Alf-verifier: An Eclipse Plugin for Verifying Alf/UML Executable Models

Elena Planas[1], David Sanchez-Mendoza[1], Jordi Cabot[2], and Cristina Gómez[3]

[1] Universitat Oberta de Catalunya (Spain), {eplanash,dsanchezmen}@uoc.edu
[2] École des Mines de Nantes - INRIA (France), jordi.cabot@inria.fr
[3] Universitat Politècnica de Catalunya (Spain), cristina@essi.upc.edu

Abstract. In this demonstration we present an Eclipse plugin that implements a lightweight method for verifying fine-grained operations at design time. This tool suffices to check that the execution of the operations (specified in Alf Action Language) is consistent with the integrity constraints defined in the class diagram (specified in UML) and returns a meaningful feedback that helps correcting them otherwise.

1 Introduction

Executable models are models described in sufficient detail so that they can be sistematically implemented/executed in the production environment. Executable models play a cornerstone role in the Model-Driven Development (MDD) paradigm, where models are the core artifacts of the development lifecycle and the basis to generate the final software implementation.

Executable models are not a new concept [7] but are now experiencing a comeback. As a relevant example, the OMG has recently published the first version of the "Foundational Subset for Executable UML Models" (fUML) standard [4], an executable subset of the UML that can be used to define, in an operational style, the structural and behavioural semantics of systems. The OMG has also published a beta version of the "Action Language for fUML" (Alf) standard [3], a concrete syntax conforming to the fUML abstract syntax, that provides a textual notation to specify the fine-grained behaviour of systems.

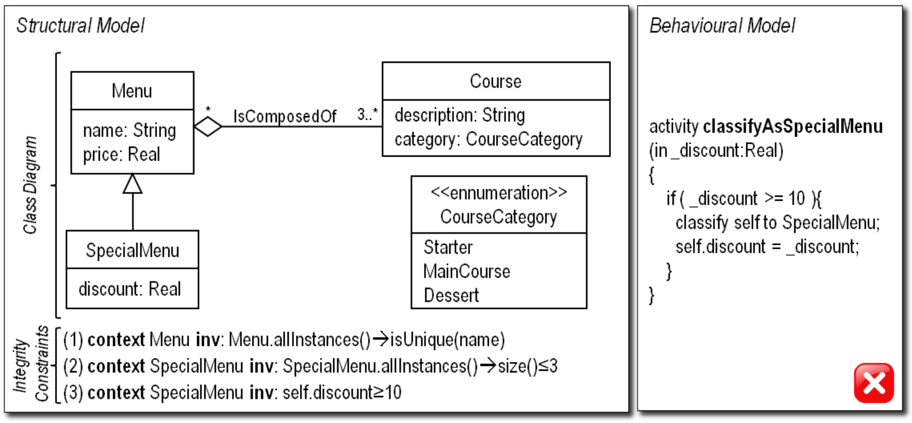
Given the increasing importance of executable models and their impact on the final quality of software systems, the existence of tools to verify the correctness of such models is becoming crucial. Unfortunately, despite the number of research works targetting the verification of behavioural specifications, their computational cost and poor feedback makes them difficult to integrate in current software development processes and tools.

In order to overcome the limitations of the existing tools, in this demostration we present an Eclipse plugin that implements a lightweight method for verifying executable models. Our tool focuses on the verification of the *Strong Executability* correctness property of operations (attached to domain classes of a UML class diagram) specified by means of Alf Action Language.

To the best of our knowledge, our tool is the first one which deals with verification of Alf specifications.

2 Executability of operations

We consider that an operation is *Strongly Executable* (SE) if it is always successfully executed, i.e. if every time we execute the operation (whatever values are given as arguments for its parameters), the effect of the actions included in the operation evolves the initial state of the system to a new system state that satisfies all integrity constraints of the structural model.



(1) **context** Menu **inv**: Menu.allInstances()→isUnique(name)
(2) **context** SpecialMenu **inv**: SpecialMenu.allInstances()→size()≤3
(3) **context** SpecialMenu **inv**: self.discount≥10

```
activity classifyAsSpecialMenu
(in _discount:Real)
{
  if ( _discount >= 10 ){
    classify self to SpecialMenu;
    self.discount = _discount;
  }
}
```

Fig. 1. Excerpt of a menus management system executable model.

As an example, consider the executable model shown in Fig. 1, meant to model the menus offered by a restaurant. Note that the operation "classifyAsSpecialMenu", which classifies a menu as special menu, is not SE since after its execution the maximum integrity constraint of class *SpecialMenu* (second constraint) may be violated (in particular, when the system state where the operation is applied contains already three special menus).

In [5] we have published the teoretical background of a lightweight method for verifying the strong executability of Alf operations. This method takes as input an executable model (composed by a UML class diagram and a set of Alf operations) and returns either a positive answer, meaning that the operation is SE, or a corrective feedback, consisting in a set of actions and guards that should be added to the operation in order to make it SE.

3 The verification tool

We developed our tool as an Eclipse plugin which can be downloaded from [6].

Figure 2 shows the general view of the tool architecture. As a first step, the designer specifies the UML executable model she wants to deal with. The executable model is composed by: (1) a UML class diagram and a set of OCL integrity constraints modelled using the graphical modelling environment provided by UML2Tools [1], an Eclipse Graphical Modeling Framework for manipulating UML models; and (2) a set of Alf operations specified in a text file with *.alf* extension. Once the executable model is provided, the designer has to click on *"Verify strong executability"* and the core of our method is invoked. This

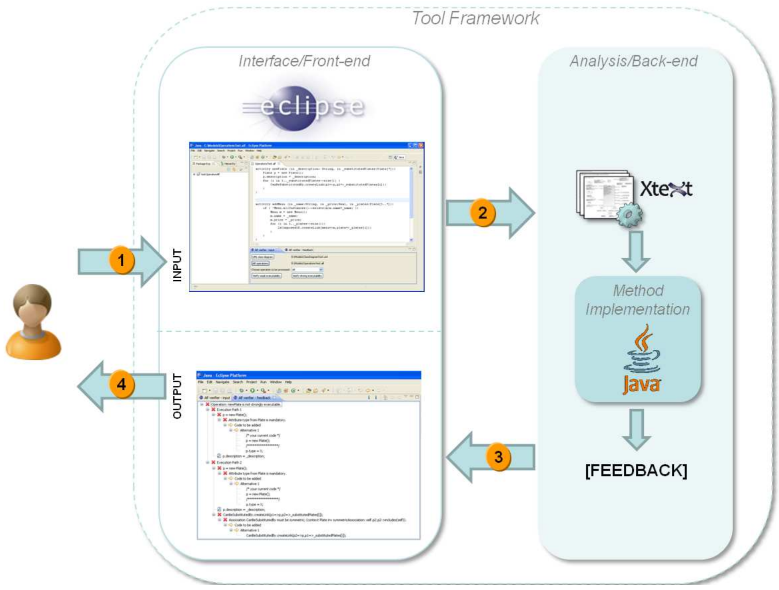
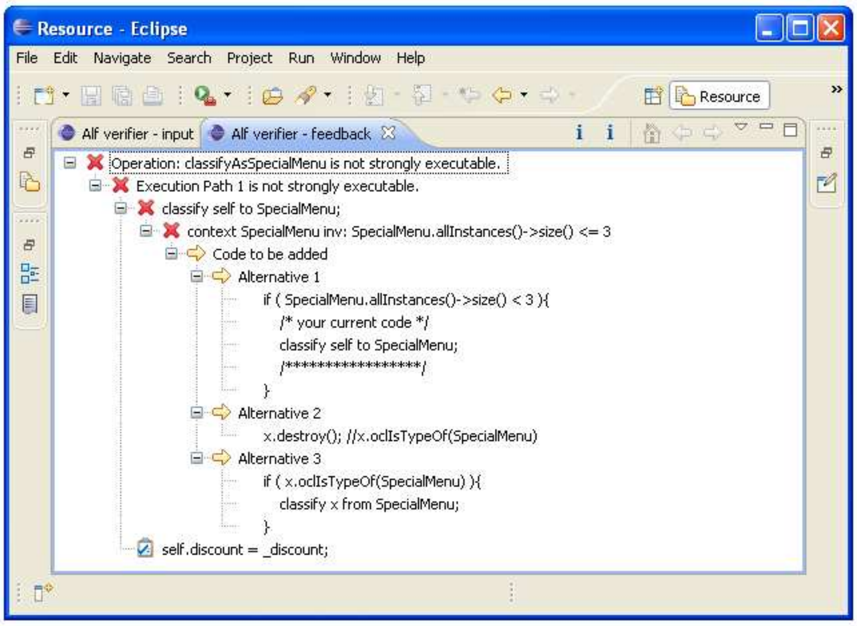


Fig. 2. General architecture or our tool.

method is implemented as a set of Java classes extended with the libraries of the UML2Tools (to interact with the input UML model) and Xtext [2] (to parse the Alf operations and instantiate them as Java classes). Finally, the feedback provided by our method is displayed, integrated into the Eclipse interface.

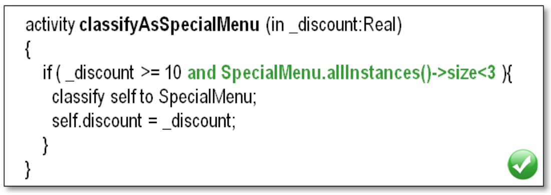


```
Operation: classifyAsSpecialMenu is not strongly executable.
  Execution Path 1 is not strongly executable.
    classify self to SpecialMenu;
      context SpecialMenu inv: SpecialMenu.allInstances()->size() <= 3
        Code to be added
          Alternative 1
            if ( SpecialMenu.allInstances()->size() < 3 ){
              /* your current code */
              classify self to SpecialMenu;
              /*****************/
            }
          Alternative 2
            x.destroy(); //x.oclIsTypeOf(SpecialMenu)
          Alternative 3
            if ( x.oclIsTypeOf(SpecialMenu) ){
              classify x from SpecialMenu;
            }
    self.discount = _discount;
```

Fig. 3. Screenshot of the feedback provided by our Eclipse plugin.

Fig. 3 shows the feedback provided by our tool when verifying the operation "classifyAsSpecialMenu" introduced in Fig. 1. This operation is not strongly executable since the action `classify self to SpecialMenu` may violate the constraint `(2) SpecialMenu.allInstances()->size()`$\leq$`3` (when the system state where the operation is applied already contains three special menus).

To avoid this violation, our tool proposes three alternatives: (1) add a guard ensuring there are less than three special menus; (2) destroy an existing special menu; or (3) reclassify an existing special menu from *SpecialMenu*. Fig. 4 shows the operation once the first alternative (hightlighted in green) has been applied. After applying this change, we can ensure every execution of the operation "classifyAsSpecialMenu" will be safe. Additional information of our tool may be found on [6].



```
activity classifyAsSpecialMenu (in _discount:Real)
{
   if ( _discount >= 10 and SpecialMenu.allInstances()->size<3 ){
     classify self to SpecialMenu;
     self.discount = _discount;
   }
}
```

Fig. 4. Operation "classifyAsSpecialMenu" repaired.

4 Summary

We have proposed an Eclipse plugin [6] for assisting the designers during the specification of executable behavioural models. Our plugin implements a lightweight method [5] that verifies the *Strong Executability* (SE) of Alf operations wrt the integrity constraints imposed by the class diagram. The main characteristics of our tool are its efficiency (since no simulation/animation of the behaviour is required) and feedback (for non-executable operations, it is able to identify the source of the inconsistency and suggest possible corrections).

References

1. UML2Tools, http://www.eclipse.org/modeling/mdt/?project=uml2tools (last visit: May 2012).
2. Xtext, www.xtext.org/ (last visit: May 2012).
3. OMG. Concrete Syntax for UML Action Language (Action Language for Foundational UML), version Beta 1, www.omg.org/spec/ALF. 2010.
4. OMG. Semantics Of A Foundational Subset For Executable UML Models (fUML), version 1.0, www.omg.org/spec/FUML. 2011.
5. E. Planas, J. Cabot, and C. Gómez. Lightweight Verification of Executable Models. In *ER*, pages 467–475, 2011.
6. E. Planas and D. Sanchez-Mendoza. Alf-verifier: A lightweight tool for verifying UML-Alf executable models. http://code.google.com/a/eclipselabs.org/p/alf-verifier/, 2012.
7. M. J. B. Stephen J. Mellor. *Executable UML: A Foundation for Model-Driven Architecture*. Addison-Wesley, 2002.