Report on a Practical Application of ASMs in Software Design

Egon Börger[1], Peter Päppinghaus[2], and Joachim Schmid[2]

[1] Università di Pisa, Dipartimento di Informatica, I-56125 Pisa, Italy
boerger@di.unipi.it
[2] Siemens AG, Corporate Technology, D-81730 Munich, Germany
{peter.paeppinghaus, joachim.schmid}@mchp.siemens.de

Abstract. ASMs have been used at Siemens Corporate Technology to design a component in a software package called FALKO. Main purpose of FALKO is the construction and validation of timetables for railway systems. For simulation the whole closed-loop traffic control system is modelled within FALKO. The railway process model part of FALKO was formally specified using the ASM approach. C++ code is generated from the formal specification and compiled together with the handwritten C++ code of the other components to obtain the FALKO executable. The project started in May 1998 and was finished in March 1999. Since then FALKO is used by the Vienna Subway Operator for the validation of the whole subway operational service.

1 FALKO

Abstract State Machines (ASMs) [Gur95,BH98] have been used at Siemens Corporate Technology to design a component in a software package called FALKO. This name is the German acronym for "Timetable Validation and Timetable Construction" describing concisely the main functionality of this tool. Detailed operational timetables including e.g. vehicle roster plan can be automatically calculated from raw data like train frequency and infrastructure of a railway system. For these calculations estimations of trip times are used. Timetables – whether constructed with FALKO or other tools – have to be validated for operability and robustness. Conventionally this is done by (physically) driving trial runs. With FALKO this costly form of validation can be replaced by dynamic simulation of operational timetables modelling quantitative aspects like velocities and trip times as accurately as needed.

To perform dynamic simulation of timetables, the whole closed-loop traffic control system is modelled within FALKO. The model is designed in a modular way with three main components: train supervision/train tracking, interlocking system, and railway process model. Software components for train supervision/train tracking and for the interlocking system are nowadays part of real train operation. The railway process model on the other hand serves to replace the real physical system (trains, signals, switches etc.) in the simulation.

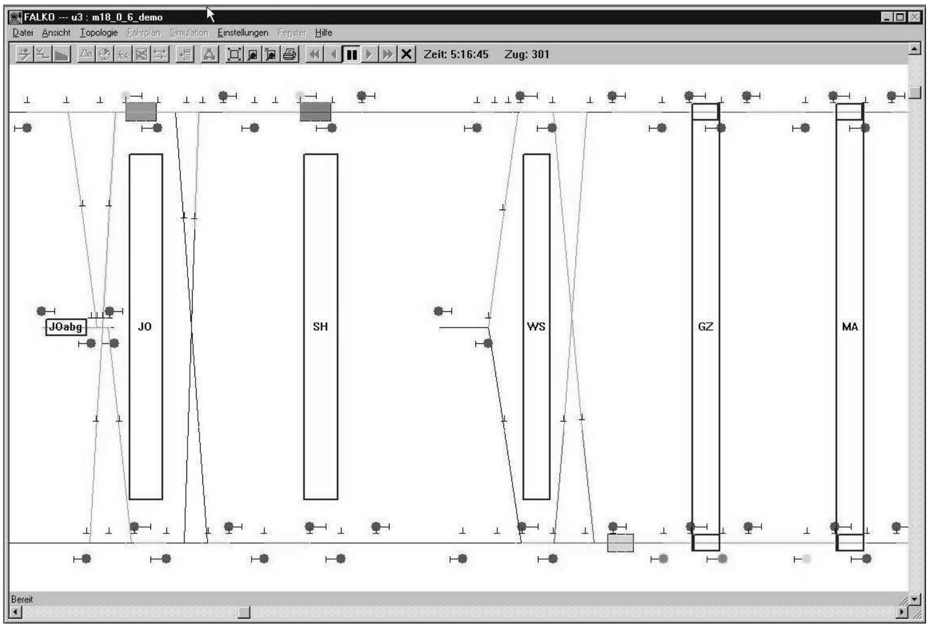

Fig. 1. Screenshot of FALKO GUI

The modelling is based on discrete event simulation. The three components communicate via events, which are tagged with certain data and stamped with the (fictitious) time of their occurrence. Let us give an example. Train supervision, e.g., sends an event at time t_0 to the interlocking system requesting route r_3. The interlocking system, possibly after having received an event which frees route r_3, sends an event at time t_1 to the railway process model requesting switch sw_5 to change its position. The railway process model then calculates the duration of rotating this switch and sends an event to the interlocking system to the effect that switch sw_3 has arrived at the requested position at time t_2, etc.

Besides the three main components modelling the traffic control system there is as a fourth hidden component the event handler, which controls the simulation by receiving events from the visible components and sending them to the respective addressee in the appropriate order.

The railway process model is based on a physical model of driving according to which the crucial numerical calculations for determining speeds, trip times etc. are done. This physical model is another hidden component of FALKO realized as a C++ library.

Further parts of FALKO are a comfortable GUI and extensive facilities to analyze and graphically display data of a simulation run. The GUI can also be used for visualization of simulation runs (see figure 1).

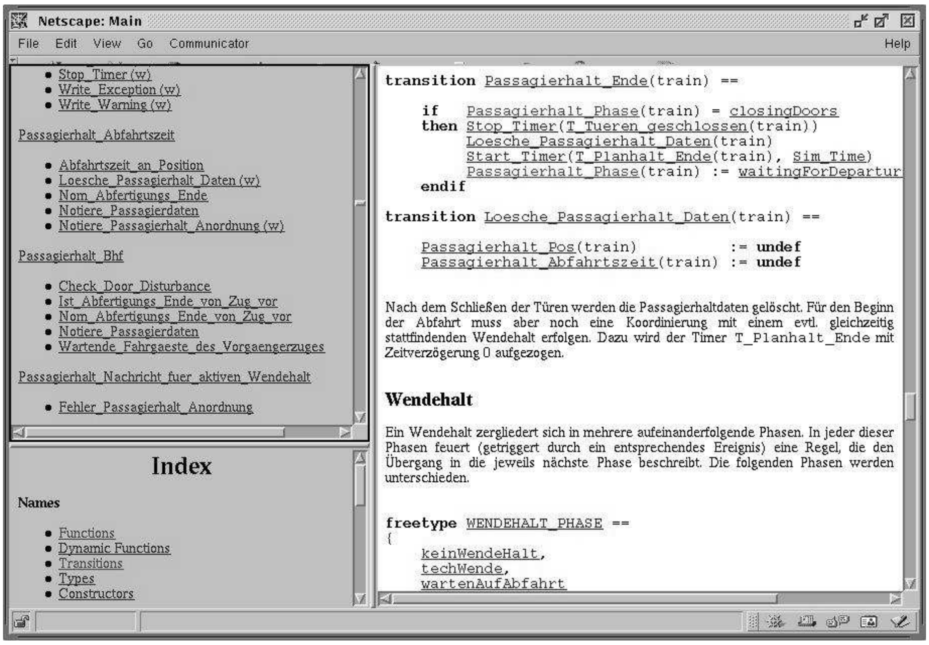

Fig. 2. Screenshot of HTML documentation

2 Design of Railway Process Model with ASMs

2.1 Design Process and Development Tools

The railway process model has been designed with ASMs [Gur95,BH98]. All the other components of FALKO have been designed and implemented conventionally with handwritten C++ code.[1] Along with this design a prototypical development environment has been built up, which can also be used to maintain this component of FALKO.

This development environment supports a seamless flow from the specification down to the executable code. The single source for FALKO's railway process model is a specification consisting of formal parts together with informal explanations. The formal parts are ASM rules and static, dynamic, derived and external functions written in ASM-SL, the language of the ASM workbench [Casar]. The specification comes as a collection of HTML documents (see figure 2). Hyperlinks to navigate between uses and definitions of rules and functions are generated automatically. The formal parts can be extracted automatically from the HTML documents to be fed into the front end of the ASM workbench, which is used for syntax and type analysis.

[1] The physical model of driving is not included in the railway process model. It has been designed conventionally as a component of its own.

In the design phase of the project the ASM workbench was also used for early tests of the ASM model(s). At the end of the design phase it was decided to attempt code generation rather than hand coding the already debugged ASM model.

In the implementation phase a code generator has been developed, automatically generating C++ code from type correct ASM-SL code. In addition some wrapper code for interfacing the generated code to the remaining components, and some low-level “library code” was hand coded.

Formal verification of the ASM model has not been attempted, this was not a goal of the project.

2.2 Effort for Design and Implementation

Design Phase

- Requirement specification based on predecessor system, developed in meetings of the design team, documented by minutes of the meetings (4 persons 2 weeks)
- Design of 1^{st} draft of executable ASM model (1 person 8 weeks)
- Several cycles of testing and debugging using the ASM workbench [Casar] (1 person 8 weeks + 1 person 11 weeks)
- Review of 2^{nd} draft of ASM model by design team plus external reviewers (6 persons 1 week)
- Several cycles of improving, testing and debugging (2 persons 5 weeks)

Implementation Phase

- Development of ASM-SL to C++ code generator (1 person 4 weeks)
- Specification and implementation of additional handwritten C++ code (1 person 2 weeks)
- Integration of FALKO system including testing and debugging (3 persons 3 weeks)
- Documentation of railway process model component and final polish (1 person 6 weeks)

Summing up that part of the above listed effort, which was spent on behalf of the railway process model component of FALKO, this yields a total effort of 66 person weeks.

It is not possible, of course, to compare this effort reliably to the corresponding effort in a conventional software design process. But a rough estimation done by an experienced programmer intimately familiar with FALKO says that the ASM design (including development of the C++ code generator) has exceeded a conventional effort by about 10%.

2.3 Size of ASM Model and C++ Code

ASM Model (source of C++ code generation)

- ca. 3 000 lines of ASM workbench code
- 120 rules
- 315 functions and relations (240 functions, 75 relations)
  - 71 dynamic
  - 69 external
  - 59 static
  - 116 derived

C++ Code

- ca. 9 000 lines of generated C++ code
- ca. 2 900 additional lines of handwritten C++ code, consisting of
  - ca. 400 lines wrapper code for interfacing to other components of FALKO
  - ca. 2 500 lines low-level library code

In the prototypical predecessor system of FALKO the railway process model consisted of ca. 20 000 lines of (handwritten) C++ code. To be fair one has to take into account, however, that this component – having been experimented with – had grown over time (and become difficult to maintain, which was the reason for redesigning it completely).

3 Experiences

It turned out that one of the main advantages of ASMs for this design was the parallel update view. This made it possible to model the railway process in such a way that each state of the ASM model corresponds to a "snapshot" of the virtual physical process taken at the occurrence of a "relevant discrete event". Due to this, one can always have a clear picture of the physical state snapshot, on which auxiliary computations (specified by static and derived functions) are based.

FALKO developers and reviewers not concerned with formal methods had no problems to understand the ASM model. The possibility of early tests by executing the ASM model with the ASM workbench was very helpful and uncovered bugs also in other components of FALKO at an early stage.

No serious attempt has been made to measure the potential performance loss due to the use of generated C++ code. Comparison to the predecessor system of FALKO has been possible only on one example, for which the data happened to be available in both systems. In this small example performance of the previous system was about 30% better than that of the current FALKO system. For the time being the performance problem has been left aside, since it turned out that FALKO's performance is good enough for the purpose the product is used for.

FALKO is used in four installations at the Vienna Subway Operator since March 1999, one of these installations being in daily use. Up to now (i.e. March

2000) the customer reported no bugs. After finishing the first version of FALKO, two bugs in the railway process model have, however, been discovered in tests during development of the second version. The FALKO developers have not yet familiarized themselves with the ASM specific tools. The generated C++ code being readable enough they chose to implement temporary fixes of the two bugs by handhacking the generated C++ code, and to postpone correction of the ASM model until the faults are analyzed more thoroughly.

Acknowledgments. We thank our colleagues of the FALKO team for their open attitude towards this pioneering use of ASMs and for a year of enjoyable and fruitful cooperation. Furthermore we thank Giuseppe Del Castillo for the permission to use early versions of his ASM workbench and for the support we obtained from him.

References

BH98. E. Börger and J. Huggins. Abstract State Machines 1988-1998: Commented ASM Bibliography. *Bulletin of EATCS*, 64:105–127, February 1998. Updated bibliography available at http://www.eecs.umich.edu/gasm.

Casar. Giuseppe Del Castillo. *The ASM Workbench – A tool environment for computer aided analysis and validation of ASM models.* PhD thesis, University of Paderborn, to appear.

Gur95. Yuri Gurevich. Evolving Algebras 1993: Lipari Guide. In E. Börger, editor, *Specification and Validation Methods*, pages 9–36. Oxford University Press, 1995.