Model Driven Software Engineering (MDSE)

Technische Universiteit Eindhoven University of Technology

Model Clone Detection in Ecore meta-models

Software Engineering and Technology (SET) Group

Jia Zhang
email: j.zhang.3@student.tue.nl

Ana-Maria Sutii
email: a.m.farcasi@tue.nl

Mark Van Den Brand
email: M.G.J.v.d.Brand@tue.nl

Introduction

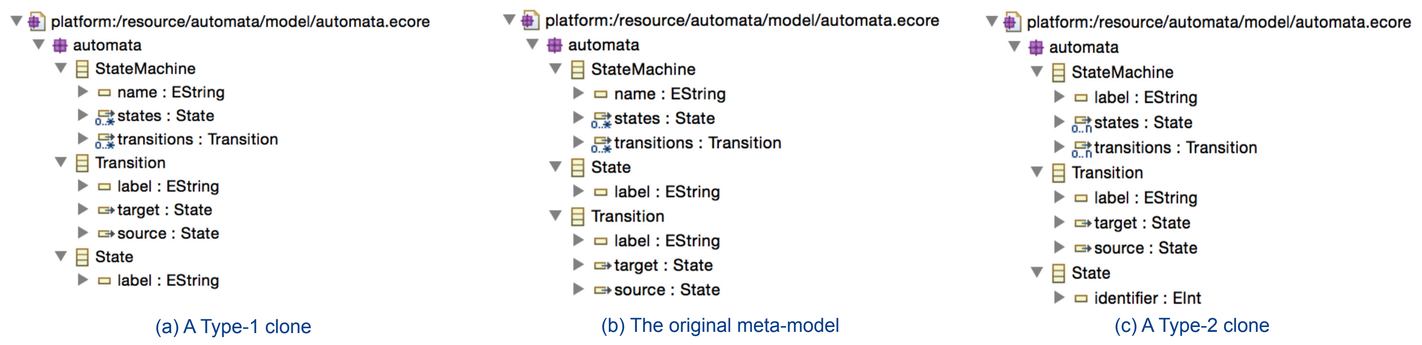
Model clones, similar to code clones, are duplicate model fragments that can be considered as a source of software defects. On the other hand, the duplicated model fragments can be considered as reusable patterns as well. As Model-Driven Engineering (MDE) is becoming more popular, model clone detection becomes an interesting topic to address. The text-based and graph-based techniques have been used to detect model clones in several kinds of models, e.g. UML models and Matlab/Simulink models [1,2].

In this work, we explore Ecore meta-model clones by defining clone types of Ecore meta-model and applying a text-based technique to detect Ecore model clones. We present tool support that was created based on the existing code clone detection tool NICAD [3].

Definition of Ecore meta-model clone types

Code clones are categorized into four types [2]. Model clone types can be derived from the four code clone types. For different types of models (e.g. Simulink models and Ecore models) the clone type definition can be different. In this work, we focus on Type-1 and Type-2 Ecore meta-model clones.

Type-1 clones: Identical model fragments except for variations in visual presentation, layout and formatting.
Type-2 clones: Structurally identical model fragments except for variations in names,multiplicity, modifiers, types,visual presentation, layout and formatting.



(a) A Type-1 clone (b) The original meta-model (c) A Type-2 clone

Figure 1. Examples of Type-1 and Type-2 clones

Experiments & Results

In the experiments, we transformed Ecore meta-models to a textual representation “Emfatic” [4] and detected clones in Emfatic model code. We considered the meta-models of the languages Ecore, OCL, OCLCST and UML. Two experiments are conducted to identify clones in/across meta-models of these languages. The first experiment is to detect clones in the meta-models of Ecore, OCL, OCLCST and UML respectively. Table 1 shows the number of clones detected in these four meta-models. In the second experiment, we compared these four meta-models pair-wise and detected cross-clones (i.e. clones that exist in different models) for each pair. Table 2 shows the number of clones detected in every pair of meta-models.

Table 1. results of experiment 1

	nr. extracted classes	nr. clone pairs	nr. clusters
Ecore	20	0	0
OCL	57	93	8
OCLCST	49	89	12
UML	243	337	29

Table 2. results of experiment 2

	Ecore		OCL		OCLCST	
	nr. clone pairs	nr. clusters	nr. clone pairs	nr. clusters	nr. clone pairs	nr. clusters
UML	21	6	324	14	240	8
Ecore	/	/	6	3	13	3
OCL	/	/	/	/	122	4

As can be seen from Figure 2, the two example clone pairs indeed satisify the definition of Type-2 model clone, namely, the classes in a clone pair are structurally identical.

Lines 2108 - 2124 of test/emfatic/uml/UML.emf

```
class Deployment extends Dependency {

  @GenModel(documentation="The specification of properties that parameterize the deployment and execution of one or more Artifacts.
<p>From package UML::Deployments.</p>")
  @subsets
  !ordered val DeploymentSpecification[*]#deployment configuration;

  @GenModel(documentation="The Artifacts that are deployed onto a Node. This association specializes the supplier association.
<p>From package UML::Deployments.</p>")
  @subsets
  !ordered ref DeployedArtifact[*] deployedArtifact;

  @GenModel(documentation="The DeployedTarget which is the target of a Deployment.
<p>From package UML::Deployments.</p>")
  @subsets
  !ordered ref DeploymentTarget[1]#deployment location;
}
```

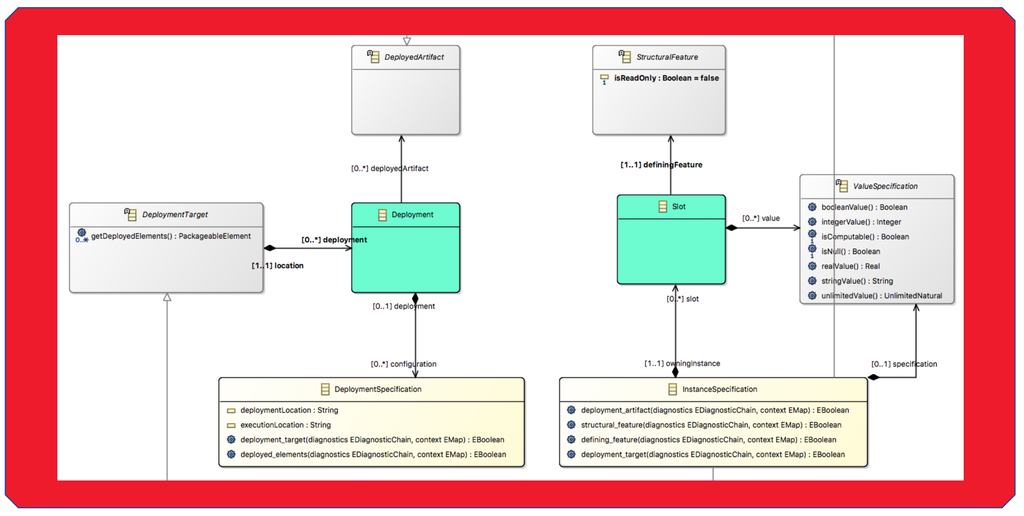
Lines 3888 - 3903 of test/emfatic/uml/UML.emf

```
class Slot extends Element {

  @GenModel(documentation="The StructuralFeature that specifies the values that may be held by the SSlot.
<p>From package UML::Classification.</p>")
  !ordered ref StructuralFeature[1] definingFeature;

  @GenModel(documentation="The value or values held by the Slot.
<p>From package UML::Classification.</p>")
  @subsets
  val ValueSpecification[*] value;

  @GenModel(documentation="The InstanceSpecification that owns this Slot.
<p>From package UML::Classification.</p>")
  @subsets
  !ordered ref InstanceSpecification[1]#slot owningInstance;
}
```



Lines 186 - 191 of test/emfatic/ocl/OCL.emf

```
class BooleanLiteralExp <C>  extends PrimitiveLiteralExp<C> {

  @GenModel(documentation="self.type.name = 'Boolean'")
  op boolean boolean_type( ecore.EDiagnosticChain diagnostics, ecore.EMap<EJavaObject, EJavaObjectt> context);
  attr Boolean booleanSymbol;
}
```

Lines 396 - 401 of test/emfatic/ocl/OCL.emf

```
class RealLiteralExp <C>  extends NumericLiteralExp<C> {

  @GenModel(documentation="self.type.name = 'Real'")
  op boolean real_type(  ecore.EDiagnosticChain diagnostics,  ecore.EMap<EJavaObject, EJavaObject>> context);
  attr Double realSymbol;
}
```

Lines 407 - 412 of test/emfatic/ocl/OCL.emf

```
class StringLiteralExp <C>  extends PrimitiveLiteralExp<C> {

  @GenModel(documentation="self.type.name = 'String'")
  op boolean string_type(  ecore.EDiagnosticChain diagnostics,  ecore.EMap<EJavaObject, EJavaObjecct> context);
  attr String stringSymbol;
}
```

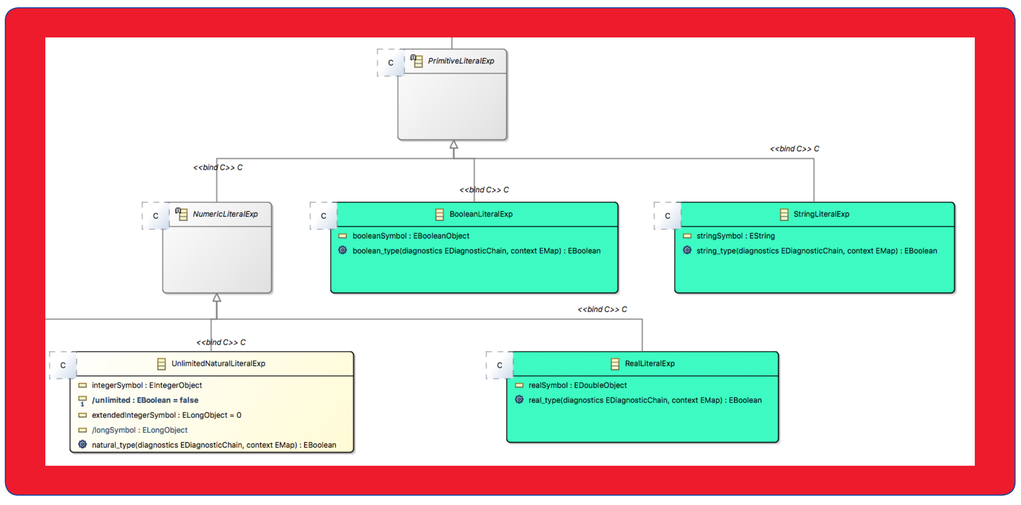


Figure 2. Example clone pairs

Limitations & Outlook

This work provides an approach to detect model clones in Ecore meta-models using code clone detection techniques. The results show that some of the model clones can be considered as patterns and may be reused in the development of DSLs. However, due to time limitations, we analysed only some of the detected clones. Thus, the number of clone pairs that are useful is unknown. In the future, we plan to analyse all clone pairs to find more patterns. Moreover, the detected clones are limited to the class level due to the non- existence of grouping facilities in Ecore. The bigger granularity (i.e. a group of classes and their relations) cannot be detected now. If this problem can be solved, Type-3 clones will be detected easier.

/ Department of Mathematics and Computer Science

[1] Alalfi, M.H., Cordy, J.R., Dean, T.R., Stephan, M., Stevenson, A. Models are code too: Near-miss clone detection for Simulink models. IEEE International Conference on Software Maintenance, ICSM pp. 295–304 (2012).

[2] Roy, C.K., Cordy, J.R., Koschke, R. Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. Science of Computer Programming 74(7), 470–495 (2009).

[3] NiCad3 Clone Detector. http://www.txl.ca/nicaddownload.html.

[4] Emfatic. https://www.eclipse.org/emfatic/