

Web-Based XML Diff Service Software Design Specification

Kaushal Mittal
(04329024)
Mtech IT , KReSIT

AnshuVeda
(04329022)
Mtech IT , KReSIT

October 18, 2004

Contents

1	Introduction	2
1.1	Purpose	2
1.2	Scope	2
1.3	Overview	2
2	System Architecture	2
2.1	Modules / Components	3
2.1.1	Web Server	4
2.1.2	Presentation Module	4
2.1.3	Delegation Module	4
2.1.4	Diff Module	4
2.1.5	Merge Module	5
2.2	Structure and RelationShips	5
3	Detailed Design	5
3.1	Presentation Module	6
3.2	Delegation module	7
3.3	Diff Module	7
3.3.1	XML Parser	7
3.3.2	XMLMetadata	8
3.3.3	DiffData	8
3.3.4	Filter	8
3.3.5	XMLDiff	8
3.3.6	OutputGenerator	8
3.4	Merge Module	9
4	Reusable Components	9
4.1	Languages and Tools	9
5	Algorithm	9
5.1	Literature survey	9
5.2	Algorithm	9
6	Design Issues and Considerations	11

1 Introduction

1.1 Purpose

To design and implement a web-based, configurable XML diff service for comparing and merging two XML documents.

1.2 Scope

Most of the diff tools with CVS etc. perform text based comparison, which is not suitable for XML documents. XML's have a document structure. It is desirable to have a tool that exploits this feature to represent the differences in two XML documents. We propose to implement an efficient diff-algorithm for the same. A Web interface is to be implemented to upload the XML files and select the settings for the type of differences to be reported. We also propose to implement an interface for merging the two XML documents. XML has become a standard for data representation in IT Industry making it worthwhile to implement such a service.

1.3 Overview

In section 2 we discuss the system architecture, brief overview of the modules and user interfaces. In Section 3 we discuss the detailed design of the system, with the clear relationship among the modules. In Section 4 we discuss the reusable components, the languages and technologies in use. In Section 5 we discuss briefly the algorithms that we are going to use. Section 6 discusses about the design assumptions.

2 System Architecture

The system is a web application. The Web Interface follows the MVC design Pattern i.e. Model View Controller Architecture. The MVC pattern is widely used in J2EE applications. JSP applications are easy to update, easy to break up (for the purpose of scaling), and easier to maintain with this method.

The view jsp or HTML will be used to get input from user which will be passed on to a controller jsp. The controller jsp will use a java bean and pass on the data to the Model java classes. The output will again be passed to the controller jsp which will invoke an appropriate view jsp for results.

This has been done so that the diff logic will totally be separate from the interface and can be re used or extended for other interfaces. The inputs from the clients are sent to the web server which hosts the controller JSP. This is delegated to the Diff or merge module based on the request.

The Client Access the application through the browser. So the browser represents the client. The interface to the client will be JSP.

2.1 Modules / Components

The system Comprises of four modules

1. WebServer Component
2. Presentation module
3. Delegation module

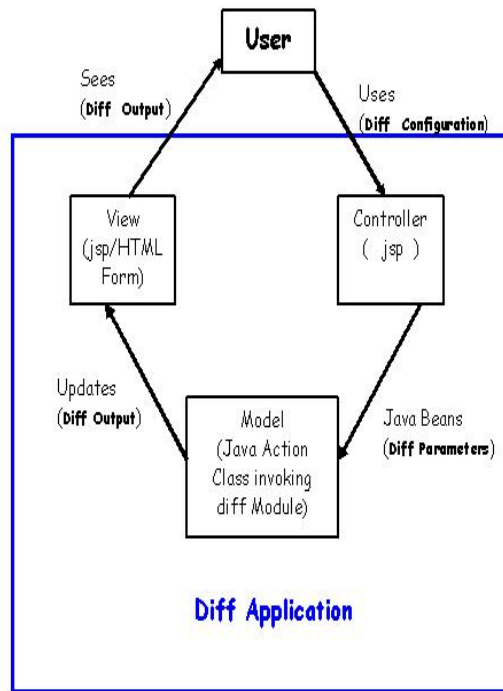


Figure 1: System Architecture

4. Diff module

5. Merge Module

2.1.1 Web Server

We are using the freely available version of Jakarta tomcat 4.0 as the web server. It hosts the presentation and the delegation module of the system.

2.1.2 Presentation Module

The presentation module comprises of the interface for the client. It consists of the JSP/HTML for taking input from the clients and displaying the output to the client.

The HTML form takes the input XML files from the user and filter settings/ Diff Configuration and passes the information to the delegation module. In the case of Merge files, the HTML form/JSP takes the base file and passes it to the delegation module.

2.1.3 Delegation Module

The Delegation Module comprises of the controller JSP and the Java Beans. Based on the input from the presentation layer a proper bean object is created and passed to the Diff module or merge module and get output from them and delegate it to the presentation module.

2.1.4 Diff Module

The diff Module is the major module of the system. It forms the backend of the system. It is implemented in Java. The diff modules consists of the diff algorithm, filter, output generator, XML parser, XMLMetadata, DiffData. We have designed our own algorithm for the diff. This recieves the filter settings and the XML files from the delegation module. The various components of the module are:

1. **XMLParser** - This component parses an XML and returns the DOM tree for the XML.
2. **XMLMetaData** - This component generates and provides the meta data for the XML's to be diff, for efficient diff algorithm.
3. **DiffData** - This component holds and provides access to the diff information that represents the differences in the two XML's.
4. **Filter** - This component represents the filter settings for the XML diff. The filter settings involves a set of flags for encapsulating the information about the type of the differences to be shown on the output. It provides access to this information to the diff algorithm and the output generator.
5. **OutputGenerator** - This component uses the diff data , and the filters to generate the HTML output for each of the XML for diff.
6. **DiffAlgorithm** - This is the major component of the diff module. It uses XML parser to generate the DOM representation of the XML files. It then uses the interface provided by the XMLMetaData component to generate the meta data for diff, and then finds the diff and generates the diff data. There is a strong cohesion between the diff algorithm XMLMetaData , DiffData and Filter component.

2.1.5 Merge Module

This module implements the merge algorithm. The input to this module is the name of the XML to be merged and the base file to be used for merging. It uses the diff data from the diff module and merges the file using the 3-Way merge algorithm. The 3-Way merge algorithm, merges the two files by taking the union of the insert and modify changes done in the two files and intersection of the delete changes done in the two files, then make these changes in the base file and returns this as the output.

There is a coupling between the Merge module and the diff module. It uses the diff data generated by the diff module for merging the files.

2.2 Structure and Relationships

As shown in the structure diagram 2.2 , The presentation module interacts with the delagation module and the client.The presentation module provides the interface for the client. The presentation module delegates the inputs from the client to the delegation module. The delegation module interacts with the diff module and the merge module.

The interaction within the diff module between the components of the diff modules are shown in fig.2.2. The various components of the diff module have an association relationship amongst them.

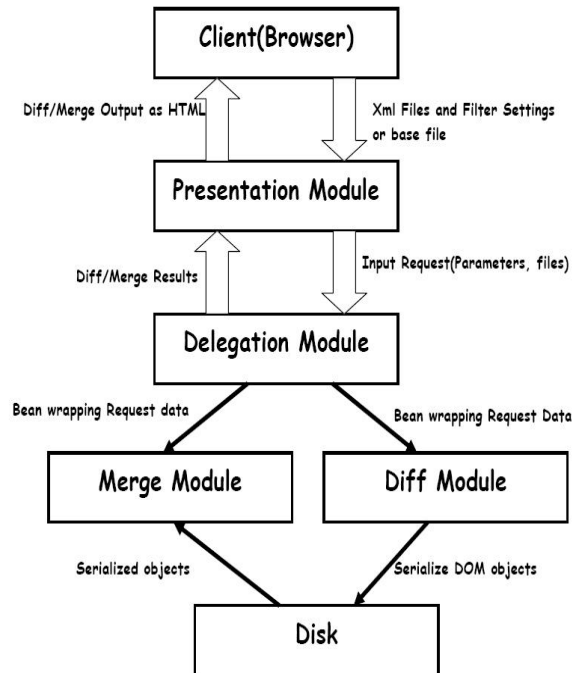


Figure 2: Structure Diagram

3 Detailed Design

In this section we present the detailed design of the various modules of the system.

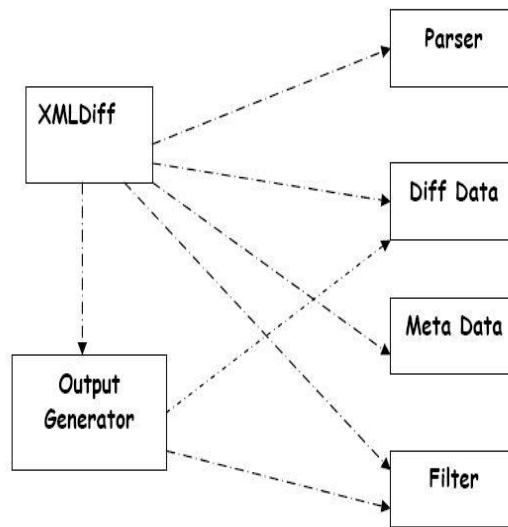
3.1 Presentation Module

This consists of HTML, JSP to provide the user interface. It needs to provide user interface for two types of requests, The diff requests and the merge requests. The user interface for the diff request will be a JSP generated HTML form. The user will first be provided with a form with :

1. Check boxes - Configuration/Filter Options
2. input - (File Type) - for uploading the XML files
3. input - (Button Type) - to submit the Request

The Filter Options includes the settings for the type of differences to be reported. The various options are :

1. Ignore Attribute difference
2. Ignore Text differences
3. Ignore Move differences
4. Ignore inserts in first Document



Diff Logic

Figure 3: Component Diagram - Diff Module

5. Ignore inserts in Second Document
6. Ignore Name Spaces Differences
7. Ignore Doctype Difference
8. Show All Differences

By default the 'Show all Differences' will be enabled. On user selecting any of the check boxes, the 'Show All Differences' check box will be deselected and disabled, and vice-versa. The submit button will lead to the submission of the request parameters along with the XML files to the controller JSP of the delegation module.

The generated HTML output will be presented to the user using the diff.jsp . The HTML output will show two XML files with each file in a scrollable frame. The differences in the XML will be shown by the different colors corresponding to the different types of the differences. The top frame will provide the mapping of the colors to the type of differences, in a tabular form. The bottom frame will give an option for merging the files. This will be a button that will delegate the request for merge to the controller JSP.

The interface for the merge option will be a HTML form generated by the JSP providing three options for selecting the base file for the merge algorithm.

1. HyperLink - file1- to select the first file as the base file.
2. HyperLink - file1- to select the Second file as the base file.
3. Input of File Type - to Browse and select the base file.

3.2 Delegation module

The Delegation Module comprises of the controller JSP and the java beans. This module separates the application logic completely from the front end interface. It wraps the user inputs received into java beans and passes on to the action classes of the logic. The output from the action class is again stored in a java bean. The controller jsp then forwards to a view jsp which uses the bean to generate the necessary mark up.

3.3 Diff Module

This is the main module of the system. The input to the module are filter settings, XML files. The output is the name of the HTML files for the two XML files. We have used the XML application framework for Java ,Dom4j for all kind of XML processing. There are significant advantages of using Dom4j. It provides random access to the nodes in the XML, It provides support for comparison and modification of XML documents. It is fast in Processing. The only limitation is that it creates a tree representation of the XML and so the Whole tree is to be in memory. This requires a lot of memory. But now a days the systems are no more having a hardware limitation of memory. This strengthened our motivation for using Dom4j. We have already listed the components of the diff module in Section 2.1.4. Here we present the detailed design of the various components of the diff module:

The various components of the diff module are as listed below :

3.3.1 XML Parser

This consist of a class *XMLParser*. It will be used by the diff algorithm to get the DOM Object corresponding to the XML file. This will be a static Class. All it methods will be static. It consist of a single method *XMLParser.getDocument(String filename)* - It returns the DOM Object for the XML file. In case of Parse Error , the return value will be *NULL*.

3.3.2 XMLMetadata

This will consist of following Classes -

1. *MetaData* - This will be a class maintaining the string representation of the hash key to be generated by the diff algorithm and the reference to the node for which the hash value has been generated. It also provides the getter and setter methods for the member variables.
2. *XMLMetaDataManipulator* - It is a class that maintains an array of XPATH's and the list of the *MetaData* objects corresponding to the nodes satisfying the XPATH. It provides method interface to create and insert the *MetaData*, sort the array on XPATH, get the list of *MetaData* Objects satisfying the XPATH.

3.3.3 DiffData

This is a class that maintains a hashmap on the *hashvalues* generated for the diff nodes, the type of the diff and the XPATH of the node. It also provides the access to the information of the diff corresponding to the input XPATH. It also consist of a public interface representing the type of diffs supported by the system.

3.3.4 Filter

This component consists of a class *XMLFilter* that maintains the information and provides the information about the XML differences to be reported or ignored. The user selected options on the HTML form get delegated to this via the Diff Algorithm. It initialises itself with this information and provides access to this information to the output generator and the diff algorithm components via the getter methods for the various fields. Few of the fields in the Class are - *ignoreAttributeDiff* , *ignoreDocTypeDiff*, *ignoreNameSpaceDiff* ,*IgnoreTextDiff* ,*IgnoreInsert_1* ,*ignoreInsert_2*. Few more can be added .

3.3.5 XMLDiff

It consists of a Class *XMLDiffMain* - that is the external interface of the module for other modules. It receives the name of the XML files as the input and the list of options for the filter configuration. It is associated to XMLFilter, XMLParser, XMLDiffAlgo class ,HTMLWriter class. The XMLDiffAlgo class takes the *Document* objects corresponding to the two XML's and generates the diff data for the differences between the two files. It is associated with the XMLFilter class, DiffData, XMLMetadata, XMLMetadataManipulator class. The algorithm for the diff has been designed by us and is presented in the coming sections.

3.3.6 OutputGenerator

This component will consist of two Classes -

1. *ColorMapper* - This class will keep the color mapping for the type of diff in the interface provided by the diffdata component. It maintains a Hashmap for this indexed on the type of Diff. The Mapping is loaded from a configuration file. In case the mapping is not provided for a diff type then by default it will be white background and blue foreground.
2. *HTMLWriter* - This will take the DOM Object as input and generate the html file representation for the XML with different color representation for the diff nodes. It interacts with the DiffData component of the Module. It uses the XMLtoHTMLConverter and ColorMapper class to generate the HTML code for the nodes.
3. *XMLtoHTMLConverter* -This class takes in the XMLRepresentation of a node as String and returns the HTML representation for the node.

3.4 Merge Module

This module consist of a class *Merge* that will use the 3-Way merge algorithm, to merge the two files. It takes in the DOM of the two input files and the name of the basefile. If the base file is different it creates a DOM model for it. The algorithm simply takes the union of inserts and modification in the two files, and the intersection of the deletes in the two files, using the DiffData. Then it simply makes these changes in the basefile. It uses the OutputGenerator to generate the merge Output as an HTML file. If the base file is one of the two files the changes reduce down to making insert and modifications in the second file to the file treated as a basefile.

4 Reusable Components

The use of MVC architecture has separated the XMLDiff module. The web interface can be replaced by the GUI interface and the diff module can be plugged into it to get the GUI based

XMLDiff product. The development of diff module involves the development of reusable components like XMLtoHTMLConverter. The Dom4j application framework for XML processing in Java is a third party component that we are using for all kind of XML processing in the system.

4.1 Languages and Tools

1. Programming Language - Java , j2sdk1.4.2.03
2. WebServer - jakarta-tomcat 5.x
3. Application FrameWork - Dom4j
4. Other Tools - JSP, HTML, XML, XPATH, JS.
5. OperatingSystem - Linux Redhat 9.0 and Compatible Versions. , (Support for Windows.)

5 Algorithm

In this section we discuss the literature survey done by us for the various algorithms and the research work done by others for XML diff. We propose our algorithm that takes a complexity of $O(N \log N + cM^2)$, where N is the number of nodes in the tree representation of the XML and M represents the maximum number of Nodes that have the same XPath at any level of the tree.

5.1 Literature survey

5.2 Algorithm

Our Algorithm detects movement of a node at any of the sibling's place at the same level. The pseudocode for the Algorithm that we are using for Diff is as follows:

1. Traverse the document of both files to generate the (*XPath, listof(Hashvalue, node(Referencetoanode))*) data for each file. This involves visiting each node of the tree representation of the XML's once. The complexity of the step is $O(N)$.
2. Sort the array containing the above information on XPATH . This takes $O(N \log N)$ worst case complexity assuming each node is a separate XPATH.
3. for each XPath , get the list of (HashValue, Nodes) pair corresponding to the nodes in the tree representation of the XML's. Let the list be A for XML1 and B for XML2 .
 - (a) Take the intersection of the A and B based on the hash value.
 - (b) If a hashvalue found in both the XML, compare the Nodes . If the Node matches delete them from the lists A and B . This results in ignoring the nodes that matches at a particular level.
 - (c) For remaining nodes
 - if moved node differences are to be ignored
 - then
 - for each node in the list A
 - compare it to node in List B .
 - If Match found
 - deletethem from the list A nad B .
 - This results in removal of moved nodes.

```

else
  for each node in the list A
    compare it to nodes in List B .
    If Match found
      delete them from the list A and B .
      This results in removal of moved nodes.
      Insert a Diff of Moved type in the Diff Data

```

- (d) For remaining nodes - get the parent hash by loosing the Last character in the hash value. for each node with same parent hash in the list B, compare the nodes and report the diff in the Diff data, and delete the nodes from the List A and B.
- (e) for remaining nodes in the List A, insert diff of insert type for XML1, and For remaining nodes in List B insert the diff of insert type for XML2.

The Algorithm for generating the hashvalue and metadata is -

```

generateMetadata( Element Node , hashvalue)

```

```

  count = 0 ;

```

```

  for each child

```

```

    append count to hashvalue ;

```

```

    get XPath

```

```

    generate MetaData( childnode , hashvalue.clone) ;

```

```

    insert Xpath, hashvalue, reference to node in the list,

```

```

    using insert (hashvalue, reference to node) in the list corresponding to the

```

XPATH given.

The algorithm for generating the output is - For each node in XML, get the hashvalue from the Meta data. if the hashvalue is in the DiffData, then generate the HTML output in the color mapped to the diff type. And if not then simply generate the HTML representation of the XML.

6 Design Issues and Considerations

The design phase for the project has ended up in the design of the algorithm for diff and the web interface for the application. The implementation phase may result in small modification to the design. The assumptions made for the application is that the input XML's are already validated against the DTD. Even if they are not validated they can be given as input but our application will not perform the validation of the XML's.