Home of the 1996 Olympic Village
Georgia Tech Research Institute
Computer Science and Information Technology Division
Information Technology and Telecommunications Laboratory

# XML DIF Experiment

David Rosenbaum

david.rosenbaum@gtri.gatech.edu

Georgia Tech Research Institute

Information Technology & Telecommunications Laboratory

Distributed Simulation Systems Group

December 9, 1998

# Goals of the experiment

- Create a program (“FEDDIFWriter”) that reads an XML OMT document and writes the corresponding FED DIF file
- Thereby:
  - Gain programming experience
    - Tools
    - Techniques
    - Level of effort
  - Verify practicability of extracting FED information from an OMT document

# Useful XML resources

- The XML Handbook
  - Prentice Hall: ISBN 0-13-081152-1
  - Good technical overview (Chapter 3)
  - Good, relatively in-depth technical introduction (Part Five)
  - Good free software list (Chapter 30)
  - Lots of material on use cases and general-interest tools
  - CD-ROM full of resources and resource pointers
- http://www.xml.org
  - The Annotated XML Specification
  - Large set of resource pointers

# Parser API standards (1 of 2)

- DOM: Document Object Model
  - Conceptually, all XML documents are trees of elements that can contain:
    - Child elements
    - Text
    - Attributes
  - Many XML parsers operate by creating a corresponding in-memory tree and providing access to this tree
  - DOM is an API for in-memory document tree access and manipulation
    - W3C Recommendation
    - Language-neutral
    - Supports user subclassing of tree nodes
    - Does not include a parser initiation interface - used after parsing is complete
    - Memory usage is roughly proportional to document size

# Parser API standards (2 of 2)

- SAX: Simple API for XML
  - The recognition of a document feature during parsing can be thought of as an “event”
    - e.g. beginning of document, end of document, beginning of element, end of element, availability of character data
  - SAX is an API for delivery of parsing events to an application program
    - De facto standard developed by members of the xml-dev mailing list
    - Currently supports Java; IDL later
    - Callbacks include startDocument(), endDocument(), startElement(), endElement(), and characters()
    - Includes a parser initiation interface
    - Minimal memory usage

# Parsers (1 of 2)

- IBM XML Parser for Java (XML4J)
  - Supports validation
  - Actively maintained
  - Committed to tracking standards, including DOM and SAX
  - Supports unparsing
  - Exposes DTD information
- Microsoft XML Parser in Java
  - Supports validation
  - Proprietary in-memory tree access API
  - SAX driver is available
  - Microsoft has discontinued improvements; attention is now focused on the Microsoft-Data Channel XML Java parser

# Parsers (2 of 2)

- Sun Java Project X: Java Services for XML Technology
  - Current release is Early Access 2
  - Supports validation
  - Supports DOM and SAX
- Several more Java parsers
- Parsers also available for C++, C, Python, Tcl, and Perl

# Implementation overview

- Used IBM XML4J
  - Wanted to use a high-profile tool
  - MS: too proprietary, no longer under active development
  - Sun: wasn’t released until after implementation had begun
- Subclassed DOM Element class
- Implemented writeFEDDIFV13() methods
- Also implemented writeOMTDIFV13() methods to create incomplete OMT DIF writer

# DOM subclassing (1 of 4)

- Memory demands of tree structure not an issue
- Tried alternate implementation based on XML4J's built-in Visitor pattern support, which proved too cumbersome
- Wasn't worried about portability; used some of XML4J's extensions
- Used sed and some scripting to generate trivial Element subclass implementations for each of the element types.
  - e.g. Element_interactionClass, Element_parameter
- Overrode default factory to create Element subclasses

# DOM subclassing (2 of 4)

- Example of trivial Element subclass

```
//
// Element_updateReflectTag.java
//

package gtri.xml.omt;

public class Element_updateReflectTag extends OMTElement
{
    // Methods

    public Element_updateReflectTag(String aName)
    {
        super(aName);
    }
}
```

# DOM subclassing (3 of 4)

- Fragment of overriden class factory

```
...
public class OMTDocument extends TXDocument
{
    // Methods

    // Factory method for elements
    public Element createElement(String aName)
    {
        TXElement lElement;

        if (aName.equals("omt"))
        {
            lElement = new Element_omt(aName);
        }
        else if (aName.equals("identification"))
        {
            lElement = new Element_identification(aName);
        }
...
```

# DOM subclassing (4 of 4)

- Fragment of non-trivial Element subclass

```
...
public class Element_interactionClass extends OMTElement
{
...
    public void writeFEDDIFV13(Indenter aIndenter)
    {
        aIndenter.println("(class " +
                          FEDDIFWriter.ToNameString(getAttribute("name"))+ " " +
                          FEDDIFWriter.ToFEDTransport(getAttribute("transport")) + " " +
                          FEDDIFWriter.ToFEDOrder(getAttribute("order")) + " " +
                          FEDDIFWriter.ToNameString(getAttribute("routeSpace")));
        aIndenter.indent();
        writeFEDDIFV13ForChildren(aIndenter);
        aIndenter.outdent();
        aIndenter.println(")");
    }
...
}
```

# Conclusions

- With XML you get:
  - Resources
  - Tools (including parsers)
  - Commitment of heavyweight companies like IBM, Sun, and Microsoft
- It’s easy to make use of XML data in your program