Trustability Improvement of an Automatic Train Protection System

Jing Liu'', Xijiao Xiong' , Dehui Du', Zuohua Ding[2], Geguang Pul

[1] Shanghai Key lab of Trustworthy Computing, East China Normal University, Shanghai, 200062, P.R.China
jliu@sei.ecnu.edu.cn, vuvaoxja,gmail.com, {dhdu,g2pul@sei.ecnu.edu.cn
[2] Center of Math Computing and Software Engineering, Zhejiang Sci-Tech University, Hangzhou, 310018, P.R.China
Zuohuangding@hotmail.com

Abstract. The trustability of control software used in automatic train protection system is crucial to ensure safety of the system. Model-driven paradigm is becoming a de-facto industrial standard for modeling and designing safety critical control systems. This has led to an increased emphasis on setting up a mechanism that can be used to guarantee the correctness of the models. In this paper, we report on a successful application of the experience with Automatic Train Protection system (ATP) modeling techniques. The ATP models are constructed and refined based on Refinement Calculus of Object Systems (rCOS) in Model Driven Architecture (MDA). The time critical interaction mechanism of components is modeled by our extended timed automata.

Keywords: formal method, rCOS, automatic train control, Timed automata.

1 Intro ductio n

With the rapid improvement and innovation of automatic control systems, the complexity of many control systems, especially safety critical systems requires the application of a battery of such techniques. One of the most promising approaches is model driven development.

Up to date, the most advanced train control system is the Communication Based Train Control (CBTC) system [1]. Its rapid deployment has enabled the primary safety features that can be gained from the Automatic Train Protection (ATP) to be introduced for the most safety critical areas [2, 3]. Thus how to model and verify the systemto fmd the design defects has become one of the key issues of ATP [4].

Our approach is sketched as follows. Firstly, an interface model of component is presented with respect to Component-Based and Model-Driven Development specification. The models are defined based on the Refinement Calculus of Object Systems (rCOS) [5, 6]. Secondly, a language-independent dynamic behavior model is constructed by adding the concept of multiple-time to timed automata. In Interface Automata, we introduce the concept of clocks and extended time constraints of timed

[1] Corresponding author

automata. These automata can model the real time interaction process directly. We apply this model to analysis of the timed behavior the ATP system.

The paper is organized as follows. Section 2 presents the architecture model of the component based system by rCOS. Section 3 elaborates on real-time analysis based on component model architecture. Section 4 states the application of the approach to our ATP system. Finally in section 5, we conclude our work.

2 Specification **and Implementation Models**

rCOS is developed to support model-driven development methods. It provides with multi-view modeling environment and combines object-oriented and component-based design and analysis techniques [7].

Class Model In rCOS, a class can be specified:

```
class          C [extends D]
invariant Inv
attr           Ti = ,            , Tk Xk= dk
method         m(T in; V return) {
               pre:    p V ... V p
               post:   A (R; . . . ;R) V . . . V (R; . . . ;R)
                       A ............
                  1* method m *1
method         m(T in; V return) { .........
```

3 Interface Automata of Components

To describe the timing features of real time system, we extend timed automata to enhance the ability of modeling multi clock system. Furthermore, we use an automaton to represent the interaction of components with its environment.

3.1 Time model

Definition 3.1 [Clock] A clock C is a tuple **(I, L, M,** **,** where

- I is a set of finite or infinite instants,
- Lis a set of labels,
- M= **I —>** L is a mapping that associates labels to instance.
- is a total, irreflexive and transitive binary relation on I, named precedence.
- u represents a unit.

To express the multiple time features, we extend the clock constraints.

Definition 3.2 [clock constraints] For a set C of clock variables, the set D(C) of clock constraints 8 is defined inductively by

- *c m* has the original meaning in timed automata, which means the time of clock *c* is less than or equal to *m*.

- *c > m* means the time of clock *c* is more than or equal to *m;*
- -< (precedence) is a binary relation on [LA , which is reflexive and transitive. Formally, let *A* and *B* be two discrete-time clocks. *A* **precedence** *B* (denoted *A -< B)* if *(et* e *4*) *(k = i dx* B *(i)) = A[k] B[k].*
- **(coincidence)=df --** n ›--.
- **-< (strictly precedence)** is a strict form of **(precedence):** *A* **StrictlyPrecede nee** *B* (denoted *A -< B)* if (Vi E /B) *(k = i dx B (i)) A[k] -< B[k].*
- **— (Alternates):** *A* **Alternates** *B* (denoted *A — B)* if (Vi E *IA) (k = idx A (0) A[k] -< B[k] AR +1].*

3.2 Interface Automata

We define interface automata to model the state transition of dynamic process.

Definition 3.3 [Interface automata] For a component *K*, an interface automaton *e[K]* = *(S,s0,E,E,C)* is a finite state automaton where,

- S is a finite set of states.
- so E *S* is the initial state.
- E is the alphabet set, where E = { *in /!out 1 in EK.Pp.M Aout E K.Pr.M** }. The event *in* represents receiving provided service from reference interface and *!out* is a fmite sequence of services.
- *E g S* x ExSx2c x o(C) is the E-labeled transition relation. An edge *(s, a1, s ',9,(3`)* represents a transition from state *s* to *s'* on input symbol *a*. The set *0 c c* gives the clock to be reset with this transition, and a e (NC) is a clock constraint, and
- *C*is a finite set of clocks.

4 Design of Automatic Train Control System

Communication Based Train Control (CBTC) is a train transportation control and protection system, which usually consists of Automatic Train Protection (ATP), Automatic Train Control (ATC) and Automatic Train Supervision (ATS) and ATP system is the core subsystem of CBTC. CBTC ensures the safe and accurate operation of trains on Automatic Train Control enabled lines [8].

4.1 System Overview

ATP ensures the safe and accurate operation of trains on Automatic Train Control enabled lines. ATP consists ofthe following devices (subsystems):

- *ATP Processing Unit (PU)* to receive train control commands from *Trainlines* and compare the train speed with limiting speed. It will also give braking command to *Braking Equipement (BE)* to apply brake.
- *Trainlines,* through which the control command and limiting speed are sent to *ATP Processing Unit.*
- *Mode Direction Handler (MDH)* for driver to select the drive mode, and each drive mode has a corresponding standard limiting speed.
- *SpeedSensor to* send actual speed of train to *ATP PU.*
- *Train Operation Display (TOD)* to display commands and speed information from *Trainlines* and other warning information from *ATP PU.*
- *Train Information Management (TIM)* for the management of train information such as drive mode and corresponding standard limiting speed.

4.2 Requirement Analysis

Model of UC1 (UC 1): Speed Supervision in RMF
The use case is modeled by the contract of the provided interface of a component: *SuperviseSpeed,* and the provided interface of a component: *SpdSpvDevicelF.*

```
class SpdSpvDevice implements SpdSpvDevicellF::
      invariant ATP t null AATP.speedSensor null A ATP.modeList t null
                  A ATP.TO.D null A TIMt null
      method initialTrain()
                  pre: ATP.lights findLight(mode) 0 null
                  post: mode' = 0 /*default drive mode: ATO */
                        ATP. lights findLight(mode).turnOn() /*ATP light turns on */
      method selectRMFMode()
                  pre: ATP.lights findLight(mode) null
                  post: mode' = 3; /*RMF drive mode*/
                        ATP. lights findLight(mode).turnOn() /*RMP light turns on */
      method acquireActualSpeed()
                  pre: true
                  post: aspd'= ATP.speedSensor.acquireSpeed()
      method acquireRestrictedSpeed(int mkt; int rspd)
                  pre: TIMmodeListfindMode(mId) 0 null
                  post: rspd'= TIMmodeListfindMode(mId).getRestrictedSpeed()
      method compareSpeed(int aspd, int rspd)
                  pre: true
                  post: dif= rspd - aspd;
      method announceWarnMsg()
                  pre: 0<dif<criticalSpeed
                  post: ATP. TOD.displayWarn()
      method brakeByHuman()
                  pre: 0<dif<criticalSpeed
                  post: ATP. CBE.brakeByHumun()
      method emergencyBrake()
                  pre: dif<=0
                  post: ATP.EBE.brake()
```

4.5 Dynamic Process Analysis

To ensure that the train is running in a safe condition, the real-time and safety properties of system should be considered during design.

Informal description The whole process of the brake by human and emergency brake with time constraint can be described as follows:

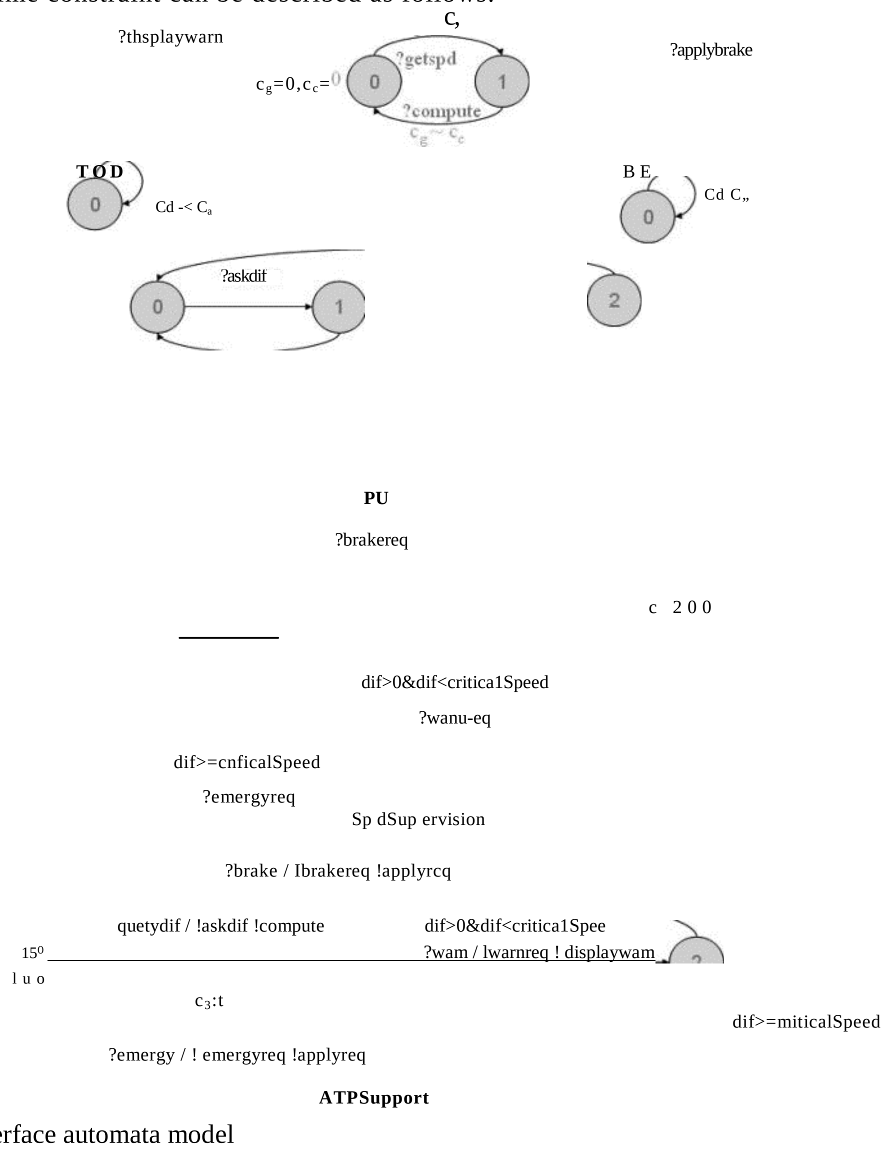

Fig. 1. interface automata model

1. When the *SpdController* asks the *PU* to compute the difference between the limiting speed and current speed, the *PU* returns the computing result, *dif.*
2. The *SpdController* informs the *ATPSupport* (here it acts like a connector) to display the warning message when *dif* is less than *criticalSpeed.*
3. The *ATPSupport* request the *TOD* to display the warning message to the train driver within 200ms. When the warn message is displayed in the *TOD*, the *TOD* will give a feedback message to the *ATPSupport,* which will also inform the *SpdController* that the warn message has displayed.
4. After the *ATPSupport* got the message from the *SpdController* (originally from the driver) to apply brake, it sends a brake request to *BE* and *BE* should finish the brake within 150 ms. Then the *BE* similarly gives a feedback message to the *ATPSupport* and the *ATPSupport* informs the the *SpdController* that brake task has been

fmished.

5. If *dif* is less than or equal to zero, which means the current speed exceeds the limiting speed, the *SpdController* informs the *ATPSupport* to send the command of applying emergency brake to *BE* and *BE* should finish the brake within 100 ms. Then the *BE* similarly gives a feedback message to the *ATPSupport* and the *ATPSupport* informs the *SpdController* that emergency task has been finished.

Interface automata Model According to the description a for *Brake by Human* and *Emergency Brake*, we obtain corresponding interface automata models in Fig. **1.**

5. Conclusion and Discussion

In this paper, we have presented an approach to model ATP System from a unified multiple views. Component based architecture is used to model networked Telemedicine System. We extended Timed Automata with multiply time characteristics by introducing clock and clock constraints. With this extension, we presented a component-based approach for the modeling of real-time system. A component-based system is a number of parallel components to realize an application by invoking services provided by other components. Interface Automata are defined for components interaction.

We will focus on the verification approach and tools base of interface automata model to ensure the correctness and safety of software system.

Acknowledgments. We would like to thank the reviewers of this paper for their valuable comments that help us to greatly improve the paper. This work is partially supported by the projects funded by the 973 program 2009CB320702, NSFC 61170084, NSFC for the Young of China 60903021, and Zuohua Ding is supported by the NSF of China under Grant No.61170015, Zhejiang NSF under Grant No.Z1090357. The Lab is supported by NSFC Creative Team 61021004 and 863 Project 2011AA010101.

References

1. IEEE Vehicular Technology Society, IEEE Recommended Practice for Communications-Based Train Control (CBTC) System Design and Functional Allocations, IEEE Std 1474.32008, 2008.
2. Goddard, E., Overview of signaling and train control systems, *Electric Traction Systems, 2006. The 9th Institution of Engineering and Technology Professional Development Course on* , pp.336-350,6-10 2006.
3. Fei Yan, Tao Tang, A formal modeling and verification approach for real-time system, *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on,* pp.204-208, 25-27 June 2008.
4. Lei Chen, Bin Ning, Tian-Hua Xu, Research on modeling and simulation of vehicle-onboard Automatic Train Protection subsystem of Communication Based Train Control system, *Vehicular Electronics and Safety, 2007. ICVES. IEEE International Conference on ,* pp.1-5, 1315 Dec. 2007.
5. J. He, X. Li, Z. Liu, rCOS: A refinement calculus for object systems, *Theoretical Computer Science*, vol.365, no.1-2, pp.109-142, 2006.
6. X. Chen, J. He, Z. Liu, N. Zhan, A model of component-based programming, Lecture Notes in Computer Science, vol.4767, 2007.
7. Z. Chen, Z. Liu, A. Ravn, V. Stolz, N. Zhan, Refinement and Verification in Component-Based Model Driven Design, *Science of Computer Programming*, vol.74, no.4, pp.168-126, 2009.
8. IEEE Vehicular Technology Society, IEEE Standard for Communications-Based Train Control (CBTC) Performance and Functional Requirements, *IEEE Std 1474.1-2004*, 2004.
9. Tom M ens, Michel Wermelinger, Separation of concerns for software evolution, Journal of Software Maintenance and Evolution: Research and Practice, vol.14, no.3, pp. 311-315, 2002.