Model-based Software Development for Automatic Train Protection System

Haifeng WANG[1]
[1]School of Electronic and Information Engineering
Beijing Jiao Tong University
Beijing 100044, China
e-mail: hfwang@bjtu.edu.cn

Chunhai GAO[2], Shuo LIU[2]
[2]Engineering Research Center of Rail Traffic Control
Beijing Jiao Tong University
Beijing 100044, China
e-mail: chhgao@bjtu.edu.cn, shliu2@bjtu.edu.cn

***Abstract*—Automatic train protection (ATP) system is a safety critical application for railway signaling with high logic complexity, and it is typically embedded and real-time demanded. As the computational complexity of ATP system, traditional software development method cannot accommodate effectively. In this paper, we propose a novel Model-Based Development (MBD) scheme for ATP system. We argue that high quality software development in ATP system should not only rely on function design, but also on safety verification. A model-based development life cycle and verification approach are described. Furthermore, we present a case study, which is using the tool of SCADE suite. The approach can shorten the development period, enhance system safety, and meet challenges in safety critical software development.**

***Index Terms*—model-based development, safety critical, ATP system, SCADE**

I. INTRODUCTION

Today, for a rapidly growing of computer-based systems in railway transportation, software has become a key factor. APT system is an important system for train control, and safety is its most rigorous property. Standard EN50128 [1] indicates that software safety integrity level (SIL) of ATP function is SIL4. ATP system belongs to the category of software-intensive systems, where software contributes essential influences to the development. Such safety critical software-intensive systems today cover so widely areas such as railway signal, nuclear power plant, aviation and spaceflight, transportation, and so on.

The observation that creating software-intensive systems is extremely difficult, classical software engineering methods are weakly to high safety requirements, the development challenge becomes even more demanding. To master the often large and complex systems, a systematic engineering approach which includes modeling as an essential design activity is required. Today, model-based development of safety critical software becomes increasingly used, thanks to a growing number of tools like Matlab/Simulink, UML tools [2], or SCADE [3, 4]. This not only allows for more formally specifying functionality of the application domain than a conventionally used artificial language, but also do verification of the designed model and the specified behavior before any code for the target platform is generated, it helps to detect design flaws in a very early development phase, which again reduces the cost for corrections.

In this paper we review ATP system and shortcomings of traditional software approaches. Thereby, we regard especially the model-based development method for safety critical system. As a case study, a model-based design of ATP system using SCADE is presented, and we also interested in the safety verification of the model design.

The paper is therefore structured as follows. In the next section, some background information of ATP system is given, section III describes the software development life cycle of safety critical system, and section IV outlines the case study of SCADE, section V discusses the conclusion.

II. AUTOMATIC TRAIN PROTECTION SYSTEM

A. Principle of ATP System

With the rapid evolution of high speed, over loading, and energy saving in railway transportation, old signal control facilities cannot adapted well to the new situation. For conventional train control systems, ensuring compliance with the signals is achieved through operating procedures, and these systems are not particularly efficient in maximizing the utilization of the rail transit infrastructure. The new ATP systems overcome limitations of conventional systems, and therefore, permit more effective utilization of the transit infrastructure, provide continuous safe train separation assurance and over speed protection.

ATP system is made up of two parts, wayside ATP equipment and on-board equipment, as shown in Fig. 1. Wayside ATP equipment is the core of the whole system, all of the logic train protection functions are performed by it. The wayside ATP equipment computes moving authority for each train on line via the information of train location, railway line data, equipment state, and route information from interlocking system.

This work was supported by the National Key Technology R&D Program of P. R. China under grant No.2006BAG02B04 (WB307001), and Beijing Municipal Science and Technology Commission under grant No.D07050601770701 (WH08001).

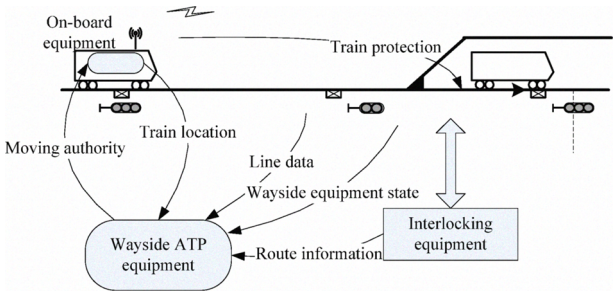

Figure 1. Structure and principle of ATP system

B. *Challenges in ATP Software Development*

At the present time, the main challenges that we face when developing ATP system or other safety-critical software can be described as follows:

(1) Mastering software complexity

The architecture and functionality of ATP system is very large and complex. It can be distributed over several control zones, has to control tens of trains and has complex interdependencies. The complexity is embodied not only to safety, but also to functionality.

(2) Avoiding multiple descriptions

Software development is divided into several phases. At each phase, it is important to avoid rewriting the software description. Such rewriting would not only be expensive, it would also be error-prone. Major risks of inconsistencies are very likely between different descriptions. This necessitates devoting a significant effort to the compliance verification of each level to the previous level.

(3) Fighting ambiguity of specifications

In conventional development, system requirements and design specifications are written in some natural language, its inherent ambiguity can lead to different interpretations depending on the readers.

(4) Finding design errors in early phase

In conventional development, many specification and design errors are only detected during software integration testing. This is too late. The cost of fixing it is much higher than if it had been detected during the specification phase.

(5) Improving verification efficiency

The level of verification for ATP software is much higher than for other non safety-related commercial software. For SIL 4 software, verification is also a bottleneck to project completion.

III. MODEL-BASED DEVELOPMENT METHOD FOR SAFETY CRITICAL SOFTWARE

A. *Safety-Oriented Development Methodology*

Firstly, we present a definition of safety as this can be found in IEC61508 [5]:

• ***Safety*** is the freedom from unacceptable risk of physical injury or of damage to the health of the people, either directly or indirectly as a result of damage to property or to the environment.

Software safety is part of the overall safety that depends on a system or equipment operating correctly in response to its inputs. For safety critical software such as ATP system, there are two types of system requirements must be explored, which are system function requirements and system safety requirements. This is why safety critical software is more complicated than normal one. Designers must take various technologies aiming at safety into account.

For safety-related system development, both standard IEC61508 and EN50128 define a V-model software development life cycle. Each phase of this software development life cycle is divided into elementary activities with the scope, inputs, and outputs specified for each phase. This cycle is presented in Fig. 2. For each life-cycle phase, appropriate techniques and measures should be used.

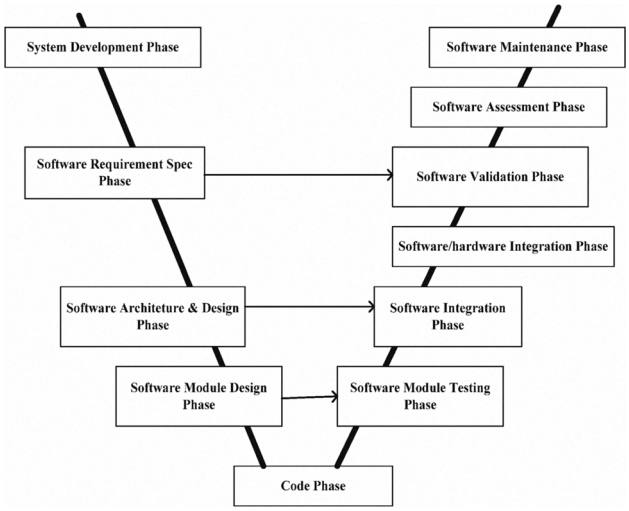

Figure 2. V-model software development life cycle

The novel model-based software development is different from conventional method, model technique dominates the overall life cycle, and it is adopted in development phase as early as possible. In this paper, we propose a safety-oriented methodology based on model technology, which is described in Fig. 3. There are three parallel processes in the development model, design process, safety process and validation process, each process has dependency relationship to the other two processes.

In design process, the function requirements are derived from system purpose and customer's prescripts. In the following phases, model design must according to this requirement specification. And a convenient advantage of model-based development is code auto-generation. In safety process, the safety requirements are derived from a risk analysis, where risks sources for system and associated environment have to be identified. This analysis determines the safety rules and safety properties. Safety is therefore a method of dealing with risks to eliminate them or reduce them to an acceptable level. Validation process of the model-based

development has no difference in essence, as software code auto-generation, the traditional module test activities are alleviated greatly.

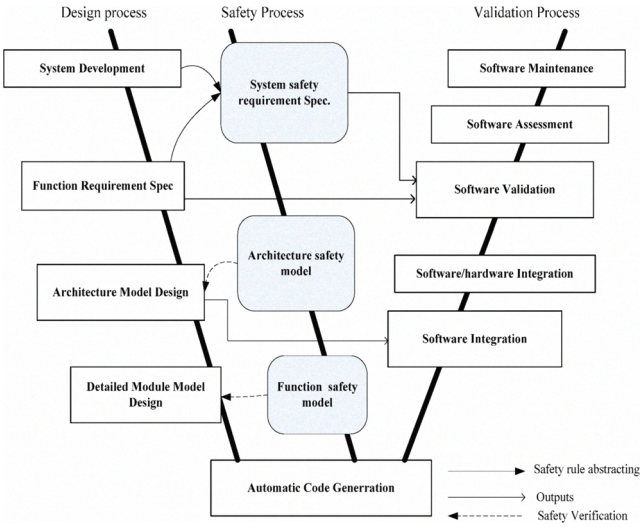

Figure 3. Safety-oriented development life cycle

B. *Model-Based Safety Verification*

Safety critical systems are usually complex, making it impossible in practice to fully determine every failure mode or test all possible behavior. It is difficult to predict the safety performance, although testing is still essential. Nowadays, it appears that model-based method can meet the challenge with an optimistic future.

Generally, many ways can be followed for verifying design, and also there are so many commercial tools for software test and verification, such as LDRA Testbed, Telelogic Logiscope, formal model checking tools--SPIN, NuSMV, and so on. But verification with special tool is usually a difficult and hard work. Here, we insist that do design and verification with a same model-method, if possible, a same tool should be used. Our verification method can be described as Fig. 4. Safety model is designed according to system safety requirement specification. To verify whether the function model is safe, we create a verification model which encompasses function model and safety model. The input and output information of function model, and other needed information from system environment are linked to the safety model as input information, the output of safety model is a Boolean value, it tells whether the function model is correct or not.

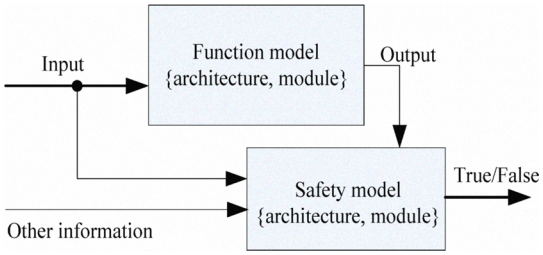

Figure 4. Model-based verification method

IV. DEVELOP USING SCADE

A. *SCADE Suite Environment*

The SCADE [3, 4] environment was developed by the Esterel Technologies. This environment, derived from the LUSTRE language, supports the synchronous data flow approach for the development of reactive systems. It allows using Lustre language features as graphical operators create user-defined operators and design hierarchical Lustre models. SCADE also has an integrated automatic model checker.

SCADE also allows creating state charts, called "safe state machines" (SSM) [6, 7]. SSM is a visual synchronous model. SSM is integrated in SCADE Suite. SSM is the commercial version of SyncCharts, an academic synchronous graphical model designed in the mid-nineties as a graphical notation for the Esterel language. As such, SyncCharts were given a mathematical semantics fully compatible with the Esterel semantics. In this presentation we use the SSM for the train behavior model design.

B. *ATP Model Design*

The basic idea of modeling the ATP system in SCADE is based on user defined operators for train protection function. In SCADE, train is modeled as operators with inputs of operating instruction, and the outputs of train operator are behaviors of the train. ATP functions can be classified into different type of tasks, such as task for main line tracing, reversing, and running through zone boundary, and so on. All of the tasks are working for train moving authority.

1) *Architecture model*

In our design, the hierarchical structure of the ATP system function model consists of four layers. Fig. 5 illustrates the architecture model of ATP system. The first layer is the overall production module. Its inputs are information from others sub-system such as interlocking subsystem, on-board subsystem, wayside equipments, and so on. The outputs are instruction or moving authority for a train. The next layer contains MA_calculate operator and Input/output_Analysis operators, which perform train behavior management and train protection logic computing. The third layer contains implementation contents of train moving authority computing. And the last layer encompasses elementary operators for its upper layer.

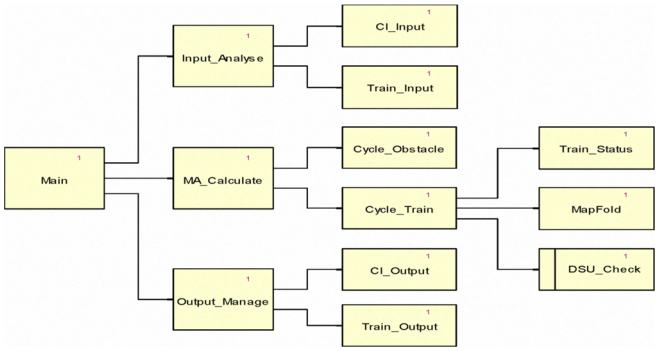

Figure 5. Hierarchical structure of the ATP model

2) *Train behavior model*

Train behavior modeling is very important in ATP system design, and is different from modeling system architecture. Architecture model can be organized in the way of dataflow with SCADE, but train behavior is a finite state process, dataflow is not suitable, so we model train behavior with SSM.

SSM is a variant of the finite state machine (FSM) model. At each instant there is one and only one active state. The initial state is the first activated state. Transitions between a source state to a target state are of two kinds: weak abortion transitions and strong abortion transitions. Labels are optionally attached to transitions and states. A transition label has two optional fields: a trigger, and an effect. A trigger may be a single signal or a combination of signals using the and, or, and not operators. An effect may be a single signal or a set of signals. A state label has only an effect field, which is a set of signals.

In ATP system, behavior of a train is a series of sequential logic action, which contains logging in, moving, reversing, takeover, exiting, and so on. The SSM has corresponding states to each behavior action; they are S_INIT, S_Moving, S_Reverse, S_Takeover, S_Handover, and S_Cancel. Fig. 6 describes the train behavior model in detail.

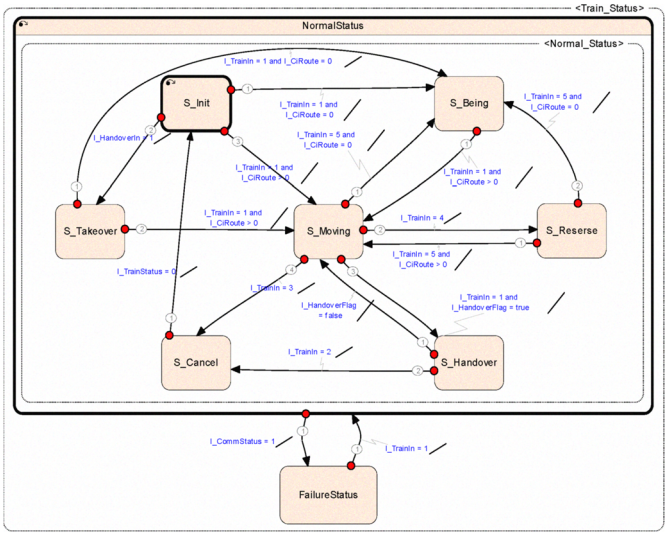

Figure 6. Train behavior model

A train behavior in the model can be denoted as a 4-tuplet:

<tran_type, trig_cond, action, next_state>

Where, tran_type can be strong abortion or weak abortion of the SSM, trig_cond is the state trigger condition, usually it is a Boolean expression, action is what the machine to do under the current state, and next_state is the aftermath of action.

Train behavior model above is only for one train, as ATP system should control tens of trains on rail line, all of the trains are in a same behavior model. Different from conventional methods, SCADE present an iteration method. We define an operator that iterates the train behavior model with Mapfold operator of SCADE, to make sure that all of the train can be control. The operator is shown in Fig. 7.

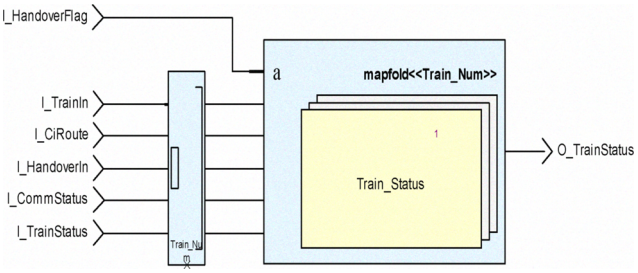

Figure 7. Train behavior model iteration

V. CONCLUSION

Mastering complexity and high demand for safety are the main challenges of safety critical system development. This paper presented a methodology of model-based development to solve the problem. The safety-oriented software life cycle and verification approach is proposed. Furthermore, we have presented a case study of ATP system based on SCADE toolset. The main advantages of this approach are:

- Fully deterministic graphical modeling technology gives a strong support to mastering software complexity.
- Help engineers to avoid multiple descriptions and ambiguity of specifications.
- Achieving verification on the basis of safety model, and finding design errors in early phase.

REFERENCES

[1] BS EN 50128: 2001, "Railway applications Communications, signalling and processing systems Software for railway control and protection systems", European Committee for Electrotechnical Standardization, May 2001.

[2] Holger Giese, Stefan Henkler, "A survey of approaches for the visual model-driven development of next generation software-intensive systems", Journal of Visual Languages and Computing, Volume 17, Issue 6, December 2006, pp. 528-550.

[3] P. A. Abdulla, J. Deneux, G. Stalmarck, H. Agren, and O. Akerlund, "Designing safe, reliable systems using SCADE", in Proceedings of ISOLA'04. Springer-Verlag, 2004.

[4] Gudemann M., Angerer A., Ortmeier F., Reif W., "Modeling of self-adaptive systems with SCADE", IEEE International Symposium on Circuits and Systems, 27-30 May 2007 pp. 2922-2925.

[5] IEC61508: Functional safety of electrical / electronic / programmable electronic safety-related systems, IEC, April 1999.

[6] Esterel Technologies, Efficient Development of Safe Railway Applications Software with EN 50128 Objectives Using SCADE Suite™, Esterel Technologies, Paris France, 2003.

[7] http://www.esterel-technologies.com.

[8] C. André, Semantics of SSM (Safe State Machines). Guyancourt, Paris France, April 2003. http://www.esterel-technologies.com.