Change Detection in XML Trees: a Survey

Luuk Peters
University of Twente
Faculty of Electrical Engineering, Mathematics and Computer Science
l.j.peters@student.utwente.nl

ABSTRACT
XML change detection has become more important in many areas of application. XML change detection algorithms provide users and applications with a more meaningful description of detected changes. Each XML change detection algorithm focuses on its own aspect of change detection and therefore has its own properties and optimizations. We provide descriptions of change detection algorithms for XML documents and describe certain types of properties that play a part in XML change detection algorithms. This paper also presents some examples of XML change detection applied in the area of merging, versioning and synchronization of XML documents. We try to explain which types properties should be considered more important when speed, correctness or minimization is the primary requirement.

Keywords
XML, change detection, tree matching.

1. INTRODUCTION
Extensible Markup Language (XML) [W3C98] is developed from Standardized Generalized Markup Language (SGML) [SGML] and was formally introduced in 1998. XML has become a standard on the web and is still gaining popularity. It is used in more and more applications as syntax for document publishing, data-exchange, enterprise integration and many more purposes.

In general, users are more interested in the changes of a document than the current version. For that purpose, Unix [UNIX], Linux [LINUX] and other Unix based systems provide us with the *diff* command [DIFF]. Diff is able to compare and output the differences between two files. Like similar programs, diff is designed to find the differences in text files and handles those files as a series of lines. Although it can find the differences in XML documents, it doesn't handle it like a tree-structured document. When XML documents are handled like tree-structured documents, more information can be gathered about differences.

In this paper we will use the term 'changes' next to 'differences', because we regard the term 'changes' to be the same as 'the differences between original and altered files'. 'Changes' are also called 'operations' when used in merging trees, in terms of 'execute the insert operation'.

3rd Twente Student Conference on IT , Enschede June, 2005.

When XML became popular and a widely used format, a renewed interest arose for computing changes in tree-structured documents. Tools were developed for change detection in tree-structured documents such as XML documents. In this paper we will discuss a selection of such tools.

As Cobéna et al. [CAM02] states, change detection "is essential for a number of applications ranging from traditional support for versions and temporal queries, to index maintenance or support for query subscriptions". Each change detection algorithm has its own advantages, its own optimizations and its own requirements. This paper tries to clarify when and in which context a certain algorithm is to be used.

This paper is structured as follows. Section 2 explains about change detection, section 3 presents some example of XML change detection applied in the area of merging, versioning and synchronization of XML documents, section 4 provides a description of each algorithm, section 5 explains about the different properties of XML change detection algorithms, section 6 provides an overview of algorithms and some of their properties. Finally, the paper will end with a conclusion containing the findings of this paper by answering the following research questions:

1. *What XML change detection algorithms exist?*
2. *What are the general properties of those algorithms?*
3. *Which properties should be considered more important when the primary requirement is speed, correctness or minimization?*

2. CHANGE DETECTION
Line based differencing compares two text based files. The algorithm finds sequences of lines common to both files, interspersed with groups of differing lines. XML diff will consider the xml (text based) file as a hierarchical structured data tree.

When considering XML documents as a structured data tree, two trees can be compared and equal nodes and equal subtrees in both XML trees can be matched. The nodes and subtrees that aren't matched are different between the two XML trees. There is a lot more to say about the context of these differences (or changes when one XML tree is a newer version of the other).

```
<example id="1">
  <name>Example one</name>
  <description>This is an example</description>
</example>
```

```
<example id="1">
  <name>Example number one</name>
  <description>This is an example</description>
</example>
```

Figure 1 Difference example, two versions of XML tree

Figure 1 describes two versions of a XML tree. The second version contains an extra word 'number'. Line based difference algorithms consider this as a changed line. When the file doesn't contain line brakes, and there is a change present, not only the line, but the whole file is considered as 'changed'. But when parsing the file as a XML tree, the algorithm only finds the `<name>` node changed. When the file contains more `<example>` nodes, each having a `<name>` node, the algorithm can use a sort of path specification for identifying the `<name>` node location.

Changes are often gathered in a delta script or delta file. The delta script contains a representation of those changes. The purpose of the delta script is that it can be used to patch a file, in other words, to apply the changes and recreate the changed file again. This is very useful when the original version is present on another location and needs to be updated without transferring the whole changed file. The other party can recreate the changed file using only the original file and the delta script.

The changes are also called operations and the most common operation types are 'insert' and 'delete'. Most algorithms specify 'updates', which can also be specified as a combination of one 'delete' and one 'insert'. Less common is 'move', which, like 'update', can also be specified as a combination of one 'delete' and one 'insert'. The advantage of both 'update' and 'move' is that they lower the size of the delta script, while the use of 'delete' and 'insert' instead would potentially lead to a large resulting delta script. The use of an operation type 'copy' would lead to an even smaller delta script, but is almost never used. When a match is found for a certain node, the algorithm can remove it from any further comparing. But if 'copy' is supported, the algorithm needs the node for comparing when trying to detect copies of the same node.

3. APPLICATIONS

When documents are saved as XML documents and used in situations where more than one version is produced, the need arises for efficient change detection algorithms that provide more information about the changes and merge algorithms that keep the XML syntax intact. This section describes some applications where such algorithms specifically designed for XML are very useful.

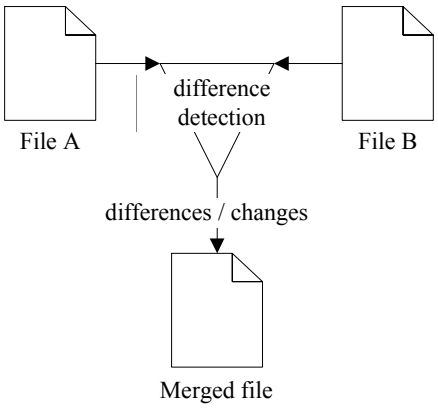

Figure 2 Two-way merge principle

3.1 Two-way merge

In parallel modifying XML documents the need for reintegration of the copies into a single XML document arises. With two-way merge, there is no notion of file versions, only the differences between two files that need to be merged are detected, as illustrated in Figure 2.

Usually in two-way merge algorithms, the user has to check most differences before they are applied, for example when deleting a part of file A, the merge algorithm will regard the same part in file B as an addition; therefore the deletion won't reach the merged file unless the user interferes.

3.2 Three-way merge

Three-way merging is a variant of merging where two replicas of a document exist that are independently edited. As Figure 3 illustrates, together with the required original document and both modified replicas, three-way merge can reintegrate the documents.

As previously described, a three-way merge tool always needs three files, the original file which is concurrently edited, and the resulting two independently modified files. The original file is first compared with one of the modified files. The differences, or changes, are gathered in a delta file which describes how the original file should be altered to create the modified file. Then the original file is compared with the second modified files, and the resulting delta file is combined with the first delta file. The combined delta file is then applied to the original file resulting in a merged file containing all modifications from both modified files.

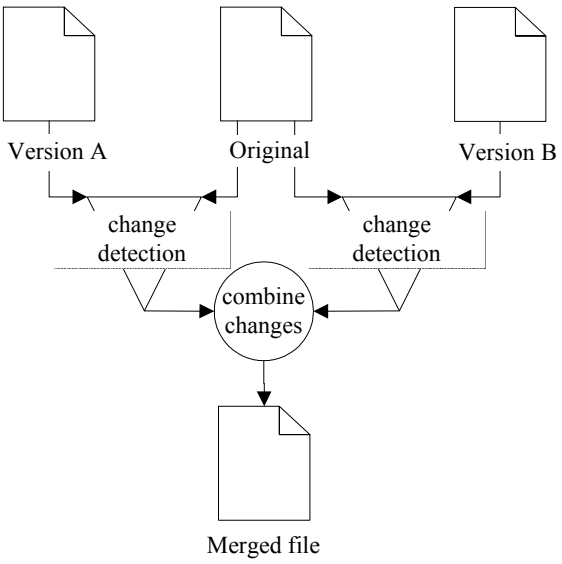

Figure 3 Three-way merge principle

Because the tools contain change detection algorithms, this paper describes some three-way merge tools in section 4. Lindholm [TL01, TL04] describes his 3DM and some scenario's in which his method would be very useful. Another merge tool is DeltaXML [MEL03, RLF03], a commercial tool, capable of both two-way and three-way merge of XML documents.

3.3 XML document versioning

File merging is used in Concurrent Versions Systems [CVS], which keeps track of file revisions and versions. Version management systems, like CVS, offer the possibility to 'go back in time' and retrieve a previous version of a certain file. But CVS and other line-based methods for version management are not efficient for structured documents like XML.

Chien et al. [CTZ01] have proposed a technique called Usefulness Based Copy Control (UBCC), which is more efficient in version reconstruction while it uses small storage overhead.

Xyleme Server [Xyleme], a native XML repository which solves the critical issues regarding storage, retrieval, and distribution of highly variable semi-structured content, is a product of the Xyleme project [ACFR01]. According to the website, Xyleme Server provides scalability, query precision, flexibility, content-centric and open architecture.

A version management system usually keeps track of changes, by means of comparing previous version with the edited version, using a change detection algorithm and need to be merged in case of multiple versions of the same file, concurrently edited by multiple users.

3.4 Website content versioning

WebDAV (Web-based Distributed Authoring and Versioning), created by the WebDAV Working Group [DAV], is a set of HTTP protocol extensions, which allows users to collaboratively edit and manage files on remote web servers.

WebDAV allows the users to perform remote web authoring operations including creation, editing, copying, moving and deleting of web resources. WebDAV also allows the users to exclusive lock or shared lock a web resource. In case of shared locking, the simultaneously edited files need to be merged.

3.5 Synchronization in mobile environment

Tancred Lindholm [TL03] gives two examples of three-way merge is a mobile environment; "the directory tree synchronization logic in a mobile file system" and "a simple shared photo library".

```
<?xml version="1.0"?>
<tree id="1">
   <directory name="tma" id="01">
   <directory name="fig" id="10">
       <file name="simple-ex.tiff" id="21" />
       <file name="photolib.tiff" id="23" />
   </directory>
       <file name="paper.lyx" id="11" />
   </directory>
   <directory name="src" id="22">
       <file name="History.java" id="08" />
   </directory>
</tree>
```

Figure 4 Lindholm's example of a directory tree

The Fuego Core mobile file system [FCP] is the first example. The file system uses Lindholm's 3DM merge algorithm and expresses the directory hierarchy as an XML document. When synchronizing the file system, a three-way merge will be performed on the local, the remote directory and the unified directory hierarchy using the 3DM merge algorithm.

The second example is a photo library applet where photos are stored in an XML index file containing metadata like: title, keywords, etc. Because the index file is stored on the mobile file system being developed in the Fuego Core research project [FCP], the index file is automatically ready to be concurrently edited. Figure 4 shows an example of such a XML index file.

3.6 Query systems

Wang et al. [WDC03] give us two examples of query system where a change detection tool can be very useful:

- Incremental Query Evaluation. With a repeated query the query system only needs to re-evaluate the query on the delta data, thus, the user doesn't receive old results.
- Trigger Condition Evaluation. A change detection tool can quickly report changes which correspond to a certain trigger condition.

4. CHANGE DETECTION ALGORITHMS

This section shows an overview of different change detection algorithms and their properties. Some algorithm have a clear connection to each other, they are used as an example for other algorithms or have been improved or adapted by other algorithms.

4.1 LaDiff

LaDiff [CRGW96] is a structural differencing tool for hierarchical structured information. As input, it takes two versions of a LaTeX [LaTeX] document and produces a LaTeX document with the changes marked. It uses an algorithm called *FastMatch*, which uses a function *equal* to compare nodes in an ordered tree.

FastMatch uses longest common subsequence computations to perform initial matching of nodes that appear in the same order, starting from the leaves of the document. According to Cobéna et al [CAM02] its cost is in $O(n*e+e^2)$, where n is the number of leaf nodes and e the sum of the number of deleted and inserted subtrees and the total size of subtrees that moved for the shortest edit script.

LaDiff handles updates, inserts, deletes and even moves. But LaDiff only supports certain LaTeX elements, therefore its *FastMatch* cannot be used as an algorithm for change detection in XML documents, but its underlying algorithms have inspired others in creating their own algorithms.

4.2 MH-Diff

MH-Diff [CGM97] can be considered as the counterpart of LaDiff and is an algorithm for *meaningful* change detection between *hierarchical* structured data trees. Its goal is to portray the changes between two trees in a succinct and descriptive way. It uses an edit script that gives the sequence of operations needed to transform the original tree into the new tree.

Besides updates, inserts and deletes MH-Diff handles moves and even copies. The copy and move operations result in an even higher quality edit script, especially when the copied or moved subtree is large.

MH-Diff is an algorithm that is very different from LaDiff [CRGW96], it is based on transforming the problem to the edge cover problem. It has, according to its authors a heuristic solution in worst-case $O(n^3)$ time, where n is the number of nodes, and an average of $O(n^2)$ time. Xu et al. [XWW02] state that MH-Diff has a time complexity of $O(n^2 \log n)$.

4.3 XMLTreeDiff

XMLTreeDiff [IBM98] is a set of Java beans, allows both command line and program access, but has retired Nov. 1998. Difference reports are presented as XML as well. XML.com [XTI98] and Kyriakos Komvoteas [KK03] provide us with a description of the algorithm.

XMLTreeDiff is designed to perform fast differentiation and update of DOM (Document Object Model) structures. Although DOM is simple to use, it is relatively slow and cannot handle large documents.

Differences are expressed as change (update), delete and insert operations. Unlike LaDiff [CRGW96] and MH-Diff [CGM97], XMLTreeDiff treats moves as a pair of delete and insert.

XML.com implies that the cost of XMLTreeDiff is at least in $O(n^2)$ time, where n is the number of tree nodes. XMLTreeDiff uses an optimal tree-differentiating algorithm together with a fast subtree matching procedure.

4.4 MMDiff and XMDiff

Sudarshan S. Chawathe [Cha99] presents "an external-memory algorithm for computing a minimum-cost edit script between two rooted, ordered, labeled trees". The algorithm XMDiff is based on the main memory version MMDiff and constructs a matrix based on the "string edit problem". MMDiff has quadratic CPU and memory cost, XMDiff has reduced memory usage, but quadratic IO costs.

Cobéna et al. [CAH02] conclude in their work that XMDiff seems to be a good choice when the requirements are pure minimality of the result, but at high computation cost.

4.5 IBM's XML Diff and Merge Tool

The XML Diff and Merge Tool is created by IBM [IBM01], but only implements basic functionality. It's a Java program, which compares a base XML document, and another XML document and doesn't support three-way merge. The tool points out the differences by use of symbols and color.

Differences can be walked through and a decision can be made whether or not to include the difference, thus, XML Diff and Merge is unable to automatically merge XML documents.

Kyriakos Komvoteas [KK03] points out that XML Diff and Merge Tool is incapable of always producing correct result, because the developers made the inaccurate assumption that the node is free of conflicts if all children of the same node have no conflicts.

4.6 3DM's matching algorithm

Lindholm's master's thesis about three-way merge [TL01], describes the 3DM merging and differencing tool for XML, which contains a XML tree matching algorithm that can be considered as a change detection algorithm. The algorithm only handles ordered tree.

Lindholm publishes his work on three-way merge for XML documents in 2004 [TL04]. He plans to extend the algorithm to handle unordered trees as well. Lindholm points out that "the issue of constructing matchings efficiently and accurately is of large practical importance when implementing XML merging for documents without unique element identifiers".

The operations supported by 3DM matching algorithm are delete, insert, update, move and copy. Lindholm has constructed the matching by heuristically matching nodes one-to-one between T_1 and T_0, as well as between T_2 and T_0, where T_0 is the original XML document and T_1 and T_2 are the concurrently edited versions of the XML document, and then trivially extending the matching to satisfy a number of conditions.

Lindholm also applies his 3DM tool in a mobile environment [TL03] where it offers a flexible way for an application to implements XML merge capabilities.

4.7 XyDiff

XyDiff [CAM02] is a tool for differencing ordered XML trees. It focuses on improving time and memory management and does so by using a very efficient algorithm that identifies large subtrees that are left unchanged between two versions of the XML document. The work of Cobéna et al. is motivated by the Xyleme project [ACFR01], in which a prototype is developed of a dynamic warehouse for XML data. XyDiff is used in the resulting Xyleme server [Xyleme] to store the changes made to XML documents as separate XML delta files.

The XyDiff algorithm computes hash values and weights for every node in the XML trees of both XML documents. It gives each node in the original XML document a unique identifier, XID. The *XidMap* identification technique used gives the list of all persistent identifiers in the XML document in the prefix order of nodes. XyDiff uses the XyDelta format [MACM01], a single XML document containing all *forward completed deltas*, to save the detected changes. But, because XyDelta uses unique identifiers, the changes do not contain context information about the changes.

XyDiff then compares the signatures of two nodes and if they are equal, the nodes are matched. A priority queue accepts child nodes of unmatched nodes, where nodes with the largest weight get the highest priority. The algorithm tries to make the tree of an exact match as large as possible by propagating the match to the respective parents of both compared nodes. When there is more than one potential candidate, it uses a few simple heuristic rules to decide which one to match.

XyDiff supports the insert, delete, update and move operations and achieves a time complexity of O(n log n). However, because of the greedy rules used, it can not guarantee an optimal result and in some cases it even mismatches subtrees.

According to Cobéna et al. [CAH02], XyDiff seems to be a good choice when high performance is required, even if it results in lower quality of the result.

4.8 VM Tools

VM Tools [VM02] is a collection of Java XML-oriented tools, created by VM Systems and is available under the open source licence. It contains a Diff and Patch tool for automatically generating differences between XML documents and can apply those differences to one of the original files. The tool focuses on minimal size of the difference file. The documentation of VM Tools currently available doesn't provide us with a clear description of properties such as types of supported operations and time complexity.

4.9 DiffXML

DiffXML [AM02] is created by Adrian Mouat and is part of a set of XML diff and patch utilities, which operate on the hierarchical structure of XML documents. The algorithm is open-source and has an independent en fully specified output format. The output format, specified in the Delta Update Language (DUL), handles the insert, update, delete and move operations. Kyriakos Komvoteas [KK03] extended the DUL output format and used it as an output format for his TreePatch algorithm.

4.10 XMiddle

XMiddle [MCZ02] is a XML-based middleware for peer-to-peer computing and provides a framework for synchronization and merging of mobile data. It "enables the transparent sharing of XML documents across heterogeneous mobile hosts, allowing online and offline access to data".

XMiddle contains a XML differencing tool and it uses DOM (Document Object Model) for representing the XML tree. XMiddle uses XMLTreeDiff [IBM98] as differencing tool.

4.11 KF-Diff+

KF-Diff+ [XWW02] is a change detection algorithm for XML documents that handles both ordered and unordered XML trees. The algorithm transforms the traditional tree-to-tree correction into comparing labeled trees without duplicate paths. KF-Diff+ achieves high efficiency with the time complexity of O(n), where XyDiff [CAM02] has a complexity of O(n log n), where n is the total number of nodes. KF-Diff+ handles insert, delete and update.

4.12 XML Diff and Patch

Microsoft created XML Diff and Patch [XDP02], a set of tools capable of comparing, differentiating and patching two XML documents. The comparison tool ignores the order attribute, ignores white spaces, considers `<q/>` the same as `<q></q>` and does not care about document encoding. It offers the option to ignore XML comments and processing instructions and some interesting options about namespaces.

XML Diff and Patch represents the differences between two XML trees in a XML Diff Language called XML Diffgram [XDL02]. It is however not an open-source tool and limits the size of XML documents to 100KB.

4.13 X-Diff

X-Diff [WDC03] can detect the optimal differences between two unordered XML trees in polynomial time. Song and Bhowmick [SB04] extended the algorithm for use in detecting changes in genomic and proteomic data (BioDiff). X-Diff integrates key XML structure characteristics with standard tree-to-tree correction techniques. The authors, Wang et al., compare X-Diff with XyDiff [CAM02].

X-Diff uses the notion of node signature together with a new matching between XML trees to find the minimum-cost matching and generate a minimum-cost delta script. The algorithm only specifies the three basic operations; insert, delete and update.

X-Diff seems to be similar to KF-Diff+ [XWW02], but according to Xu et al. [XWW02], besides the opinion that X-Diff may not satisfy users' need, the filtering of X-Diff is not efficient, because it can only reduce the size of the tree by removing equivalent second-level subtrees.

4.14 DeltaXML

Robin La Fontaine (Monsell EDM ltd.) [MEL03, RLF03] created DeltaXML, which is a commercial tool. It is capable of comparing, merging and synchronizing XML documents. DeltaXML handles both ordered and unordered XML trees, although the maximum tree size is 50MB. It must be explicitly stated which type of tree, ordered or unordered, is used. The merge part supports both two-way and three-way merge. DeltaXML uses a tree-matching algorithm, which runs in linear time and memory. It supports the insert, delete and update operations.

Cobéna et al. [CAH02] conclude in their work that DeltaXML seems to be the best choice as a change detection algorithm, because "it runs extremely fast and its results are close to the minimum".

4.15 TreePatch

TreePatch [KK03] is an open-source XML Diff and Patch Tool, designed and created by Kyriakos Komvoteas, as subject of his master's thesis. Komvoteas extended the DiffXML algorithm created by Adrian Mouat [AM02].

TreePatch describes changes using Extended Delta Update Language (EDUL), which is an extension of Adrian Mouat's DUL [AM02]. The EDUL output format describes the update, insert, delete and move operation.

4.16 TreeMatch

TreeMatch [YZ04] is a fast tree pattern-matching algorithm for XML Query. The goal of TreeMatch is to "directly find all distinct matchings of a query tree pattern in XML data sources". The algorithm is not applicable when detecting changes in XML documents because it is designed as a tree pattern-matching algorithm and not as a tree-matching algorithm.

4.17 BioDIFF

Based on X-Diff, BioDiff [SB04] is an algorithm specially designed for detecting changes in genomic and proteomic data specified as XML. It reduces the size of the set of biological data and focuses on the special semantics of the XML elements.

5. ALGORITHM PROPERTIES

Each algorithm has a number of advantages; the algorithms are optimized to certain properties. These properties are explained below.

5.1 Ordered or unordered tree

XML trees can be divided into two categories, ordered and unordered trees. In both types of trees, the parent-child relationship is significant, but in ordered trees, the left-to-right ordering among siblings is also significant.

Usually, applications do not mind if the nodes in a XML tree aren't ordered. Applications usually care about the contents of the XML trees, not about order, unless, for example, a XML document represents a web page, then the change detection algorithm needs to handle it as an ordered XML tree.

```
<examples>
  <example id="1">
    <name>Example one</name>
  </example>
  <example id="2">
    <name>Example two</name>
  </example>
</examples>
```

```
<examples>
  <example id="2">
    <name>Example two</name>
  </example>
  <example id="1">
    <name>Example one</name>
  </example>
</examples>
```

Figure 5 XML tree treated different as unordered tree

Table 1 Overview of change detection algorithms and properties

	Reference	Time complexity		Memory	Supported operations	Ordered / unordered	Section	Notes
LaDiff	[CRGW96]	linear	$O(ne+e^2)$	linear	basic, move	ordered	4.1	for LATEX elements
MH-Diff	[CGM97]	quadratic	$O(n^2 \log n)$	?	basic, move, copy	unordered	4.2	
XMLTreeDiff	[IBM98]	quadratic	$O(n^2)$	quadratic	basic	ordered	4.3	
MMDiff	[Cha99]	quadratic	$O(n^2)$	quadratic	basic	ordered	4.4	
XMDiff	[Cha99]	quadratic	$O(n^2)$	linear	basic	ordered	4.4	quadratic I/O cost
IBM's XML Diff and Merge Tool	[IBM01]	?	?	?	basic	?	4.5	commercial tool
3DM's matching algorithm	[TL01, TL03, TL04]	linear	O(n)	?	basic, move	ordered	4.6	
XyDiff	[CAM02]	linear	O(n log n)	linear	basic, move	ordered	4.7	
VM Tools	[VM02]	?	?	?	?	unordered	4.8	
DiffXML	[AM02]	linear	$O(ne+e^2)$	linear	basic, move	ordered	4.9	
KF-Diff+	[XWW02]	linear	O(n)	?	basic	both	4.11	
XML Diff and Patch	[XDP02]	?	?	?	?	both	4.12	commercial tool
X-Diff	[WDC03]	quadratic	$O(n^2)$	quadratic	basic	unordered	4.13	
DeltaXML	[MEL03, RLF03]	linear	?	linear	basic	both	4.14	
TreePatch	[KK03]	linear	$O(ne+e^2)$	linear	basic, move		4.15	
BioDIFF	[SB04]	quadratic	$O(n^2)$	quadratic	basic	unordered	4.17	for genomic and proteomic data

The difference for the change detection algorithm is that the algorithm has to detect a move (or insert + delete) when a node changes position while remaining child of the same parent.

Figure 5 shows an example of a XML tree where the difference between treating the tree as ordered tree and treating the tree as unordered tree becomes clear. Treat it as an ordered tree and there is a change present, the first example node is moved below the second, treat it as an unordered tree and both trees are the same, no change will be detected.

In three-way merging, algorithms that handle XML trees as ordered trees have a greater chance of generating conflicts. Similar to the example in Figure 5, a conflict would arise when two trees were to be merged where a node, present in both trees, has moved to different positions, while remaining child of the same parent. There would arise no conflict when the trees were treated as unordered XML trees.

5.2 Performance and Complexity

For database applications and dynamic services, good performance and low memory usage is required, especially with large amounts of data.

5.3 Correctness

We can assume that most algorithms are correct, that they find a set of operations which can transform the old version into the new version of the XML document. Some algorithms can match nodes that should not be matched. The final result after applying the delta script to the old version is still the new version, but the delta script is not optimal. This type of mismatching occurs when an algorithm applies certain criteria such as "the nodes are 90% similar, so they must be the same", and the algorithm stops searching for a better match. This is useful when time is an issue, but it degrades the meaning of the *changes* in the delta script, because multiple nodes can be mismatched, generating more *changes* or other types of *changes* then would be generated with an algorithm keeps searching for a better match.

5.4 Memory usage

An algorithm can improve memory usage by removing matched nodes from the XML trees. These nodes do not need further comparison, because they are already matched. Especially in large amounts of data, this can be useful.

5.5 Input size

Some algorithms are specially designed for handling large amounts of data. Others, like XMLTreeDiff [IBM98] and XMiddle [MCZ02] use the Document Object Model, therefore limited by DOM's maximum input file size. Microsoft's XML Diff and Patch [XDP02] limits it's file size to just 100KB while DeltaXML's [MEL03, RLF03] maximum file size is 50MB.

5.6 Semantics

The semantics of XML data can be used in some algorithms to quickly and correctly match nodes. Some may consider keys, defined as ID in the DTD, and match nodes with the same key and tagname.

5.7 Minimal delta scripts

Minimal delta scripts are an important factor in change detection algorithms. If the algorithm and delta script is used in

a version management system, the delta script needs primarily to be correct; secondary is the size of the delta script. But when transfer sizes of delta scripts are important, the algorithm needs to find a minimal delta script.

The use of 'move' and 'copy' operations can have a great impact on the size of the delta script. The 'move' operation reduces the file size, but has an impact on the complexity and performance of the algorithm. The 'copy' operation has an even more impact and is therefore even less used in change detection algorithms. Thus, most algorithms only support the basic operations 'update', 'insert' and 'delete'.

6. ALGORITHMS OVERVIEW

XMiddle is not present in Table 1, because it has no change detection algorithm of its own, but uses XMLTreeDiff, which is present in Table 1. TreeMatch is an algorithm specially designed for XML Query and therefore not present in Table 1.

For some algorithms, there was not sufficient information available. IBM's XML Diff and Merge Tool [IBM01] is a commercial tool, unless used for non-commercial purposes. There is no detailed documentation available for the algorithm. The same problem occurred with Microsoft's XML Diff and Patch [XDP02].

Cobéna et al. have delivered us information about the memory aspect of some algorithms, but not all, thus, there is some information missing.

For most algorithms, information about time complexity, supported operations and whether the algorithm supports ordered and/or unordered XML trees, is present in Table 1.

Table 1 describes the update, insert and delete operation as *basic*, because all algorithms support these operations.

7. CONCLUSION

This paper has described some algorithms specifically designed for change detection, and others where the change detection algorithm is just a part of the algorithm. There exists clear relations between some algorithms, for example BioDIFF [SB04] extends the X-Diff [WDC03] algorithm and Mouat's DiffXML algorithm is improved by Komvoteas' TreePatch.

Most properties described in this paper have a great impact on the design of most change detection algorithms. Should the algorithm disregard left-to-right ordering among siblings when handling the trees as unordered trees? Or should it regard some changes as a *move* operation, thus handling the trees as ordered trees?

Some algorithms use a combination of *insert* and *delete* instead of *move*, which has a positive impact on the size of the delta script. The MH-Diff algorithm even handles the *copy* operation, therefore generating a potentially even smaller delta script. The *move* and *copy* operation also have a negative impact on the time complexity of the algorithm. Especially with the *copy* operation, where more comparisons need to be made, the algorithm is slower than without the *copy* operation.

Some fast change detection algorithms do not provide optimal delta scripts, because they match nodes that are partly similar and stop searching for a better match. This is useful when time is an issue, but it degrades the meaning of *changes* in the delta script.

When the speed of the algorithm is the primary requirement, the correctness will become a secondary requirement. For example in large amounts of data speed is usually more important. On the other hand, when the algorithm's correctness is the primary requirement, for example in data synchronization, the speed will become secondary. Last but not least, when transfer speeds are an issue, for example in a mobile environment, the primary requirement should be the minimization of the delta script, which leads to a decrease in the speed of the algorithm.

ACKNOWLEDGMENTS

I would like to thank M.M. Fokkinga and K. Sikkel for helping me clarify research goals and research questions.

REFERENCES

[CVS] Concurrent Versions System: The open standard for version control. http://www.cvshome.org, accessed May 2005.

[W3C98] Extensible Markup Language (XML) 1.0, Feb. 98. http://www.w3c.org/TR/1998/REC-XML-19980210, accessed May 2005.

[SGML] Standard Generalized Markup Language (SGML), International Organization for Standardization, ISO 8879, 1986. http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=16387, accessed May 2005

[UNIX] The UNIX System Home Page, The Open Group, http://www.unix.org/, accessed May 2005

[LINUX] The Linux Home Page, http://www.linux.org/, accessed May 2005.

[DIFF] Diffutils, *The GNU Operating System*, http://www.gnu.org/software/diffutils/diffutils.html.

[LaTeX] LaTeX, *A document preparation system*, http://www.latex-project.org/.

[Xyleme] Xyleme, http://www.xyleme.com/.

[CRGW96] Sudarshan S. Chawathe, Anand Rajaraman, Hector Garcia-Molina, Jennifer Widom: Change Detection in Hierarchically Structured Information. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, p. 493-504.* June, 1996

[CGM97] Sudarshan S. Chawathe and Hector Garcia-Molina, Meaningful change detection in structured data, *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, *p. 26-37.* May. 1997.

[IBM98] XML TreeDiff by IBM alphaWorks, retired Nov. 1998. Idea of David Epstein, designed and implemented by Francisco Curbera. http://alphaworks.ibm.com/tech/xmltreediff/, accessed May 2005.

[XTI98] XML TreeDiff by IBM alphaWorks, retired Nov. 1998. http://www.xml.com/pub/r/536, accessed May 2005.

[Cha99] Sudarshan S. Chawathe, Comparing Hierarchical Data in External Memory, *Proceedings of the 25th International Conference on Very Large Data Bases*, Sept. 1999

[CTZ00] Shu-Yao Chien, Vassilis J. Tsotras , Carlo Zaniolo, Version Management of XML Documents, *Selected papers from the Third International Workshop WebDB 2000 on The World Wide Web and Databases, p. 184-200,* May 2000.

[IBM01] XML Diff and Merge Tool by IBM alphaWorks, last update Mar. 2001, http://www.alphaworks.ibm.com/tech/xmldiffmerge , accessed May 2005.

[CTZ01] Shu Yao Chien, Vassilis J. Tsotras, Carlo Zaniolo. XML Document Versioning, *ACM SIGMOD Record, Volume 30, Issue 3, SPECIAL ISSUE: Special section on advanced XML data processing, 46-53,* Sept. 2001.

[TL01] Tancred Lindholm, A 3-way merging algorithm for synchronizing ordered trees – the 3DM merging and differencing tool for XML, Master's thesis, Helsinki University of Technology, Dept. of Computer Science, Sept. 2001. http://www.cs.hut.fi/~ctl/3dm/thesis.pdf, accessed May 2005.

[MACM01] Amélie Marian, Serge Abiteboul, Grégory Cobéna, Laurent Mignet, Change-Centric Management of Versions in an XML Warehouse, *Proceedings of the 27th International Conference on Very Large Data Bases, p. 581-590.* Sept. 2001.

[ACFR01] Serge Abiteboul, Sophie Cluet, Guy Ferran, Marie-Christine Rousset, The Xyleme Project, *Gemo Report number 248*, Nov. 2001.

[CAM02] Gregory Cobéna, Serge Abiteboul, Amélie Marian: Detecting Changes in XML Documents. *Proceedings of the 18th International Conference on Data Engineering, 41-52.* Feb. 2002.

[VM02] VM Tools by VM Systems, last release Feb. 2002. http://www.vmsystems.net/vmtools/, accessed May 2005.

[MCZ02] Cecilia Mascolo, Licia Capra, Stefanos Zachariadis and Wolfgang Emmerich, XMIDDLE: A Data-Sharing Middleware for Mobile Computing. *In Personal and Wireless Communications Journal 21(1). Kluwer*. Apr. 2002.

[AM02] Adrian Mouat, Diffxml, June 2002, http://diffxml.sourceforge.net, accessed May 2005.

[XWW02] Haiyuan Xu, Quanyuan Wu, Huaimin Wang, Guogui Yang, Yan Jia. KF-Diff+: Highly Efficient Change Detection Algorithm for XML Documents. *On the Move to Meaningful Internet Systems, DOA/CoopIS/ODBASE, p. 1273-1286*. Nov. 2002

[XDP02] XML Diff and Patch, Microsoft Corporation, http://apps.gotdotnet.com/xmltools/xmldiff/, 2002, accessed May 2005.

[XDL02] XML Diff Language, Microsoft Corporation, http://apps.gotdotnet.com/xmltools/xmldiff/XmlDiff Language (Diffgram).html, accessed May 2005.

[CAH02] Grégory Cobéna, Talel Abdessalem, Yassine Hinnach A comparative study for XML change detection, *Gemo Report number 221,* 2002. http://www-rocqbis.inria.fr/verso/Gemo/PUBLI/all-byyear.php, ftp://ftp.inria.fr/INRIA/Projects/gemo/gemo/GemoReport-221.pdf, accessed May 2005.

[WDC03] Yuan Wang, David J. DeWitt, Jin-yi Cai. X-Diff: An Effective Change Detection Algorithm for XML Documents. *Proceedings of the 19th International Conference on Data Engineering, 519-530.* Mar. 2003.

[MEL03] Monsell EDM Ltd. Merging XML Changes with DeltaXML, http://www.deltaxml.com/, accessed May 2005.

[RLF03] Robin La Fontaine, DeltaXML, Change Control for XML: Do It Right, (Director of Monsell EDM Ltd.), *XML Europe 2003*, May 2003.

[KK03] Kyriakos Komvoteas. XML Diff and Patch Tool. *Master's thesis, Master of Science in Distributed and Multimedia Information Systems,* Sept 5. 2003 http://treepatch.sourceforge.net/, accessed May 2005.

[TL03] Tancred Lindholm, XML three-way merge as a reconciliation engine for mobile data, *Proceedings of the 3rd ACM international workshop on Data engineering for wireless and mobile access*, *93-97*, Sept. 2003.

[FCP] Fuego Core Project, Jan. 2002 to Dec. 2004, continued Jan. 2005 to Dec. 2007. http://www.hiit.fi/fuego/fc/.

[YZ04] J.T. Yao, M. Zhang. A Fast Tree Pattern Matching Algorithm for XML Query. *2004 IEEE/WIC/ACM International Conference on Web Intelligence, 235-241.* Sept. 2004

[TL04] Tancred Lindholm, A three-way merge for XML documents, *Proceedings of the 2004 ACM symposium on Document engineering*, *1-10*, Oct. 2004.

[SB04] Song, Bhowmick. BioDIFF An Effective Fast Change Detection Algorithm for Genomic and Proteomic Data. *Proceedings of the Thirteenth ACM conference on Information and knowledge management.* 146-147. Nov. 2004.