Design Pattern Based Transformation of Dynamic UML Models for Quantitative Analysis

HUSZERL Gábor
Budapest University of Technology and Economics, Dept. of Measurement and Information Systems
(huszerl@mit.bme.hu)

Towards the dynamic analysis for performance, performability and timing constraints of Unified Modeling Language (UML) models a previous paper [1] presented the transformation of Guarded Statecharts (GSC) – a type of statecharts (SC) – to Stochastic Reward Nets (SRN) – a class of Petri nets. For performance evaluation and analysis Petri nets offer a mathematically well-defined methodology with precise semantics and a theoretical background. Now the input model of the above mentioned transformation has been extended in special consideration of event processing, state hierarchy and transitions. The accordingly extended transformation is defined by a series of design patterns.

In order to support timing analysis the original UML models are extended by timing information, and Petri net (SRN) patterns implementing possible semantics of "timed and guarded" state transitions are defined. These patterns complete the above mentioned ones.

When working with statecharts including events, one important question is the semantics of event dispatching. Two patterns for event dispatchers are defined in this paper. One is selecting events from the queue non-deterministically. It is easy to implement with SRNs, and covers all potential behavior. An other dispatcher is elaborated also, selecting events in the order of arrival (FIFO). It is the way how one would imagine an event "queue", but it is not trivial to implement it by SRNs.

UML statecharts are hierarchical, where states can contain substates or concurrent sub-machines. When several transitions are enabled, some of them may fire simultaneously. SRNs have no hierarchy of places thus the priority of transitions needs extra constructions in the patterns: The UML transitions (triggered by the same event) are ordered by priority, and this state hierarchy is represented by a tree-like structure. When the token representing the dispatched event leaves the tree, all selected UML transitions of the step have already completed their firing.

When introducing timing in UML the delays are assigned to the transitions (and not to the states). Three different patterns for transformation of guarded and timed transitions are defined here: the selection of transitions is irrespective of timing, the fastest wins, or the guard has to be true during the delay else the transition will be deselected (preempted), respectively.

The resulted SRN implements the phases of the step semantics of a UML Statechart: selecting an event; firing the appropriate transitions (one after the other) – i.e. exiting the actual state and superstates (on demand), sending events and entering all target states –; exiting states, of which superstates are no more active and updating the last stable state. The last stable state of the SC have to be stored to the correct evaluation of the guards of later firing transitions, and it have to be updated after completion of all firings. During the steps, there are states of the SRN, which have no valid counterpart in the SC, but these are transient states vanishing before the completion of the step.

A UML Statechart model can be transformed by the help of the above mentioned patterns, and the arising SRN can be analyzed by commercial Petri net tools. Performance measures (throughput, utilization) can be directly derived by using these tools, while dependability analysis requires explicit modeling of erroneous states and faulty behavior (unintended state transitions, loss of messages). The analysis of the probability of erroneous states leads to reliability or availability figures (without or with modeling of repair respectively).

Reference

[1] M. Dal Cin, Huszerl G., K. Kosmidis, "Quantitative Evaluation of Dependability Critical Systems Based on Guarded Statechart", HASE99, Fourth IEEE International Symposium on High Assurance Systems Engineering, November 17-19., Washington DC, 1999