The Use of Complex Stateflow-Charts with KIEL— An Automotive Case Study

Steffen Prochnow and Reinhard von Hanxleden
Real-Time and Embedded Systems Group, Dept. of Computer Science
Christian-Albrechts-University of Kiel, Olshausenstr. 40, D-24118 Kiel, Germany
{spr, rvh}@informatik.uni-kiel.de

Abstract: Modeling systems with Statecharts has become standard practice in the design of reactive embedded devices. However, the modeling of realistic applications with the paradigms established so far often results in models that are difficult to comprehend and maintain, which severely compromises their practical use. The Kiel Integrated Environment for Layout (KIEL) is a modeling environment for the exploration of alternative editing and representation paradigms. We here report on an adaptation of KIEL to MATLAB Simulink/Stateflow, and on an automotive case study.

1 Introduction

A commonly touted advantage of graphical formalisms such as Statecharts is their intuitive use and the good level of overview they provide. Graphical models may be convenient to browse; however, compared to textual entry, they are rather cumbersome to construct and maintain, as designers spend a significant fraction of their time with tedious drawing and layout chores.

The Kiel Integrated Environment for Layout (KIEL) environment has been designed as a generic framework for the efficient model-based design of complex systems. In this paper we report on the experiences gained from adapting KIEL to MATLAB Simulink/Stateflow[1], and present a case study using an application from the automotive sector.

The rest of the paper is organized as follows. The remainder of the introduction presents related work, including further details on KIEL. Section 2 compares KIEL and the Stateflow environment and describes how the two are interfaced. The automotive application that is the basis of the case study has originally been developed with MATLAB Simulink/Stateflow; Section 3 discusses the experiences with importing and simulating this application in KIEL. The paper concludes in Section 4.

1.1 Related Work

Castelló *et al.* [CMT02] have developed a framework for the automatic generation of layouts of Statecharts based on floor planning. Harel and Yashchin [HY02] have investigated the

[1] `http://www.mathworks.com/products/stateflow/`

optimal layout of *blobs*, which are edge-less hierarchical structures that correspond to Statecharts without transitions. Rational Rose[2] supports the modeler by re-arranging the manually constructed Statechart. The underlying layout algorithm uses horizontal layers to place states of upper hierarchy level (inner states will not be touched) and performs a middle affine placement of polygon or spline curve transitions. KIEL offers several layout mechanisms, some employ the GraphViz [GN00] layout framework, others are developed from scratch.

Köth and Minas [KM01] have applied "semantic focus-and-context" representations to visualize complex class diagrams with DIAGEN. Our approach of dynamic Statecharts [PvH06] is an extension of this concept, which provides dynamically changing views on a System under Development (SUD) according to the simulation state.

In addition to the model import capabilities, as the Stateflow import mechanism used in our case study, KIEL provides a built-in, structure-based editor, and can also synthesize graphical models from textual descriptions [PTvH06, PvH07]. A customizable checking framework checks for compliance to *robustness rules* [PSBvH06]. Apart from the interface to Stateflow described here, KIEL currently supports the Statechart dialects of Esterel Studio and the UML via the XMI format, as, *e. g.*, generated by ArgoUML[3].

2 Interfacing KIEL and Stateflow

Stateflow is a commercial modeling environment that is routinely used by control engineers to design and simulate discrete controllers. It provides the ability to model hierarchical parallel Statecharts. The Stateflow model can be simulated within the MATLAB framework for a desired time period, which in effect steps through the state machine for a specific input trace. The responses from the state machine can be plotted graphically and the trajectory of the state machine can be observed visually, as well as recorded for postanalysis.

We have implemented an interface that (1) translates the KIEL representation of Statecharts into a Stateflow model, (2) translates the Stateflow model into KIEL representation of Statecharts, and (3) controls the non-interactive simulation of Stateflow models using KIEL. Our interface is built on the MATLAB API, using the MATLAB scripting language (M-file) for procedurally creating and manipulating Stateflow models. In doing so KIEL works as a wrapper for the simulation with Stateflow, which works in the background.

3 An Automotive Case Study

The application that served as our case study is a window wiper controller. It was kindly provided by Daimler Chrysler AG, Research REI/SM, and is an example without any reference to a production design. The model describes a complete wiping system and allows sensor-triggered wiping, interval wiping, wiping interruption due to engine starter activity, *etc.* It consists of 36 States and pseudo states (nine of them are hierarchical OR states, and one is an AND state) and 82 transition. The chart is distributed to eleven sub-charts. Figure 1

[2] `http://www-306.ibm.com/software/de/rational/design.html`
[3] `http://argouml.tigris.org/`

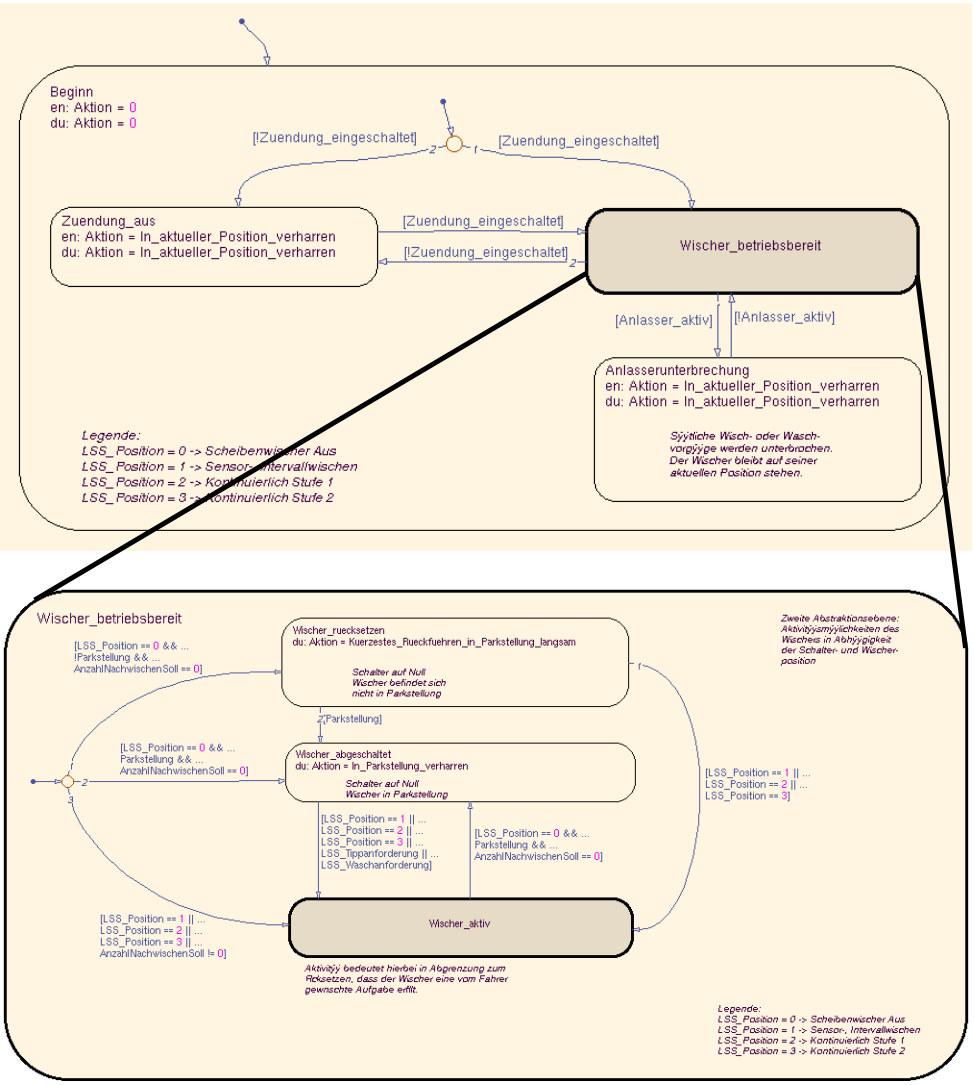

Figure 1: Change of hierarchy level using Stateflow. The upper view provides the top level view of a Statechart. It shows two simple states, and the hierarchical state Wischer_betriebsbereit ("wiper operational"). To show its inner states, the modeler has to double-click with the mouse on the state's border, and the view on the inner states (lower window) replaces the top-level view.

presents exemplary views of the original model, using the Stateflow modeling environment. For comparison, Figure 2 shows a screen shot of the model imported in KIEL.

In general the automatic Statechart layout and the dynamic simulation produced satisfying results. Dynamic Statecharts reduce the size compared to a manual layout. The dynamic Statecharts allow to view comparatively large models in full without losing the detail. Of

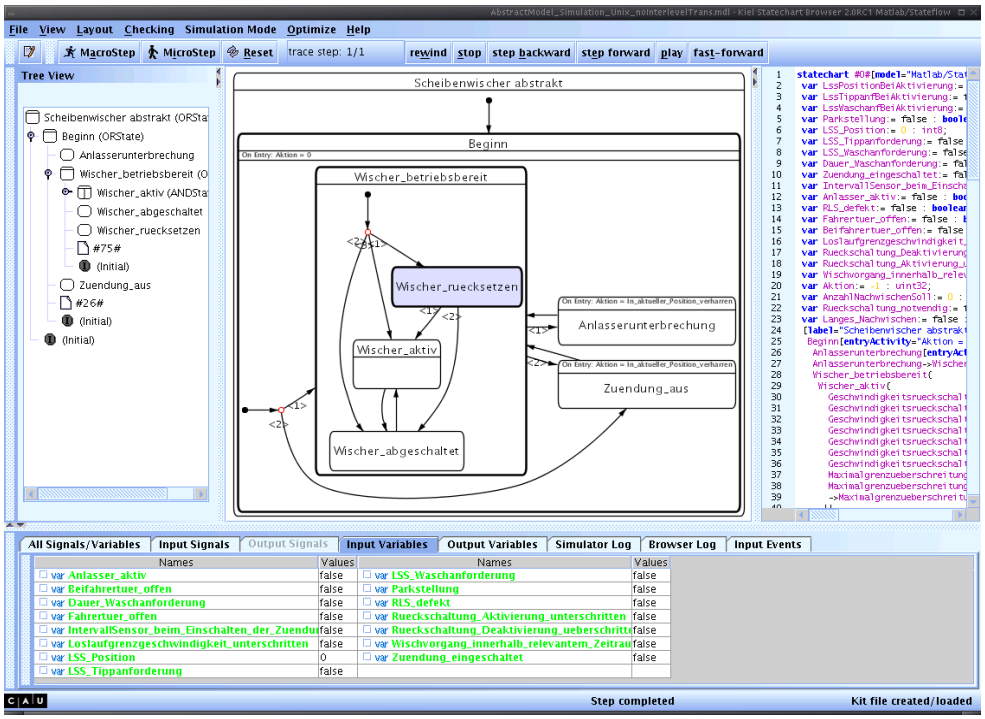

Figure 2: The KIEL view on the same system as in Figure 1 during simulation. The automatic Statechart layouter expands the hierarchy levels. During the simulation the inner states of inactive hierarchical states are hidden. The KIEL tool provides three alternative views on the system: a tree structure, the graphical Statechart browser, and an alternative textual representation.

course, there comes a point when details become unreadable, and the automotive example was complex enough to contain such configurations as well. However, the full-model dynamic Statechart view provided a very valuable overview of where the “hot spots” of a model were during simulation, which is difficult with the traditional Statechart animation approach.

To assess the efficiency of KIEL’s automated layout mechanisms, we have instrumented it to measure computation times. The size of the model did not pose any difficulties, the layout computation was always well in the sub-second range. However, in contrast to academic examples, the window wiper example exposed some weakness of the resulting layouts, in particular when long labels were present. Due to transition labels that consist of more than 20 characters and often embrace more than three lines, the element placement was often unsatisfying and the resulting chart was difficult to read. We therefore consider to make the visibility of labels optional. We also observed that especially states that are arranged around a central Statechart element produce long transitions. We suppose that another layout method, *e. g.*, a force directed layout approach like a spring embedder, would reduce transition lengths.

4 Conclusions and Future Work

Overall, the generic concept and infrastructure of KIEL have proven to be very flexible, the adaptation of the KIEL environment to Stateflow did not pose particular problems. Due to the fairly complex semantics of Stateflow we have opted to not build an internal simulator, as had been the case for Safe State Machines, but instead to build a gateway to the original Stateflow simulator. To that end, the powerful and reasonably well documented Stateflow API has been helpful.

The automotive case study has been very valuable for validating the KIEL/Stateflow adaptation, and for assessing the efficacy of KIEL as a platform for modeling complex reactive systems. As the application has not been developed within KIEL, but rather imported from another tool, we did not get to investigate the effectiveness of KIEL's built-in editing capabilities, but we gained insights on the layout and simulation capabilities. Overall, we did see our original hypotheses on the value of automated layout and adaptive views during simulation confirmed. However, we also identified several areas for further improvements, such as making the view management more flexible.

References

[CMT02] Rodolfo Castelló, Rym Mili, and Ioannis G. Tollis. A Framework for the Static and Interactive Visualization for Statecharts. *Journal of Graph Algorithms and Applications*, 6(3):313–351, 2002.

[GN00] Emden R. Gansner and Stephen C. North. An Open Graph Visualization System and its Applications to Software Engineering. *Software—Practice and Experience*, 30(11):1203–1234, 2000. `http://www.research.att.com/sw/tools/graphviz/GN99.pdf`.

[HY02] D. Harel and G. Yashchin. An Algorithm for Blob Hierarchy Layout. *The Visual Computer*, 18:164–185, 2002.

[KM01] Oliver Köth and Mark Minas. Structure, Abstraction and Direct Manipulation in Diagram Editors. In M. Hegarty et. al., editor, *LNAI 2317*. Springer Verlag, 2001.

[PSBvH06] Steffen Prochnow, Gunnar Schaefer, Ken Bell, and Reinhard von Hanxleden. Analyzing Robustness of UML State Machines. In *Proceedings of the Workshop on Modeling and Analysis of Real-Time and Embedded Systems (MARTES'06), held in conjunction with the 9th International Conference on Model Driven Engineering Languages and Systems, MoDELS/UML 2006*, Genua, October 2006.

[PTvH06] Steffen Prochnow, Claus Traulsen, and Reinhard von Hanxleden. Synthesizing Safe State Machines from Esterel. In *Proceedings of ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'06)*, Ottawa, Canada, June 2006.

[PvH06] Steffen Prochnow and Reinhard von Hanxleden. Comfortable Modeling of Complex Reactive Systems. In *Proceedings of Design, Automation and Test in Europe (DATE'06)*, Munich, March 2006.

[PvH07] Steffen Prochnow and Reinhard von Hanxleden. Statechart Development Beyond WYSIWYG. In *Proceedings of the ACM/IEEE 10th International Conference on Model Driven Engineering Languages and Systems (MODELS'07)*, Nashville, September 2007.