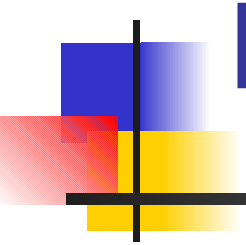
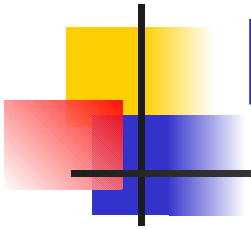


FUJABA

A Generic Difference Algorithm for UML Models

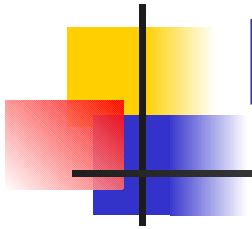
Sherif Luka



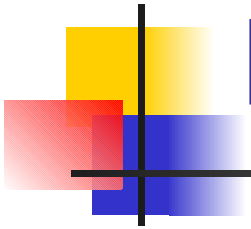
Presentation Overview

- Related Work
- X-Diff
- FUJABA Difference algorithm
- Demo
- References

Related Work



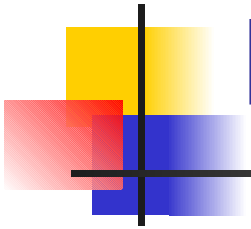
1. GNU diff utility uses the LCS (Longest Common Subsequence) algorithm to compare two plain text files.
2. CVS (GNU utility) uses diff to detect differences between two version of programs.
   - Why don't we simply use these tools?
3. AT & T Internet Difference Engine uses Html Diff. Why not use this for XML? Markups in XML provide context and contents within different markups can't be matched.



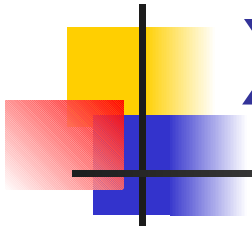
Related Work

3. Zhang and Sasha proposed a fast algorithm to detect changes in XML documents using ordered labeled trees. (They use minimum cost editing distance). They find an optimal edit script in O(n1 * n2 * min (depth(T1), leaves(T1)) * min (depth (T2), leaves (T2)))
4. Chawathe et al, presented a heuristic algorithm, MH-Diff, to detect change in unordered structured documents (edit script as an edge cover of a bipartite graph). Worst case running time : $O(n^3)$

Related Work



1. XML TreeDiff: May not produce an optimal result, and it uses Z and S (4) algorithm, and it works with ordered trees.
2. Cobena et al proposed XyDiff which uses a greedy approach and thus can not guarantee any form of optimal or near optimal result.

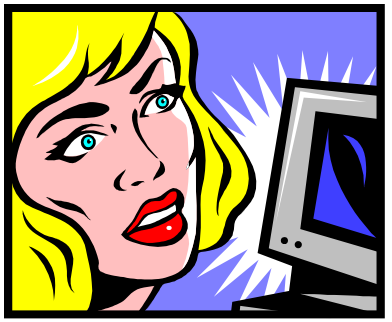


X-Diff (XML differences)

- XML has become the standard format for web publishing and data transportation.
- Previous work in XML change detection used an ordered tree model.
- X-Diff uses an unordered model. It produces more accurate results although the algorithm is substantially harder than in ordered models. (NP-Complete)
- But because XML documents have certain features it is possible to compute the optimal difference between two XML documents in polynomial time.

Example

- Assume that you have an online auction site equipped with a search engine and a change detection tool.
- A parent is interested in buying books for his child.



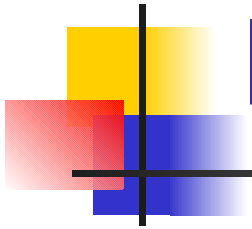
```
<Books>
  <Book>
    <Title>Harry Potter and the Sorcerer's Stone</Title>
    <Author>J.K. Rowling</Author>
    <Seller>
      <ID>Mike</ID>
      <Rating>30</Rating>
    </Seller>
    <First_Bid>$5.00</First_Bid>
    <Current_Bid Time_Left = "36 hrs.">$8.50</Current_Bid>
    <Bidder>
      <ID>Steve</ID>
      <Rating>25</Rating>
    </Bidder>
  </Book>
  <Book>
    <Title>The Adventures of Tom Sawyer</Title>
    <Author>Mark Twain</Author>
    <Seller>
      <ID>Sean</ID>
      <Rating>100</Rating>
    </Seller>
    <First_Bid>$2.00</First_Bid>
    <Current_Bid Time_Left = "4 hrs.">$3.50</Current_Bid>
    <Bidder>
      <ID>Tim</ID>
      <Rating>5</Rating>
    </Bidder>
  </Book>
</Books>
```

Figure 1.1 A piece of auction data of old version

```
<Books>
  <Book>
    <Title>The Adventures of Tom Sawyer</Title>
    <Author>Mark Twain</Author>
    <Seller>
      <ID>Sean</ID>
      <Rating>100</Rating>
    </Seller>
    <First_Bid>$2.00</First_Bid>
    <Current_Bid Time_Left = "2 hrs.">$4.50</Current_Bid>
    <Bidder>
      <ID>Tim</ID>
      <Rating>5</Rating>
    </Bidder>
  </Book>
  <Book>
    <Title>Harry Potter and the Sorcerer's Stone</Title>
    <Author>J.K. Rowling</Author>
    <Seller>
      <ID>Mike</ID>
      <Rating>30</Rating>
    </Seller>
    <First_Bid>$5.00</First_Bid>
    <Current_Bid Time_Left = "34 hrs.">$10.00</Current_Bid>
    <Bidder>
      <ID>Mark</ID>
      <Rating>125</Rating>
    </Bidder>
  </Book>
</Books>
```

Figure 1.2 A piece of auction data of new version

Advantages of a Change Detection Tool like X-DIFF



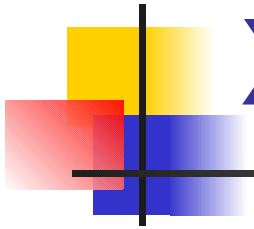
1. Incremental Query Evaluation:

   When a user has a standing query against a time-varying data source, a change detection tool can provide the query engine with delta data (Much faster!).

3. Trigger Condition Evaluation:

   Continuous query s/s, condition of firing is dependant on specific data changes.

X-Diff (Tree Representation of XML Documents)



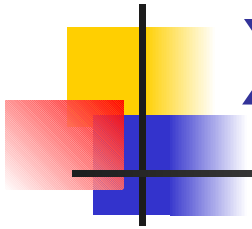
- XML documents have a hierarchical structure. Based on DOM, an XML document can be presented as a tree.
- There are three kind of nodes in DOM tree:
  1. Element Nodes: non-leaf nodes with name.
  2. Text Nodes: leaf nodes with value.
  3. Attribute Nodes: leaf nodes with name and value.
- Two Trees are isomorphic if they are identical except for the ordering of siblings. X-Diff considers two trees are equivalent if they are isomorphic.

X-Diff (Edit Operations)

1. Insert(x(name,value),y)
2. Delete(x)
3. Update (x,new_value)
4. Insert (Tx,y)
5. Delete (Tx)

Note:

- No need to specify which position among y's child nodes to insert node x.
- There are no "move" operations, which transfer a node or a subtree from one position to another (replace with a combination of delete and insert operations)



X-Diff (Edit Scripts)

- A sequence of basic edit operations that convert one tree into another.

X-Diff (Edit Scripts Example)

Example:

E(T1 → T2) = Delete(5),Insert(5(B, λ),3),Update(6, ω).

E'(T1 → T2) = Update(5, λ), Delete(5), Insert(5(B, λ),3),
Update(6, ω)

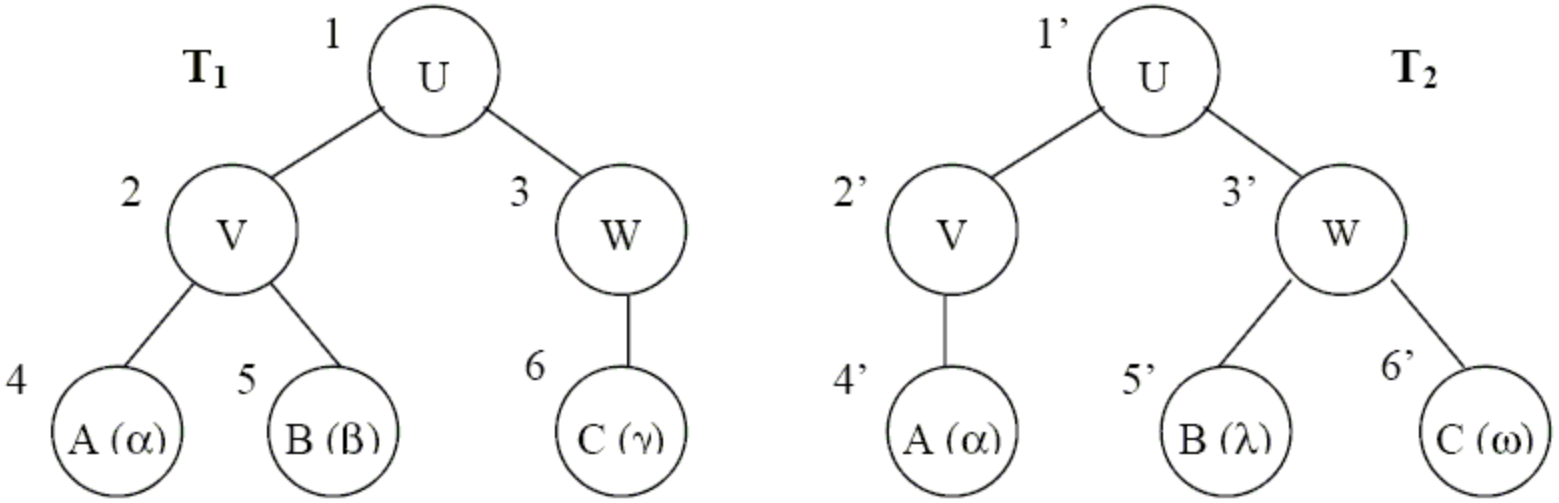
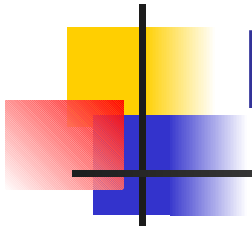


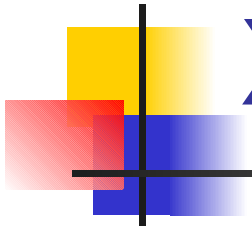
Figure 3.1 An example for edit scripts

X-Diff (General Cost Model for Edit Scripts)



Given an edit script E:

Cost (E) = n, where E = O1 O2 O3 … On and Oi is a basic edit operation.

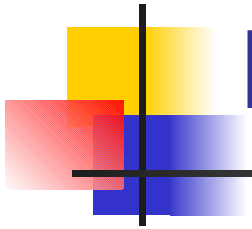


X-Diff

Definitions:

2. E is a minimum-cost edit script (optimal edit script) for (T1 → T2) iff for all edit scripts E' of (T1 → T2) cost (E') ≥ cost (E)
3. Editing distance: Dist (T1,T2) = Cost (E), where E is a minimum-cost edit script for (T1 → T2)

X-Diff (Node Signature and Minimum-Cost Matching)



- It is not a good idea to match every node in the first tree to every node in the second tree because each node in XML has its own context.
- Also nodes with different names and with different node types shouldn't be matched.
- Is it sufficient to only match nodes with the same name and type to decide if they match?

X-Diff (Node Signature and Minimum-Cost Matching)

- Given a DOM tree T:

  Root (T): root of T

  Type (x): node type of x

  Name (x): node name of x

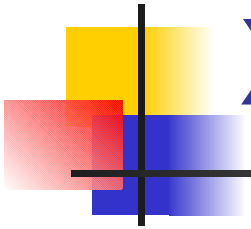
  Value (x): node value of x

Signature (x) = /Name(x1)/…/Name (xn)/Name (x)/Type (x) where x1 is the root of T, (x1, x2, … xn, x) is the path from root to x. if x is a text node,

Signature (x) = /Name(x1)/…/Name (xn)/Type (x)

X-Diff (Node Signature and Minimum-Cost Matching)

- A set of node pairs (x, y), M, is called a Matching from T1 to T2, iff
  1. (x, y) ε M, x ε T1, y ε T2, Signature (x) = Signature (y).
  2. For all (x1, y1) ε M, and (x2, y2) ε M, x1=x2 iff y1=y2 (one to one to correspondence)
  3. M is prefix closed, i.e., given (x, y) ε M, suppose x' is the parent of x, y' is the parent of y, then (x, y') ε M.
     1. Suppose (x1, y1) ε M, (x2, y2) ε M, x1 is an ancestor of x2 iff y1 is an ancestor of y2.
     2. M is a matching from T1 to T2, M={} iff (Root(T1), Root(T2)) is not ε M.



X-Diff (Algorithm)

Input:

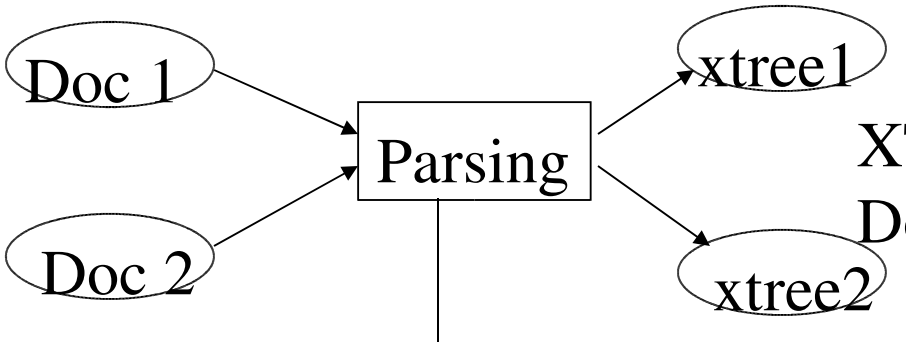
- Doc1 and Doc2 (two XML documents)

Algorithm:

3. Parsing and Hashing
4. Matching
5. Generating Minimum-Cost Edit Script

X-Diff (Algorithm)

1. Parsing and Hashing:



XTree is a subset of Dom Tree.

During the parsing process X-Diff uses a special Hash function (XHash) to compute a hash value for every node on both trees. Two Isomorphic trees have the same XHash value for their nodes (each node's hash value represents the entire subtree). Running time: O(|T1| * log (|T1|) + |T2| *log(|T2|))

X-Diff (Algorithm)

1. **Matching:**

   Reduce matching space: filter out equivalent subtrees between two root nodes by comparing the XHash values of second level child nodes.

   1. Compute the editing distance for each of the remaining subtree pairs and obtain a minimum-cost matching.
   2. Compute the editing distance between T1 and T2 and obtain minimum-cost matching.

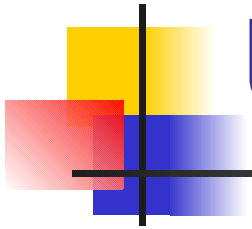
   Dynamic programming and minimum-cost maximum flow algorithms are used to compute Dist(T1, T2), starting from the leaf node pairs and moving upwards.

   Running time: O(|T1| * |T2| * max {deg(T1),deg(T2)} * log (max{deg(T1),deg(T2)})

3. **Generating Minimum Cost Edit Script:**

   Done recursively from root to leaves.

   Running time : O(|T1| + |T2|)



UML-Diff by FUJABA

Motivation:

OMG → MDA

MDA (PIM → PDM)

UML

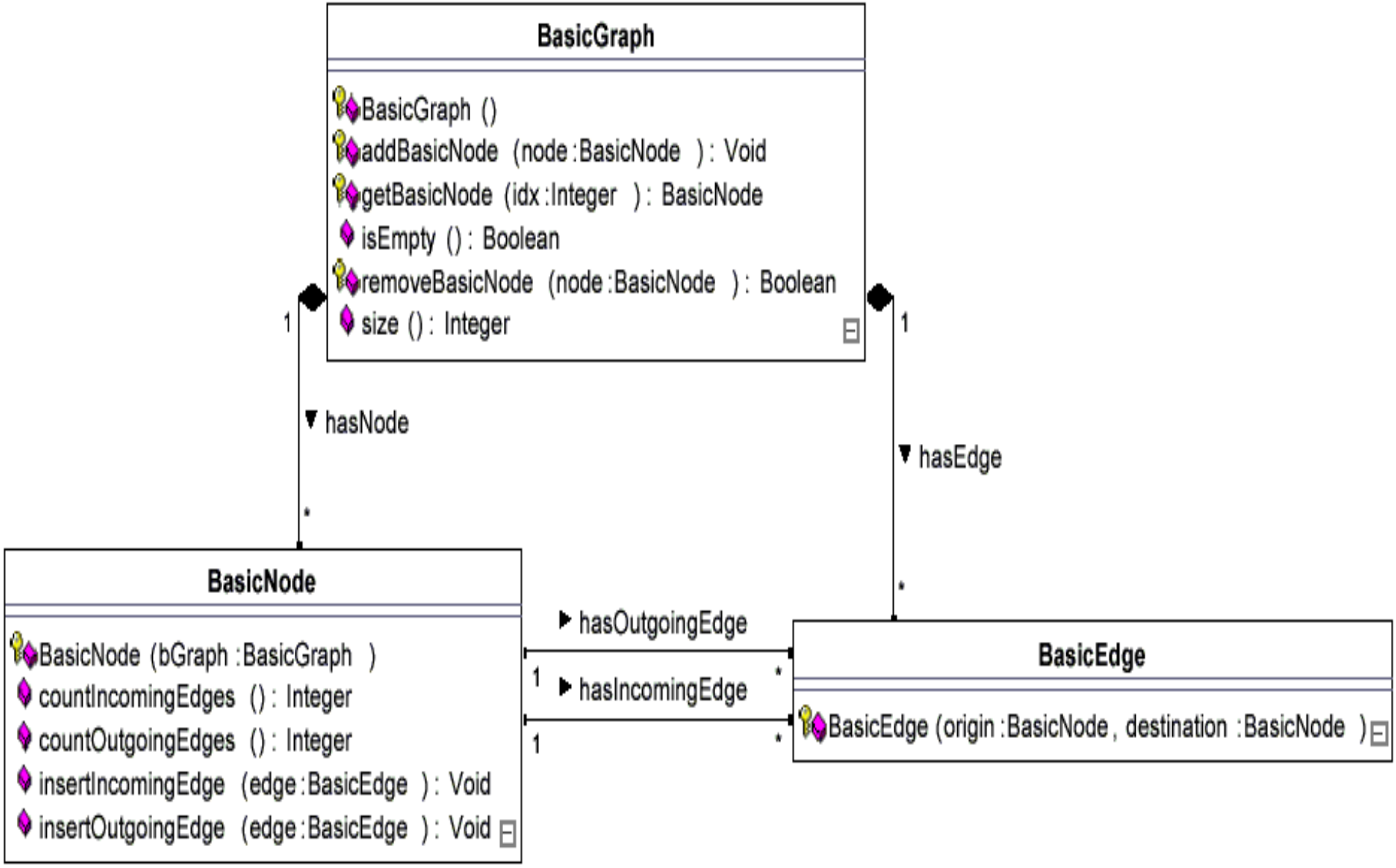


Figure 1: Initial model of BasicGraph

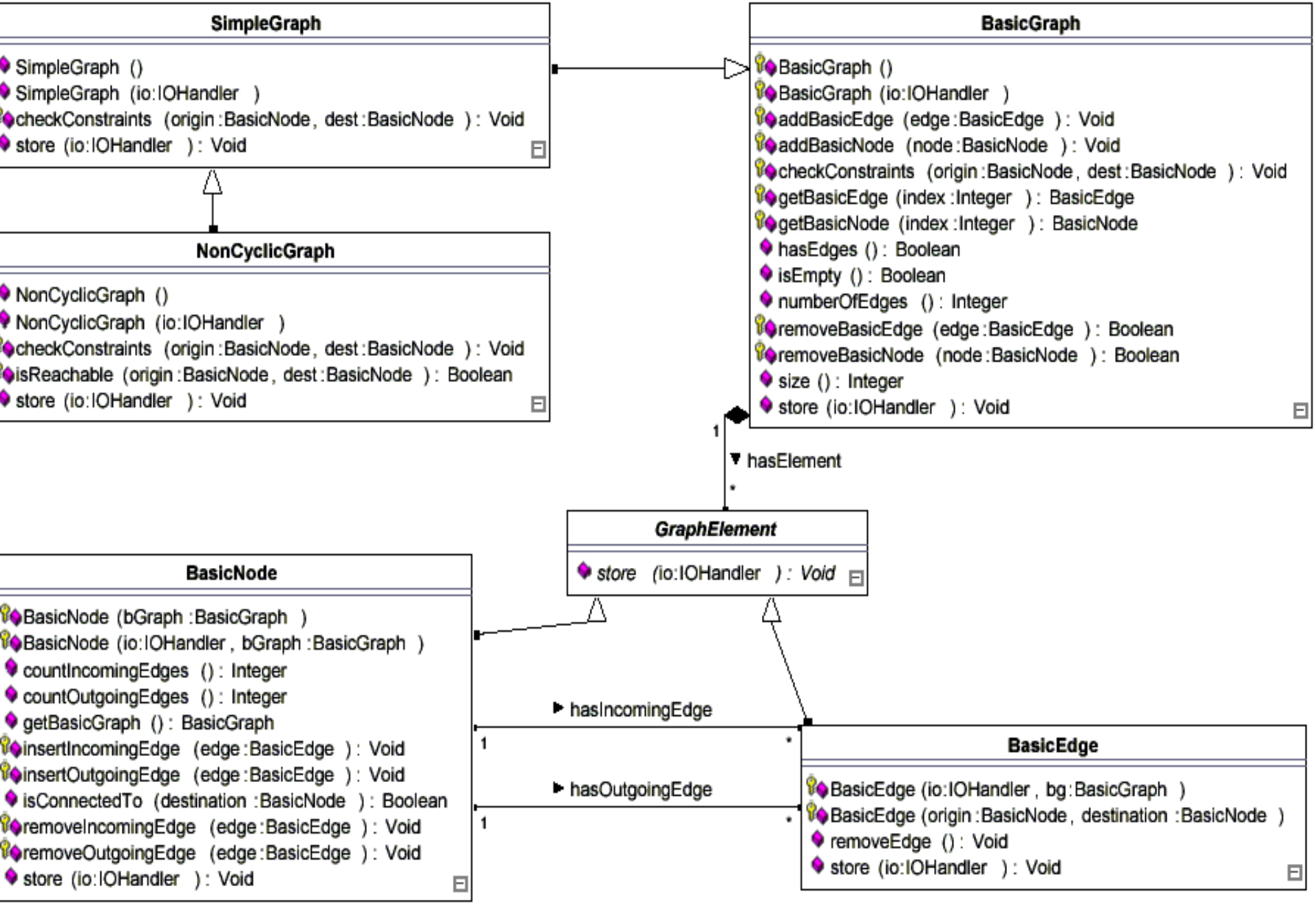


Figure 2: Final model of BasicGraph

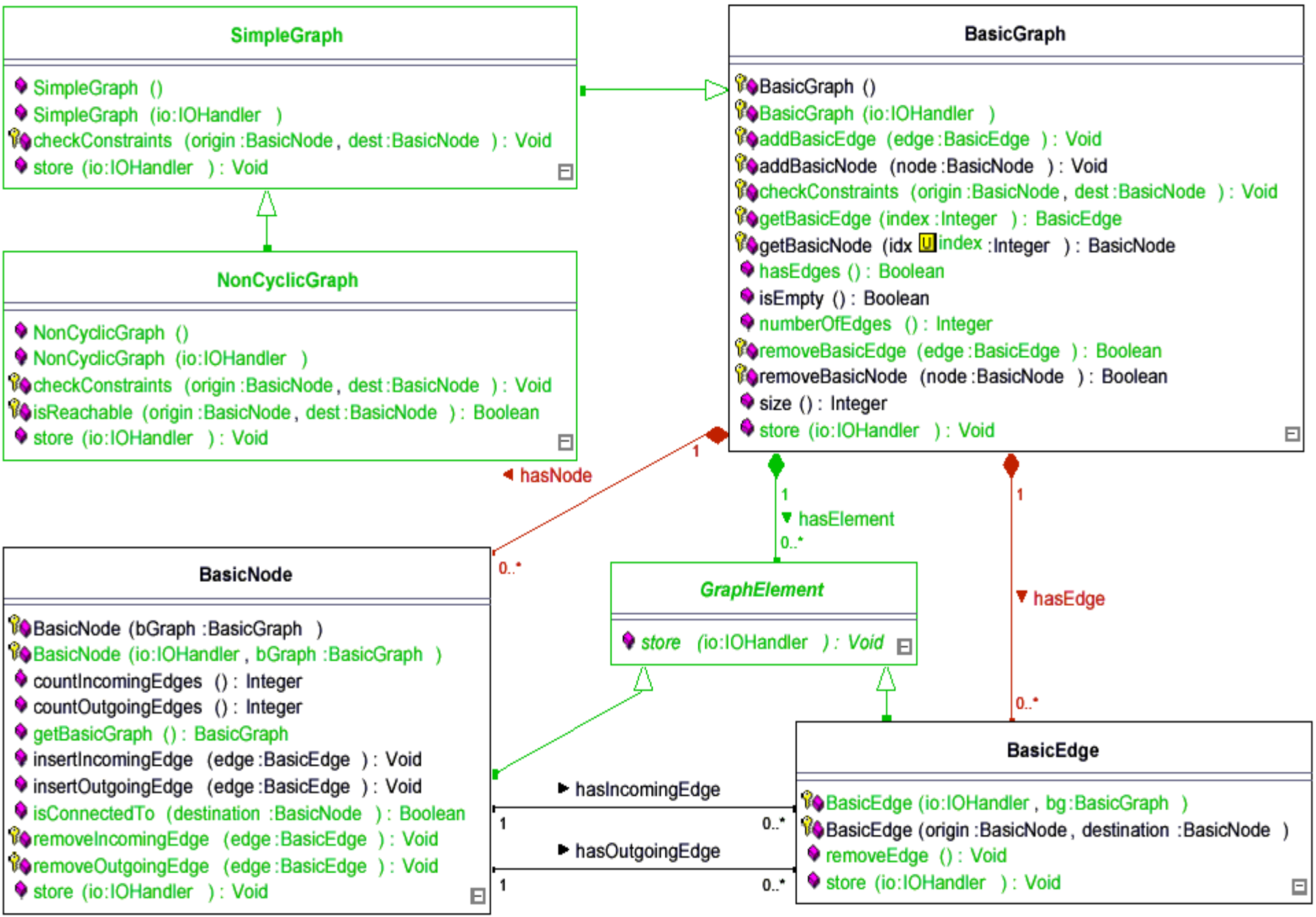
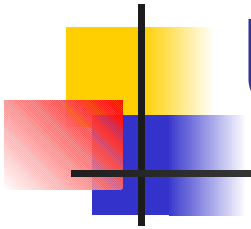


Figure 3: Document with difference information



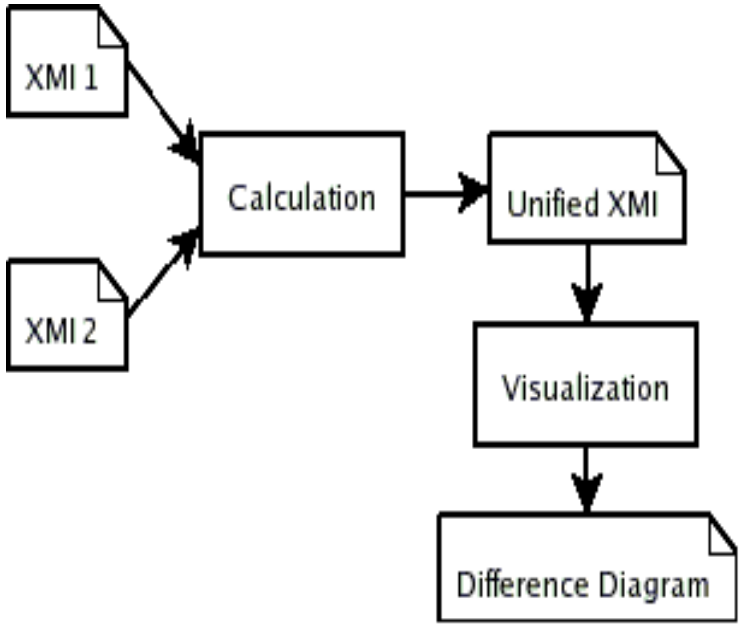
UML-Diff

1. First store the UML models as XMI files. Why can't we simply use a tool that compares XMI files?

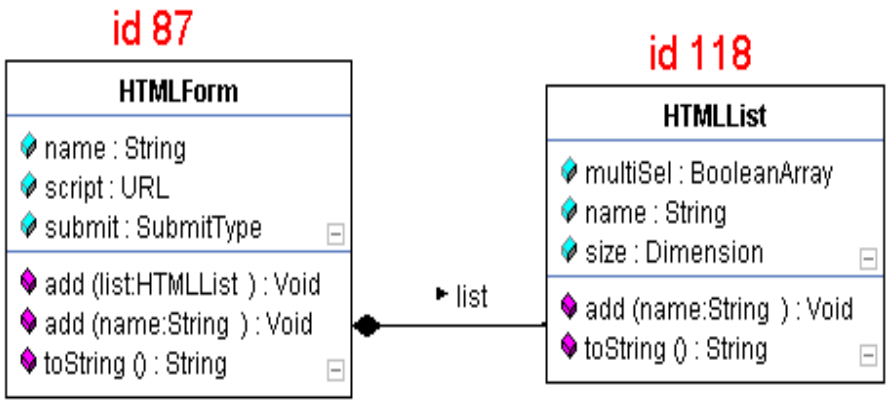
   Because XMI files can contain tool specific and other auxiliary data and the order in which elements are stored in XMI files depend on tool used for conversion, this leads to many irrelevant textual differences.

UML-Diff

1. Second we interpret XMI files as graphs, the main structure of the graph is a tree which contains references (idrefs in XMI). See next slide!
2. We then perform the difference on the trees, and generate an XMI file containing difference information.



```
  </UML:Association>
- <UML:Association name="list" namespace="id26" xmi.id="id199">
  - <UML:Association.connection>
    - <UML:AssociationEnd aggregation="composite" isNavigable="false" name="hTMLList" ordering="unordered" visibility="public" xmi.id="id191">
      - <UML:AssociationEnd.multiplicity>
        - <UML:Multiplicity>
          - <UML:Multiplicity.range>
              <UML:MultiplicityRange lower="1" upper="1"/>
            </UML:Multiplicity.range>
          </UML:Multiplicity>
        </UML:AssociationEnd.multiplicity>
      - <UML:AssociationEnd.participant>
          <UML:Classifier xmi.idref="id87"/>
        </UML:AssociationEnd.participant>
      </UML:AssociationEnd>
    - <UML:AssociationEnd aggregation="none" isNavigable="false" name="hTMLForm" ordering="unordered" visibility="public" xmi.id="id192">
      - <UML:AssociationEnd.multiplicity>
        - <UML:Multiplicity>
          - <UML:Multiplicity.range>
              <UML:MultiplicityRange lower="1" upper="1"/>
            </UML:Multiplicity.range>
          </UML:Multiplicity>
        </UML:AssociationEnd.multiplicity>
      - <UML:AssociationEnd.participant>
          <UML:Classifier xmi.idref="id118"/>
        </UML:AssociationEnd.participant>
      </UML:AssociationEnd>
    </UML:Association.connection>
  </UML:Association>
```



UML-Diff (Data Model)

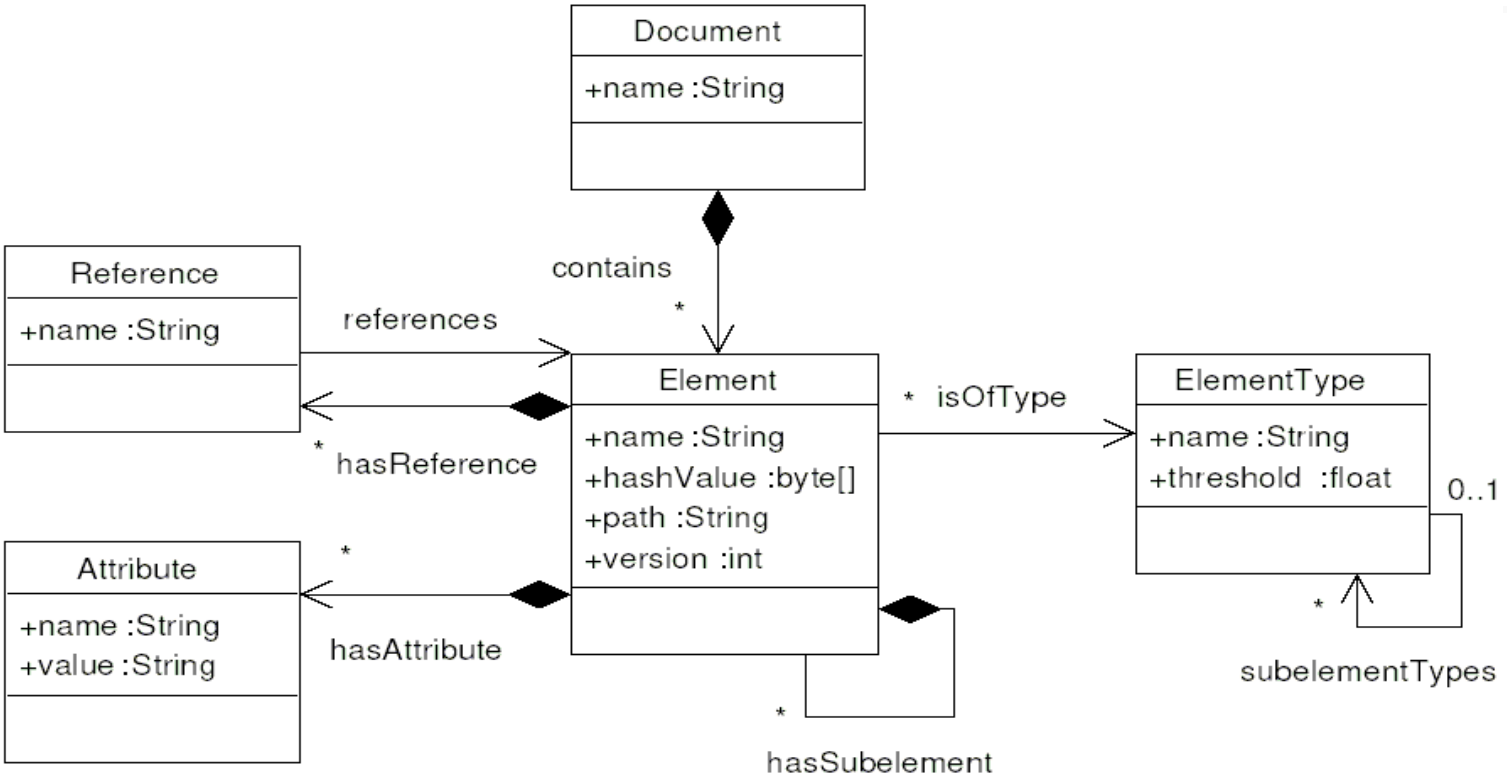


Figure 4: The data model of the difference algorithm

UML-Diff (Difference Algorithm)

- Two phases:

II. Bottom-Up:

   I. First we compare all inferior elements. Classifier elements (Class element) are compared. Elements with unique similarity to exactly one other element are matched. Similarity is noticed if its value is greater than a threshold value that is specified for each element type. In figure 5, no match was found at the classifier, parameter and operation level. Only at the Class level. In such a case we switch to phase II

UML-Diff (Difference Algorithm)

I. Top-Down:

We start with the last match in the bottom-up phase, and we propagate down to the children elements (Composite structure of our data model). Order of similar elements can differ from order of bottom up phase due to the fact that parent elements have been matched and eventually referenced elements.

We stop when all the elements have been compared in the bottom-up phase.

The result is a corresponding table consisting of matching element pairs.

UML-Diff (Algorithm phases)

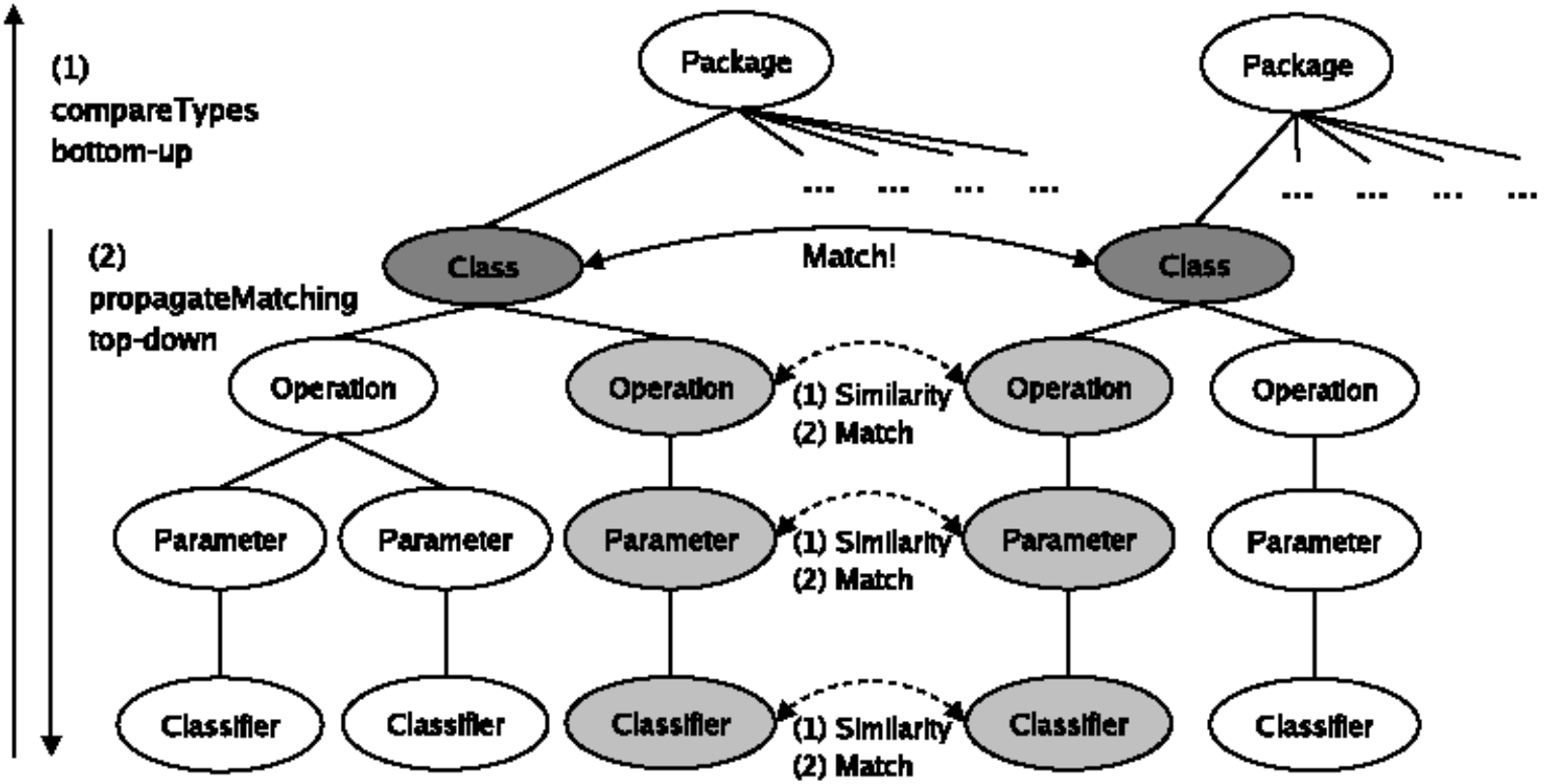


Figure 5: The bottom-up and top-down phase of the algorithm

UML-Diff (Similarity Function)

- We set up some criteria for our similarity function in a configuration file.
- Elements of the same type are compared and they are given a similarity value [0,1], where 0 means no similarity and 1 means mostly similar.
- $Sim_{e1,e2} = \sum_{c \ \varepsilon \ C} w_c \ . \ Compare_c \ (e1, e2)$

Element Type	Threshold	Criterion	weight
Class	0.4	Similarity of the class names	0.4
		Ratio of similar or matched Operations	0.2
		Ratio of similar or matched Attributes	0.2
		Generalization targets match	0.1
		Packages match	0.1

UML-Diff (Output)

- The output is simply a correspondence table consisting of all the matched element pairs as well as A unified document containing all elements in both documents exactly once is created.
- We can then simply compute the differences.
- Types of differences:

4. Structural difference (SD):

Elements that have no entry in the correspondence table.

5. Attribute difference (AD):

Corresponding elements that differ in their attribute values get an AD obtaining both, the old and the new value.

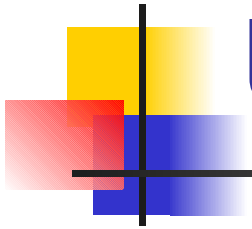
6. Reference difference (RD):

Corresponding elements whose references are different in the two original documents have a reference difference.

7. Move difference (MD):

Elements that appear to change their parent element.

UML-Diff (Optimization)

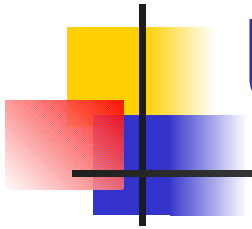


- Complexity: $O(n^2)$ where n is the number of elements in both XMI documents.
- Pre-phase: Use hashing similar to the X-Diff algorithm. We calculate the path of each element regarding to the composite structure of the data-model.
  - The determination of paths has complexity O(n) and takes place during parsing (from XMI to data-model).
  - Finding elements with identical paths takes O (n*log(n)).
  - The disadvantage of this optimization is that moves cannot be detected (different element paths!).

UML-Diff (Evaluation)

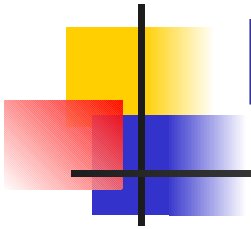
	Quantity			Det. Differences				Errors		
Testdata	**Elem.**	**Cl.**	**SD%**	**Σ**	**OD**	**SD**	**HM%**	**α**	**β**	**RT(s)**
HTMLPackage										
V0 vs. V1	171	17	55,56%	97	2	95	71%	0	0	0,5
V0 vs. V2	204	20	65,69%	136	2	134	66%	0	0	0,5
V1 vs. V2	185	23	21,08%	39	0	39	84%	0	0	0,5
UMLdiff Packages 26.04.04-15.07.04										
diagModel	804	32	30,85%	270	22	248	79%	0	0	0,9
compItems	595	35	46,55%	291	14	277	72%	0	0	0,7
calculator	1545	66	69,45%	1088	15	1073	78%	0	0	1,5
altogether	2929	99	57,87%	1740	45	1695	78%	0	0	3,7
Fujaba Packages 01.01.04-21.07.04										
asg	1755	102	7,69%	148	13	135	94%	0	0	1,1
fsa	7020	199	2,76%	208	14	194	99%	0	0	2,9
basic	8629	237	28,98%	2551	50	2501	87%	0	0	12,69
uml	13973	284	6,13%	971	114	857	93%	-	-	14,95
Ritterspiel V0=21.01.04, V1=29.03.04, V2=27.07.04										
V0 vs. V1	621	36	43,32%	282	13	269	59%	1	1	0,9
V0 vs. V2	1057	44	66,70%	723	18	705	56%	1	1	1,3
V1 vs. V2	1276	48	34,17%	449	13	436	87%	0	1	1,5
Fujaba BasicPackage Series										
01.01.-04.01.	11054	230	0,00%	4	4	0	100%	0	0	3,95
22.01.-25.01.	8648	237	28,05%	2435	9	2426	98%	0	0	5,15
09.02.-12.02.	6224	240	0,16%	18	8	10	98%	0	0	3,2
19.03.-22.03.	7666	267	0,03%	13	11	2	98%	0	0	3,70

Table 2: Test results



UML-Diff (Demo)

Demo!



References

- X-Diff Paper:

  Yuan Wang, David J. DeWitt, and Jin-Yi Cai. *X-Diff: An Effective Change Detection Algorithm for XML Documents*. In 19th International Conference on Data Engineering, March 5 - March 8, 2003 - Bangalore, India, 2003.

- UML-Diff paper:

  The following paper has been accepted but yet to be published, special thanks to Jőrg Niere who made it available for me:

  Udo Kelter, Jörg Niere. *A Generic Difference Algorithm for UML Models*.

- FUJABA:

  Thomas Klein, Ulrich~A. Nickel, Jörg Niere, and Albert Zündorf. *From UML to Java And Back Again.* Tech. Rep. tr-ri-00-216, University of Paderborn, Paderborn, Germany, September 1999.

- FUJABA Web site:

  http://www.cs.upb.de/cs/fujaba/index.html