

XML Application in the TrEPS System

Janine Lafayette

www.csm.ornl.gov/~janine

Overview

- ❑ RAM Project Scope
- ❑ ITS Requirements
- ❑ Enabling Technologies
 - XML
 - Applicability to TrEPS
- ❑ Conclusion (Recommendation)
- ❑ Summary
- ❑ Next Steps...

ITS Environment

- ITS is a system of systems
 - Communications
 - Multi-jurisdiction, multi-organization, multi-system, multi-protocol
 - Software systems
 - Multi-platform, non-standardized
 - Surveillance systems
 - Multi-technology
 - Management & control systems
 - Multi-jurisdiction, multi-agency, multi-system
 - Etc.

TrEPS Technical Needs

- ❑ Link traditional data exchanges to ITS subsystems
- ❑ Provide a means for collaborating agencies to quickly and easily to interface with each other
- ❑ Create TrEPS-based services based on smart documents
- ❑ Enable a low cost server and client based solutions

TrEPS Needs

□ Data interoperability

- Standardized TrEPS data definitions
 - Defining data structures and content
 - Exchanging data independent of format
- Platform-independent data transfer
- Seamless operation of systems that interface with TrEPS

RAM Project Goals

- ❑ Learn XML
- ❑ Determine the maturity and application potential of XML technology for TrEPS application
- ❑ Prepare report & presentation on the findings

What is XML?

- ❑ Short for eXtensible Markup Language, XML is a method for putting structured data into a text file
- ❑ Standards established by the World Wide Web consortium (W3C)
- ❑ Similar to HTML, XML was derived from the Standard Generalized Markup Language (SGML)
- ❑ XML is a metalanguage, designed to support the definitions of unlimited number of languages for specific industry applications
- ❑ XML is a powerful data modeling tool with no limits on namespace or structural depths

Example of HTML

```
<h1>Rhubarb Cobbler</h1>
```

```
<h2>Maggie.Herrick@bbs.mhv.net</h2>
```

```
<h3>Wed, 14 Jun 95</h3>
```

Rhubarb Cobbler made with bananas as the main sweetener.
It was delicious. Basically it was

```
<table>
```

```
<tr><td> 2 1/2 cups <td> diced rhubarb (blanched with boiling water, drain)
```

```
<tr><td> 2 tablespoons <td> sugar
```

```
<tr><td> 2 <td> fairly ripe bananas sliced 1/4" round
```

```
<tr><td> 1/4 teaspoon <td> cinnamon
```

```
<tr><td> dash of <td> nutmeg
```

```
</table>
```

Combine all and use as cobbler, pie, or crisp.

Related recipes: [Garden Quiche](#GardenQuiche)

Example of XML

```
<recipe id="117" category="dessert">  
  <title>Rhubarb Cobbler</title>  
  <author><email>Maggie.Herrick@bbs.mhv.net</email></author>  
  <date>Wed, 14 Jun 95</date>  
  <description>  
    Rhubarb Cobbler made with bananas as the main sweetener. It was delicious.  
  </description>  
  <ingredients>  
    ...  
  </ingredients>  
  <preparation>  
    Combine all and use as cobbler, pie, or crisp.  
  </preparation>  
  <related url="#GardenQuiche">Garden Quiche</related>  
</recipe>
```

XML is “Syntax, not semantics”

- ❑ Tags have no predefined meaning
- ❑ Unlike HTML, XML by itself conveys only content and structure, not presentation, behavior, or meaning
- ❑ The meaning of XML languages must be specified outside of XML itself
 - Operational semantics: programs, servlets, applets, scripts, stylesheets,...
 - Definitional semantics: prose, namespaces, ontologies, UML diagrams,...

XML Applications

There are already hundreds of serious applications of XML.

XHTML

W3C's XMLization of HTML 4.0. Example XHTML document: `<?xml version="1.0" encoding="UTF-8"?>`

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head><title>Hello world!</title></head>
  <body><p>foobar</p></body>
</html>
```

CML

Chemical Markup Language. Example CML document snippet:

```
<molecule id="METHANOL">
  <atomArray>
    <stringArray builtin="elementType">C O H H H H</stringArray>
    <floatArray builtin="x3" units="pm">
      -0.748 0.558 -1.293 -1.263 -0.699 0.716
    </floatArray>
  </atomArray>
</molecule>
```

XML Applications (cont.)

TMML

Traffic Management Markup Language:

```
<?xml version="1.0"?>  
<tmml>  
  <network>  
    <nodes>...</nodes>  
    <links>  
      <segments>  
        <lanes>...</lanes>  
      </segments>  
    </links>  
  </network>  
</tmml>
```

What's an XML Document?

- ❑ A XML document is data that you can read

XML Documents

❑ Document-centric

- are characterized by **irregular structure**, larger grained data and lots of **mixed content**
- Generally for human consumption
- Email, books, invoice

❑ Data-centric

- are characterized by fairly **regular structure**, fine-grained data and little or **no mixed content**
- Generally documents/data for Machine exchange
- Sales order, Flight schedules

<Orders>

```

<SalesOrder SONumber="12345">
  <Customer CustNumber="543">
    <CustName>ABC Industries</CustName>
    <Street>123 Main St.</Street>
    <City>Chicago</City>
    <State>IL</State>
    <PostCode>60609</PostCode>
  </Customer>
  <OrderDate>981215</OrderDate>
  <Line LineNumber="1">
    <Part PartNumber="123">
      <Description>
        <p><b>Turkey wrench:</b><br />
        Stainless steel, one-piece construction,
        lifetime guarantee.</p>
      </Description>
      <Price>9.95</Price>
    </Part>
    <Quantity>10</Quantity>
  </Line>
  <Line LineNumber="2">
    <Part PartNumber="456">
      <Description>
        <p><b>Stuffing separator:</b><br />
        Aluminum, one-year guarantee.</p>
      </Description>
      <Price>13.27</Price>
    </Part>
    <Quantity>5</Quantity>
  </Line>
</SalesOrder>

```



KEY

Parent

Children

Grandchildren

Great Grandchildren

Great Great grandchildren

</Order>

Creating an XML Document

□ External/Internal Data Definition

- **DTD (Data Type Definition)**

- It defines the legal building blocks of a XML document
- Provides a common ground interchanging data
- Can be used for validating
- Can be written in Attributes or Elements

- **Schema**

- Extends DTD
- Offers more data types
- Offers extensible data types
- Written in XML

Creating an XML Document (cont.)

Element Type name='NETWORK'>

<complexType>

<sequence>

<Element ref='t:ID'/>

<Element ref='t:NAME'/>

<Element ref='t:CITY'/>

<Element ref='t:STATE'/>

<Element ref='t:DESCRIPTION'
minOccurs='0' maxOccurs='unbounded'/>

<Element ref='t:METRIC_UNITS'/>

<Element ref='t:LINK_COUNT'/>

<Element ref='t:ZONE_COUNT'/>

<Element ref='t:OD_PAIR_COUNT'/>

<Element
ref='t:SIGNALIZED_INTERSECTION'/>

<Element ref='t:NODES'
maxOccurs='unbounded'/>

<Element ref='t:LINKS'
maxOccurs='unbounded'/>

</sequence>

</complexType>

**!ELEMENT network (NAME CITY STATE DESCRIPTION
METRIC_UNITS LINK_COUNT ZONE_COUNT
OD_PAIR_COUNT SIGNALIZED_INTERSECTION NODES
LINKS)>**

<!ELEMENT NAME (#PCDATA)>

<!ELEMENT CITY (#PCDATA)>

<!ELEMENT STATE (#PCDATA)>

<!ELEMENT DESCRIPTION (#PCDATA)>

<!ELEMENT METRIC_UNITS EMPTY>

<!ELEMENT LINK_COUNT (#PCDATA)>

<!ELEMENT ZONE_COUNT (#PCDATA)>

<!ELEMENT OD_PAIR_COUNT (#PCDATA)>

<!ELEMENT SIGNALIZED_INTERSECTION (#PCDATA)>

Advantages of XML

- ❑ XML is a framework for developing unlimited number of special-purpose data languages for various industry (e.g. TrEPS)
- ❑ XML allows sharing of TrEPS data and work out an open solution to data exchange problem
 - Without interference from third parties
 - Without dependence on large software vendors
 - Without bindings to specific tools
 - Without language restrictions
 - In a way that lets anyone with a similar problem use the same solution

Advantages of XML (Contd.)

- Enables Separation of Data from Processing:
 - The XML approach for publishing data decouples data from processing
 - XML approach isolates changes in large systems, making them more flexible and reliable
 - A system based on XML makes it well suited to transaction processing in an heterogeneous, asynchronous, distributed environment (like the web and ITS)

Advantages of XML (Contd.)

- ❑ XML provides a standard framework for making agreements about communications
 - Transportation Industry - TrEPS DTDs
 - Transportation Industry - TrEPS schemas
 - Transportation Industry - TrEPS namespaces
- ❑ But we still have to make those agreements happen!!

My Work

- ❑ Analyze the application of XML to TrEPS
- ❑ Define the DTD and Schemas for TrEPS
 - Define DTD by elements
- ❑ Converted portions of TrEPS database to XML documents (Networks)
- ❑ Validated the documents against the DTD
 - Data is represented hierarchically as defined in the DTD
- ❑ Parsers
 - IE

XML Trade-off

- ❑ We give up
 - Performance
 - Centralized control
 - Uniformity
- ❑ In order to get
 - Persistence
 - Distributed control
 - Asynchronicity
 - A data structure that is obvious to both humans and machines
 - A certain kind of readability
 - Very low cost entry
 - Globalize the use of TrEPS generated data

Conclusions: XML Supports TrEPS Needs

- XML meets Data interoperability Requirements of TrEPS application
 - It has the capability to standardize TrEPS system generated data
 - Allows definition of data structures and content
 - It allows platform-independent data transfer
 - It allows seamless operation of other ITS sub-systems with TrEPS

Summary

- ❑ Learnt XML
- ❑ Created 2 different types of DTD & Schemas
- ❑ Used the TrEPS database to create the XML doc.
- ❑ Tested the codes

Next steps....

- ❑ Finish the schema with the data
- ❑ Test computational performance of creating, transferring and processing of XML documents
- ❑ Exploring XML family of tools
 - Databases, parsers, namespaces, DOM
- ❑ Converting legacy TrEPS data to XML Documents
- ❑ ...