Simulation and Verification of UML-based Railway Interlocking Designs

Yuen Man Hon[1] Maik Kollmann[2]

Institute of Information Systems
Technical University Braunschweig
38106 Braunschweig, Germany

Abstract

The development of safety critical systems such as railway interlocking systems demands the application of formal methods in order to verify the operativeness and the safety of the system. In this contribution we outline our approach of developing a UML-based Railway Interlocking System. The feasibility of the basic functionality is demonstrated by simulation. In order to verify the system under development, we apply model checking for the verification of specifications that belong to a single object. We tackle the state space explosion problem in those cases in which a number of objects is related to a checking condition by the application of multi-object checking. As multi-object checking is only limited by the state space of the largest object (a point in our case), multi-object checking will be applied successfully to verify the system (e.g ensure the absence of conflicting routes). First results of a case study promise the near future of fully automatic verification of complete station layouts and all desired routes.

Keywords: Railway Interlocking System, UML, Simulation, Verification, Model Checking, Multi-Object Checking.

1 Introduction

Railway **I**nterlocking **S**ystems (RIS) [Pac04,Bon01] are responsible for establishing safe routes for trains that are scheduled to pass through or stop at a railway station. Safe routes ensure that trains cannot be driven into tracks that are occupied or may become occupied by other trains. This can be ensured by proper settings of infrastructure elements like points and signals in the station.

RIS are often designed specificially according to the layouts of stations [vDFK+98]. When the layout of a station is changed, the corresponding RIS has to be modified. This causes high costs for resources and time. In order to reduce the effort in modifying RIS, one can develop RIS under the object oriented approach. Under this approach, geographical interlocking is used as the interlocking logic of a RIS. Objects, events and actions can be captured by using state modeling. The suitable tool for state modeling are the **U**nified **M**odeling **L**anguage (UML) state

[1] Email: Y.Hon@tu-bs.de
[2] Email: M.Kollmann@tu-bs.de

machines [Gro05]. RIS that are developed by applying the geographical interlocking and using UML as specification tool are called **UML-based R**ailway **I**nterlockings (UML-based RI).

In order to verify and validate the correctness of a UML-based RI in establishing safe routes, one can apply several methods for this purpose. One is verification and validation by simulation and animation, the other is verification by model checking. The first method is supported by Rhapsody [IL,HG96]. It is a software development environment that supports users in simulating and in animating the behavior of a UML-based RI in response to the environment events. However, it is difficult to cover all the possible situations by test cases for verification and validation. Verification by model checking can be used to handle this problem. In this work, the state machines of infrastructure objects are first transformed into **F**inite **S**tate **M**achines (FSM) and the specifications of interest, e.g. two conflicting routes must not exist in any state of the system, are put into NuSMV. This approach in verifying the model leads to the state explosion problem. Multi-object checking [EKP03] is now applied to investigate the feasibility in handling this problem.

2 Railway Interlocking Systems

Before a train is allowed to pass through or stop at a terminal in a railway station, a route request is sent to the RIS to indicate where the train should stop or pass through. The RIS establishes a safe route for this route request by coordinating the setting of infrastructure elements of railway stations, such that safe movements of every train through a station are guaranteed. Figure 1 shows the railway station that is used to check the feasibility of developing a UML-based RI. There are different kinds of infrastructure elements that are located within this station: tracks, points and signals. The basic infrastructure elements are tracks that trains run on, e.g. *GA1*. Instead of moving only straight ahead, trains can also turn to another track via a turnout that is composed of movable points, e.g. *W1*. A point can be set either to the left or to the right position. Signals, e.g. *A*, are used to control the train movements and control the speed of trains.

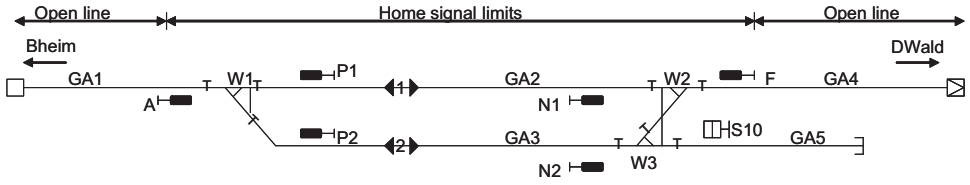

Fig. 1. Layout of a railway station

The main purpose of establishing safe routes for train movements is to ensure no train collisions would take place while trains drive along the tracks based on these routes. A route is described as a safe route if it satisfies the following requirements: The corresponding infrastructure elements along the route are used exclusively by a single train. In addition, there is a proper setting of points along the routes such that no other trains can be driven into the safe route from the divergent direction. Such a flank protection that can be provided by points or signals ensures that no trains can be driven into a route through turnouts or crossings (cf. [Pac04]).

When an interlocking system receives a route request from the approaching train, it will develop the route and check this route against the above two requirements. If this route fulfills those requirements, then this route and the infrastructure elements along this route will be locked exclusively for this train. No other train can use the same route, so that collisions can be avoided. The mentioned safety requirements can be ensured by executing two procedures: checking the availability of infrastructure elements to ensure the exlcusive usage of infrastructure elements and searching flank protection for each point along the route.

3 Modeling of UML-based RI

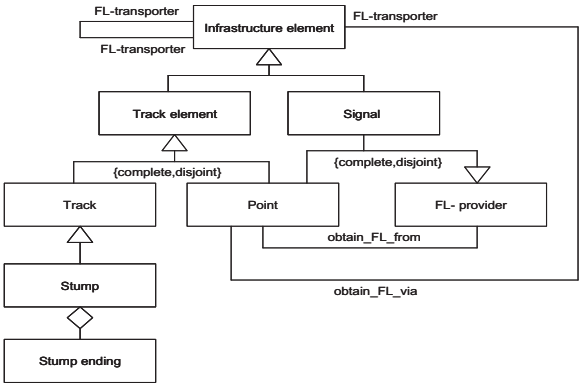

Fig. 2. UML class diagram of infrastructure objects

As mentioned in section 1, infrastructure elements can be viewed as objects, called infrastructure objects. Figure 2 summarizes objects that participate in the system world and the structural relationships among the objects. The objects communicate among each other with messages according to their neighbor tables to function as an interlocking system.

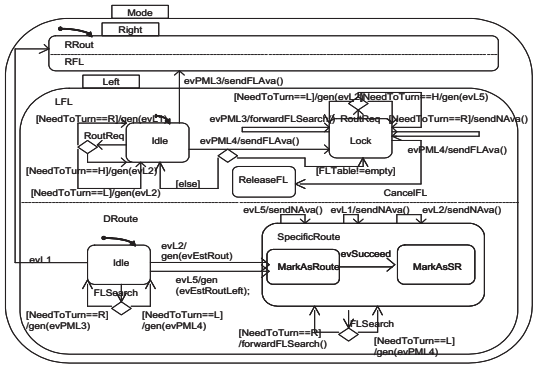

Fig. 3. Substate *mode* of state machine *Point*

As a result, an infrastructure object has both static and dynamic behavior. The static behavior consists of the properties of the object, e.g. points can either be set to the left or to the right position. The dynamic behavior of the object is determined by the actions that are described in the collaboration algorithm [Hon06]. This behavior of objects is modeled in UML state machines (see figure 3).

4 Simulation and Verification of UML-based RI

As RIS are safty critical systems, it is very important to check their correctness in developing safe routes, such that no unsafe route will be issued by the system. Simulation and animation by Rhapsody is used as the first method for this purpose. State machines of objects are specified in Rhapsody. Events form the test cases as inputs and Rhapsody simulates and animates the behavior of object according to the specified models (see figure 4).

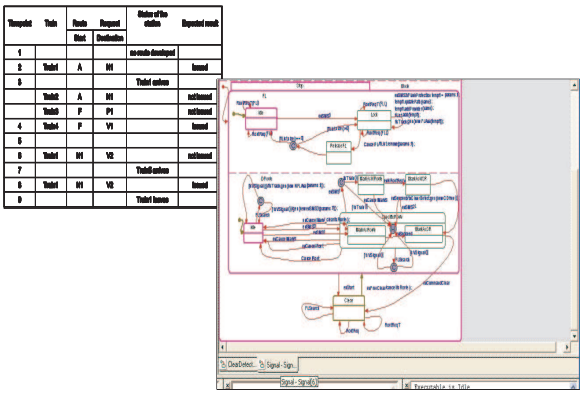

Fig. 4. Simulation and animation by Rhapsody

Simulation enables a system developer to become familiar with the system under development. In addition, simulation offers a way to validate a model on the test case level. However, it is difficult to cover all possible situations in the test cases for simulation.

Model checking is the second method used in this work to verify the UML-based RI. Model checking searches the state space of the UML-based RI and checks whether conflicting routes can exist at the same time in a state. NuSMV is used as the model checker in this work. The state machines of objects are transformed into FSM. The concurrent states and substates of an object are specified as `MODULE`s. Events, actions and transitions are defined as state variables. Further translations are based on the concepts of [Ara03,EW00]. The specifications of interest, e.g. the absence of conflicting routes during the life time of the system, are defined in Temporal Logic. FSM and specifications are put into NuSMV. Unfortunately the verification of the complete interlocking system by applying NuSMV directly to our translation scheme is impossible. After seven days of building up BDDs/ADDs we stopped this approach. The state space is attempted to be reduced by defining invariants in the FSM, e.g. such that only events of interest will be taken into account when the state space is built by NuSMV. This results in the state space explosion within a short time (less than 30 minutes) as the system ran out of memory. This can prove that the first conventional method in verifying the state based model led to the state space explosion problem because of the size of the Cartesian product of state variables.

5 Conclusions and Future Work

Our first results as to the modeling and the verification of a UML-based Railway Interlocking applying simulation, model checking and finally multi-object checking are promising. In order to verify the complete system, we will use multi-object checking. Multi-object checking is known to scale well using model checking as the underlying verification technique. Only slight adaptations to the elements have to be applied and a mapping of messages among elements has to be prepared. The translation of checking conditions into multi-object logic formulas [EC00] is nearly straightforward. Specifications involving a number of objects may require defining further objects in the checking condition. As the state space of the largest object (a point) can be traversed in minutes, multi-object checking is applicable to verify multi-object specifications. We look forward to use multi-object checking successfully in order to verify the absence of conflicting routes and to prove the existence of flank protection for each established route.

Future work will incorporate a larger set of infrastructure elements, the creation of overlap etc. and finally a verified creation of complete and safe routes. In addition, we will concentrate on a toolkit to generate counterexample traces of the involved objects, if multi-object checking detects an error.

Acknowledgements

Many thanks are due to Hans-Dieter Ehrich and Jan-Tecker Gayen for their valuable hints during various discussions.

References

[Ara03] Saeid Arabestani. Umschreibung von Zustandsdiagrammen der UML für das Model-Checking. In E. Schnieder, editor, *Entwurfsmethodik, Modellbildung, Werkzeuge und Anwendungen, 8. Fachtagung Entwurf komplexer Automatisierungssysteme EKA 2003*, pages 63–79, June 2003.

[Bon01] C. F. Bonnett. *Practical Railway Engineering.* Imperial College Press, 2001.

[EC00] H.-D. Ehrich and C. Caleiro. Specifying communication in distributed information systems. *Acta Informatica*, 36(Fasc. 8):591–616, 2000.

[EKP03] H.-D. Ehrich, M. Kollmann, and R. Pinger. Checking Object System Designs Incrementally. *Journal of Universal Computer Science*, 9(2):106–119, 2003.

[EW00] Rik Eshuis and Roel Wieringa. Requirements-level semantics for UML statecharts. In *Fourth International Conference on Formal methods for open object-based distributed systems IV*, pages 121–140, Norwell, MA, USA, 2000. Kluwer Academic Publishers.

[Gro05] Object Management Group. Unified modeling language superstructure specification - v2.0. http://www.omg.org, 2005.

[HG96] David Harel and Eran Gery. Executable object modeling with statecharts. In *ICSE '96: Proceedings of the 18th international conference on Software engineering*, pages 246–257, Washington, DC, USA, 1996. IEEE Computer Society.

[Hon06] Y. M. Hon. Modeling and Verification of Railway Interlockings. Master's thesis, Technische Universität Braunschweig, 2006.

[IL] Telelogic I-Logix. I-Logix Rhapsody. http://www.ilogix.com.

[Pac04] J. Pachl. *Railway Operation and Control.* VTD Rail Publishing, New Jersey, 2004.

[vDFK+98] Fokko van Dijk, Wan Fokkink, Gea Kolk, Paul van de Ven, and Bas van Vlijmen. EURIS, a specification method for distributed interlockings. *LNCS*, 1516:296–??, 1998.