Toward developing a Decentralized Railway Signalling System Using Petri Nets

Xinhong Hei[1,2] , Sei Takahashi[1], Nakamura Hideo[1]
1: College of Science and Technology
Nihon University
Funabashi, Chiba, Japan
heixinhongjp@yahoo.co.jp
2: State Key Laboratory of Rail Traffic Control and Safety (Beijing Jiaotong University), China
Beijing, China

***Abstract*— Railway interlocking systems ensure the safe operation of trains in stations by controlling sets of devices and equipment. Component-based decentralized railway interlocking system (CBDRIS) is a recently presented railway signalling in which component-based technology is applied. For such a safety-critical real-time system, fail-safe and fault-tolerance have to be ensured. In this paper, a Petri net-based development strategy of CBDRIS is proposed. The development strategy separates the development process into dynamical process (Standardizing hardware as well as control flow of interlocking device components) and static process (Converting current interlocking table of a specific station to a unified format for CBDRIS). In addition, hardware and soft-ware fault-tolerance measures adopted in CBDRIS are described respectively.**

***Keywords*—Decentralized, Railway signalling system, Component-based, Petri nets, Fault-tolerance.**

I. INTRODUCTION

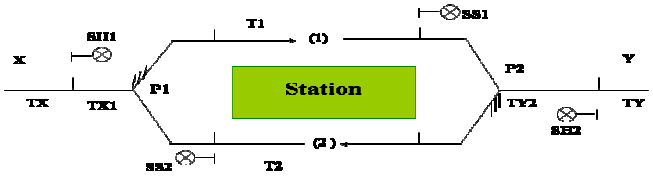
Railway interlocking technology has been developed over the long history of railways, and has been vital in ensuring the safe operation of trains. However, computers have been used into such safety-critical systems only relative recently [1]. Such computerized interlocking systems have demonstrated high level of safety and reliability.

One drawback of existing interlocking systems, however, is that they require the development of different soft-ware for different stations, which tends to introduce unreliable factors. One possible solution to overcome this problem is to apply component-based technology to railway interlocking systems [2]. Component-based technology has seen remarkable progress and has already been adopted in many fields, such as mechanics, control and software engineering [3][4]. Component-based decentralized railway interlocking system (CBDRIS) was such a railway signal-ling system in which component-based technology is adopted to standardize signaling devices both in hardware and software. In CBDRIS, the specification data are related with specific stations, while the software and hardware are standardized, i.e. the control flow and design of devices need not to be developed from start when constructing a new station. The advantages of CBDRIS include higher safety, higher reliability, and lower development costs.

In this paper, we describe a development strategy of CBDRIS. We separate the development process into dynamic process and static process. The dynamic process mainly tries to establish standardized control flows for each kind of interlocking device components, and the static process deals with the interlocking logic relations between interlocking device components for a specific station. We mainly deal with the dynamic process, i.e. designing and modeling the specifications of device components in CBDRIS in this paper.

In order to ensure CBDRIS safety and reliability, fault-tolerance technologies have to be adopted both in hard-ware design and software design. In hardware, we mainly use the Triple Modular Redundancy (TMR) [5] and redundancy policies. In software, an effective time constraint is used to detect the device or communication failures.

II. COMPONENT-BASED DECENTRALIZED RAILWAY INTERLOCKING SYSTEM (CBDRIS)



A part of Interlocking table

Type	Name	Signal	Locking	Signal control	Route locking	Approach locking
Home Signal	X-1	SH1	P1-N,P2-N	TX1,T1	TX1	TX
Start Signal	1-Y	SS1	P2-R	TY2,TY	TY2	T1
Home Signal	Y-2	SH2	P2-N,P1-N	TY2,T2	TY2	TY
Start Signal	2-X	SS2	P1-R	TX1,TX	TX1	T2

Figure 1. A simple station

A. Railway interlocking system

Railway signalling system is used to control train safety. In railway signalling, interlocking systems protect train safety by controlling sets of devices and equipments in a station. Typically, these devices include signals, points and track units. Signals indicate whether the train can run or not by displaying green or red. Points are devices for controlling turnouts which determine the direction in which trains move. Track units detect whether or not there is a train on the track. If there is,

*: The project is supported by the State Key Laboratory of Rail Traffic Control and Safety (Beijing Jiaotong University), China.

then other trains are prohibited from entering this section of track until the first train leaves.

A simple station is shown in Fig.1. There are four routes, two home signals (SH1 and SH2), two start signals (SS1 and SS2), two points (P1 and P2), and two platforms in the station. The table below is part of the interlocking table for this station. Each row of the table defines the conditions for signals changing to green. For instance, home signal SH1 indicates the route X-1, which shows whether a train can move into platform (1). Signal SH1 can change to green only when point P1 and P2 are in normal position ("N") and when there is no train on track units TX1 and T1. Start signal SS1 indicates whether a train can depart from the platform (1) through route 1-Y.

B. Component-based Decentralized Railway Interlocking System (CBDRIS)

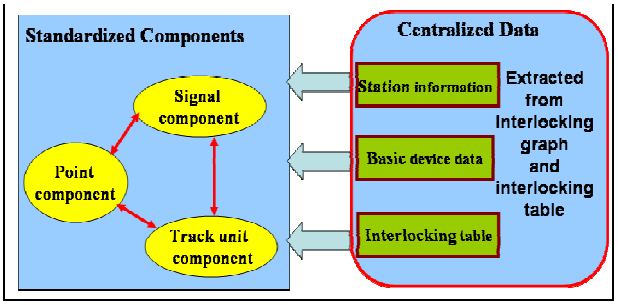


Figure 2. Architecture of CBDRIS

The system architecture and process routine of CBDRIS are shown in Fig.2. An interlocking device in past signalling systems is designed as an interlocking device component. All devices components in CBDRIS are autonomous and intelligent, and they communicate with each other directly without centralized interlocking computer. Each kind of interlocking device components has the same functions and same control logic. So they can be standardized both in hardware and software. These standardized components combine logic relations and data of a specific station to execute defined controls. The logic relations and data are extracted from interlocking graph and interlocking table of the station. For instance, signal components include execution flows which are designed before-hand, such as receiving route request messages, preparing destination devices of communication, sending messages to related devices, receiving the response messages and analyzing them, executing route setting and canceling route request, etc.

When interlocking components are initialized, the extracted centralized data are downloaded to the device components, where upon the device components start to operate and interact with each other to complete the route interlocking.

The internal structure of interlocking components is shown in Fig 3. An interlocking device component typically consists of four modules: communication module, computing module, monitor module and control module. Centralized data as well as status of device components are stored in the local memory. The communication module allows the device component to communicate with other device components or other systems, such as automatic train stop (ATS) system [6]. The computing module performs logic computing and judgment and decides whether the logic status is correct, what status the device component should be in, and so on. The control module is in charge of device control output. The monitor module monitors the status of the device component by periodically polling the related devices and issuing an alarm when the status is abnormal. All these modules can access local memory via an internal system bus.

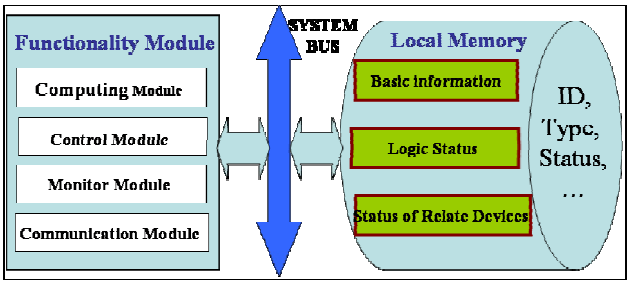


Figure 3. Internal structure of interlocking component

III. DEVELOPMENT STRATEGY OF THE CBDRIS

The development of CBDRIS can be performed from two aspects: dynamical development process and static development process.

The dynamic development process deals with standardized hardware and software for the interlocking device components. Control flows of the interlocking device components are based on their function specification. The static development process deals with centralized data such as logic relations between interlocking device components in a specific station.

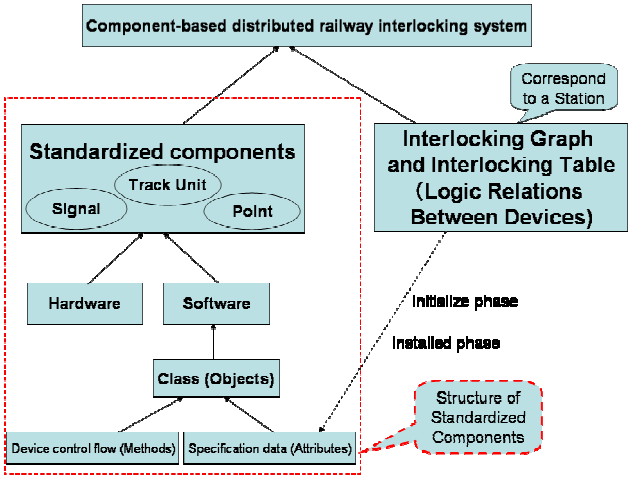


Figure 4. Development strategy of the CBDRIS

The development strategy of the CBDRIS is shown in Fig. 4. Structure of standardized device components consists of hardware part and software part. The hardware specifications include device board design, CPU, digital circuit, input/output, etc. The software specifications include the four modules design of typical interlocking device components. Each module in Fig. 3 is similar with a class or object which consists of attributes and methods. The device control flows are expressed by methods; while centralized data related to a specific station are expressed by attributes.

When the device components are initialized, the centralized data will be loaded into the components, and then the components operate based on these data.

Once the device components have been verified safe enough, they can be ordered and produced when a new station is constructed. What engineers just need to do is analyzing the logic relations and allocating some basic attributes such as device ID to each device component.

Each module in Fig. 3 is modeled with G-nets which are Petri-nets extended with object concept [7]. Thus the whole system will become a G-net system. By using analysis tools of Petri-nets, verification and analysis of software part can be performed.

G-net is a Petri-net based on multi-level executable specification model which incorporates the concepts of module and system structure. A G-nets system comprises of a number of G-nets, each of which represents an independent module. These modules communicate with each other through well-defined interfaces, that is, the methods of G-nets. A G-net is composed of two parts: a special place called Generic Switch Place (GSP) and an Internal Structure (IS). The GSP provides the abstraction of the module, and serves as the only interface between the G-net and other modules. The IS, a modified Petri-net, represents the detailed internal design and realization of the module. The methods and attributes for a G-net can be identified. In the internal structure, places represent primitives, while transitions, together with arcs, represent connections or relations among the primitives. A set of special places called Goal Place (double circle) represents the final state of the execution, and the results (if any) to be returned.

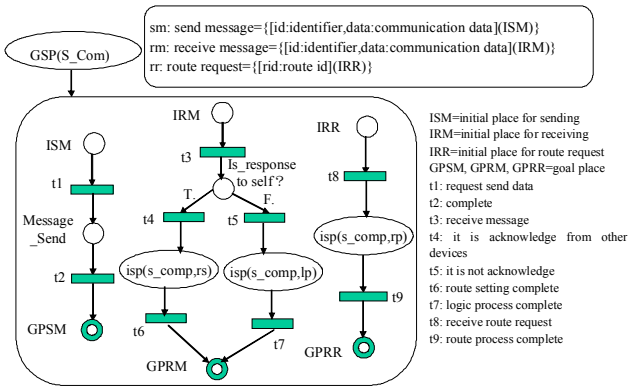


Figure 5. G-net representing communication module of signal component

Fig 5 is the G-net model of communication module of signal component. The G-net name is "S_Com". There are 3 methods are defined: rm (receive message), sm (send message) and rr (route request). The communication modules of point and track unit component have not the route request method. Parameters in method sm and rm are id and data. Id represents network id of destination devices; and data represents communication data, which maybe comprises logic status information. Parameter of method rr is rid, which indicates the id of requested route.

Message_Send and If_is_ack are primitives. They can be executed without calling other methods. The return value of primitive If_is_ack is used to judge whether the received message is acknowledgment from other devices, i.e. whether it is an acknowledgment message. If it is, then call rs (route setting) method of module S_Comp, which changes signal to permission status when all acknowledgments from route-related devices are received. Otherwise, the received message must be an order that indicates what status this signal component should be. In this case, method lp (logic process) of module S_Comp is called.

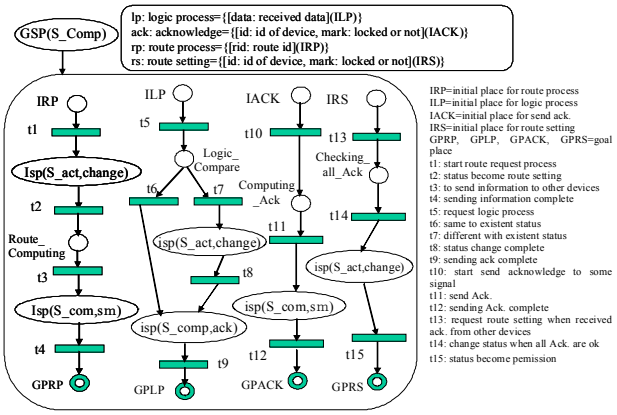


Figure 6. G-net representing computing module of signal component

Fig. 6 is the G-net model of computing module. There are 4 methods in module S_Comp. When signal receives route request via method rr of module S_Som, method rp changes its status to route setting, and computes the status information of other route-related devices, then sends the result to these devices by isp(S_com,sm), which call sm method of module S_Com. Method lp executes logic processing. There 2 circumstances are considered when a message is received. One is that the logic status coming from network is the same to current logic status, in this case, transition t6 is fired; the other one is different. In this case, it is necessary to change logic status and lock it, then devices acknowledge messages to the signal that starts route setting by invoking method ack of module S_Comp. Method rs starts when acknowledgment is received from other devices. When all responses are received and are ok (primitive Checking_all_Ack checks the acknowledgment message), the signal component change itself

```
Class DeviceSignal
{private:
        Int SignalId;
        Char SignalName[10];
        Int SignalType;    //home:0; start: 1;
        Int route;
        Class relatedevice RelateDevice[ ];
Public:
        DeviceSignal();                 //Initialize
        //Communication module
        Int SendMessage();
        Int RecieveMessage();
        Int RouteRequest();             //ReadRouteMessage()
                                        //then call RouteComputing
        //Control module
        Int ChangeStatus();             //Call SendMessage()
        //Computing module
        Int LogicProcess();
        Int RouteComputing();
        Int Routesetting();
        Int Ack();
        //Monitor module
        Int poll();
}
```

Figure 7. Sample class structure of signal component in C++

to permission (display green).The computing modules of point and track unit components are different from that of signal components. They have not method rp (route process) and rs (route setting).

We also established the G-nets models of control module and monitor module of signal component as well as these modules of other interlocking device components including point components and track unit components [8].

Fig 7 shows a sample class structure of signal components by C++. There are some attributes and methods are defined. The methods are divided into 4 modules shown in Fig 3. All signal components have the same attributes and methods as Fig. 7. When the device components initialize, the class will be instantiated to device component objects.

IV. SAFETY TECHNOLOGIES OF CBDRIS

A. *Safety technologies in Hardware*

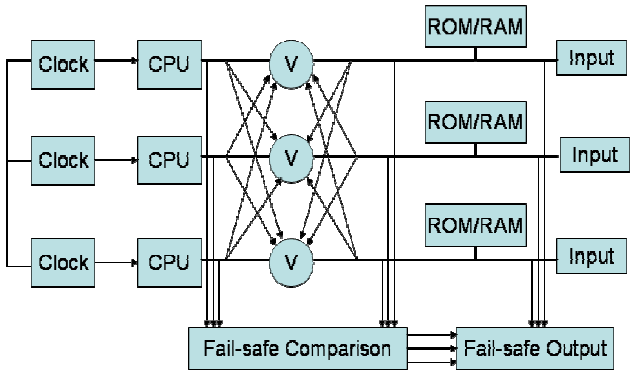


Figure 8. Fault-tolerance configuration of hardware in CBDRIS

Redundancy management is believed a useful measure to fault-tolerance of safety-critical systems. Triple Modular Redundancy (TMR) has been used in computerized interlocking system SMILE in order to satisfy fail-safe [1]. In CBDRIS, hardware fault-tolerance also adopts TMR. Fig. 8 shows the fault-tolerance configuration of hardware in CBDRIS.

The "two-out-of-three" voting circuit is used in CBDRIS. All critical hardware units have three separate copies. Even one fails, the other two still can make comparison and out put the correct result.

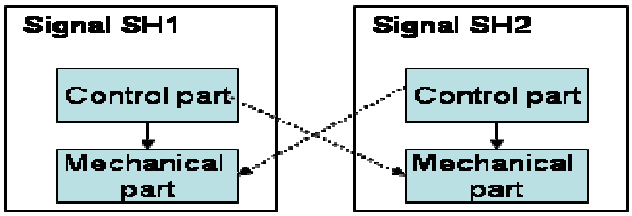


Figure 9. Device component level redundancy

Further, device component level redundancies also are adopted in CBDRIS. A device component can be divided into control part and mechanical part. Two of same kind device components which are near geographically can be connected together to realize fault-tolerance. For instance, signal component SH1 and SH2 in Fig.1 can be connected as shown in Fig. 9. If control part of SH1 fails, the control part of SH2 can control the mechanical part of the failed device component.

Of course, the two device components need to communicate and backup operation status of the other one. This method is based on the assumption that control part is apt to fail than mechanical part.

B. *Safety technologies in software*

In software, control flow standardization, software validation, development process standardization as well as fault-tolerance are main concerns. With standardization of software control flow, the reliability of CBDRIS can be improved significantly. Software validation deals with the correctness of software. International standardization IEC 62278 [9], IEC 62279 [10] describe guideline of safety management technologies for railway applications.

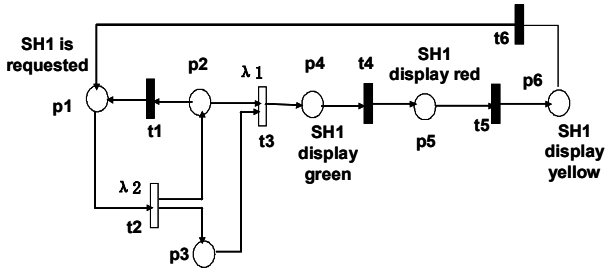


Figure 10. A DSPN model with fault-tolerance

Besides, because the time is a very important factor in real-time distributed control systems, we add a time restraint to communication or action of device components. This can be realized by using DSPNs (Deterministic Stochastic Petri-Nets) in which two kinds of transitions: deterministic transitions and timed transitions [11]. A deterministic transition fires in a deterministic time once it is enabled. A timed transition fires after a delay time which is assumed to be exponentially distributed. The max waiting time can be set with a deterministic transition. If an interlocking device component does not receive response messages in a deterministic time, it can be believed the data package lost and some measures have to be carried out.

Fig. 10 illustrates the DSPN model for signal component SH1 requesting route in Fig.1. Deterministic transitions are expressed with black box, and timed transitions are expressed with white box.

Place p1 means that signal component SH1 is requested. Then SH1 send messages to other route-related device components (here point P1, P2, track unit TX1 and T1) by timed transition t2. After signal SH1 receive response messages from these device components (timed transition t3 fires), signal SH1 displays green to permit train enter the platform. SH1 displays red which means there is a train in the platform after train enters the platform. SH1 displays yellow which warns following trains to reduce their speed when the former train leaves the platform and enter next track unit. Finally transition t6 fires when the train leaves the next track unit.

Here, deterministic transition t1 is designed to provide fault-tolerance by adding a time limit. In case that some device

component fails or communication fails, signal SH1 can not receive response messages. The failure must be detected and dealt with immediately. Timed transition t2 and t3 are exponentially distributed with mean waiting time 1/ λ 2 and 1/ λ 1 respectively.

With the DSPN model, system analysis of structural property and dynamic property is possible [12], including liveness, fairness, deadlock etc. Further, steady state solutions of the DSPN model can be computed with a time and space efficient algorithm [13]. Based on the solution, system performance evaluation is possible, such as mean delay associated with transitions, mean time for tokens staying in some places.

V. CONCLUSION

A development framework for component-based decentralized railway interlocking system is presented. The framework divides the development of CBDRIS into dynamic process and static process. G-nets are used to model and verify control flow of device components. In order to ensure the system reliability and safety, several measures are adopted in CBDRIS. Hardware redundancies both in CPU level and device level are the main concerns. In software, considering that the time is important in such a decentralized real-time system, an effective time constraint is used to detect device or communication failures and a DSPN model is presented. The presented approach standardizes the attributes and methods of component class, using Petri-net-based modeling method (G-nets) to design and evaluate CBDRIS, shortening the development process and curtailing the numbers of error brought by human.

Now, a simulation tool for a simple station is being implemented with C++. The tool adopts standardized methods and attributes to realize the interlocking component. The tool can be considered as a prototype to standardizing interlocking component.

In future research, converting the interlocking graph and interlocking table to a unified data format suiting to each kind of device components will be conducted. The conversion equivalence of interlocking data should be verified with formal methods.

REFERENCES

[1] K. Akita, T. Watanabe., H. Nakamura., I. Okumura.: "Computerized Interlocking System for Railway Signaling Control; SMILE". IEEE Trans., May 1985: Ind., 1A-21.

[2] X. Hei., S. Takahashi., H. Nakamura., M. Fukuda, K. Iwata & K. Sato.: "Improving Reliability of Railway Interlocking System with Component-based Technology". Journal of Reliability Engineering Association of Japan, Vol.28, December 2006: pp.557-568.

[3] S.M. Yacoub, B. Cukic, H.H Ammar.: "A Component-based Approach to Reliability Analysis of Distributed Systems", In Proc. Of the 18th IEEE Symposium on Reliable Distributed Systems, 1999: pp.158-167.

[4] D. Hamlet, D. Mason, D. Woit : "Theory of software reliability based on components". In the 23th International Conference on Software Engineering (ICSE'01), 2001, pp.361-370.

[5] R. E. Lyons, W. Vanderkulk.: "The Use of Triple-Modular Redundancy to Improve Computer Reliability", IBM Journal of Research and Development, Vol. 6, No. 2. April 1962: pp. 200-209.

[6] Railway Electrical Engineering Association of Japan, "Signalling: ATS・ATC", Japan: 2001: pp.2-pp.100.

[7] A. Perkusich and J. Figueiredo: "G-Nets: A Petri Net Based Approach for Logical and Timing Analysis of Complex Software Systems". The Journal of Systems and Software, Vol.39, No.1, October 1997: pp.33-59.

[8] X. Hei, H. Mochizuki, S. Takahashi. & H. Nakamura: "Modeling distributed railway interlocking system with object-oriented petri-net". In 10th International Conference on Computer System Design and Operation in the Railway and Other Transit System, Prague, Czech Republic, 2006, pp.309-318.

[9] IEC 62278, Railway applications –Specification and demonstration of reliability, availability, maintainability and safety (RAMS). 2002.

[10] IEC 62279, Railway applications – Communications, signalling and processing systems –Software for railway control and protection systems. 2002.

[11] M. Marsan and G. Chiola.: "On Petri Nets with Deterministic and Exponentially Distributed Firing Times", Lecture Notes in Computer Science, vol.266, 1987. pp. 132-145.

[12] T. Murata: "Petri Nets: Properties, Analysis and Applications", Proceedings of IEEE, Vol. 77, No. 4, 1989. pp. 541-580.

[13] G. Ciardo and C. Lindemann. Analysis of deterministic and stochastic petri nets, in Proc. of the Fifth Int. Workshop on Petri Nets and Performance Models (PNPM93), Toulouse, France, Oct. 1993.