12 Fujaba4Eclipse Real-Time Tool Suite

Claudia Priesterjahn, Matthias Tichy, Stefan Henkler,
Martin Hirsch, and Wilhelm Schäfer

Software Engineering Group, Department of Computer Science,
University of Paderborn, Paderborn, Germany
{cpr,mtt,shenkler,mahirsch,wilhelm}@uni-paderborn.de

Abstract. The Fujaba Real-Time Tool Suite supports modeling and verification of software in mechatronic or embedded systems. It also addresses the specification of advanced systems which reconfigure part of their structure and behavior at runtime. The Fujaba Real-Time Tool Suite requires a rigorous development process concerning the use of the different (partially refined) UML diagrams. All diagrams have a formally and well-defined semantics which allow to check models for given safety properties. Further, the tool suite provides a tight integration with software tools used by control engineers like CaMEL-View and Matlab to enable the simulation of production code of a complete system.

12.1 Introduction

Fujaba is an Open Source UML CASE tool project which was kicked off by the software engineering group at the University of Paderborn in 1997. Current major contributors to Fujaba are research groups at the University of Paderborn, the University of Kassel, the Technical University of Darmstadt, the Hasso-Plattner Institute at the University of Potsdam, the University of Bayreuth, the Technical University of Dresden and the University of Antwerp. Minor contributions come from a number of other places like Tampere and Victoria. In 2002, Fujaba has been redesigned and became the Fujaba Tool Suite with a plug-in architecture allowing developers to add functionality easily while retaining full control over their contributions. There are different Fujaba tool suites available, consisting of the Fujaba Core with different sets of plug-ins, each of which supports modeling and analysis for different domains.

One of the above mentioned tool suites is the Fujaba Real-Time Tool Suite which supports modeling and verification of software in mechatronic or embedded systems. The Fujaba Real-Time Tool Suite requires a rigorous development process concerning the use of the different (partially refined) UML diagrams. All diagrams have a formally and well-defined semantics which allow to check models for given safety properties. Further, the tool suite provides a tight integration with software tools used by control engineers like CaMEL-View and Matlab and a transformation from domain-spanning models of the early development phases to domain-specific models of the Fujaba Real-Time Tool Suite.

In 2008, this tool suite received an IBM Real-Time Innovation Award. In addition, another FUJABA tool suite (supporting the teaching of object-oriented

H. Giese et al. (Eds.): MBEERTS, LNCS 6100, pp. 309–315, 2010.
© Springer-Verlag Berlin Heidelberg 2010

concepts in undergraduate education) was acknowledged by an IBM Eclipse Innovation Grant in 2004.

In this paper, we present an overview of the features and the corresponding development process of the Fujaba4Eclipse Real-Time Tool Suite[1]. The whole approach is called MUML (cf. [1]). We further show how the MUML was employed to develop the software of a prototype of a new type of public transport system, i.e. a non trivial case study.

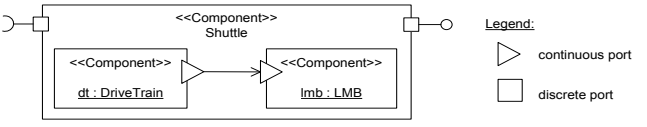
12.2 Features

The software of mechatronic systems is characterized by hard real-time constraints and the integration of controllers to control the dynamics of mechanical components. Hence, our approach supports the modeling and formal verification of so-called hybrid systems and the specification of timed behavior. As formal verification techniques like model checking suffer from the state space explosion problem, we developed a modular and compositional verification approach (cf. [2]). Code synthesis (for C++ and Java real-time) takes the specified real-time requirements into account such that the code exhibits the specified time constraints [3].

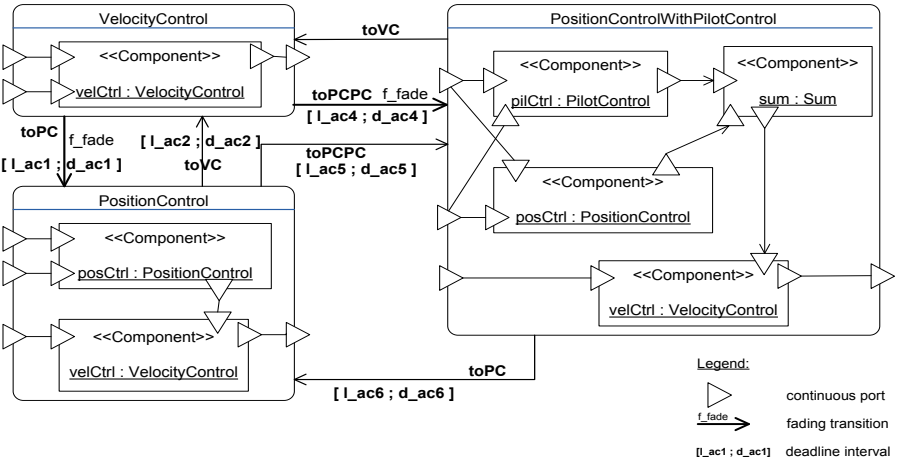
In more detail, the structure (architecture) of the system is specified by software components (as well as the relevant parts of the physics of the system, e.g. for the control engineers) and connectors between them using a slightly refined and formally defined UML 2.0 component model. The behavior of each port to port connection between components is specified by so-called **real-time coordination patterns**. Using an extended version of timed automata called **Real-time Statecharts**, which includes a number of additional syntactical constructs, the pattern specifications define, besides the particular communication protocols, all related required time constraints like invariants, guards, worst case execution times (WCET), and deadlines. The coordination patterns consist of different roles, which correspond to a particular port's behavior. Each pattern is individually verifiable concerning safety properties, specified using ATCTL, using the model checker UPPAAL.

The complete behavior of **components**, consisting of a number of ports as depicted in Figure 12.1(a), is automatically composed of all port roles of corresponding patterns. The user just sets a few parameters like eliminating nondeterministic choices in the pattern definition, i.e. synchronizing all behavior using internal events which do not affect the external behavior. A well defined refinement relation, which is checked by the tool, guarantees that the already verified properties still hold after composition without the need to check them again. The only check remaining is to make sure that the composed automaton does not include any deadlocks [2].

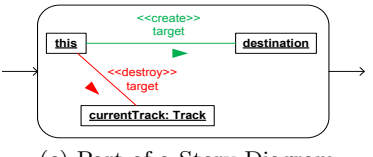
[1] The Fujaba4Eclipse Real-Time Tool Suite is available for download at http://wwwcs.uni-paderborn.de/cs/fujaba/projects/realtime/index.html



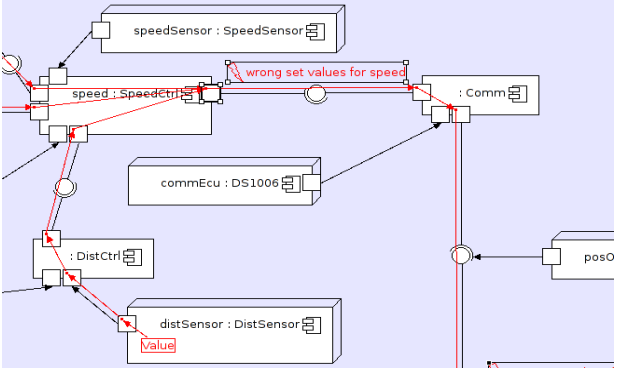
(a) Component Diagram



(b) Hybrid Reconfiguration Chart



(c) Part of a Story Diagram



(d) Qualitative Hazard Analysis

Fig. 12.1. Development with the Fujaba4Eclipse Real-Time Tool Suite

To integrate **controllers** into the component model, they are embedded into hierarchical component structures. **Hybrid reconfiguration charts** (s. Figure 12.1(b)) are used to specify the different controller modes. The provided embedding concept enables the specification and modular verification of reconfiguration, i.e. the activation and deactivation of software as well as hardware components, across multiple components. Simple consistency checks ensure again that the verified real-time constraints of the coordination patterns are still valid in spite of the embedding. Thus, a verification of the whole system is not necessary, because the verification results of the individual patterns and components hold for the complete system [4].

Advanced systems may change their structure during runtime, i.e. a part of or even a complete component may be replaced or removed. These structural changes which may correspond e.g. to a removal of a pattern or the addition of a component, are also specified at design time. As a usual infinite state space has to be specified, we employ a grammar-based formalism in the sense of a generator definition. In more detail, a graph transformation system defines all valid and usually infinitely many system configurations by a finite set of rules. The correctness of such a rules set concerning safety properties and reachability of only valid system configurations is automatically verifiable [5] and [6, 7, 8]. Timing constraints of the execution of rules may also be specified such that the real time constraints of a system reconfiguration are expressable and analyzable on the model level as well.

We employ a special formalism to define the transformation rules called Story Diagrams [9] as depicted in Figure 12.1(c). We have shown in [10] that this formalism supports the computation of worst case execution times such that the time constraints defined on the model level correspond to the real execution time on a particular implementation platform. This approach "only" assumes that the implementation platform guarantees time constraints for basic operations like adding or deleting a graph node or edge resp.

The tool suite further supports a **hazard analysis** [11]. That analysis identifies random faults by propagating the impact of component errors through the whole system architecture. A qualitative analysis, as shown in Figure 12.1(d), determines which hazards result from a given set of basic errors (bottom up) or which basic errors have to occur in order to make a given hazard happen (top down). This qualitative analysis is accompanied by a quantitative analysis which computes the hazard's probability. The hazard analysis furthermore supports the analysis of reconfigurable systems.

If the hazard analysis shows that the required hazard probability is not satisfied, we apply **fault tolerance patterns** [12]. We again use Story Diagrams for their specification (s. Figure 12.1(c)).

Input to code synthesis are the hybrid reconfiguration charts and the graph rules defining reconfiguration. As not all system properties and the whole system behavior can be checked on the model level, the resulting code is executed using an advanced simulation system, which is partly based on the integration of a commercially available control engineering tool. The key point of this

approach is that the simulated code is the same as the production code which is driving the real system, thus avoiding variations of behavior in simulation and implementation.

12.3 Case Study: RailCab

In terms of ecological values, public transport by bus or railway is deemed superior to individual transport by car. Unfortunately, individual transport clearly provides more flexibility and comfort for the passenger. The RailCab project[2] was founded at the University of Paderborn in 1998 in order to develop a new railway system that features the advantages of both techniques in terms of cost and fuel efficiency as well as flexibility and comfort. The novel system is characterized by autonomous vehicles operating on demand instead of trains being determined to a fixed schedule. RailCabs exhibit self-adaptive properties and operate in a safety-critical domain. Consequently, they provide an excellent case study for the Fujaba4Eclipse Real-Time Tool Suite.

We used the Fujaba4Eclipse Real-Time Tool Suite in different scenarios in the RailCab project. These scenarios include the active steering, the suspension/tilt [13] and the air gap adjustment system [14].

One particular problem is to reduce the energy consumption due to air resistance by coordinating the autonomously operating RailCabs in such a way that they build **convoys** whenever possible. Such convoys are built on-demand and require a small distance between the different RailCabs such that a high reduction of energy consumption is achieved. The convoy operation is clearly safety-critical. It requires a rigorous development approach for the real-time coordination between the RailCabs as well as for the integration of feedback controllers.

The first step is to **specify the structure and behavior** of the system [15]. The structure reflects typically the relevant parts of the physics. In our case study, we specify a RailCab component which embeds further components like a drive component. Like the RailCab component, the drive component could be composed of multiple other component instances. This leads to an architectural description of the RailCab, consisting of multiple layers. The coordination behavior of a convoy is specified by a real-time coordination pattern. We further embed the controllers of the RailCab – the distance and velocity controllers – into the internal component behavior and specify the reconfiguration between the different controller modes.

After building the model, we **verify** it applying two different techniques. First, we verify for the convoy scenario the real-time protocol behavior as well as the internal real-time component behavior through model checking, thus ensuring the safety critical property that the RailCabs will not collide [15]. Therefore, we check that all RailCabs are driving in convoy mode simultaneously. Second, we check structural properties as the correct instantiation of patterns and the consistent reconfiguration. For the first property we check, whether RailCabs

[2] http://www.railcab.de/en/index.html

driving on consecutive tracks apply the appropriate real-time coordination pattern to ensure they keep enough distance between each other. For consistency in reconfiguration, we check the correct embedding of controllers [3].

For **hazard analysis**, we consider the hazard of a RailCab driving in convoy mode at a wrong speed, which might result in collision. Therefore, a value failure on the output of the speed control is specified. Then, we employ the top down analysis to determine the errors that result in the hazard, e.g. wrong values from the speed sensor. The next step is to compute the hazard's probability by quantitative analysis. Thereafter, we determine the error's propagation paths leading to the hazard (see Figure 12.1(d)) by bottom up analysis thereby obtaining improvement points in the system's architecture [12].

After obtaining the final model, the tool **generates source code** that integrates the continuous and the discrete behavior. Then, we validate it through **simulation** that uses the same code as the final implementation, thus avoiding variation of behavior in simulation and implementation.

12.4 Conclusions and Future Work

We applied the Fujaba4Eclipse Real-Time Tool Suite successfully in the development of the RailCab's software. Up to now, we focused mainly on the forward engineering of the software. We are currently complementing our approach by a reverse engineering part [16]. This enables the integration of legacy software into our rigorous development approach. Due to space constraints, a discussion of related work is only contained in the cited papers.

References

[1] Burmester, S., Tichy, M., Giese, H.: Modeling Reconfigurable Mechatronic Systems with Mechatronic UML. In: Aßmann, U. (ed.) Proc. of Model Driven Architecture: Foundations and Applications (MDAFA 2004), Linköping, Sweden, pp. 155–169 (June 2004)

[2] Giese, H., Tichy, M., Burmester, S., Schäfer, W., Flake, S.: Towards the Compositional Verification of Real-Time UML Designs. In: Proc. of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT international symposium on Foundations of software engineering (ESEC/FSE-11), pp. 38–47 (September 2003)

[3] Burmester, S., Giese, H., Henkler, S., Hirsch, M., Tichy, M., Gambuzza, A., Müch, E., Vöcking, H.: Tool support for developing advanced mechatronic systems: Integrating the fujaba real-time tool suite with camel-view. In: Proc. of the 29th International Conference on Software Engineering (ICSE), Minneapolis, Minnesota, USA, pp. 801–804. IEEE Computer Society Press, Los Alamitos (May 2007)

[4] Giese, H., Burmester, S., Schäfer, W., Oberschelp, O.: Modular Design and Verification of Component-Based Mechatronic Systems with Online-Reconfiguration. In: Proc. of 12th ACM SIGSOFT Foundations of Software Engineering 2004 (FSE 2004), Newport Beach, USA, pp. 179–188 (November 2004)

[5] Becker, B., Beyer, D., Giese, H., Klein, F., Schilling, D.: Symbolic Invariant Verification for Systems with Dynamic Structural Adaptation. In: Proc. of the 28th International Conference on Software Engineering (ICSE), Shanghai, China, pp. 72–81. ACM Press, New York (2006)
[6] Burmester, S., Giese, H.: Visual Integration of UML 2.0 and Block Diagrams for Flexible Reconfiguration in Mechatronic UML. In: Proc. of the IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2005), Dallas, Texas, USA, pp. 109–116. IEEE Computer Society Press, Los Alamitos (September 2005)
[7] Tichy, M., Henkler, S., Holtmann, J., Oberthür, S.: Component story diagrams: A transformation language for component structures in mechatronic systems. In: Postproc. of the 4th Workshop on Object-oriented Modeling of Embedded Real-Time Systems (OMER 4), Paderborn, Germany (2008)
[8] Hirsch, M., Henkler, S., Giese, H.: Modeling Collaborations with Dynamic Structural Adaptation in Mechatronic UML. In: Proc. of the ICSE 2008 Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2008), Leipzig, Germany, pp. 33–40. ACM Press, New York (May 2008)
[9] Fischer, T., Niere, J., Torunski, L., Zündorf, A.: Story diagrams: A new graph rewrite language based on the unified modeling language. In: Ehrig, H., Engels, G., Kreowski, H.-J., Rozenberg, G. (eds.) TAGT 1998. LNCS, vol. 1764, pp. 296–309. Springer, Heidelberg (2000)
[10] Burmester, S., Giese, H., Seibel, A., Tichy, M.: Worst-case execution time optimization of story patterns for hard real-time systems. In: Proc. of the 3rd International Fujaba Days 2005, Paderborn, Germany, pp. 71–78 (September 2005)
[11] Giese, H., Tichy, M.: Component-Based Hazard Analysis: Optimal Designs, Product Lines, and Online-Reconfiguration. In: Górski, J. (ed.) SAFECOMP 2006. LNCS, vol. 4166, pp. 156–169. Springer, Heidelberg (2006)
[12] Tichy, M., Henkler, S., Meyer, M., von Detten, M.: Safety of component-based systems: Analysis and improvement using fujaba4eclipse. In: Companion Proceedings of the 30th International Conference on Software Engineering (ICSE), Leipzig, Germany, pp. 1–2 (May 2008)
[13] Burmester, S., Giese, H., Oberschelp, O.: Hybrid UML Components for the Design of Complex Self-optimizing Mechatronic Systems. In: Braz, J., Araújo, H., Vieira, A., Encarnacao, B. (eds.) Informatics in Control, Automation and Robotics I, Springer, Heidelberg (March 2006)
[14] Henkler, S., Hirsch, M., Kahl, S., Schmidt, A.: Development of self-optimizing systems: Domain-spanning and domain-specific models exemplified by an air gap adjustment system for autonomous vehicles. In: ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, New York, USA, ASME, August 3-6, pp. 1–11 (September 2008)
[15] Burmester, S., Giese, H., Hirsch, M., Schilling, D., Tichy, M.: The fujaba real-time tool suite: Model-driven development of safety-critical, real-time systems. In: Proc. of the 27th International Conference on Software Engineering (ICSE), St. Louis, Missouri, USA, pp. 670–671. ACM Press, New York (May 2005)
[16] Giese, H., Henkler, S., Hirsch, M.: Combining Compositional Formal Verification and Testing for Correct Legacy Component Integration in Mechatronic UML. In: de Lemos, R., Di Giandomenico, F., Gacek, C., Muccini, H., Vieira, M. (eds.) Architecting Dependable Systems V. LNCS, vol. 5135, pp. 248–272. Springer, Heidelberg (2008)