Towards Efficient Comparison of Change-based Models

Alfa Yohannis¹  Horacio Hoyos Rodriguez*¹
Fiona Polack**²  Dimitris Kolovos¹

¹Department of Computer Science, University of York, United Kingdom
²School of Computing and Maths, Keele University, United Kingdom

{ary506, dimitris.kolovos}@york.ac.uk
*horacio_hoyos_rodriguez@ieee.org
**f.a.c.polack@keele.ac.uk

ECMFA 2019
Wednesday, 17 July 2019
Eindhoven, the Netherlands
Introduction

• A model can have many versions or variants
• Persisted as snapshots, in state-based format (e.g., XMI)
• Model comparison highlights their differences
• Comparing versions in a state-based format can be computationally expensive
• They need to be loaded in their entirety before their elements can be matched and diffed
Introduction

• **Change-Based Persistence (CBP)** persists the complete history of changes of a model instead of its eventual state.

<table>
<thead>
<tr>
<th>Math</th>
</tr>
</thead>
<tbody>
<tr>
<td>abs()</td>
</tr>
<tr>
<td>mean()</td>
</tr>
<tr>
<td>pow()</td>
</tr>
</tbody>
</table>

(a) origin

Listing 1 – The simplified XMI of the model in Fig. 1a.

```
1  <uml:Class id="x" name="Math">
2    <operation id="a" name="abs"/>
3    <operation id="b" name="mean"/>
4    <operation id="c" name="pow"/>
5  </uml:Class>
```

Listing 2 – The pseudo-formatted CBP of the model in Fig. 1a.

```
1  create x type Class
2  set x.name to "Math"
3  create a type Operation
4  set a.name to "abs"
5  create b type Operation
6  set b.name to "mean"
7  create c type Operation
8  set c.name to "pow"
9  add a to x.operations at 0
10 add b to x.operations at 1
11 add c to x.operations at 2
```
Main Goals

• Use the persisted changes to optimise:
  • Incremental model management
  • Model differencing, conflict detection, and merging

• Support Collaborative Modelling
  • Support common text-based Version Control Systems (e.g. Git, SVN) to persist changes

• Facilitate Model Analytics

• This paper addresses Model Differencing
Example

State-based Comparison:
1. Load models,
2. Match elements, and
3. Diff
EMF Compare applies a Longest Common Subsequence (LCS) algorithm to identify these differences:

\(d_{s1}:\) change \(x.name\) from “MathLib” to “MathUtil”
\(d_{s2}:\) delete \(b\)
\(d_{s3}:\) add \(d\) to \(name.operations\) at 1
\(d_{s4}:\) move \(c\) in \(name.operations\) from 1 to 2

Figure 3 – A model comparison of the left and right models in Listings 3 and 4.
Change-based Model Differencing (1)

1  <uml:Class id="x" name="Math">
2   <operation id="a" name="abs"/>
3   <operation id="b" name="mean"/>
4   <operation id="c" name="pow"/>
5  </uml:Class>

Listing 1 – The simplified XMI of the model in Fig. 1a.

Listing 5 – The appended changes made by Bob to produce the model in Fig. 1b (left version).

12  set x.name from "Math" to "MathLib"
13  create d type Operation
14  set d.name to "sqrt"
15  add d to x.operations at 1
16  remove b in x.operations at 2
17  delete b

Listing 6 – The appended changes made by Alice to produce the model in Fig. 1c (right version).

Listing 3 – The simplified XMI of the left model in Fig. 1b.

1  <uml:Class id="x" name="MathLib">
2   <operation id="a" name="abs"/>
3   <operation id="d" name="sqrt"/>
4   <operation id="c" name="pow"/>
5  </uml:Class>

Listing 4 – The simplified XMI of the right model in Fig. 1c.
Change-based Model Differencing (2)

- **Steps:**
  1. Change Event Loading
  2. Element Tree Construction
  3. Diff Computation

- **Change Event Loading**
  1. Load events from files into memory

Listing 5 – The appended changes made by Bob to produce the model in Fig. 1b (left version).

Listing 6 – The appended changes made by Alice to produce the model in Fig. 1c (right version).
Change-based Model Differencing (3)

• Element Tree Construction

• Left

Listing 5 – The appended changes made by Bob to produce the model in Fig. 1b (left version).

Listing 3 – The simplified XMI of the left model in Fig. 1b.

Figure 5 – The elementTree after processing all left change events.
Change-based Model Differencing (4)

- Element Tree Construction
- Right

Listing 6 – The appended changes made by Alice to produce the model in Fig. 1c (right version).

```
12 move a in x.operations from 0 to 2
13 set x.name from "Math" to "MathUtil"
```

Listing 4 – The simplified XMI of the right model in Fig. 1c.

```
1 <uml:Class id="x" name="MathUtil">
2   <operation id="b" name="mean"/>
3   <operation id="c" name="pow"/>
4   <operation id="a" name="abs"/>
5 </uml:Class>
```

Figure 6 – The elementTree after processing all left and right change events.
Change-based Model Differencing (5)

Diff Computation
Diffs = a set of operations to make right side equal to left side

- $D_{C1}$: change $x$.name from "MathLib" to "MathUtil"
- $D_{C2}$: delete $b$
- $D_{C3}$: add $d$ to $\text{name.operations}$ at 1
- $D_{C4}$: move $a$ in $\text{name.operations}$ from 2 to 1

Figure 6 – The elementTree after processing all left and right change events.
Change-based Model Differencing (6)

• Change-based:

  \( d_{c1} \): change x.name from “MathLib” to “MathUtil”
  \( d_{c2} \): delete b
  \( d_{c3} \): add d to name.operations at 1
  \( d_{c4} \): move a in name.operations from 2 to 1

• State-based:

  \( d_{s1} \): change x.name from “MathLib” to “MathUtil”
  \( d_{s2} \): delete b
  \( d_{s3} \): add d to name.operations at 1
  \( d_{s4} \): move c in name.operations from 1 to 2
Performance Evaluation: Mixed Operations

- Original model
  - 1.6 million elements
  - 224 MBs (XMI file)

- Two original models were randomly modified
  - 1.1 million changes to each model
  - set, move, add, delete types of changes
  - 1 change can produce more than 1 events

- Ratio

Figure 8 – total elements, affected elements, and diffs
Performance Evaluation: Mixed Operations

- Change-based vs. State-based Model Differencing
Performance Evaluation: Mixed Operations

- Breakdown view of comparison time

(a) change-based comparison time

(b) state-based comparison time
Performance Evaluation: Mixed Operations

- Breakdown view of memory footprint

(c) change-based memory footprint

(d) state-based memory footprint
Performance Evaluation on Comparison Time: Homogenous Operations

(a) add-only

(b) delete-only

(c) move-only

(d) change-only
Performance Evaluation on Memory Footprint: Homogenous Operations

(a) add-only

(b) delete-only

(c) move-only

(d) change-only
Conclusions and Future Work

• **Conclusions**
  - Proposed change-based model comparison
  - It exploits the nature of change-based persistence; only compare the last set of changes between two versions
  - It can compare models faster than state-based model comparison
  - **Drawback**: It needs to load change events; may requires more memory than state-based approach
  - Arguably, diff and merge operations are usually performed on smaller deltas

• **Future Work**
  - Conflict detection and merging of change-based models
• Prototype: [https://github.com/epsilonlabs/emf-cbp](https://github.com/epsilonlabs/emf-cbp)

```plaintext
12  set x.name from "Math" to "MathLib"
13  create d type Operation
14  set d.name to "sqrt"
15  add d to x.operations at 1
16  remove b in x.operations at 2
17  delete b
```

**Figure 5** – The elementTree after processing all left change events.

**Figure 6** – The elementTree after processing all left and right change events.