Animation of UML-B State-machines

Vitaly Savicks, Colin Snook, Michael Butler

University of Southampton, Southampton SO17 1BJ, UK
{vs02, cfs, mjb}@ecs.soton.ac.uk

1 Introduction

Animation of formal models lets us test whether we constructed the model we intended. For State-based languages such as Event-B[1], animation is the primary mechanism for performing validation. The model is 'executed' so that the modeller can check that the state changes as expected and that the correct events are enabled ready for the next animation step. Since this process relies on conveying information to humans, visualisation can make the assessment task much easier. UML-B[2, 3] already provides visualisation of Event-B models for the purposes of editing a model. Here we describe a new plug-in feature that provides visual animation of UML-B state-machine diagrams. This feature has been developed as part of the Deploy project[1] in response to requests from our industrial partners.

State-machine Animation is a plug-in feature for the Rodin[4] platform and requires the UML-B graphical front-end and ProB[5] model checker/animator to also be installed. It brings the UML-B state-machine diagrams to life by exploiting the ProB Animator. The state-machine animation plug-in feature is available as a prototype that can be installed into the Rodin platform (v1.1) using the usual update site mechanism.[2]

2 Functionality

To start animating state-machines, a user selects specific state-machines from the UML-B project explorer and right-clicks to select a pop-up menu action, *Animate Statemachines*. The tool reads the selected UML-B diagrams and creates the corresponding animation diagrams. It automatically initiates the ProB Animator on the Machine that has been generated by UML-B from the selected state-machines, switching into the Pro-B perspective. The currently active states are highlighted by colouring and enabled transitions are indicated by emboldening. An event can be fired by clicking on its corresponding transition. If it has any parameters the possible values for these will appear in a pop-up menu for selection in a style similar to the ProB Animator.The state-machines may not contain all the information in the animated Event-B machine and the normal Pro-B interface should be used in parallel with it. For example, some events

[1] DEPLOY: EU Project IP-214158, www.deploy-project.eu
[2] users.ecs.soton.ac.uk/cfs/downloads/ac.soton.umlb.umlbMetamodel.stateDiagram.animation.site/

may not be represented as transitions and these will need to be invoked from the Pro-B events viewer instead. The state of ancillary variables that are not visible on the state-machine can be seen in the Pro-B variables view. If multiple state-machines are animated simultaneously, they are opened in separate editors and the animation can be observed on all of them running in parallel, thus the constraints and dependencies between the states of different state-machines can be verified by observing the behaviour of animation. Both nested and refined state-machines and elaborated transitions are supported by the tool.

UML-B classes introduce to Event-B machines the notion of sets of instances which can be (re-)created and deleted during ProB animation. If animated state-machines belong to a class, each instance of the class may be in a different state of the state-machine. The tool allows instances to be manipulated by selecting them as parameters in the same manner as other parameters are selected. After an instance is created it appears as a token inside its active state and moves from state to state when manipulated by selecting one of the enabled transitions. Multiple instances moving between states of animated state-machines can be observed allowing observation of their interaction. Fig 1 shows a screenshot of a state-machine during animation.

Fig. 1. State-machine Animation Example

3 Implementation

Like UML-B, the tool is based on the Eclipse Modelling Framework (EMF)[6] and Graphical modelling Framework (GMF)[7]. Initially we attempted to extend the UML-B meta-model to include the information necessary to model an animation. Due to difficulties in providing a retrospective and independent extension we decided to model the animation separately. The domain model for animation diagrams replicates a small subset of the UML-B model, sufficient for the animation purposes. GMF was then used to create the diagram definitions and generate the implementation, resulting in a set of working plug-ins for creation and manipulation of the animation diagrams. The interface with UML-B is only used to create the animation diagram models and to record details of the translation options used by the UML-B, Event-B translator. Once this launch stage has completed, there is no further interface with UML-B during animation.

The interface with the ProB Animator is provided by the Eclipse's plug-in extension mechanism through the ProB's animation listener extension point definition. The Pro-B API is used to invoke events that correspond to selected transitions. In response Pro-B informs the registered animation listener that a new state is available. The output format is an API that returns the machine state in the form of pairs of strings: variable name and the variable value, which are represented in a standard B notation. The main difficulty involved is to

translate the state values received from the ProB into corresponding animation diagram objects. The UML-B translation must be taken into account. Depending on the UML-B state-machine's translation type a state-machine is translated in Event-B as either a collection of variables representing each state or a single variable representing the whole state-machine. This significantly changes the variable's type and output format. The following example shows the differences between the functional and set translation of the same state-machine for a state-machine called sm with states s1 and s2 and belonging to a class which currently has two instances ci1 (in state s1) and ci2 (in state s2).

For, state-function translation:

```
sm = {(ci1 |-> s1),(ci2 |-> s2)}
```

For state sets translation:

```
s1 = {ci1}
s2 = {ci2}
```

State-machines that are not lifted by classes have a different translation. Nesting of state-machines inside states also affects the resulting variable's type. All the cases thus are treated separately with different assumptions on the output format of the information received from ProB.

4 Conclusion

There are some difficulties in managing the display of several state-machines simultaneously. It can be confusing knowing which state-machine is which in the nesting hierarchy. (This is a problem inherited from UML-B) and we are looking into ways to manage the system of state-machines better in both tools.

The state-machine animation tool is an effective means of visualising the behaviour of a model especially where a *family* of state-machines is modelled via class-lifting. The ability to animate several refinements simultaneously is particularly effective. While testing the protoype tool we discovered a bug in the test model which was a fully proven refinement and assumed to be correct. The bug concerned the elaboration of a self-loop transition which was not performing its intended transition in the refinement. The bug could not be discovered by proof since the bug resulted in a valid refinement, however, the behaviour was certainly not what was intended which immediately apparent by animation.

References

1. Metayer, C., Abrial, J.R., Voisin, L.: Event-B Language. Rodin deliverable 3.2, EU Project IST-511599 -RODIN (May 2005)
2. Snook, C., Butler, M.: Uml-b: Formal modeling and design aided by uml. ACM Trans. Softw. Eng. Methodol. **15**(1) (2006) 92–122

3. Snook, C., Butler, M.: Uml-b and event-b: an integration of languages and tools. In: The IASTED International Conference on Software Engineering - SE2008. (February 2008)
4. Butler, M., Hallerstede, S.: The rodin formal modelling tool. BCS-FACS Christmas 2007 Meeting - Formal Methods In Industry, London. (December 2007)
5. Leuschel, M., Butler, M.: ProB: A model checker for B. In Araki, K., Gnesi, S., Mandrioli, D., eds.: FME 2003: Formal Methods. LNCS 2805, Springer-Verlag (2003) 855–874
6. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: Eclipse Modeling Framework. 2nd edn. The Eclipse Series. Addison-Wesley Professional (December 2008)
7. Online: Graphical modeling framework (GMF). Project Website: http://www.eclipse.org/modeling/gmf/