WEBVIGIL: MONITORING MULTIPLE WEB PAGES AND PRESENTATION OF XML PAGES

Sharavan Chamakura, Alpa Sachde, Sharma Chakravarthy and Akshaya Arora
Information Technology Laboratory
Computer Science and Engineering Department
The University of Texas, Arlington, TX 76019
{chamakura, sachde, sharma, aarora}@cse.uta.edu

Abstract

In the case of large-scale distributed environments such as the Internet, users are interested in monitoring changes to a particular web page (XML or HTML). There are many instances in which the change detection is required at a finer granularity, such as changes to links, images, phrases or keywords in a page. WebVigiL is a general-purpose, change monitoring, notification and presentation system for web pages. It handles specification, management, and propagation of customized changes as requested by a user in an efficient way. When the user specifies an URL, it may contain frames instead of a single content page; this entails additional pages to be fetched from the frames for detecting changes. In this paper we propose different approaches for detecting changes over pages containing frames and generalizing it in terms of depth. We also address the case of monitoring multiple web pages for different change types with a single request. As there are no presentation semantics and tags, manual identification of the detected changes is difficult on a XML web page. Once detected, changes have to be notified to the user in a meaningful manner. In this paper we also propose various presentation schemes to display the changes computed on XML pages.

Keywords

Change Detection, Multiple Web Pages, Presentation.

1. Introduction

The World Wide Web has outdone traditional media such as television, to become an indispensable source of information. There is data related to each and every context on the Web, and this data is increasing at a rapid pace. This plethora of information often leads to situations wherein users looking for specific pieces of information are flooded with irrelevant data. For example, an investment banking professional who is only interested in news related to his areas of interest, is flooded with news on many topics when he goes to any popular news website. In some cases users are concerned in knowing about the updates happening to a particular web page of their interest such as the keywords, links or other change types. For example, students might want to know about the updates that are made to the course websites regarding any new projects or assignments. There may be some situations where users are interested in tracking changes across multiple web pages. For example, a user of a discussion board related to technical topics might be interested in knowing about posts being made regarding a particular set of topics. In this case, the user wants to be notified of the changes related to various topics present on different web pages. In general, the ability to specify changes to arbitrary documents and get notified in different ways will be useful for reducing the inefficient navigation.

WebVigiL [1-3] is a change detection and notification system developed to provide users with the ability to monitor web pages (XML/HTML) according to a profile. Users’ place a request by giving a set of specifications like the page to be monitored, the type of change to be checked for, when to fetch the page and how to notify the changes. The user request is in the form of a sentinel that is used for change detection and presentation. The changes are detected and stored in the change repository. The stored changes are then notified to the user. Figure 1 summarizes the high level architecture of WebVigiL [1]. Information from the sentinel is extracted, validated and stored in a knowledge base and is used by other modules in the
system. The presentation module provides the various presentation schemes for detected changes to present them in a well-organized and readable format. The fetch module is responsible for fetching the pages in an efficient/optimal manner for detecting changes. The version control module stores different versions of a page until it is no longer needed for detecting changes and for notification of changes. The change detection module interacts with various modules for detecting changes in an efficient manner and placing the results of changes in the knowledge for notification by the presentation/notification module.

Many a times, users are interested in monitoring multiple URLs or even entire web sites. Users may place a sentinel on a page without realizing that it is a frame and not an ordinary page. In this case, we want the WebVigiL system to transparently process the page with frames taking the user intent into account. Support for monitoring multiple pages in general allows us to treat frames as a special case of multiple pages. We use the notion of depth to specify how many levels one needs to consider from the root.

This paper extends upon our previous work in customized content monitoring of web pages. The contributions of this paper are: a) the improved Change Detection Graph (CDG) which supports the monitoring of multiple URLs for either similar or different change types. As a subset, this includes the monitoring of web pages containing frames, b) Presentation schemes for the semi structured XML pages. The rest of the paper is organized as follows. Section 2 discusses the overview of the related work. Section 3 and 4 will discuss, respectively, a mechanism to monitor multiple URLs, which includes the class of web pages with frames and the schemes for presentation of detected changes for XML webpage. Section 5 provides the conclusion and future work.

2. Related Work

Most of the research has been done in change detection systems for web pages. In WebVigiL the change detection is based on an expressive user specification.

Delta XML [4] developed by Mosnell provides a plug-in solution for detecting and displaying changes to contents between two versions of an XML document. They represent changes in a merged delta file [5] by adding additional attributes such as insert = delete to the original XML document.
resolving the issues for presentation of the detected changes in a legible format.

The Current WebCQ [10] system supports change detection over various primitive change types such as links, images, keywords, lists, and tables but there is no provision to give composite change requests. Also, the current WebCQ system does not support the monitoring of multiple web pages with a single request. Users cannot combine multiple URLs with different change types. It provides the interface to display the web page specified by the user, if the web page contains frames user have the flexibility to select the particular frame for the change detection, if the frame is not selected then the change will only be detected on the main webpage. The base page can be assumed to be present at level zero; all the pages pointed by the base page at level one and so on. If there are links on the frames WebCQ cannot monitor changes on the web page corresponding to that links means monitoring the web pages in depth is not possible but in WebVigiL the user only specifies the URL and the changes are detected on all the frames if exists of the webpage specified by the user. Unlike WebCQ, WebVigiL is capable of detecting changes to one level in depth of a web page.

3. Monitoring Multiple Web Pages

Our earlier work [14] addressed monitoring individual web pages (both HTML and XML) using a change detection graph (CDG). However, as many pages contain frames, a general change detection approach for multiple pages and frames is needed.

3.1. Change Detection Graph

We need a data structure that will allow us to asynchronously feed fetched pages for change detection, allow parallelism where possible, optimize the computation by grouping sentinels over URL’s and change types, facilitate composite change detection [1, 14]. In order to achieve the above functionalities we construct a graph. This graph is referred to as the change detection graph (CDG). Consider the scenario where sentinel S1 on www.uta.edu monitors composite changes to Links AND Images, another sentinel S2 on www.uta.edu monitors Links and there is one more sentinel S3 on www.uta.edu monitors Images. The change detection graph constructed for sentinels S1, S2 and S3 are shown is Figure 2.

![Change Detection Graph](image)

The different types of nodes in the CDG are:
1. URL node: A URL node is a leaf node that denotes the page being monitored.
2. Change type node: All level 1 nodes are said to be change type (links, keywords, images, phrases) nodes. These nodes perform the change detection over the respective types.
3. Composite node: A Composite node represents a combination of change types. All nodes above level 1 belong to this type.

The URL nodes are notified regularly after fetching the required pages with the specified periodicity. To reduce redundant fetching, a page is fetched only if it’s last modified time (LMT) or the checksum of its content changes. When a page has changed, it is propagated to all the different change type nodes corresponding to it. The change type nodes perform the action of detecting changes according to their properties. For example, a links node detects changes happening to any of the links on that page. If a change type node detects a change, it propagates the detected change to the sentinel monitoring that particular page. The sentinel then notifies the user of the detected change. If there is a composite node involving the change type node, the change type node propagates the page to the composite node. In the graph, notification mechanism is used to capture the relationship between the nodes. The URL node contains the list of references of the change types. All the higher-level nodes contain the list of interested sentinels.

3.2. Frames

A frame is a tag used in html pages to display more than one web page in the same browser
window. Each HTML document is called a frame, and each frame is independent of the others. The frame tag specifies the actual location of the constituent pages and also the layout of the pages.

The main web page just acts as a placeholder for the links to the actual web pages that constitute the page. In the current system, when a web page is specified for monitoring, the fetching of the web page is done by sending a request and retrieving it. There is no processing done to determine the type of content the web page contains before the fetching starts. This leads to detecting no changes at all when a request is made to monitor a web page containing frames as the frame references do not change. When the user places a request to monitor a page containing frames, the URL part of the request will contain the address of the base page that holds the frames. As a result of this, the system will monitor the base page for changes, which does not truly reflect the user intent. The user's intent is to monitor the constituent pages, which are present as frames. This problem can be addressed using the approach presented in this paper by extracting the different constituent pages from the base page, and placing individual requests on each page. But, this will not truly reflect the user intent and will make the management of the current change detection graph very difficult.

3.3. Types Of Monitoring

Monitoring of multiple web pages for changes can be classified into the following two categories:

3.3.1. Assorted Monitoring

There might be cases in which users might want to monitor multiple web pages for changes using a single request. For example, a user might be interested in knowing about changes to links on http://www.cnn.com and changes to keywords “rangers” on http://www.espn.com. The user might want to be notified only when both the pages are updated with the corresponding changes. These requests map to monitoring multiple web pages for different change types. The requirements for an efficient mechanism for assorted monitoring are as follows:

1. Users should be able to specify any number of URLs and any change type for each URL.

2. Users should be able to combine the URL and change type pairs using the operators OR, AND and NOT.

3. For every request, only one set of ECA rules should be created to start and stop the sentinel.

4. For every request, only one CDG should be constructed, with multiple leaf nodes.

5. Fetching of all the specified URLs should be done using only one fetch rule.

6. A request should have only one notification mechanism for every change.

Based on the above requirements, whenever a request for assorted monitoring is received, only one set of ECA rules are generated using the existing mechanism, by treating the first URL in the request as the base page. The way the CDG is constructed for these requests is illustrated in Figure 3.

3.3.2. Linked Monitoring

There are many instances in which information present on a web page is categorized into different sections and each section is placed on a different page. In such cases, all the different pages are linked from one main page. These are the cases in which users might be interested in monitoring multiple web pages by giving a single request on the page that links all the pages. For example, for the website like http://itlab.uta.edu, there will be many different sections on different web pages but all the sections will be linked from the home page. A user might be interested in monitoring posts made on different sections of such a web site. In this case, the base page can be assumed to be at a depth 0 and all the pages linked from it to be at a depth 1. For web pages containing frames, the different pages present...
as frames are automatically mapped to depth of 1. When a sentinel with a request for linked monitoring is received, all the pages till the given depth are crawled and the required URLs extracted. The crawler needs to handle circular references wherein a link on a web page points to a web page that has already been crawled. This might result in an infinite loop if not handled properly. In the WebVigiL system, a breadth first search is used to crawl the required pages. Starting from the base page, all the pages at level 1 are crawled, followed by all the pages at level 2 and so on (checking and avoiding circular references).

The construction of CDG for these requests can be done using the following two approaches:

3.4. Only-OR Approach

Web pages containing frames can be treated as a special case of linked monitoring where the user does not provide any depth but a default depth of 1 is assumed. Therefore, requests on web pages with frames are converted to requests for linked monitoring with a depth of 1 so that the base page and all the pages pointed by the frames can be monitored. The mapping of frames to multi-URLs is explained using Figure 4. It shows the web page corresponding to the URL http://itlab.uta.edu. The actual base page that represents this web page is main.html having the URL http://itlab.uta.edu/main.html. The main page consists of two frames, which contain the pages contents.html and home.html having the URLs http://itlab.uta.edu/contents.html and http://itlab.uta.edu/home.html, respectively. A request to monitor this URL actually maps to monitoring main.html, contents.html and home.html. Therefore, whenever a change happens to any of these three pages, the sentinel should be notified.

To handle requests on multi-URLs, one approach for constructing the CDG is to use many ‘OR’ nodes. This is illustrated in Figure 5. To construct the CDG using only ‘OR’ nodes, two URLs are first joined using a binary ‘OR’ node. The output from this node is combined with the next URL node using another ‘OR’ node. This process is repeated for all the URLs present in the specification. The number of OR nodes is one less than the number of pages in the frame. With this framework, whenever a change is detected for a URL, it is propagated to the higher-level ‘OR’ node corresponding to it. The propagation is repeatedly done till the notification reaches the node at the topmost level, at which point the sentinel is notified of the changes. The problems associated with this approach are discussed below:

1. Graph Size: With the only-OR approach, the size of the CDG greatly increases. The number of ‘OR’ nodes and the number of dummy sentinels required will be directly proportional to the number of URLs that are being monitored.
2. Change propagation time: The time required to propagate the detected changes to the sentinel will increase as the size of the graph increases. The propagation time is proportional to the depth of the CDG, which in turn is proportional to the number of URLs being monitored.
3. Modification complexity: The CDG has to dynamically adjust based on the changing links in web pages for multi-URL requests. If new links are added, additional nodes should be added for monitoring and if links are deleted, the corresponding URL nodes have to be deleted. The complexity of modifying the CDG using only ‘OR’ nodes is very high.
4. If the user is interested in knowing the depth of the URL where a change is detected, it is
not possible to know that with the binary OR approach in a straightforward manner. As the depth of a web page is relative to the base page for every request, the depth information should be maintained in an external source and accessed while notification. To overcome the problems associated with this approach, a new operator is proposed in the next section.

3.5. Extended-OR (EOR) Approach

The ‘extended-OR’ (EOR) is an N-ary operator designed to efficiently handle multi-URL requests. The design details of the ‘EOR’ node are presented in Figure 6. The binary operator ‘OR’ has two child nodes but the number of child nodes an ‘EOR’ can have is variable. The semantics of the ‘EOR’ node is the same as that of a binary ‘OR’ node in terms of propagation of changes. Whenever a child node propagates a change, the ‘EOR’ node notifies the sentinel of the change. To easily perform the action of adding and deleting sentinels to the EOR node, it maintains all the child sentinels as a linked list. The EOR node also consists of a hash table, which has a mapping from the URL of every child to its corresponding relative level. For example, the entry in the hash table for the base page will be 0, while the entries for pages linked from the base page will be 1. When a linked monitoring request is given the base page is parsed till the required depth and all the required URLs are extracted. The traversal is done in a breadth-first manner to maintain the information regarding the relative depth of every URL. Each of the extracted URLs is checked to see whether or not a URL node corresponding to that URL already exists (to avoid traversing cyclic references).

After creating all the change type nodes, the list of URLs nodes and the corresponding level information is used to create the EOR node. Finally, the main sentinel is inserted into the EOR node, which in turn is cloned and inserted at every child node of the EOR node.

The advantages of the EOR node with respect to change detection and propagation are:

1. Using EOR nodes for graph construction results in a smaller CDG than the one resulting with binary OR operators. There is only one EOR node created for every
request and there are no dummy sentinels with a request.

2. There is no delay involved in the propagation of detected changes with EOR nodes. Any change that is detected gets propagated through only one level and the sentinel is notified.

3. Since all the child nodes are stored in a vector at the EOR node, addition and deletion is done in a constant time with simple operations.

4. The EOR node maintains the relative depth information for all the child sentinels, which makes it easy to propagate this information to the sentinel.

4. Change presentation for XML

In this section we describe the issues related to the presentation of detected changes on XML pages and proposed two schemes for the same with their advantages and limitations.

XML is gaining popularity to become the new standard for semi-structured data exchange over the Internet. With XML, it is possible to deliver higher-level services, which were being difficult to support through current HTML technology. The self-descriptive characteristic of XML gives us some useful information on the semantics of the document and enables us to detect changes at a finer granularity level.

In WebVigiL, we are interested in detecting content changes and not structural changes. The assumption here is that the structure does not change that often. The emphasis on the content is based on the observation that users do not see the structure (or not aware of the structure) and are interested in monitoring changes to contents. Change presentation is the last phase of web monitoring where the detected changes are presented to the user. The presentation method selected should clearly show the detected differences between two web pages to the user. Therefore, computing and displaying the detected differences in a meaningful format is very important.

XML does not contain presentation tags whereas in HTML the data is embedded in the presentation tags [11]. Unlike HTML, the element names in XML have no presentation semantics. Unless associated with a presentation style, the web browsers would display the XML document in its source format, which is difficult for a common user to understand. Hence, the presentation of an XML document is dependent on a stylesheet. HTML browsers are hard-coded. Some browsers do base their formatting on Cascading Style Sheets (CSS), but they still contain hard-coded conventions for documents which do not provide a stylesheet.

For presentation, XML data can be stored inside HTML pages as "Data Islands", where HTML is used only for formatting and displaying the data. The standard stylesheet language for XML documents is the Extendible Stylesheet Language (XSL). It provides a comprehensive model and vocabulary for writing the stylesheets in XML syntax. Stylesheets are used to express how the content of XML should be presented. XSLT (XSL Transformation) accepts the XML document and the XSL for that document and presents it based on the intention of the designer of that stylesheet.

An XML document fetched from the Internet for monitoring may or may not have a stylesheet linked to it or the stylesheet may not even be publicly accessible, making it difficult to be fetched along with the XML document. In such cases, we would have to create a new style-sheet for presentation. Even if the stylesheet is available, we need to modify the stylesheet to highlight the detected changes.

As an example, in Figure 8, the node "J.K.Rowling" appears twice for the same context. If only one of the nodes results in either an insert or delete, it would be difficult to identify and highlight that particular node without modifying the existing structure.

XML does not contain presentation tags whereas in HTML the data is embedded in the presentation tags [11]. Unlike HTML, the element names in XML have no presentation semantics. Unless associated with a presentation style, the web browsers would display the XML document in its source format, which is difficult for a common user to understand. Hence, the presentation of an XML document is dependent on a stylesheet. HTML browsers are hard-coded. Some browsers do base their formatting on Cascading Style Sheets (CSS), but they still contain hard-coded conventions for documents which do not provide a stylesheet.

For presentation, XML data can be stored inside HTML pages as "Data Islands", where HTML is used only for formatting and displaying the data. The standard stylesheet language for XML documents is the Extendible Stylesheet Language (XSL). It provides a comprehensive model and vocabulary for writing the stylesheets in XML syntax. Stylesheets are used to express how the content of XML should be presented. XSLT (XSL Transformation) accepts the XML document and the XSL for that document and presents it based on the intention of the designer of that stylesheet.

An XML document fetched from the Internet for monitoring may or may not have a stylesheet linked to it or the stylesheet may not even be publicly accessible, making it difficult to be fetched along with the XML document. In such cases, we would have to create a new style-sheet for presentation. Even if the stylesheet is available, we need to modify the stylesheet to highlight the detected changes.

As an example, in Figure 8, the node "J.K.Rowling" appears twice for the same context. If only one of the nodes results in either an insert or delete, it would be difficult to identify and highlight that particular node without modifying the existing structure.

XML does not contain presentation tags whereas in HTML the data is embedded in the presentation tags [11]. Unlike HTML, the element names in XML have no presentation semantics. Unless associated with a presentation style, the web browsers would display the XML document in its source format, which is difficult for a common user to understand. Hence, the presentation of an XML document is dependent on a stylesheet. HTML browsers are hard-coded. Some browsers do base their formatting on Cascading Style Sheets (CSS), but they still contain hard-coded conventions for documents which do not provide a stylesheet.

For presentation, XML data can be stored inside HTML pages as "Data Islands", where HTML is used only for formatting and displaying the data. The standard stylesheet language for XML documents is the Extendible Stylesheet Language (XSL). It provides a comprehensive model and vocabulary for writing the stylesheets in XML syntax. Stylesheets are used to express how the content of XML should be presented. XSLT (XSL Transformation) accepts the XML document and the XSL for that document and presents it based on the intention of the designer of that stylesheet.

An XML document fetched from the Internet for monitoring may or may not have a stylesheet linked to it or the stylesheet may not even be publicly accessible, making it difficult to be fetched along with the XML document. In such cases, we would have to create a new style-sheet for presentation. Even if the stylesheet is available, we need to modify the stylesheet to highlight the detected changes.

As an example, in Figure 8, the node "J.K.Rowling" appears twice for the same context. If only one of the nodes results in either an insert or delete, it would be difficult to identify and highlight that particular node without modifying the existing structure.

XML does not contain presentation tags whereas in HTML the data is embedded in the presentation tags [11]. Unlike HTML, the element names in XML have no presentation semantics. Unless associated with a presentation style, the web browsers would display the XML document in its source format, which is difficult for a common user to understand. Hence, the presentation of an XML document is dependent on a stylesheet. HTML browsers are hard-coded. Some browsers do base their formatting on Cascading Style Sheets (CSS), but they still contain hard-coded conventions for documents which do not provide a stylesheet.

For presentation, XML data can be stored inside HTML pages as "Data Islands", where HTML is used only for formatting and displaying the data. The standard stylesheet language for XML documents is the Extendible Stylesheet Language (XSL). It provides a comprehensive model and vocabulary for writing the stylesheets in XML syntax. Stylesheets are used to express how the content of XML should be presented. XSLT (XSL Transformation) accepts the XML document and the XSL for that document and presents it based on the intention of the designer of that stylesheet.

An XML document fetched from the Internet for monitoring may or may not have a stylesheet linked to it or the stylesheet may not even be publicly accessible, making it difficult to be fetched along with the XML document. In such cases, we would have to create a new style-sheet for presentation. Even if the stylesheet is available, we need to modify the stylesheet to highlight the detected changes.

As an example, in Figure 8, the node "J.K.Rowling" appears twice for the same context. If only one of the nodes results in either an insert or delete, it would be difficult to identify and highlight that particular node without modifying the existing structure.

XML does not contain presentation tags whereas in HTML the data is embedded in the presentation tags [11]. Unlike HTML, the element names in XML have no presentation semantics. Unless associated with a presentation style, the web browsers would display the XML document in its source format, which is difficult for a common user to understand. Hence, the presentation of an XML document is dependent on a stylesheet. HTML browsers are hard-coded. Some browsers do base their formatting on Cascading Style Sheets (CSS), but they still contain hard-coded conventions for documents which do not provide a stylesheet.

For presentation, XML data can be stored inside HTML pages as "Data Islands", where HTML is used only for formatting and displaying the data. The standard stylesheet language for XML documents is the Extendible Stylesheet Language (XSL). It provides a comprehensive model and vocabulary for writing the stylesheets in XML syntax. Stylesheets are used to express how the content of XML should be presented. XSLT (XSL Transformation) accepts the XML document and the XSL for that document and presents it based on the intention of the designer of that stylesheet.

An XML document fetched from the Internet for monitoring may or may not have a stylesheet linked to it or the stylesheet may not even be publicly accessible, making it difficult to be fetched along with the XML document. In such cases, we would have to create a new style-sheet for presentation. Even if the stylesheet is available, we need to modify the stylesheet to highlight the detected changes.

As an example, in Figure 8, the node "J.K.Rowling" appears twice for the same context. If only one of the nodes results in either an insert or delete, it would be difficult to identify and highlight that particular node without modifying the existing structure.
notification of these changes may have to be sent to different devices that have different storage and communication bandwidths. At present, we are proposing various schemes to present the changes for a meaningful interpretation by the user. The various schemes [12] and its advantages and disadvantages are discussed below:

4.1. Only Change Scheme

In this scheme only the changes are presented. The traditional method would be a tabular structure with the types of changes (insert/delete/move) as different columns of the table. We can also present the changes by displaying the XPath expressions of the nodes that have changes. Here we would just need one stylesheet for any XML document. This scheme does not involve going through the contents of the files associated with the changes.

We need to store the whole object of change detected, which might be the list of links, or keywords based on the user specification. As the number of elements (list of links or keywords etc.) in the change object cannot be determined we store the whole object using a BLOB (Binary Large Objects), which makes it simpler. In order to insert the changes in database we convert the change object in the binary format using ObjectOutputStream Java API. In the next step empty BLOB object is created. Then this empty BLOB object is retrieved and updated with the binary form of change object. The changes are then retrieved from the lists and placed in HTML tables. The presentation is generated on the fly when requested by the user.

<table>
<thead>
<tr>
<th>Entry</th>
<th>OldCount</th>
<th>NewCount</th>
<th>Change</th>
<th>Signature [Path]</th>
</tr>
</thead>
<tbody>
<tr>
<td>TV</td>
<td>0</td>
<td>1</td>
<td>Insert</td>
<td>Electronic/Products/NewItem</td>
</tr>
<tr>
<td>DVD</td>
<td>1</td>
<td>0</td>
<td>Delete</td>
<td>Electronic/Products/NewItem</td>
</tr>
</tbody>
</table>

Figure 9: Only Change Scheme

This scheme will be advantageous for large documents and documents where the number of changes is also large. Only Change scheme will be a better scheme to have for devices such as PDA’s where there is a constraint on the amount of data transmitted over a limited bandwidth. But it may be somewhat difficult for the user to interpret the changes when the changes are more. Figure 9 shows the only change scheme of presentation.

Here only the changes are presented as some content is being inserted and some is removed.

4.2. Dual Frame Scheme

In this scheme we show both the documents side by side to highlight the changes. This scheme has an advantage over the other scheme of being easy to interpret. But here we need our own stylesheets or embed them in HTML as data islands for us to highlight or strike out the changed contents which will not have the same look and feel as the original document. This scheme is difficult to implement but very appealing from the user point of view.

Changes detected for XML pages are stored in ChangeList. The insert, delete and move lists in ChangeList are retrieved for the change that has to be presented. Insert list-containing objects (links, images, words, phrases) that are inserted. Delete list contain the objects that are deleted. Move list contain objects that are moved. A primitive change is associated with a single ChangeList whereas a composite (for e.g., LINKS AND IMAGES) is associated with multiple sets of ChangeList, that is, the collection of the ChangeList of all the constituent events. This collection is called ListOfChangeLists.

The associated version information is obtained as well. The old and new versions of the XML page are parsed using the Xerces Parser [13] and converted to a DOM [8] tree for traversing through it to locate the content that has changed. As the changes inserted in the lists are sorted in the order of their position in the tree (old or new), the tree is traversed only once to cover all the elements in the list. As the tree is traversed, for each node the position and signature in the tree is computed. The list is traversed to find a match until a position greater than the computed position is found. If a match is found a node for the type of change (insert/delete/move) is inserted between the leaf and the parent. Rather than traversing through the list of changes for each node, it is traversed only until the position of the node in the list is equal or greater than that computed while traversal. Once both (old and new) the trees are traversed and the changes have been made, an XSL is used to render them.

Figure 12 shows the dual frame scheme of presentation of two XML files, which are shown in Figures 10 and Figure 11. Here we assume that the user is interested in the keywords “midnight” and “December” in the
sample versions of an XML page shown in Figure 10 and Figure 11. When the versions are given to change detection module, we derive that the keyword “midnight” is deleted from the old version and “December” is inserted in the new version. As discussed we also get the signature and the position of the nodes that had the above mentioned changes in the XML tree. The signature of the node where “midnight” is deleted is “POEM-STANZA-VERSE” and the position is “4”. Similarly, in case of an insert of “December” the signature and position of the node containing the word is “POEM-STANZA-VERSE” and “12” respectively in the XML tree. Hence, while traversing the old XML tree we insert a node called “delete” as a child of the node that contains “midnight”.

```
<xml version="1.0"/>
<POEM>
  <TITLE>The Raven</TITLE>
  <AUTHOR>Edgar Allan Poe</AUTHOR>
  <DATE>1845</DATE>
  <STANZA>
    <VERSE>Once upon a midnight dreary...<VERSE>
    <VERSE>Over many a quizzing, and curious volume of forgotten lore...<VERSE>
    <VERSE>And I nodded, nearly napping, suddenly there came a tapping...<VERSE>
    <VERSE>Awoke some visitor, I answered, “tapping at my chamber door”...<VERSE>
    <VERSE>Only this, and nothing more...<VERSE>
  </STANZA>
</POEM>
```

Figure 10: Old Version of XML file involved in Change Detection

Similarly, we insert a node called “insert” as a child of the node containing “December” in the new XML tree. The old and new XML trees with the “insert” and “delete” nodes are printed into temporary XML files. Both the temporary XML files are then given to the XSLT (XML Stylesheet Language Transformer) along with the generic XSL (XML Stylesheet Language) to display the XML with the highlighted nodes.

Complete nodes containing the change are highlighted, as finding the words in the node would be an additional overhead in case of large number of changes. Duplicate nodes are also handled, as we not only take into account the signature but also the position of the node in the XML tree involved in a change. In case of phrases the tree is realigned by the change detection algorithm, so the position and signature of the changed nodes with respect to the realigned tree is given as output. Hence it is not possible to traverse the tree formed by the original files and find the exact position and signature of change.

```
<xml version="1.0"/>
<POEM>
  <TITLE>The Raven</TITLE>
  <AUTHOR>Edgar Allan Poe</AUTHOR>
  <DATE>1845</DATE>
  <STANZA>
    <VERSE>Once upon a midnight...<VERSE>
    <VERSE>Over many a quizzing...<VERSE>
    <VERSE>Nearly napping...<VERSE>
    <VERSE>And I nodded...<VERSE>
    <VERSE>Awoke some visitor...<VERSE>
    <VERSE>Answered, “tapping at my chamber door”...<VERSE>
    <VERSE>Only this, and nothing more...<VERSE>
  </STANZA>
</POEM>
```

Figure 11: New Version of XML file involved in Change Detection
5. Conclusion & Future Work

WebVigiL is a change monitoring system for the web that supports specification, management of sentinels, change detection for HTML and XML pages, and provides presentation of detected changes in many ways. The presentation module handles the presentation of changes in a manner that is easy for the users to decipher the changes. The schemes used for the presentation of XML web pages are: Change-Only Scheme and Dual-Frame Scheme. The manner in which the presentations for HTML and XML are made is totally different as the tags in HTML are used for presentation and the tags in XML define the content. The current WebVigiL system is skilful in detecting changes over frames which is a subset of detecting the change on a website or multiple URLs and also capable in monitoring to level one in depth of a web page using different approaches. Extended – OR (EOR) approach is used by the system which overcomes the limitations of Only OR approach which is proposed initially.

There are some issues, which still need to be addressed. The presentation of images is not handled currently as the images are not being fetched and stored by the system. The presentation of images will pose many problems that need to be investigated. The storage of images for every version of the page and then taking care of deletion will have a significant overhead on the system.

6. Acknowledgements

This work was supported, in part, by the Office of Naval Research & the SPAWAR System Center–San Diego & by the Rome Laboratory (grant F30602-01-2-05430), and by NSF (grant IIS-0123730).

7. References