2006

DIS-XML, XML Representations of Distributed Interactive Simulation Packets

McGregor, Don
Monterey, California: Naval Postgraduate School.

http://hdl.handle.net/10945/37847
DIS-XML

XML representations of Distributed Interactive Simulation Packets

Don McGregor, Naval Postgraduate School (mcgredo at nps.navy.mil)
Don Brutzman, Naval Postgraduate School (brutzman at nps.navy.mil)
DIS

DIS is a well-understood binary IEEE standard for exchanging information in military simulations.

But the binary information is not well suited for working with other tools--the custom format works against archiving, use in databases, further processing, etc. The data format makes it an island cut off from other tools.
Information Representations

In practice there is a DIS “ infoset” of data that usually has at least two representations:

- A IEEE-DIS standard binary format PDU
- A programming language object that corresponds to a PDU

The information contained in the representations is very nearly identical, but the different formats are each useful in their own context.

It is very common for the information to be transferred between these representations.
Often the information represented in a format is shuffled to another format, and back again--the information is read from the wire in IEEE format, then transformed to a Java or C++ object. The same data is in both representations, just in a different format.
Other Representations

Why not other representations for the same information, such as XML? The alternative representations can work better in other contexts.
XML Information Representations

XML is a valuable way to represent DIS data because information in XML can be used by many tools—parsers, databases, transformation tools, web services, data mining, etc.

This means that the data in DIS is no longer a special case, just more XML data.
DIS in XML Format

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DIS>
    <EntityStatePdu capabilities="0" entityAppearance="0"
        forceID="0" numberOfArticulationParameters="0">
        <PduHeader pduType="1" protocolFamily="1" timestamp="0"/>
        <EntityID/>
        <Entity/>
        <AlternativeEntity/>
        <EntityLinearVelocity/>
        <EntityLocation x="1.0" y="2.0" z="3.0"/>
        <EntityOrientation/>
        <DeadReckoningParameters
            otherParameters="00000000000000000000000000000000">
            <EntityLinearAcceleration/>
            <EntityAngularVelocity/>
        </DeadReckoningParameters>
        <EntityMarking characterSet="0" characters="00000000000000000000000000000000"/>
    </EntityStatePdu>
</DIS>
Transformations

It is obvious that we will need to shift information from one representation to another. We should choose a starting point for the representations that makes our job easier.

It turns out a natural starting point is to write an XML schema that describes DIS PDUs. Using standard industry tools we can then automatically generate programming language objects that know how to marshal themselves to XML, and then manually write a minimal amount of code to marshal the objects to IEEE DIS format.

We have already written an XML schema for DIS.
Transformations

The example schema for DIS written by NPS describes the DIS-1998 standard
An example fragment from the NPS XML Schema for DIS:

```xml
<xsd:complexType name="EntityIDType">
  <xsd:attribute name="site" type="xsd:unsignedShort"/>
  <xsd:attribute name="application" type="xsd:unsignedShort"/>
  <xsd:attribute name="entity" type="xsd:unsignedShort"/>
</xsd:complexType>
```
Transformations

We use JAXB from Sun to automatically generate Java programming language objects from the schema. These objects know how to marshal themselves to and from XML for “free”.

```java
public class EntityId
{
    short site;
    short application;
    short entity;

    public short getSite()
    {
        ...
    }
}
```
Transformations

The automatically generated programming language code can read and write XML documents. This means that once we have the XML schema, we need only write code to marshal and unmarshal to IEEE DIS format.

Marshalling to IEEE DIS is relatively straightforward. We have implemented this code for the most popular PDUs.
The portions in red must be manually implemented; we have already written the XML schema for DIS and some of the marshalling code. The rest is “free”
Benefits

This gives us the ability to archive DIS data in XML format, and to take advantage of all the tools in the XML ecosystem, including databases, transformational tools, query tools, web services, etc. It also allows us to easily shift data between representations. The DIS information is no longer a “special case” but just more XML data that can be chewed on by the same tools that are used for other XML data.
Objective

There would be obvious benefits to having a standard for XML representations of DIS data. We would like DIS XML documents to be interchangeable among vendors, just as IEEE DIS is.

This implies a standard, agreed-upon DIS XML schema.

We want others to review our DIS XML schema, make suggestions, and agree upon a standard.
Getting Code

The code is available at sourceforge via CVS or package download. Open source, BSD license.

To download a source code snapshot:
https://sourceforge.net/projects/xmsf, click on the download button, and select the “disxml” package

You can also download the latest from the CVS repository—instructions at the sourceforge page

You should also have Ant installed. Get it from http://ant.apache.org

Install xj3d, an X3D browser, available at http://www.xj3d.org/
Running Code

There are various simple demo programs that can be run from the command line. Open up an Xj3d browser window and load sliderExample.x3dv, a simple 3D scene.

Start up a packet generating tool:

```
ant runslider
```

This opens a Java Swing application. Moving the sliders generates IEEE DIS packets that are sent over the wire and move the 3D object.
Screen Shots

Xj3d Browser with 3D object controlled from DIS

Swing application that sends dis-xml binary PDUs
Running Code

You can also generate XML-format PDUs:

```
ant run
```

This writes some PDUs in XML format to the file `xmlPdus.xml`. The example code, in `TestHarness.java`, also reads PDUs back from an XML file.
Running Code

An example program that sends XML format DIS PDUs across an IETF-standard Extensible Messaging and Presence Protocol (XMPP) instant messaging backbone is also available. This is in the process of being expanded to a servlet/JSP-based web application.
Contact Points

Don McGregor, Naval Postgraduate School
mcgredo at nps.navy.mil, 831-656-7605
Source code at sourceforge.net/projects/xmsf
Mailing list: http://movesinstitute.org/mailman/listinfo/dis-xml