Analysing differences in tree-structured data by Visual Analytics

Johannes Lichtenberger

University of Konstanz, Germany
`Johannes.Lichtenberger@uni-konstanz.de`

Abstract. Current state of the art line-based diffing algorithms and visualizations thereof aren't tree aware. Furthermore almost all recent tree-aware diff-visualizations aren't able to explicitly highlight inserted-, deleted-, replaced- as well as updated-nodes. Thus, we propose different solutions based on a SunburstView.

1 Motivation

Current state of the art line-based diffing algorithms and visualizations on tree-structured data aren't sufficient. XML, which is a tree-structured meta markup-language, adds further constraints.

In a nutshell line based LLCS algorithms such as `GNU-diff`, `(g)vimdiff` etc.pp. are not tree-aware and have no knowledge about XML semantics which results in

- Mismatches of element nodes with different `QNames`.
- Matches across node-boundaries.
- `Attribute`- and `namespace`-order changes are considered as a change.
- Whitespace between attributes or namespaces are considered as a change.

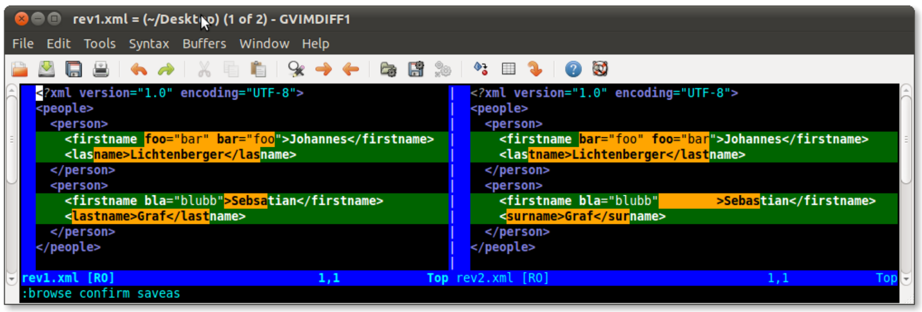
Figure 1 illustrates some of these drawbacks on two revisions of a simple XML-document. For larger XML instances it is mandatory to scroll, thus additionally losing the overview about the whole file is another drawback.

1.1 Application domains

Motivating examples are manyfold. To meantion just a few of them:

- Even though companies and the relationship among employees in general form a directed acyclic graph (DAG) most of them are organized in a hierarichal tree-structure [1] which evolves over time. Appropriate representations of the changes enables to track the career of certain persons or might reveal how sub-organizations are evolving very well while others might not due to the success or failure of products.

[1] also referred to as a connected acyclic graph (CAG)


rev1.xml = (~/Desktop) (1 of 2) - GVIMDIFF1
File Edit Tools Syntax Buffers Window Help

```
<?xml version="1.0" encoding="UTF-8">               | <?xml version="1.0" encoding="UTF-8">
<people>                                             | <people>
  <person>                                           |   <person>
    <firstname foo="bar" bar="foo">Johannes</firstname> |     <firstname bar="foo" foo="bar">Johannes</firstname>
    <lasname>Lichtenberger</lasname>                 |     <lastname>Lichtenberger</lastname>
  </person>                                          |   </person>
  <person>                                           |   <person>
    <firstname bla="blubb">Sebsatian</firstname>     |     <firstname bla="blubb"        >Sebastian</firstname>
    <lastname>Graf</lastname>                        |     <surname>Graf</surname>
  </person>                                          |   </person>
</people>                                            | </people>
rev1.xml [RO]                   1,1            Top rev2.xml [RO]                   1,1            Top
:browse confirm saveas
```

Fig. 1. Comparsion of two revisions of a simple XML-document with gvimdiff.

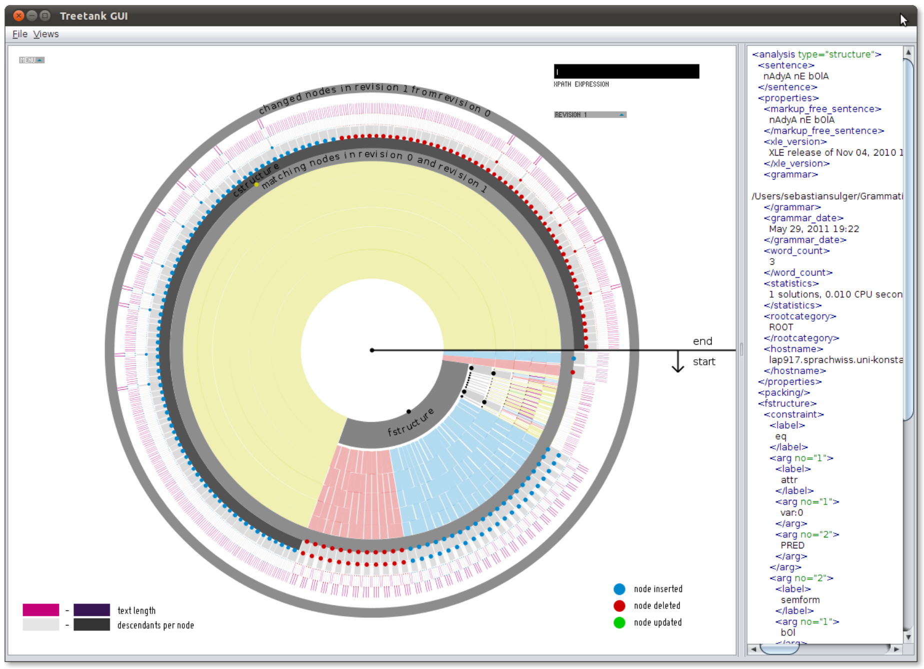
- The hierarichal structure of filesystems and the recently upcoming abilities to store Snapshots, which is based on the COW (copy-on-write) implementation of recent filesystems like BtrFS and ZFS. Users might want to quickly determine when and where directories/files have been moved, when they have been deleted or created and so forth. An example application which recently reached end users through Apples' OS X is TimeMachine. Furthermore permission rights of files can be tracked through time, which might be of great value to security engineers.
- Humans which are interested in detailed changes of several versions of some products and compare their attributes and features. This analysis might even reveal how companies prioritise the development and enhancement of certain features.
- Tracking changes in software development based on a low level Abstract Syntax Tree (AST) comparsion. This might reveal code splits, merges, deletions and inserts. At a greater scale constant unmodified code can be tracked through several revisions revealing stable code in contrast to new features. The latter usually is subject to many changes.
- Biologists have to understand structural differences of phylogenetic/evolutionary trees which in general can be very large. An efficient comparsion algorithm is needed as well as some kind of visual interface to asses biologists in locating and exploring tree differences.

2 Proposed visualizations

2.1 SunburstView

In contrast our first proposed visualization (Figure 2) used on a XML-document with about 800 nodes displays detailed differences of the tree data itself and is aware of XML semantics. Two revisions of a Treetank-`resource` can be compared. The *SunburstItems*, which represent nodes and are located between the inner ring and the root node (the center) contain nodes which haven't been changed at all. Changed nodes are projected to the outer ring. The colors denote different operations in which a node has been changed. A legend is available in the bottom right corner.

It is roughly based on [4], but now adheres to the well known MVC-architecture, whereas the model(s) in conjunction with a special `SunburstDescendantAxis` interact with Treetank. Furthermore besides the addition of the whole diff-view, our visualization in contrast uses an offscreen-buffer for all `SunburstItem` instances and was basically rewritten from scratch to adhere to good object oriented practices and design patterns. Left-over are some drawing mechanisms (which are now implemented based on the strategy pattern) such as the drawing of the *SunburstItems* and the connection splines to emphasize the `parent/child`-relationship.



```
<analysis type="structure">
  <sentence>
   nAdyA nE bOlA
  </sentence>
  <properties>
    <markup_free_sentence>
     nAdyA nE bOlA
    </markup_free_sentence>
    <xle_version>
     XLE release of Nov 04, 2010 1
    </xle_version>
    <grammar>

/Users/sebastiansulger/Grammati
    </grammar>
    <grammar_date>
     May 29, 2011 19:22
    </grammar_date>
    <word_count>
     3
    </word_count>
    <statistics>
     1 solutions, 0.010 CPU secon
    </statistics>
    <rootcategory>
     ROOT
    </rootcategory>
    <hostname>
     lap917.sprachwiss.uni-konsta
    </hostname>
  </properties>
  <packing/>
  <fstructure>
    <constraint>
      <label>
       eq
      </label>
      <arg no="1">
        <label>
         attr
        </label>
        <arg no="1">
         var:0
        </arg>
        <arg no="2">
         PRED
        </arg>
      </arg>
      <arg no="2">
        <label>
         semform
        </label>
        <arg no="1">
         bOl
```

Fig. 2. Comparsion of two revisions of an XML-document with approximately 800 nodes.

Currently our visualization is able to differentiate between the following `edit`-operations.

- `insertion` of a node.
- `deletion` of a node.
- `update` of a node, which is updating the `QName` in case of an element-node or the value in case of a text-node.

- `replace` of a node with another node.

2.2 SunburstView challenges

Challenges while implementing the *SunburstView* were manyfold:

- Items require a `start-angle`, an `extension` or `end-angle`, `descendant-or-self-count`, `modification-count` and the kind of change (`inserted`, `deleted`, `updated` or `replaced`). The `modification-count` denotes how many modified nodes exist in the subtree rooted at the current node.
- Agglomeration of two revisions into one tree-representation.
- Pruning of the tree to allow the visualization to scale for even the largest XML instances.
- Provide a global enlargement of subtrees with many changed nodes to further help tracking differences which also shrinks unchanged subtrees.
- Computation of the maximum level in the tree of unchanged nodes.
- Computation of the `descendant-or-self-count` per node in the new revision including deleted nodes.
- Integration of views based on processing.org [3] which can be used just like any other Java library in a Swing-GUI (but problems arise due to heavyweight vs. lightweight components).
- Providing labels if items are big enough.
- Write all `SunburstItem`s into an offscreen buffer.
- Improved mouseover-effect to display further details about currently hovered nodes and to determine at which position the text has to be plotted (right or left to the cursor, dependent on the space left to display the information).
- Integration of zooming and a fisheye-transformation.
- Three normalizations are available for the `SunburstItem` color which currently denotes either how many `descendant-or-self` nodes follow in the subtree of the node, or the String-length of text node values.

Most of these challenges culminated in a `SunburstModel` and a specialized `SunburstDescendantAxis` which is used to create `SunburstItem`s, the segments in the *SunburstView* which represent the nodes in the tree. The agglomeration of two revisions into one tree requires the iteration over the newer of both revisions but to add deleted nodes at the right positions[2]. Thus the Treetank `read-`transaction has to switch to the old revision whenever a deleted node is encountered and to switch back if no further deleted nodes follow which is illustrated in figure (3). Input to the `SunburstDescendantAxis` besides a `read-transaction` on the new revision is a list of diffs. Each element incorporates the kind of diff, the depth of the node in the old revision and the node in the new revision and the two nodeKeys of the nodes which have been compared. Each time `hasNext()` is called the first element of the list is removed.

[2] iteration over the old revision but to add inserted nodes would have been possible as well

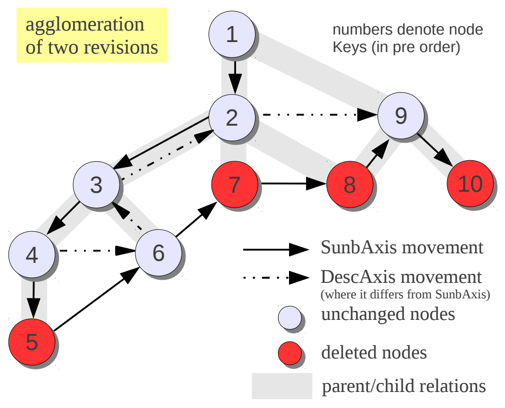

agglomeration of two revisions
numbers denote node Keys (in pre order)
1
2
3
4
5
6
7
8
9
10
SunbAxis movement
DescAxis movement
(where it differs from SunbAxis)
unchanged nodes
deleted nodes
parent/child relations

Fig. 3. Illustrates some challenges while developing the SunburstDescendantAxis which agglomerates two revisions into one tree.

One of the core challenges is to adjust the stacks for the `start-angle`, `extension`, `descendant-or-self-count` and the `modification-count` of each `SunburstItem` appropriately which is closely related to the challenge of fusing both trees together. If the transaction is located at node 4 it still has to push values onto the different stacks and remember this step. The `next-NodeKey` which would be usually 6 in the `DescendantAxis` is saved in a temporal variable to move to whenever no further deleted nodes are encountered[3]. Furthermore the next call to the iterator method `hasNext()` must change the current transaction which is a `read-transaction` on the new revision to a `read-transaction` on the old revision and modify member variables such as the `start-angle`, `extension`, and so on according to the previous step before the next `SunburstItem` is added. Located at node 6 the current active transaction must only call `pop()` once from all stacks which is actually one of the first things done the next time `hasNext()` is called. If an unchanged node would follow after node 6 (node 9 without considering deleted nodes) the stacks would be immediately adapted, e.g. `pop()` would have been called two times from all stacks since the next node (node 9) is located two levels above node 6. Another case occurs at node 9. If the new revision has no further nodes the diff-list has to be examined if deleted nodes are left.

Furthermore to get the maximum level for the nodes which haven't changed at all and to draw the ring at $maxLevel + 1$ the algorithm iterates over the old revision but skips nodes with higher levels, which are actually deleted in the new revision.

[3] in the running example based on figure 3 after node 5

2.3 TextView

The TextView is synchronized with the SunburstView (and the SmallMultiplesView) and provides pretty printed serialized XML as well as syntax highlighting and is scalable due to the fact, that initially only the viewport and some extra value is filled. Scrolling down adds more text depending on the scrollbar value change. Before a `StAXSerializer` which can be used for many other purposes has been developed.

2.4 SmallMultiplesView

SmallMultiples divide the screen space into regions whereas each region displays a *SunburstView* with different parameters to enable the analysis of the evolution of a tree. Currently three variants of a *SmallMultiplesView* are available.

- An incremental view illustrating differences between successive revisions e.g. the differences between revision 0 and 1 illustrated in one SunburstView, the differences between revision 1 and 2 in another side by side and so on and so forth.
- A differential view illustrating differences between the first and each successive revision. Assuming revision 0 is loaded and the `resource` has 10 revisions the comparsion of revision 0 and 1, 0 and 2, 0 and 3 and revision 0 and 3 are displayed through succesive *SunburstViews*). Therefore at most four regions with a SunburstView are available just like in the other variants. However this is only a parameter which can easily be changed to 6 or 8 or another even value such that the screen space of larger monitors can effectively be utilized.
- A hybrid view which displays the whole tree after the last revision (which is shown on screen) with every change made in the revisions in between. Then any changes which didn't occur in each successive incremental SunburstView are blackened.

Figure 4 depicts the last hybrid variant.

2.5 SmallMultiplesView challenges

First of all further abstractions have been made to adhere to good software practices. A few of the challenges are:

- Since all (public) methods in every model[4] are multithreaded the notification of observers (the GUI) about state changes have to be synchronized.
- Multithreading also requires sorting of offscreen buffers such that the SmallMultiples are sorted from left to right to bottom.
- In the hybrid view changes which haven't occured in an incremental diff between two successive revisions have to be blackened, respectively.

[4] one component of the MVC architecture

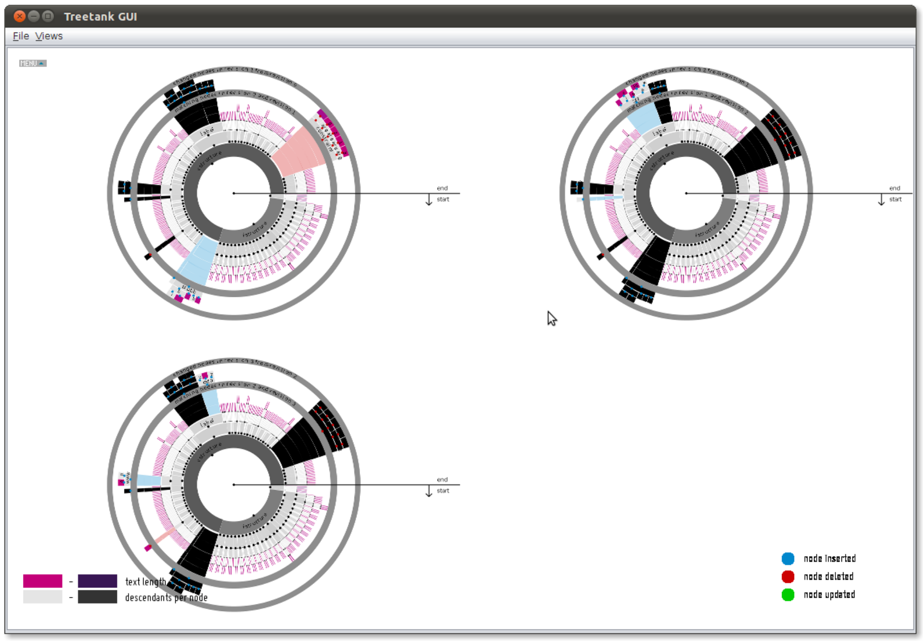


Fig. 4. Comparsion of four successive revisions of an XML-document based on the hybrid variant.

The third challenge is particular interesting. It involves the computation of differences between the first and last revision which can be displayed on screen (usually revision 0 to 4 with four different regions and incremental change views, the *SunburstViews*). Next the `SunburstItems` created during the incremental diff as well the comparsion between the first and last revision have to be sorted according to the `NodeKey` which is unique for each node. Then initially all nodes are blackened meaning their status is set to `BlackState.YES`. Thereafter a search is made to get the intersection of nodes. If nodes are equal according to the `java.lang.Object` method `boolean equal(Object paramItem)` nodes are colored ordinary, which means their greyout status is set to `BlackState.NO`. All nodes which are still in the status `BlackState.YES` are drawn in black.

3 Tree-to-Tree Differences

To compute the differences between two revisions initially two kinds of algorithms have to be distinguished, namely ID-less- and ID-based-algorithms. Figure 5 points out how these two kinds of algorithms are used in our approach. An ID-based algorithm has been developed and is used to compare revisions of a *resource* in Treetank since each node has a unique `NodeKey` which identifies this node through time/all revisions even after updates. To speed up the algorithm, it optionally uses postorder-hashes which are per default generated during the shredding of an XML-document. If the hashes and the nodeKeys of both nodes match the whole subtree can be considered to be equal and therefore skipped. Thus the `read`-transactions move to the following nodes in both revisions. Yet this feature can't be used for the diff-view since each individual node has to be set to an explicit `diff`-state and the axis which has been developed needs to know the explicit diff-state for each node.

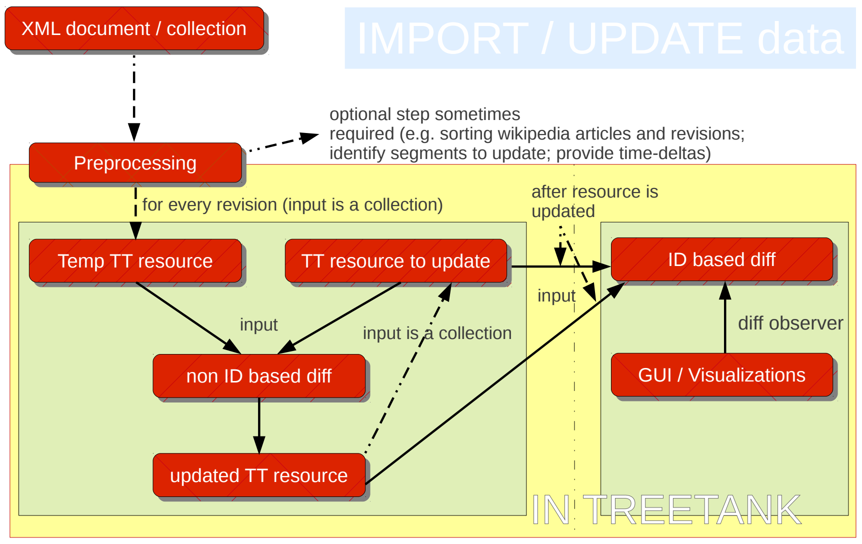

XML document / collection
IMPORT / UPDATE data
Preprocessing
optional step sometimes required (e.g. sorting wikipedia articles and revisions; identify segments to update; provide time-deltas)
for every revision (input is a collection)
after resource is updated
Temp TT resource
TT resource to update
ID based diff
input
input
input is a collection
diff observer
non ID based diff
GUI / Visualizations
updated TT resource
IN TREETANK

Fig. 5. Illustrates when and how ID-less and ID-based diff-algorithms are used.

To import XML documents in Treetank in the first place the data sometimes has to be preprocessed. The following steps were necessary for the preprocessing of the Wikipedia dump with revision history of each article:

- Replace WikiText markup with well formed XML.
- Sorting of articles and revisions based on timestamps with the MapReduce Framework of Apache Hadoop [2].
- Import based on different time-intervals.

An ID-less algorithm has to be used, which is usually slower due to the initial matching between nodes. Heuristics have to be used to speed up the matching algorithm since it's CPU time complexity to find "optimal" matches is at least $O(n^2)$ and above. Optimal matches are based on a cost model, which incorporates edit script costs. An edit script is a sequence of edit operations (usually insert, delete, update and move). Each edit operation is assigned a cost, which might be unit cost. An optimal matching between nodes yields a minimum edit script, which is an edit script whereas no other edit script exists with less overall cost, the sum of edit operation costs. During this initial phase of an algorithm nodes which are approximately equal also have to be matched which is illustrated in figure 6. Node 20 has been inserted but nontheless nodes 3 and 14 should match. Most of the algorithms use a *Longest Common Subsequence* (LCSS) algorithm to match leaf nodes and sometimes also match inner nodes.

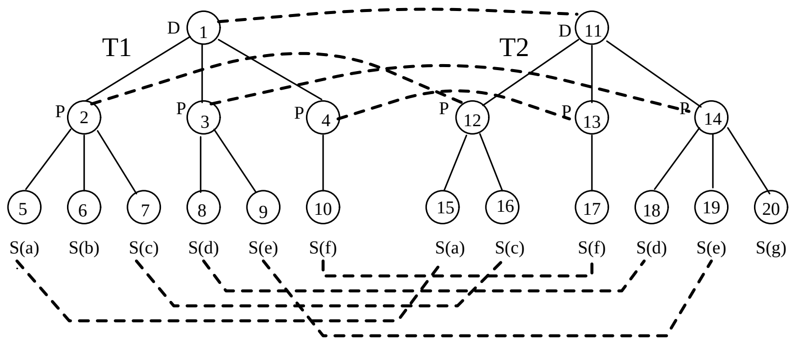


Fig. 6. Illustrates matching nodes in two revisions by dashes (from the paper "Change Detection in Hierarichally Structured Data" by Chawathe [1]).

3.1 Tree-to-Tree Differences - Challenges

In a nutshell the main challenges were the implementation of:

- Moves in Treetank with merging of adjacent text nodes.
- A copy operation to insert whole subtrees (based on the visitor pattern).
- One of the many existing ID-less algorithms (implemented the FMSE algorithm from Chawathe [1], whereas the algorithm in the paper seems to be inaccurate in two cases). The implementation is roughly based on the work of Daniel Hottinger and Franziska Meyer [5], which have used DOM.
- A LevelOrderAxis, which provides a breath first traversal to support the FSME algorithm.
- An ID-based algorithm developed to compare revisions in Treetank.

Note that it is hard to track bugs in the FMSE algorithm since almost all code-units are dependent on each other.

4 Future work

Moves have recently been implemented in the backend (Treetank) to support FMSE and other ID-less difference algorithms which might be implemented in the future. To support moves in the visualizations the following three steps have to be implemented:

1. Optional support of moves in the ID-based algorithm.
2. Add `dummy`-nodes at the places where the nodes have been moved from (to speed up the algorithm which currently traverses the trees in a single preorder-traversal).
3. Splines to connect the dummy-nodes and moved nodes in the *SunburstView*.

Second the *SmallMultiplesView* has to be synchronized with the *SunburstView*.

Furthermore to support sophisticated analytical tasks involving several revisions a specialized `XPathAxis` has to be implemented.

References

1. S. Chawathe, A. Rajaraman, H. Garcia-Molina, and J. Widom. Change detection in hierarchically structured information. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, pages 493–504. ACM, 1996.
2. H. P. M. Committee. Apache hadoop. `http://hadoop.apache.org`.
3. B. Fry and C. Reas. Processing.org. `http://processing.org`.
4. J. L. C. L. Hartmut Bohnacker, Benedikt Groß. Generative-gestalung : Sunburst-tool. `http://www.generative-gestaltung.de/M_5_5_01_TOOL`, 2009.
5. D. Hottinger and F. Meyer. Xml-diff-algorithmen. `http://www.infsec.ethz.ch/education/projects/archive/XMLDiffReport.pdf`.