Optimising Change-based Model's Loading Performance

Alfa Yohannis
Computer Science Department
University of York
York, United Kingdom
ary506@york.ac.uk

Dimitris Kolovos
Computer Science Department
University of York
York, United Kingdom
dimitris.kolovos@york.ac.uk

Fiona Polack
Computer Science Department
University of York
York, United Kingdom
fiona.polack@york.ac.uk

ABSTRACT

This paper provides a sample of a LaTeX document which conforms, somewhat loosely, to the formatting guidelines for ACM SIG Proceedings.[1]

CCS CONCEPTS

• **Software and its engineering → Model-driven software engineering**; *Software performance*; Software architectures;

KEYWORDS

ACM proceedings, LaTeX, text tagging

ACM Reference format:
Alfa Yohannis, Dimitris Kolovos, and Fiona Polack. 2018. Optimising Change-based Model's Loading Performance. In *Proceedings of International Conference on Software Engineering, Gothenburg, Sweden, May-June 2018 (ICSE'18),* 2 pages.
https://doi.org/10.475/123_4

1 INTRODUCTION

2 CHANGE-BASED MODELS

3 EVENTS

4 MODEL HISTORY

5 OPTIMISATION PRINCIPLES

5.1 The Last Value is All that Matters

Lst. 1 shows an EOL code to create a model. It performs set and unset operations to an attribute `name` of an object `node`. When executed, this code produces a CBP XML representation consists of 6 events shown in Lst. 2. The 1st event registers the package `node` and is followed by other events that create an object, with id "0", as an instance of a `Node` class (line 1) and then add the object to the model's resource to make the object a part of the model (line 2). The code then sets the attribute `name`'s value to "Old Name!" (line 3), then nullify/unset it (line 4), and finally sets it to "New Name!" as its latest value (line 5). Lst. 1 shows that the last value's of `node.name` is all that matters (line 4). Any previous value assignment to the `name` attribute can be ignored (line 1 and 2), since the final version of the model will always be the same.

Listing 1: The EOL code for Setting Attribute.

```
1  var node = new Node;
2  node.name = "Old Name!";
3  node.name = null;
4  node.name = "New Name!";
```

Listing 2: The generated change-based representation of Lst. 1.

```
1  0 <register epackage="node"/>
2  1 <create eclass="Node" epackage="node" id="0"/>
3  2 <add-to-resource position="0"><value eobject="0"/></add-to-
       resource>
4  3 <set-eattribute name="name" target="0"><value literal="Old
       Name!"/></set-eattribute>
5  4 <unset-eattribute name="name" target="0"/>
6  5 <set-eattribute name="name" target="0"><value literal="New
       Name!"/></set-eattribute>
```

All events that involve the object "0" are recorded in an object history. The object history records the line number of occurrence of each event (Lst. 3). For example, `SetEAttributeValue` event contains line 3 and 5 based on its occurrence.

Listing 3: The event-line relationships recorded by an EObjectHistory of Lst. 1.

```
1  EObject: 0
2      CreateEObjectEvent = [[1]]
3      AddToResourceEvent = [[2]]
4      EAttribute:
5          name:
6              UnsetEAttributeEvent = [[4]]
7              SetEAttributeEvent = [[3], [5]]
```

The history enables us to track which lines can be ignored for optimisation so that not every event is executed to produce the same model. From Lst. 3, we can deduce that line 3 and 4 can be ignored since the `SetEAttributeValue`'s last event always sets the end value of the `name` attribute.

Listing 4: The optimised change-based Representation of Lst. 1.

```
1  0 <register epackage="node"/>
2  1 <create eclass="Node" epackage="node" id="0"/>
3  2 <add-to-resource position="0"><value eobject="0"/></add-to-
       resource>
4  5 <set-eattribute name="name" target="0"><value literal="New
       Name!"/></set-eattribute>
```

5.2 Unset Ignores Itself and Its Previous Sets and Unsets

Listing 5: The EOL code for Nullyfying/Unsetting Attribute.

```
1  var node = new Node;
2  node.name = "Old Name!";
```

[1]This is an abstract footnote

ICSE'18, May-June 2018, Gothenburg, Sweden

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

```
3   node.name = null;
4   node.name = "New Name!";
5   node.name = null;
```

Listing 6: The generated change-based representation of Lst. 5.

```
1   0 <register epackage="node"/>
2   1 <create eclass="Node" epackage="node" id="0"/>
3   2 <add-to-resource position="0"><value eobject="0"/></add-to-
        resource>
4   3 <set-eattribute name="name" target="0"><value literal="Old
        Name!"/></set-eattribute>
5   4 <unset-eattribute name="name" target="0"/>
6   5 <set-eattribute name="name" target="0"><value literal="New
        Name!"/></set-eattribute>
7   6 <unset-eattribute name="name" target="0"/>
```

Listing 7: The event-line relationships recorded by an EObjectHistory of Lst. 5.

```
1   EObject: 0
2       CreateEObjectEvent = [[1]]
3       AddToResourceEvent = [[2]]
4       EAttribute:
5           name:
6               UnsetEAttributeEvent = [[4], [6]]
7               SetEAttributeEvent = [[3], [5]]
```

Listing 8: The optimised change-based Representation of Lst. 5.

```
1   0 <register epackage="node"/>
2   1 <create eclass="Node" epackage="node" id="0"/>
3   2 <add-to-resource position="0"><value eobject="0"/></add-to-
        resource>
```

6 PERFORMANCE

Listing 9: The EOL code for creating deep tree of nodes.

```
1   var eRoot = new Node;
2   eRoot.name = "0";
3   for(i in Sequence{1..40}){
4       var node = new Node;
5       node.name = i.asString();
6       eRoot.valNodes.add(node);
7             eRoot = node;
8   }
```

6.1 Loading Deep Tree

7 RELATED WORK

8 CONCLUSIONS

[1].

ACKNOWLEDGMENTS

REFERENCES

[1] Leslie Lamport. 1986. *LaTeX: A Document Preparation System.* Addison-Wesley, Reading, MA.

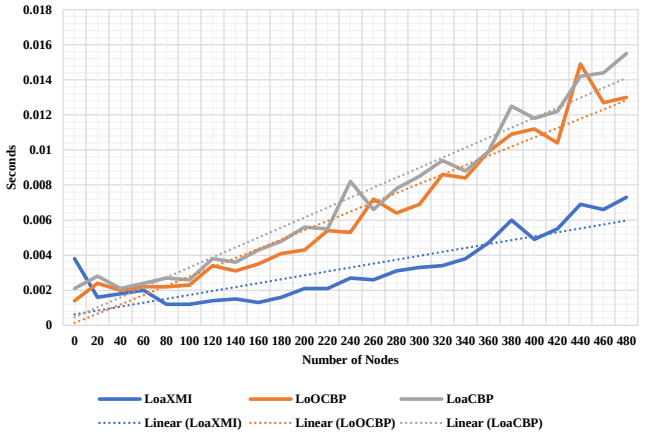

Figure 1: Loading performance between loading XMI, optimised CBP, and CBP.