Assurance Analysis by Scenario-based UML-Modeling

Konstantinos Kosmidis and Matthias Sand
University of Erlangen-Nuremberg, Department of Computer Science 3 (Computer Structures)
[kosmidis|sand]@informatik.uni-erlangen.de

1. Introduction

Adequate models of complex interacting systems need to be of a high degree of detail. It is however a common observation that a sufficiently high level of detail decreases the intelligibility of the models whereas it increases the complexity of numerical analysis beyond reasonable and realistic borders. This leads us to the idea of scenario-driven modeling, which focuses on the most interesting scenarios of the system to be modeled (e.g. high assurance scenarios, typical interactions, fault situations, etc.).

2. Scenario-based Modeling and Analysis

A scenario is a single interaction of the components of a communicating system reflecting the calls and/or message passing between them over a restricted period of time or leading the system from one well-specified overall situation to another. Usually a scenario represents a typical interaction the system has to realize e.g. to fulfill a required use case or to meet some non-functional requirements.

We consider two different approaches to scenario-based modeling. The first one is starting at the behavioral model of an interacting system and is leading to scenarios, driven by overall system situations. This is achieved by first laying out the system using a well-defined formalism with clear semantics. In this context, modeling languages containing state machine related formalisms should offer the necessary expressive power. The behavior of all the different entities of the system is to be specified as well as their interaction. Afterwards stochastic annotations like fire rates for transitions can be added to the model. Based on this model, the initial and the target situation of the scenario must be provided. Such a situation usually corresponds to a state configuration of at least some of the specified machines; its form mainly depends on the modeling formalism.

Now it is possible to automatically find all (interesting) scenarios (i.e. paths) leading from the given source to the target configuration e.g. by using a search algorithm that implements the dynamic semantics of the modeling language. The results have to be represented in some form adequate for further processing - be it manual or automatic.

They can either be provided as some kind of sequence chart or be directly transformed to a formalism that allows numerical analysis (like GSPN), respectively. In the latter case all scenarios resulting from one search should be cumulated in one single graph, on which then different kinds of stochastic analyses can be performed (cf. Fig. 1).

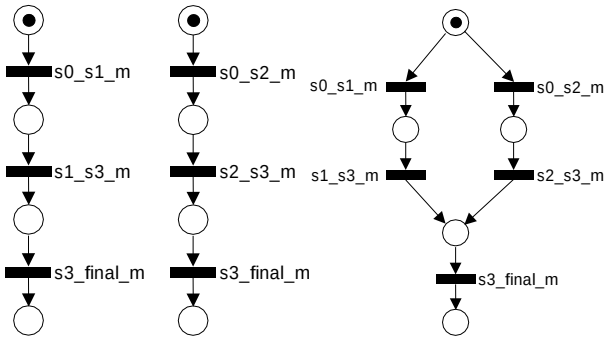

Fig.1 Separate and merged path petri nets

The second approach takes the opposite direction: some given, manually pre-modeled scenarios are checked against the system model. Like above the system has to be set up using a strict formal modeling language. But now, also the scenarios - expressed in a sufficiently formal notation - must be specified by the modeler. To this end, a formalism from the sequence chart family could again provide both formal and comprehensional adequacy. After these preliminaries the system can be checked whether it is able to perform the scenarios.

3. Ongoing Work

Due to its increasing popularity and the large variety of existing tools we decided to use the Unified Modeling Language (UML) [1] for system modeling purposes.

The entities of the system are modeled as classes, the behavior of which is specified using state charts (see Fig. 5). It is however important to mention that the syntax as well as the (dynamic) semantics of the UML are not strictly enough specified to generally allow the design of models for automatic analyses. We thus had to restrict the syntax and cleanly define the execution semantics of UML state charts [2] e.g. by proposing a formal language for evaluable guard expressions and for executable action

sequences.

In future work we intend to define a UML profile [3] for quantitative analysis based on executable models. The runtime set up of the system specified so far then has to be defined by instanciating the classes. In the case of checking the system against scenarios the instances are given by the object nodes of the interaction diagrams (i.e. sequence or collaboration diagrams, see Fig. 3 and 4) which are used to lay out scenarios [4]. For the scenario-generating approach, the instances of the running system as well as the source and target configuration can be specified using UML object diagrams [2].

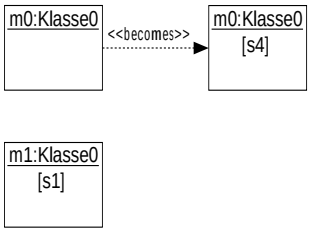

Fig. 2: Source and target configuration for a specific scenario

Results of the analysis should be represented either as interaction diagrams for visualization or - leaving the possibilities of the UML - as GSPNs [5], [6].

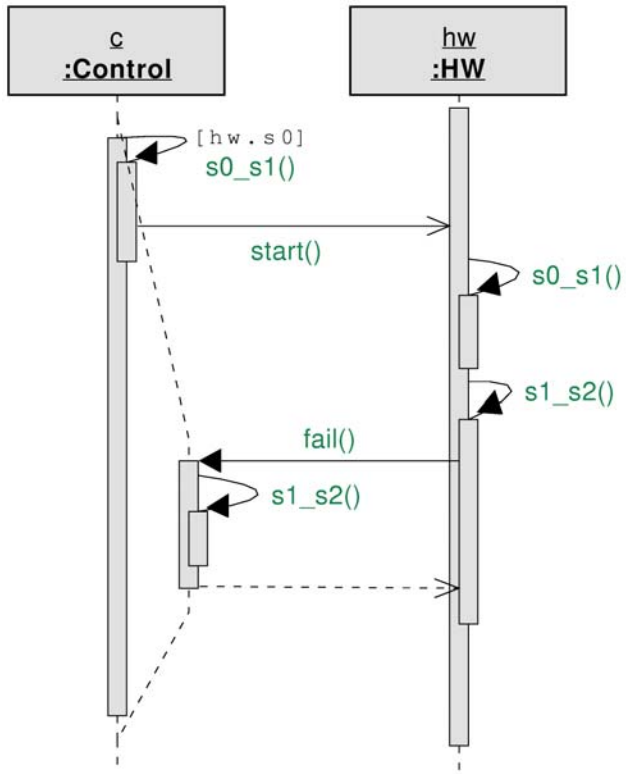

Fig. 3: Sequence chart of a scenario

4. Tool Support

The two approaches are currently elaborated on the basis of the CASE-tool Innovator [7]. The restrictions of this tool forced us to modify some of the modeling principles (e.g. system configurations cannot be specified with object diagrams, activity diagrams have to be used instead) but not the general idea.

The multi-platform tool Innovator provides a scripting API based on Tcl/Tk, which is used to perform the checks in the cross-checking approach. As the main analysis component of the scenario-generating module is implemented in Java, both a platform and tool independent deployment are possible. This module can easily be extended to accept several in- and output formats, e.g. an XMI interface is prepared.

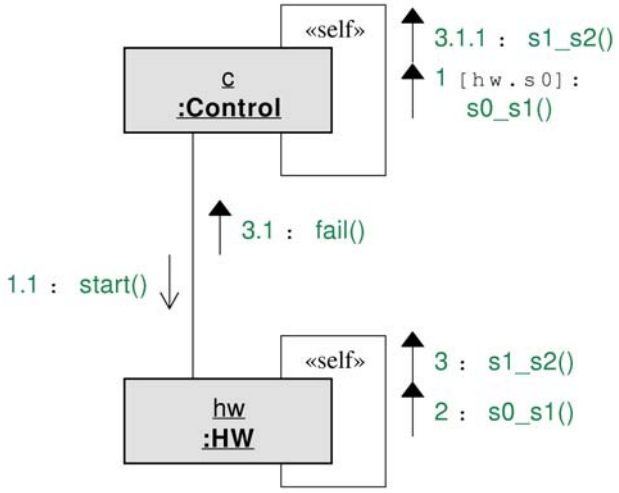

Fig. 4: Collaboration diagram of a scenario

5. References

[1] OMG UML Specification, Version 1.3: ftp://ftp.omg.org/pub/docs/formal/00-03-01.pdf

[2] Sand, M.: IXNOΣ - a pathfinder tool for interacting UML-State Machines, Master Thesis FAU Erlangen-Nuremberg Department of Computer Science 3, 07/2001

[3] S.Cook; The UML Family: Profiles, Prefaces and Packages, 3rd International Conference on the Unified Modeling Language: UML 2000, LNCS 1939

[4] Ghieltsch D.: Conception and implementation of a evaluation component for semantic consistency of dynamic UML-Models. Master Thesis, Department of Computer Science 3, 08/2001

[5] Dal Cin, M.; Huszerl G.; Kosmidis, K.: Evaluation of Dependability Critical Systems based on Guarded Statechart

Models. Proc. High Assurance System Engineering Symposium HASE 1999, p. 37-45

[6] Kosmidis, K; Huszerl, G.: UML Extensions for Quantitative Analysis. Workshop "Dynamic Behaviour in UML Models: Semantic Questions" UML 2000, York UK. In: Workshop Proceedings LMU-München Insitut für Informatik Bericht 0006, Oktober 2000

[7] http://www.mid.de

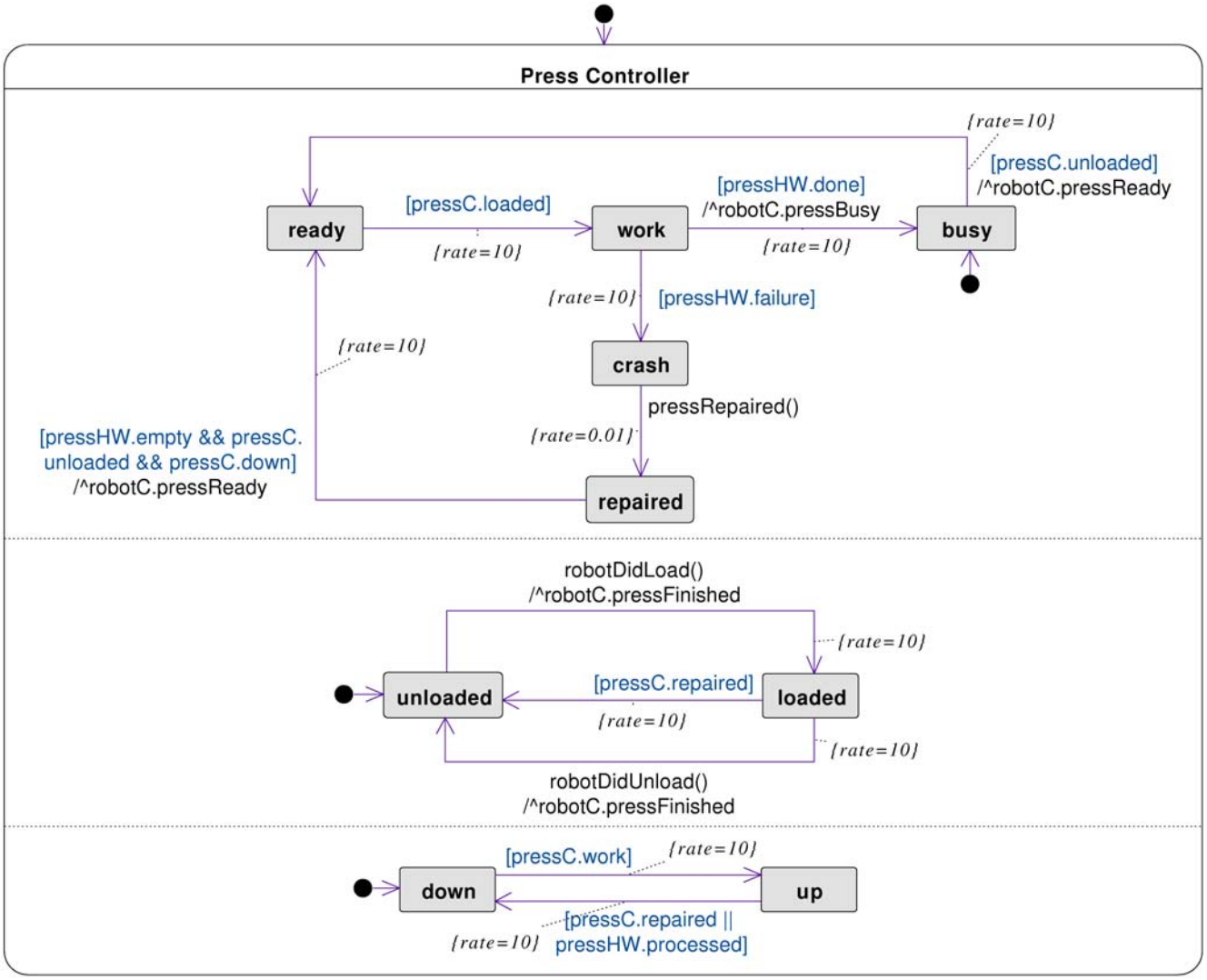

Fig. 5: State chart of a hardware component