implementation from NPS is available. [See http://web.nps.navy.mil/~brutzman/vrtp/dis-java-vrml]

The ESPDUs for a given entity each have a unique identifier called out in the header of the packet. The object or model in the virtual world can then "listen" for its unique identifier and is driven around the virtual world based on the entity state communicated in the data packet. The protocol has safeguards for lossy networks, such as a five-second dead reckoning and virtual world course correction for temporary periods when no packets have been received. The header information is encoded from the field values shown in Figure 10.

Server-Side Support

As AUVs and other data collectors become more ubiquitous, it will be increasingly challenging to manage all these data. On recent exercises such as Millennium Challenge '03, the data collection was done by handing an operator a diskette with the mission data. It was then processed by a technician and uploaded to a local computer. This process, while robust, was labor intensive.

With XML-based mission results, this process can be automated fairly easily. In the AUV Workbench, the results are gathered from multiple sources and simply pushed up via the web to a server. Because these files contain sufficient meta-data, the server can automatically sort, file and even validate the information based on a Schema. From this web archive, mission results can be presented in easily understood ways, such as HTML pages and X3D replay of missions. Large scale mission repositories can be made available for research and tactical exploitation.

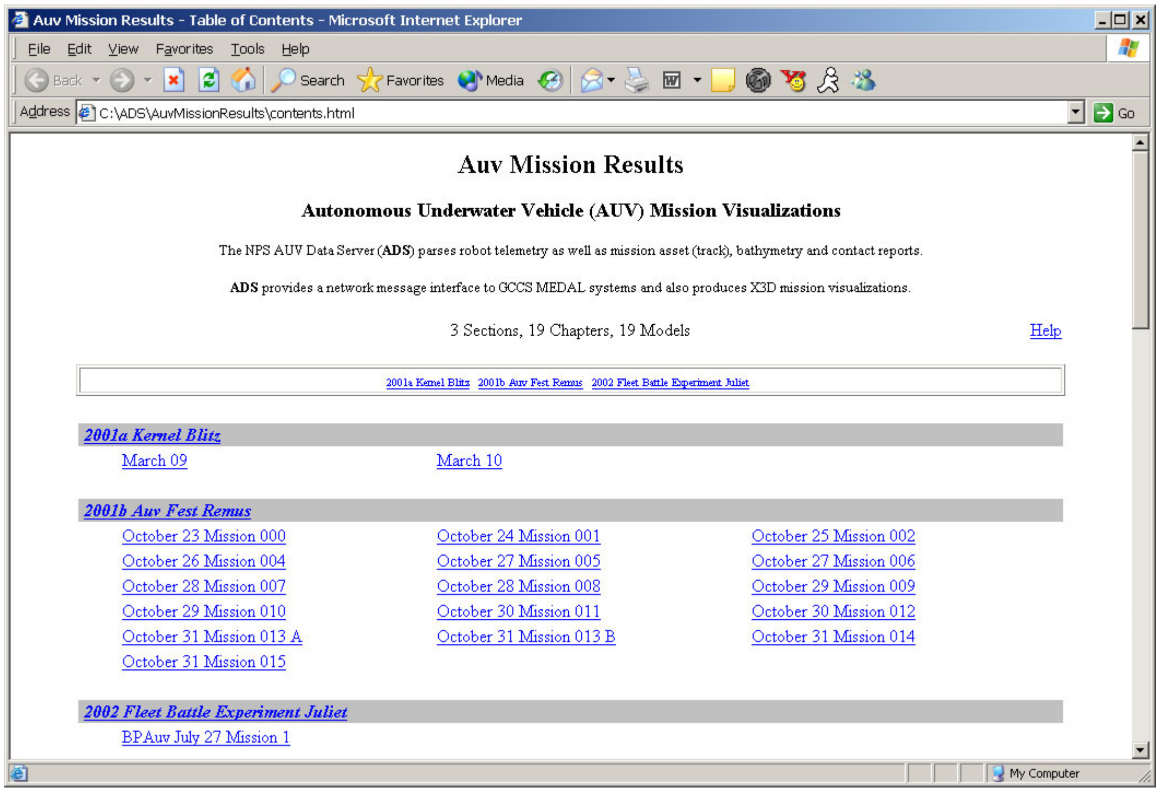
Real World Results

Currently, there are no data-rich AUV mission archives widely available on the World Wide Web. The data are ephemeral. Archival information is often limited to MATLAB® generated plots and HTML pages with imagery, but none of the actual mission data. This limits the ability of others to scrutinize and analyze the data. Adopting a common AUV Tactical Mission Language would allow for the auto-generation of web-based archives. (Figure 12)

This web-based archive could be viewed in a web browser, the data analyzed either by machine or by humans in 2D or 3D; it could be transformed via XSLT to other data formats and even imported into other mission planning tools for further experimentation. Of course, it could also be obscured and encrypted, by any means that is commonly available for other web-based information.

Web-based Replay of Missions

Figure 11 is an example of one web-based mission replay, available in the SAVAGE Model Archive at the Naval Postgraduate School (NPS). There are over 800 model components and models available there, including many AUVs and underwater mines. These models are typically formatted such that they can be automatically inserted into a computer generated scene. As such, they can greatly aid in the analysis by creating visual cues for the analyst. [See http://www.stl.nps.navy.mil/~brutzman/Savage/contents.html]


Auv Mission Results - Table of Contents - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\ADS\AuvMissionResults\contents.html
Go
My Computer

Auv Mission Results

Autonomous Underwater Vehicle (AUV) Mission Visualizations

The NPS AUV Data Server (**ADS**) parses robot telemetry as well as mission asset (track), bathymetry and contact reports.

ADS provides a network message interface to GCCS MEDAL systems and also produces X3D mission visualizations.

3 Sections, 19 Chapters, 19 Models Help

2001a Kernel Blitz 2001b Auv Fest Remus 2002 Fleet Battle Experiment Juliet

2001a Kernel Blitz

March 09	March 10	

2001b Auv Fest Remus

October 23 Mission 000	October 24 Mission 001	October 25 Mission 002
October 26 Mission 004	October 27 Mission 005	October 27 Mission 006
October 28 Mission 007	October 28 Mission 008	October 29 Mission 009
October 29 Mission 010	October 30 Mission 011	October 30 Mission 012
October 31 Mission 013 A	October 31 Mission 013 B	October 31 Mission 014
October 31 Mission 015		

2002 Fleet Battle Experiment Juliet

BPAuv July 27 Mission 1

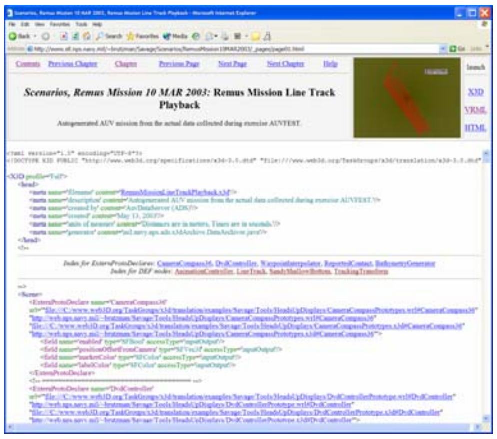
Figure 12 Various AUV Mission Results in a Web-based Archive

Related Efforts

The SAVAGE project is a long-term research program seeking to push the frontier of Web-based 3D scenario authoring and visualization. In addition to the numerous models and case-study examples previously described, directions for the work include:

- Expanding the palette of models and events that can be inserted into a scenario, including representation of control measures and other non-physical concepts in the battlespace.
- Creating scenarios of greater complexity depicting the interplay of represented land, air, sea, and littoral objects and operations. Include the interaction of operations with control measures.
- Creating branching flows in the scenario script to present decision points that engage the user in the unfolding scenario.
- Investigating assignment of behaviors to scenario objects; for example, from XML libraries of pre-scripted actions (see Lacy and Pearman, 2000) with adaptation mechanisms to the current situation.
- Developing techniques for rapidly generating battlespace terrain, to include representation of built-up areas and vegetation cover, for use in the Web3D environment.
- Automating extraction of scenario information from doctrinal operations orders and plans.
- Integrating Web3D graphics into an overarching Web-based Modeling and Simulation framework.

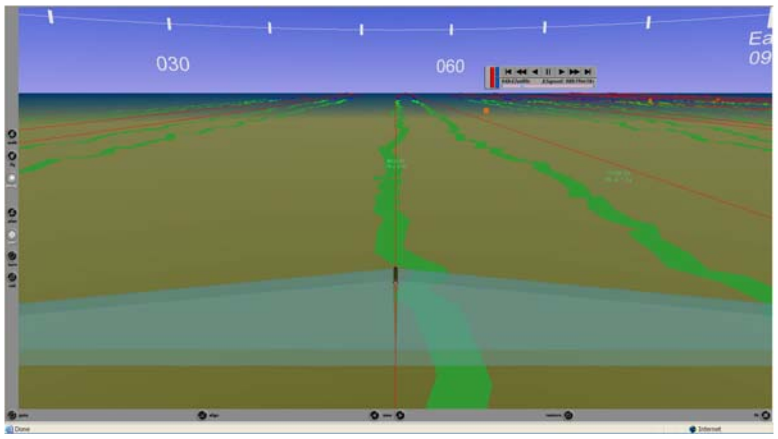
The SAVAGE Model Archive and related tools have proven to be a flexible and powerful base for modeling and simulation visualization. [Blais et al. 2001 and 2002]



Scenarios, Remus Mission 10 MAR 2003: Remus Mission Line Track Playback

Autogenerated AUV mission from the actual data collected during exercise AUVFEST.

[illegible]



XML-based Tactical Chat (XTC)

Text-based communications over Internet Protocol (IP) or chat has been around for many years. It is widely adopted by most Internet users and has proven to be useful in tactical situations. There are several popular, but proprietary tools for chat. None are currently officially supported by the US Navy, because of security issues and licensing concerns. XML Tactical Chat (XTC) over comes some of these restrictions because it complies with the principles of Open Standards. It is free to use, open source software that bridges multiple platforms and protocols. It can be inspected, modified and improved without the encumbrances of commercial software and since it is XML-based, the messages themselves are readable by machines and humans alike. Unlike commercial chat clients, it doesn't rely on specialized servers. It operates on any web server and can cross network boundaries using standard protocols. It is also client agnostic, meaning that XSLTs can be performed to suit any specific client application. It supports individual or group messaging, and can be used to transmit binary (imagery) as well as plain text.

Agents can also use XTC to interact with agent-based software. This allows for an element of Artificial Intelligence to be interjected into the tactical chat environment. For instance, in an information-rich, tactical environment, agents could be assigned to watch targets of interest and when these targets show a change of intention (moving off a station, for instance), an agent could send out a tactical chat message to the analyst. This makes it possible for the analyst to track many more targets than otherwise possible.

Extensible Modeling and Simulation Framework (XMSF)

The Extensible Modeling and Simulation Framework is defined as a composable set of standards, profiles and recommended practices for web-based modeling & simulation (M&S). XML-based markup languages, Internet technologies and Web Services will enable a new generation of distributed M&S applications to emerge, develop and interoperate. XMSF integrates several high-level requirements derived from years of experience with M&S frameworks, and the challenges of their effective deployment across diverse networks and systems.

XMSF must enable simulations to interact directly and scalably over a highly distributed network, achieved through compatibility between a web framework and networking technologies. XMSF must be equally usable by human and software agents. Clearly XMSF must support composable, reusable model components. XMSF use must not be constrained by proprietary technology or legally encumbering patents, since such barriers discourage the free, open, ad hoc development of interconnected tactical models and simulations. [See http://www.movesinstitute.org/xmsf/xmsf.html and http://netlab.gmu.edu/xmsf]

WHAT IS XMSF?

- XMSF provides the technical basis for transformational interoperability via XML interchange, profiles, and recommended practices for web-based modeling & simulation.
- Broad technical interoperability is provided by open standards, XML-based markup languages, Internet technologies, and cross-platform Web services.
- Supports diverse distributed modeling and simulation applications. Also enables simulations to interact directly and scale appropriately over a distributed network through composable and reusable model components.
- Employs mainstream practices of enterprise-wide software development.
- Provides support for all types and domains of modeling and simulation (constructive, live, virtual, and analytical).
- Excellent support for ISO Extensible 3D (X3D) Graphics Specification with industry-academic-government activity in multiple standards consortia

WHAT ARE THE ADVANTAGES OF XMSF?

- Supports Open Standards in Web, Internet, and XML technologies. Web services allow self validating syntax and semantics to achieve cross-cutting interoperability in modeling and simulation.
- Development and acceptance of common data and metadata standards provides semantic consistency among systems and services.
- Profiles are specification suites based on international standards, which define common capabilities for content production user/application support.
- Data-driven conversion capabilities and application ubiquity provides both best business case and best technical case on a DoD-wide scale.

As with other efforts, XMSF uses best business practices from the World Wide Web, strong institutional support and cutting edge technology to solve some of the grand challenges of modeling and simulation. [Brutzman and Zyda 2002]

Conclusion

Interoperability through open-standards, loosely coupled data and web-technologies is crucial to the success of multiple vehicular and multi-national operations. Open source applications such as the AUV Workbench benefit from collaborative efforts, across international and technological boundaries. It is componentized and scaleable to meet changing needs. The user, not the developer, sets the requirements under these conditions.

Through the use of X3D, users can rehearse, replay and create virtual reality. This ability to “see” the vehicle, when it would otherwise be obscured by the depths to the ocean during replay and to test new ideas through simulation is critical to success. Any failure, while on a real mission, can be catastrophic. While simulation plays a limited role in mission planning across most defense-related activities, its applicability to AUVs and their operation in the harsh underwater environment is clear.

The related efforts touched upon here reveal a larger strategy: when open source, open standards, combine with best e-business practices, every effort then becomes related, synergies are created and tasks that once seem daunting, become achievable.

References

Blais, C.L., Brutzman, D., Harney, J.W., & Weekley, J. (2002a). “Web-based 3D reconstruction of scenarios for limited objective experiments”, Proceedings of the 2002 Summer Computer Simulation Conference, San Diego, 17-19 July. Available at http://www.movesinstitute.org/Publications/S192_Blais.pdf

Brutzman, D. and McGregor, D., Naval Postgraduate School, and Hudson, A., Yumetech Inc., “XML Binary Serialization using Cross-Format Schema Protocol (XFSP) and XML Compression Considerations for Extensible 3D (X3D) Graphics”, Binary Interchange Workshop of the W3C, 2003. Available at http://www.w3.org/2003/08/binary-interchange-workshop/40-BrutzmanXmlBinarySerializationUsingXfspW3cWorkshopSeptember2003.pdf

Brutzman, D and Zyda, M., “Extensible Modeling and Simulation Framework (XMSF) Challenges for Web-Based Modeling and Simulation” http://www.movesinstitute.org/xmsf/XmsfWorkshopSymposiumReportOctober2002.pdf

Distributed Interactive Simulation Working Group of the Web 3D Consortium www.web3d.org/WorkingGroups/vrtp/dis-java-vrml

Hamming, R., *The Art of Doing Science and Engineering, Learning to Learn,* Gordon and Breach, 1997

Hawkins, D. and Van Leuvan B**., “**An XML-based mission command language for autonomous underwater vehicles (AUVs)”**,** Master’s Thesis, Naval Postgraduate School, Monterey, CA June 2003

Neushul, J., “Interoperability, data control and battlespace visualization using XML, XSLT and X3D”, Master’s Thesis, Naval Postgraduate School, Monterey, CA 03 September, 2003. Available at http://www.movesinstitute.org/Theses/Neushulthesis.pdf

Johnson, J.**, “**Parameter Identification of the Aries AUV”, Master’s Thesis, Naval Postgraduate School, 2001. Available at http://stinet.dtic.mil/cgi-bin/fulcrum_main.pl?database=ft_u2&searchid=0&keyfieldvalue=ADA397498&filename=%2Ffulcrum%2Fdata%2FTR_fulltext%2Fdoc%2FADA397498.pdf

Serin, E., “Design and Test of the Cross-Format Schema Protocol (XFSP) for Networked Virtual Environment”, Master’s Thesis, Naval Postgraduate School, Monterey, CA March 2003

XML in 10 points, http://www.w3.org/XML/1999/XML-in-10-points

XML Schema Part 1: Structures, http://www.w3.org/TR/xmlschema-1