Intelligent Correction and Validation Tool for XML

Abhishek Shivadas

Masters Project Defense

Jan 30, 2004

Committee:

Dr. Susan Gauch

Dr. John Gauch

Dr. Ed Meyen

What is it ?

Intelligent:

Deduces the right action to be performed on the right element

Correction:

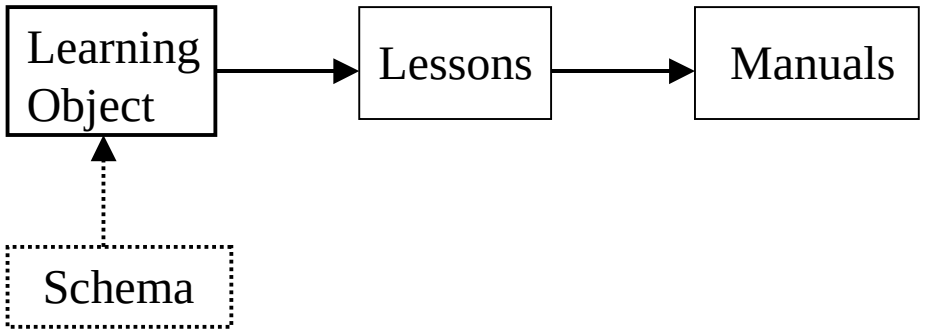
Corrects the incorrect constructs in the XML document.

Validation:

Makes sure that the XML conforms to the schema.

IKME – Create Learning Objects

- **Learning Objects provided by Jim Ritter.**
  - These Learning Objects conform to version 1.0
- **Created by the *creation tool*.**
  - The tool assumes that the schema will not change over the life of the learning objects.



But…*Changes ???*

Change is a fundamental aspect of data centric systems

What if there is …

- change in user requirements
- change in real world operation
- mistake in early design
- need for incremental maintenance

Kinds of changes

- Element Renames
- Order Changes
- Enumeration Changes
- Element Additions
- Element Deletions
- Attribute Additions
- Attribute Deletions
- Element moved from one parent to another.

How to bring Learning Objects up-to-date

Option 1: Manually effect the changes

A team member sat down and manually brought the documents up to date. This took him about a week to effect changes on 100 to 150 documents.

Option 2:

Design a software that would automatically identify the changes and correct the XML documents.

Goals

- Intelligently identify changes in schema.
- Suggest changes in the XML learning object.
- Perform changes.
- Develop a grammar that allows editing to be performed on XML documents.

Design Considerations

Implement the changes as primitives

Why Primitives ?

Many of the schema changes can be implemented in permutations of some of these primitives.

Design Considerations *continued*

- **Consistency Preserving**

These set of primitives do not affect the systems integrity in terms of well formedness or validity.

- **Minimize Data Loss**

The system is designed in such a way that the data loss in minimized.

- **Complete**

All the changes that have taken place since the inception of the project have been considered.

The Set of XML Data Change Primitives

	XML Data primitive	description
1	add-new(name, pos)	Add an empty element to position pos
2	move(pos1, pos2)	Move element form pos1 to pos2
3	delete(name)	Delete the element and its sub- elements from the XML document
4	delete-att(attname)	Delete the attribute attname from the element.
5	modify-att-value(new value)	Modify the contents of the attribute to new value.
6	modify-tag-value(new value)	Modify the contents of the tag to new value.
7	rename(old name, new name)	Rename the name of the element from old name to new name.

Design Considerations *control files*

Control File:

A file that is automatically generated when a learning object is validated by the system.

Advantages of having a control file:

- Allows the user to control the operation performed.
- Acts a like a script with the primitives being the commands.
- This feature of the system can be extended in future to accommodate batch processing.

Implementation

Software Requirements:

- Apache Server
- Perl V 5.0 and Later
- Additional Perl Modules

  XML::Twig

  CGI

The system is implemented using OOP in PERL and additional PERL modules such as XML::Twig and CGI

Implementation

The system is implemented in three phases.

Phase 1: (Generate Table)

- Breaks down the hierarchical structure of the schema into a flat comma delimited file.
- Repeated every time the schema gets changed

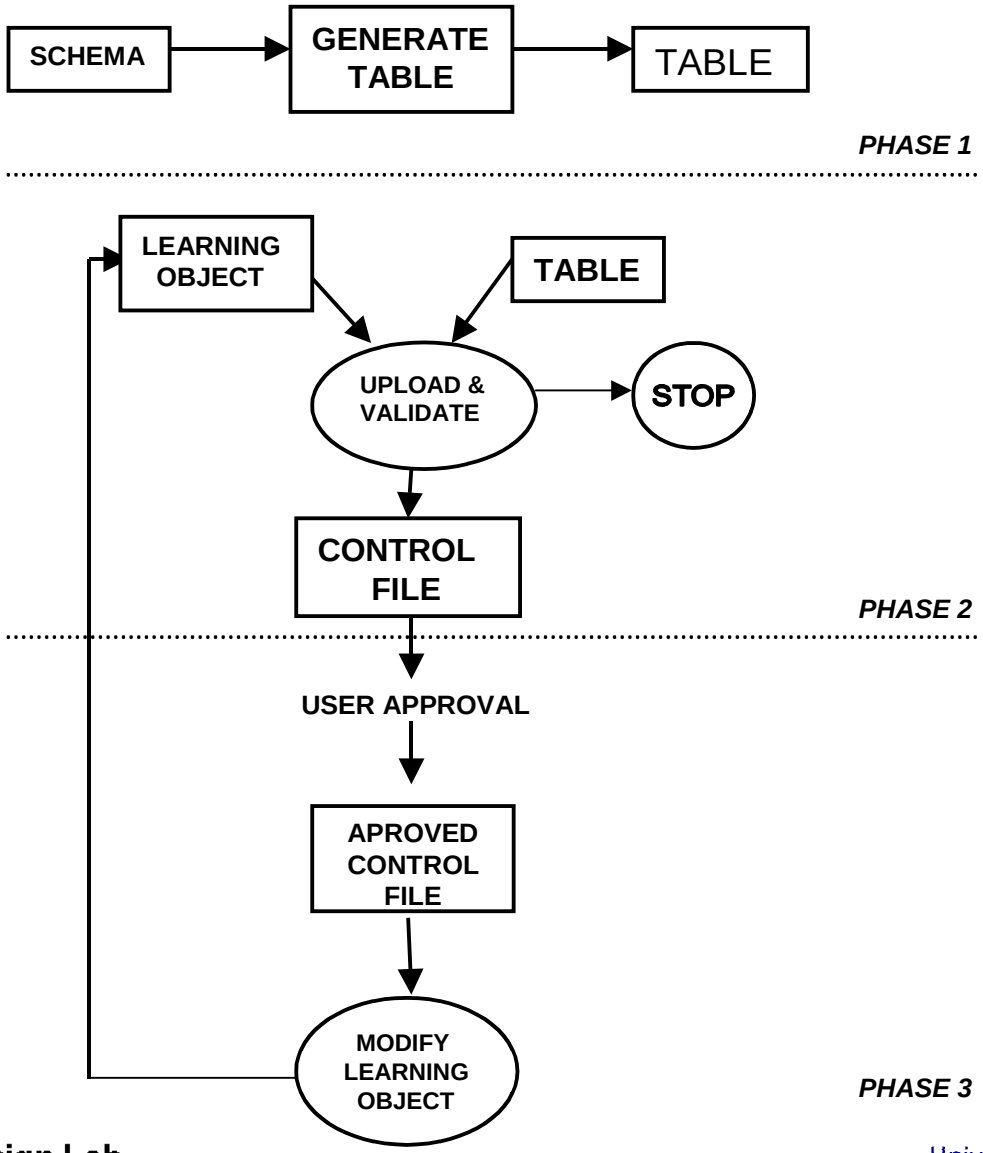
Phase 2: (Validate Learning Objects)

- Uploads the learning object.
- Validates the learning object
- Generates primitives.

Phase 3: (Modify Learning Objects)

- Modifies the learning object according to the control file.

System Design



Generate Table (Phase 1)

- Input: The schema provided by the client.
- Output: A comma delimited file.

The file contains an entry for each element along with all the "immediate" properties of the element.

The advantage of having such a table is that it can be hashed and an *O(1)* access can be performed to extract all the relevant details of the context tag.

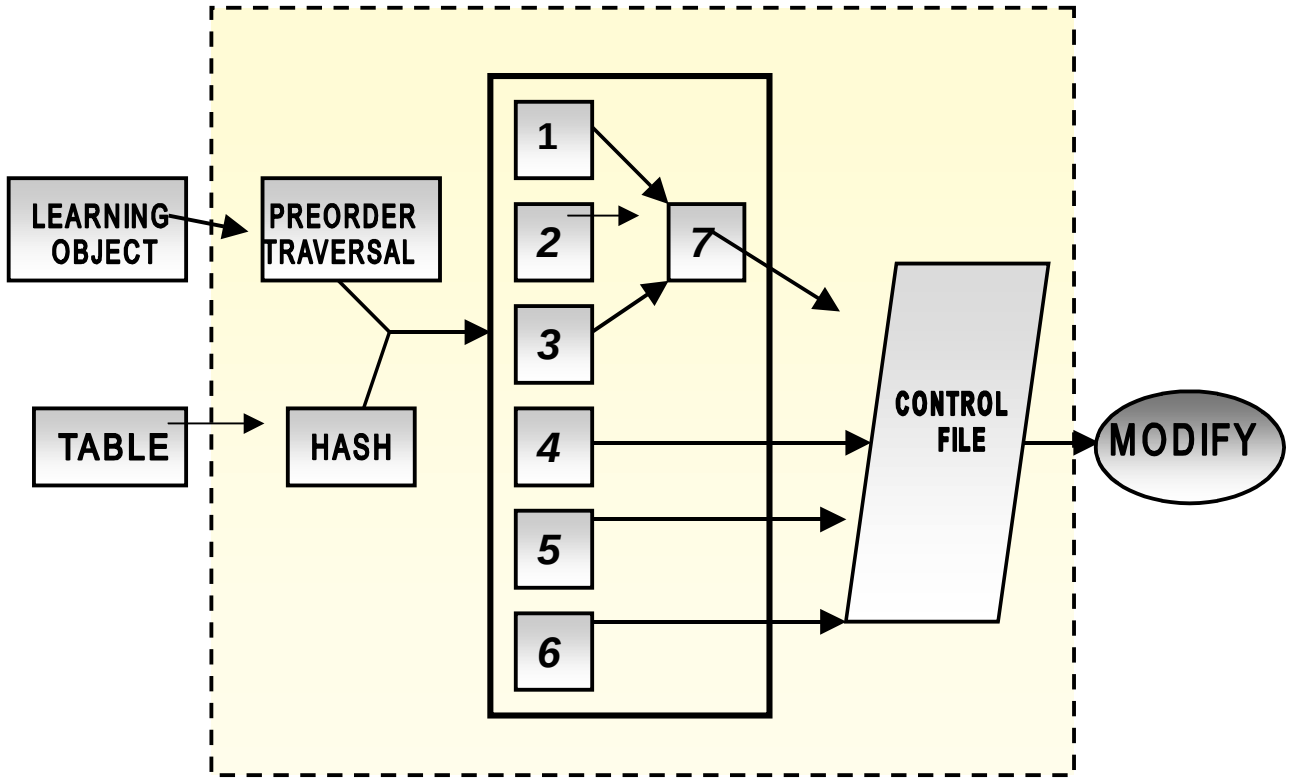
```
<xml version = “1.0” encoding = “UTF – 8”>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="object">
        <xs:element name = “metadata”>
          <xs:complextype>
           <xs:sequence>
              <xs:element name = “classificaton”>
                 <xs:simpleContent>
               <xs:extension base = “xs:string”>
                  <xs:attribute name = “type” use = “optional”>
                       <xs:simpleType>
                       <xs:restriction base = “xs:string”>
                           <xs:enumeration value= “Unclassified”/>
                               <xs:enumeration value =“Top Secret” />
                <xs:element name = “restriction” minOccurs =“0” maxOccurs = “unbounded”>
                       <xs:complexType>
                       <xs:simpleContent>
                               <xs:extension base=“xs:string”>
                                  <xs:attribute name = “type” use = “optional”/>
```

Path	Required	Enumeration	Min	Max	Sequence	Attribute	Bool
/object	metadata	-	1	1	metadata	-	f
/object/metadata	classification	-	1	1	classification restriction	-	f
/object/metadata/classification	-	Unclassified Top Secret	1	1	-	type	t
/object/metadata/restriction	-	-	0	u	-	type	f

Validate Learning Object (Phase II)

- Uploads the learning object into a working directory.
- Validates the learning object. Thus each element is checked if:
  - The element contains all the required sub tags.
  - The element has a legal enumeration value.
  - The element has occurred at least the minimum number of times.
  - The element is in the right position with respect to its parent.
  - The element has the right attribute.
  - The element does not occur more than the maximum number of times.
- Generates primitives if inconsistencies are found.
- The checks are done in sequence in a single pass. Therefore, occasionally a learning object will need multiple passes to fix all problems.

Block diagram



KEY:

1. _check_ required
2. _check_sequence
3. _check_min
4. _check_max
5. _check_attributes
6 _check_enumeration
7. _find_right_position

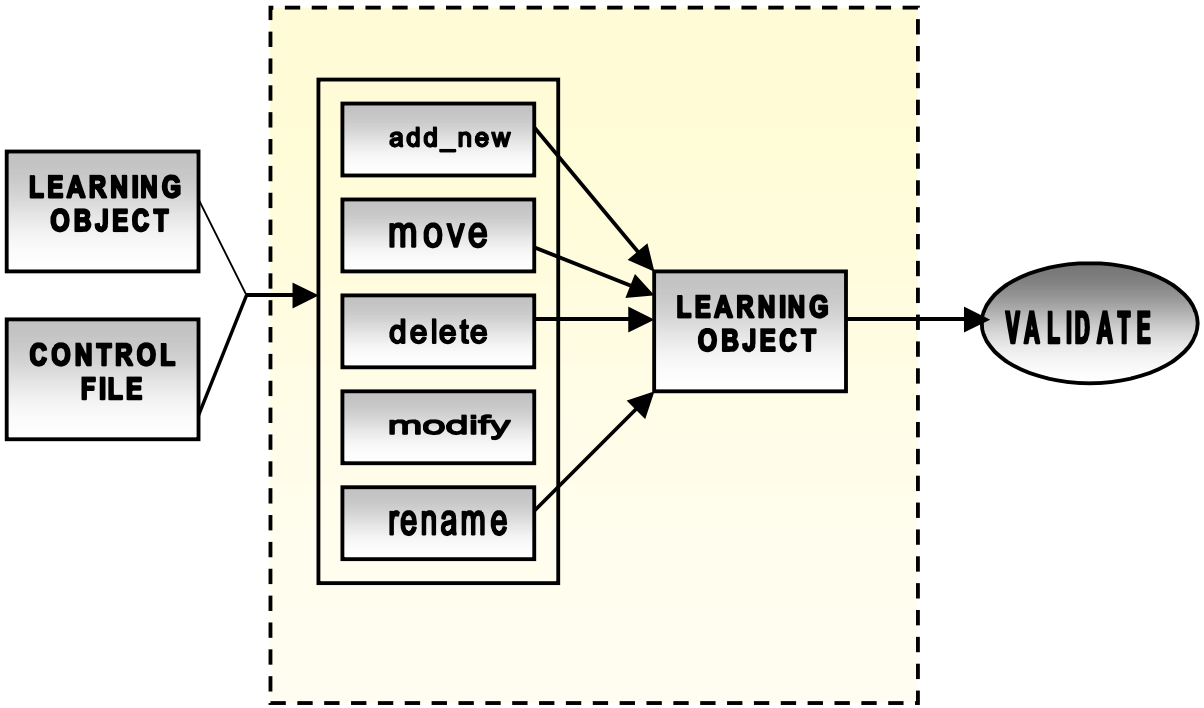
Modify Learning Objects

- Is called by *"Validate Learning Object"* when there are inconsistencies in the learning object that are to be corrected.

- The modules performs all the changes it is instructed to do by the control file.

- The module calls *Validate Learning Objects* once it completes.

Block Diagram



Evaluation

The system has the ability to identify changes in the schema and is also capable of performing modifications to the learning objects. The changes that the software is capable of identifying are:

	Type Of Change	**Primitives**
1	Adding a new tag	add-new
2	Deleting a tag	delete
3	Change the sequence of a tag	move
4	Modifying the contents of a tag	modify
5	Modifying the contents of an attribute	modify-att
6	Deleting an attribute	delete-att

Evaluation

The system cannot identify certain changes they are:

- When a tag is moved from a local parent to a different parent. These kind of moves are more than one level deep. Example

Evaluation *continued*

- The software cannot identify tag renames as well. The table below lists the type of changes that the software cannot identify automatically.

	Type Of Change	Primitives
1	Tag Renames	rename
2	Tag moves from one parent to another	move

Thus, apart from the table generated automatically, the users can create a table of changes they want, that is input to the system.

Evaluation – Test 1

Objective: The objective of test 1 is to assess how well the software performs when an invalid and almost zero length learning object is provided as input.

```
<?xml version="1.0" encoding ="UTF-8"?>
    < object>
        <metadata/>
        <content/>
    </object>
```

Evaluation Test 1 *continued*

The schema states that:

- The tag <*object*> has five required elements *metadata*, *tracking*, *views*, *reconstructions*, *content* in the same order
- The tag <*metadata*> has four required elements *security*, *source*, *description and applicability* in the same order.
- Therefore, we would expect the above statements to be present in the control file.

Evaluation – Sample Output 1

The following are the suggested changes in the Learning Object you wish to upload

	COMMAND	SOURCE	DESTINATION
☑	move	/object/metadata/description/topic/object_c	after /object/metadata/description/service
☑	add-new	/object/tracking	after /object/metadata
☑	add-new	/object/views	after /object/metadata
☑	add-new	/object/reconstructions	before /object/content
☑	add-new	/object/metadata/security	before /object/metadata/source
☑	add-new	/object/metadata/source	after /object/metadata/security
☑	add-new	/object/metadata/description	after /object/metadata/source
☑	add-new	/object/metadata/applicability	after /object/metadata/description
☑	add-new	/object/metadata/file	after /object/metadata/applicability

Correct them!

Internet

Evaluation – Test 2

Objective: The objective of Test 2 is to assess how the software performs when a learning object provided by the army is fed as input.

```
<?xml version="1.0" encoding="UTF-8"?>
<object type="Content - Doctrine" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <metadata>
                <security>
                        <portionmark>U</portionmark>
                        <classification type="US Security">Unclassified</classification>
                        <restriction type=""/>
                </security>
                <security>
                        <portionmark>U</portionmark>
                        <classification type="US Security">Unclassified</classification>
                        <restriction type=""/>
                </security>
                <header level="object">
                        <header/>
                        <title>Overview of the Urban Triad</title>
                        <subtitle/>
                </header>
                <description>
                        <obj_description/>
                        <service/>
                        <proponent>J7</proponent>
                        <doc_version>Version 1.0</doc_version>
```

Evaluation – Test 2 *continued*

The Schema states that:

- The tag <*security*> occurs only once in the schema.
- The tag <*classification*> under security should occur before the tag <*portionmark*>
- The tag <*doc_version*> does not occur in the schema.
- A Tag <*object_type*> is required before the tag <*proponent*>.

Evaluation – Sample Output 2

The following are the suggested changes in the Learning Object you wish to upload

	COMMAND	SOURCE	DESTINATION
☑	rename	/object/metadata/description/doc_version	/object/metadata/description/object_versio
☑	move	/object/metadata/description/topic/object_c	after /object/metadata/description/service
☑	rename	/object/tracking/action/docversion	/object/tracking/action/object_version
☑	rename	/object/tracking/action/status	/object/tracking/action/action_status
☑	delete-att	/object	xmlns:xsi
☑	move	/object/metadata/security[1]/portionmark	after /object/metadata/security[1]/classificat
☑	move	/object/metadata/security[1]/classification	before /object/metadata/security[1]/portionr
☑	modify-att-value	/object/metadata/security[1]/classification	
☑	delete	/object/metadata/security[2]	
☑	move	/object/metadata/security[2]/portionmark	after /object/metadata/security[2]/classificat
☑	move	/object/metadata/security[2]/classification	before /object/metadata/security[2]/portionr
☑	modify-att-value	/object/metadata/security[2]/classification	
☑	add-new	/object/metadata/description/object_type	after /object/metadata/description/proponer
☑	move	/object/metadata/description/proponent	after /object/metadata/description/service
☑	delete	/object/metadata/description/doc_version	
☑	move	/object/metadata/description/vital_record	before /object/metadata/description/rev_de
☑	move	/object/metadata/description/topic/topic_ac	after /object/metadata/description/topic/top
☑	delete	/object/metadata/description/topic/object_c	
☑	move	/object/metadata/description/topic/taxonpa	after /object/metadata/description/topic/top
☑	delete	/object/tracking/action/docversion	
☑	move	/object/tracking/action/actiondate	before /object/tracking/action/actor
☑	move	/object/tracking/action/actor	after /object/tracking/action/actiondate
☑	delete	/object/tracking/action/status	
☑	move	/object/tracking/action/actiondesc	after /object/tracking/action/actor
☑	move	/object/content/block/list	after /object/content/block/para[2]
☑	move	/object/content/block/para[2]	after /object/content/block/title

Correct them!

But Then….loss of data??

- The control file suggests the removal of *security[2]* and retain *security[1]*. What if the user wants it the other way??
- The control file suggest modifying the attribute value of <*classification* > to an empty string. What if the user wants to modify it to some other value?

The system is designed in a such a way that the user has the ultimate control of modifications performed in learning object. Therefore, an edit box is thrown up allowing the user to manually edit the learning object.

Sample Output 3 *The edit window*

eLearning Design Lab
Dole Center
University of Kansas

The changes have been made and are stored in a temporary file.

Click Here to view the changes in the file

Click here to hand edit the file

Hit the Button to save the changes in the original file

Save

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 3 U (http://www.xmlspy.com) by
(dole) --><object type="Content - Doctrine">
  <metadata>
    <security>
      <classification type="">Unclassified</classification>
      <portionmark>U</portionmark>
      <restriction type=""/>
    </security>
    <header level="object">
      <header/>
      <title>Overview of the Urban Triad</title>
      <subtitle/>
    </header>
    <source type="">
      <source_document>
        <header>This information was extracted from:</header>
        <title/>
        <publication>JP 3-06</publication>
        <edition/>
        <publication_date>May 2002</publication_date>
        <source_comment/>
      </source_document>
      <actual_event>
        <event/>
        <event_date/>
        <location/>
        <about_whom/>
        <event_comment/>
      </actual_event>
    </source>
```

Viewing Changes

The software is equipped with a tool that enables the user to view the changes that have taken place.

Future Work

- **n – gram analysis.**
  - Tag renaming?
  - Enumerations
- **Schema comparisons.**
  - Tag renaming?
  - Move statements
- **User History.**
  - Selective Delete
- **Batch processing.**
  - Migrating the whole collection