Modeling Safety-Critical System Requirements with Hierarchical State Machine

Zheng Wang[1], Chen-ge Geng[1*],Xiang-xian Chen[1], Dong Wang[1], Hai Huang[1], Ai-ai Guan[1]
1 Department of Instrument Science and Engineering
Zhejiang University
Hangzhou, China
Email: gengcg@gmail.com (*Corresponding author)

***Abstract*—Automatic Train Protection (ATP) system is a safety-critical system; it is widely used to ensure trains running safely. During its development lifecycle, there are many safety problems which are derived from the requirements. In order to make descriptions of the requirements accurate and consistent, we introduce requirement models in the development lifecycle. The requirement models are built based on the modified state machine with text descriptions and the introduction of Super-State. In these models, the limitations of transitions are well defined and the complexity of the models can be reduced effectively. With this approach, the requirement of train localization function of ATP system is described clearly and strictly. Besides, this requirement model is easy to understand and read for developers.**

Keywords- Automatic Train Protection (ATP) system; safety-critical sysem; lifecycle; requirement modeling; hierarchical state machine model

I. INTRODUCTION

Due to the growing traffic intensity, railway signaling systems are widely used with their characteristics of high-speed and high-safety in recent years. The automatic train protection (ATP) system, which is used to ensure a safe distance between successive trains, plays an important role in railway signaling systems. It is a software-intensive and safety-critical system with the safety integrity level (SIL) of SIL4 [1]. To develop such a system is very difficult and a long development lifecycle is required. What's more, the developers usually need to fix a large amount of errors during the development process. A study from Tavolato and his coworkers shows that 56% of the errors which are discovered during the development lifecycle can be traced back to the requirement phase. Their study also indicates that the later an error is found in the development lifecycle, the higher cost it is to fix [2]. Therefore, it is necessary to consider the reliability, integrality and accuracy of the requirements of ATP system.

So far, formal methods have been studied and used in the development of the ATP system [3-8]. Formal method is a particular kind of mathematics-based technique; it describes the requirements with mathematical formulas, logical illations or deductive proof. By using formal methods, the requirements of ATP system can be expressed clearly and unambiguously. However, it is difficult to understand and read them for the developers, especially those with no experience or professional training. Compared with formal methods, Unified Modeling Language (UML) is more comprehensive, simple, visual, standardized, and easier to master by developers [9]. Beyond that, because the developers don't need extensive training, the efficiency of development can be improved sharply and the costs can thus be reduced greatly [10-12].

In view of these, a requirement modeling method based on modified state machine for ATP system is presented by us. It is an object-oriented method and can accurately describe the requirements of ATP system. The inconsistencies and ambiguities of the requirements will be effectively decreased, and the accuracy and reliability of the system will be markedly improved.

This paper is organized as follows: The functions of ATP system will be described in brief first and then the requirement models, following are the knowledge of UML and the modification of state machine models. The method is applied into the requirement modeling of ATP system later. A conclusion will be made at the end of this article.

II. ATP SYSTEM AND REQUIREMENT MODEL

ATP system plays a significant role in the Automatic Train Control (ATC) system. It is a core part of the carborne control system, performing all the safety protection functions for trains. Through the calculations of the received signals, such as the semaphore signals, switch statuses, interlocking logic signals and et al., ATP system can always ensure that trains run with a safe speed [13]. Besides, it can monitor all the statuses of the devices on the carborne, including doors, sensors, train brake system and et al.

It is clear that the software of ATP system must be high-safe and high-reliable, otherwise any failure of the system will be dangerous, even fatal or catastrophic. During the software development lifecycle of ATP system, a requirement specification which can clearly describe all the software functions is demanded intensively. The functions involved in the requirement specification should be defined completely and unambiguously. What's more, the fact is inevitable that the developers may misunderstand the requirements during the software development if the requirements are written in natural language.

For the reasons above, we introduce requirement models in ATP software development. The requirement models are established with UML on the basis of requirement specifications. The models are expressed by graphic symbols

978-1-4799-3197-2/14/$31.00 ©2014 IEEE

and text descriptions. The graphic symbols represent mathematic and logic relationships between input data and output data, the text descriptions are the supplement including the definition, rule and limitation. Therefore, it is easy for the developers to analyze them and transform them into other forms.

Fig.1 shows a V-model lifecycle of the software development. It is noted in the EN 50128 Standard. The Requirement Models are created based on the analysis of the requirements in the Software Requirements Spec Phase, which is represented by the solid arrowhead. During other phases, including Software Architecture & Design Phase, Software Integration Phase and Software Validation Phase, the models are the most important bases and references for the developers, which are represented by the dash arrowhead.

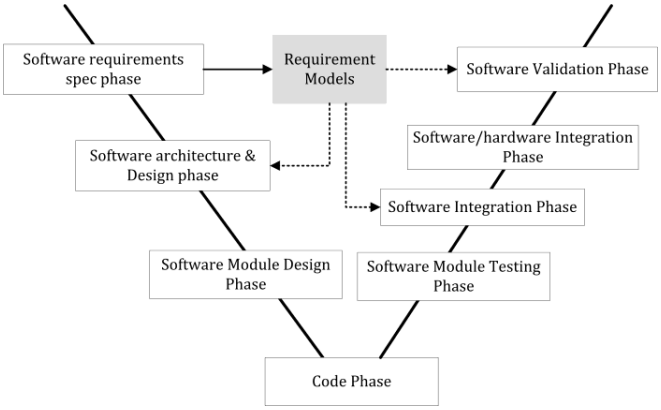

Fig.1 Requirement models in the V-model lifecycle

III. HIERARCHICAL STATE MACHINE MODEL

A. Unified Modeling Language

Unified Modeling Language (UML) is a semi-formal modeling language including a set of graphic notation techniques. It integrates the notations of the Booch method, the Object-Modeling Technique (OMT), and Object-Oriented Software Engineering (OOSE). It also supports other object-oriented technologies [14, 15]. In addition, multiple types of graphic representations such as Use Case Diagram, Class Diagram, Sequence Diagram, State Diagram, Activity Diagram, etc., can be used in various phases of the system development lifecycle. Therefore, The UML can create visual models for the software of a highly intensive and complex system like ATP system and it can be applied into all processes during the software development lifecycle.

There is no doubt that the requirement models of ATP system illuminated above can be built easily by UML techniques. The models make the system architecture visual, clear, and legible. More importantly, they can be transformed into other graphic representations which can be applied in the latter phases. But there exists a problem that the number of the models could be too large or the models are too complicated to read, because the ATP system involves all the safety-related functions and it has various mathematic and logic calculations.

B. Modified State Machine Model

State machine is one of the UML notations; its state describes a behavioral node for the software which can transform to another node when the transition is triggered [16]. Apart from that, state machines can be created in a hierarchical structure, which can reduce the complexity and quantities of the states. The hierarchical nested state is named Super-State and it includes a series of independent states. The Super-State is treated as a single state in high-level models, while in low-level it expands into a series of states.

With the introduction of the Super-State, the requirement models of ATP system can be obtained in the form of modified state machine models. As the state machine model is a semi-formal model, the transitions of the model lack strict formal definitions. In order to describe the functions of the ATP software clearly and accurately, the limitations of the text description are required, which are described by expressions and shown as follows.

$$T\_[Event1\ 1, Event2, \ldots] \quad (1)$$

The expression includes some events which describe input data statuses. The expression is valid when all of the events are eligible. Each transition in the models is confined by one or some expressions and it is triggered only when one of these expressions is eligible.

For instance, the state machine with an expression showed in Fig.2 consists of two states, including *St_1* and *St_2*. The *St_1* will transfer to the *St_2* only if the expression *T_[A, ~B, ~C]* is eligible, that is event A is true and event B, event C both are false.

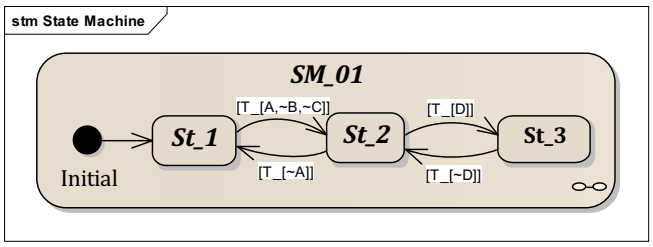

Fig.2 A example of state machine with an expression

IV. IMPLEMENTATION

A. Requirements of Train Localization Function

Train localization function is one of the most important requirements of ATP system. The train receives the beacon information when it runs across a beacon. The information includes identification, beacon data, max impulse count, and min impulse count. Based on the train speed and the beacon position from the track map, the train position can be calculated. When the train comes by the other beacons, the train position can be checked and calculated again. In this way, it is possible to obtain the train position instantaneously and accurately. In addition, the safety characteristic is considered. The

localization process is divided into several states which are listed as follows.

- LOC_NOT: The train position is unknown;
- LOC_INIT: ATP system is ready for locating;
- LOC_REV1:ATP system receives the information of first beacon;
- LOC_REV2: ATP system receives the information of second beacon;
- LOCAL: The train position is known;
- FAULT: Some unsafe failures occur ;

B. *Analysis of Data flows and Events*

According to the definitions, the data flows of train localization function are analyzed and shown in Fig.3. The external input data includes *BTM_INFO* received from beacons, *SPD_INFO* from the Speed Measure Module of ATP system, *LOC_INFO*, and *TRACK_INFO* which contains all the positions and attributes of beacons. *SPD_INFO* consists of maximum and minimum train speeds which are named *MAX_SPD* and *MIN_SPD,* respectively. *LOC_INFO* includes the train position and impulse counts of the last cycle. *OUT_STATE* and *OUT_POS* are exported after calculation. *OUT_STATE* is the state defined above and *OUT_POS* is the position of the train.

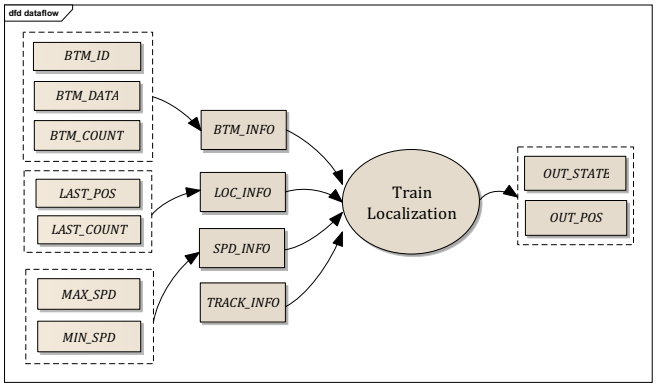

Fig.3 Data flow diagram of train localization function

The events which determine the triggers of the transition between states can be summarized after the analysis of the data flows, which are listed in Table I. The transitions of localization states are triggered by different combinations of these events. According to the definitions of these states, the essential and prerequisite conditions of each mode are determined, which is shown in Table II.

In addition, the input data, output data, and some rules are defined in the text descriptions. The rules in following should be followed when changing state and thus the safety of this function can be improved.

- Only one state exists in every cycle;
- The state should be checked in every cycle;
- The transition between these states is triggered by a unique condition.

TABLE I. A EVENTS LIST OF TRAIN LOCALIZATION FUNCTION

num	Event
1	The train receives the first beacon information.
2	The train receives the second beacon information.
3	The train speed of Speed Measure Modul is usable.
4	The train doesn't stop.
5	The train doesn't go astern.
6	The directions of the train and beacons are inconsistent.
7	The beacon information is correct.
8	The information of track map is correct.
9	The difference of maximum and minimum position is accredited.
10	The train performs emergency brake.
11	The train integrality is normal.
12	The train misses information of two continuous beacons.
13	The speed sensors break down.

TABLE II. THE ESSENTIAL AND PREREQUISITE CONDITIONS OF STATES

N	*LOC_ NOT*	*LOC_I NIT*	*LOC_ REV1*	*LOC_ REV2*	*LOCAL*	*FAULT*
1	/	/	R	/	R	/
2	/	/	/	R	R	/
3	/	R	R	R	R	/
4	/	R	R	R	R	/
5	/	/	R	R	R	/
6	/	/	/	R	R	/
7	/	/	R	R	R	/
8	/	/	R	R	R	/
9	/	/	/	/	R	/
10	/	/	/	/	/	/
11	R	R	R	R	R	/
12	/	/	/	/	/	/
13	/	/	/	/	/	R

(Notes: (a) R—required condition; (b) /—not required condition)

To simplify the requirement model, *LOC_REV1*, *LOC_REV2*, and *LOCAL* can be seen as a Super-state *LOC_PRO*. The prerequisites of these sates are the same. The hierarchical structure of simplified states is shown in Fig.4.

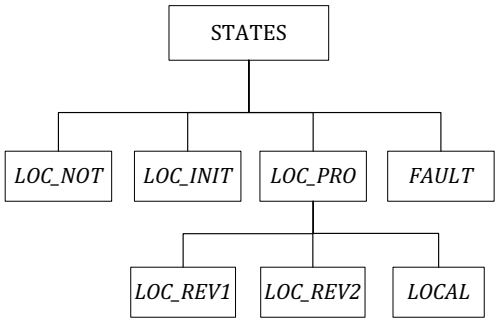

Fig.4 The hierarchical structure of localization states

C. Software Requirement Model of ATP

The requirement model of the train localization function can be obtained by using modified state machine model, which has a hierarchical structure (Fig.5). The transition is triggered only if the conditions are eligible. *LOC_PRO* state is Super-State, which contains several independent states.

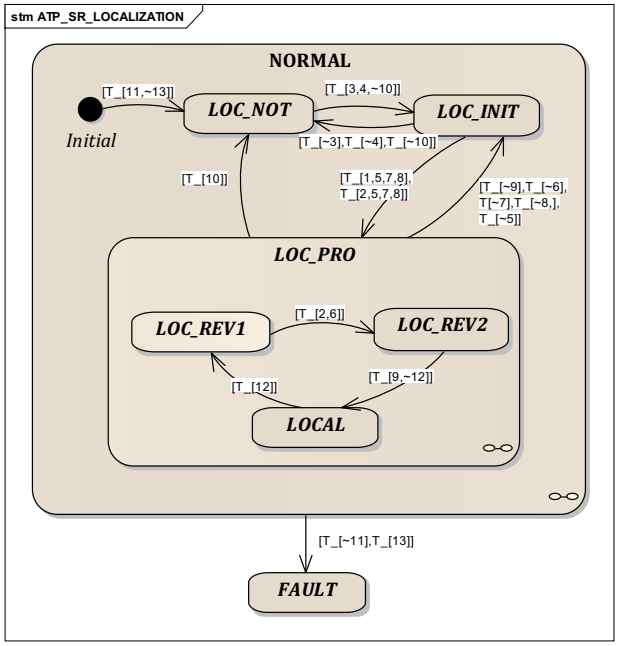

Fig.5 Requirement model of the driving mode management

V. CONCLUSION

This paper presented an approach to model safety-critical system requirements with modified state machine model for ATP system. With the introduction of the Super-State, the models can be built hierarchically and their complexity is reduced a lot, compared with formal methods modeling. In this way, the requirement models of ATP system are visual and clear. There is no inconsistency and ambiguity in these models. Thus, the hierarchical state machine not only avoids the shortcomings of formal methods modeling and natural language, but it also makes the developer easily understand them and transform them into other forms. Human errors can also be avoided to the maximum.

In addition, the models can be used in other phases to shorten the development lifecycle and improve the safety of software, such as Software Architecture & Design Phase, Software Integration Phase and Software Validation Phase. In fact, the models could be applied into automatic generation of software in safety-critical systems, such as ATP system, which may maximum reduce human errors.

VI. ACKNOWLEDGMENT

This work was supported by the Science and Technology Program of Zhejiang province, China (Grant NO.: 2012C21065), and also supported by the National Key Technologies R&D Program (Grant NO.: 2011BAG01B03).

REFERENCES

[1] CENELEC, "EN50128:Railway Applications - Communications, signaling and processing systems - Software for railway control and protection systems," EN 50128: 2001 N IEC 62279: 2002, pp. 17-44.

[2] P. Tavolato and K. Vincena, "A Prototyping Methodology and Its Tool," In Approaches to Prototyping, 1984, pp. 434-436.

[3] S. Liu, M. Asuka, K. Komaya, and Y. Nakamura, "An approach to Specifying and Verifying Safety-Critical Systems with Practical Formal Method SOFL", In Proceedings of Fourth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'98), California, USA, 1998, pp. 100.

[4] H.J. Jo, J.G. Hwang, and Y.K. Yoon, "Formal Requirements Specification in Safety-critical Railway Signaling System," IEEE T&D Asia 2009, pp.1-4.

[5] C. Livadas, "Formal Verification of Safety-Critical Hybrid Systems," master's thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Sept.1997.

[6] J.G. Hwang, H.J. Jo, and J. H. Lee, "Model checker for railway signalling communication protocol," WIT Transactions on The Built Environment, vol 88, 2006, pp. 675-682.

[7] H. Wang, C. Gao, and S. Liu, "Model-based Software Development for Automatic Train Protection system," IEEE Computer Society, 2009, pp. 463-466.

[8] P. Behm and P. Benoit, "A Successful Application of B in A Large Project," FM99-Formal Methods, 1999.

[9] W. Schafer, H. Wehrheim, "Model-Driven Development with Mechatrinic UML," Springer-Verlag Berlin Heidelberg, 2010, pp. 533-554.

[10] X. Niu and Q. Liu, "Survey of Requirement Modelling in Software Engineering," Microcompuier Applications, 2006(11).

[11] E. Hull, K. Jackson, and J. Dick, "Requirements Engineering. Springer," 2010, pp.3, 6, 47-76.

[12] Y. Shi, "Application of UML in Requirement Modeling of Spacecraft Attitude and Orbit Control Software," Aerospace Control and Application, 2008(6).

[13] IEEE Recommended Practice for Communications-Based Train Control (CBTC) System Design and Functional Allocations, IEEE 1474.3, 2008.

[14] G. Booch, J. Rumbaugh, and I. Jacobson, "Unified Modeling Language User Guide," Addison-Wesley Professional ©2005, ISBN:0321267974.

[15] G. Booch, J. Rumbaugh, and I. Jacobson, "The Unified Modeling Language for Object-Oriented Development," Rational Software Corporation, 1996.

[16] J. Lilius, IP, Paltor, "Formalising UML state machines for model checking," The Uinfied Modeling Language: Beyond the Standard, v1723, 1999, pp.430-445.