Synchronizing Automata and Petri Net based Controllers

Mustafa Seçkin Durmuş, Uğur Yıldırım, Oytun Eriş, Mehmet Turan Söylemez

Control Engineering Department, Istanbul Technical University, Maslak, Istanbul, Turkey
durmusmu@itu.edu.tr, yildirimu@itu.edu.tr, erisoy@itu.edu.tr, soylemezm@itu.edu.tr

Abstract

In designing safe and reliable interlocking systems for railways which are compatible with the related CENELEC (*European Committee for Electrotechnical Standardization*) standards semi-formal methods and diverse programming techniques are highly recommended (HR). EN 50128 (where methodologies to build failsafe software for railway applications are defined) recommends the use of Automata and Petri Nets (PNs) as semi-formal methods to build failsafe interlocking software for railway applications. In this paper interlocking software design which is achieved by using automata and PNs is explained where they were used synchronously as a voting system.

1. Introduction

Along with hardware development, software development process of an interlocking system is one of the toughest and most crucial parts in building a railway signalization system. For this purpose, EN 50126/8/9 standards are developed by CENELEC specifically for railway applications. EN 50126, where mainly RAMS (Reliability, Availability, Maintenance and Safety) analysis is described, is the general standard for all kinds of railway applications. EN 50128 defines methodologies to build failsafe software for railway applications, whereas EN 50129 determines requirements for the hardware of electric, electronic and programmable devices that are to be used in railway applications. On the software part, as it is mentioned above, designers have to fully concern with EN 50128 standards where the steps of safety critical software development process are defined [1-4].

These standards bring out a very important concept, that is; the probability for the system to execute the safety functions required in all specified input conditions within a specified time interval. Depending on this probability the Safety Integrity Level (SIL) [5] of a system can be determined.

SIL level has to be at least at level 3 (SIL3) for railway interlocking systems [1]. At SIL3 level, the range of failure per hour (which is symbolized with λ) must be between 10^{-8} and 10^{-7} ($10^{-8} \leq \lambda < 10^{-7}$). This also means that the interlocking system has to work in average 1000 years without falling into a dangerous failure state. The easiest way to provide SIL3 level on hardware is to use COTS (Commercial off the shelf) products which are already certified [6-8].

Automata [9] and Petri Nets [10] are the most popular modeling tools for Discrete Event Systems (DES). Railway systems are also considered as DES because of the similarity in their behaviors and features [11]. Even PNs have some advantages [12] in comparison to Automata theory both in graphical and mathematical representation; Automata theory still preserves its own popularity. Since both methods decrease the possible logical errors, they are very useful as semi-formal methods on the modeling part of a DES. Several applications including both methods can be found in [13-17].

Both IEC 61508 and EN 50128 standards regarded Automata and Petri Nets as semi-formal modeling tools. Sometimes, programs designed using Automata and Petri Nets have to work together in a synchronized manner in the control of a master controller unit. Recommendation of EN 50128 about semi-formal methods and modeling methods can be seen in figure 1.

Table A.4 – Software Design and Implementation (clause 10)

TECHNIQUE/MEASURE		Ref	SWS IL0	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1.	Formal Methods including for example CCS, CSP, HOL, LOTOS, OBJ, Temporal Logic, VDM, Z and B	B.30	-	R	R	HR	HR
2.	Semi-Formal Methods	D.7	R	HR	HR	HR	HR
3.	Structured. Methodology including for example JSD, MASCOT, SADT, SDL, SSADM and Yourdon.	B.60	R	HR	HR	HR	HR

Fig. 1. Part of EN50128 Software Design and Implementation table.

This can be useful especially in providing a *Diverse Programming* design as mentioned in EN 50128 [4].

In this study, a railway station controlled by a decision unit and a voting system is explained. The voting system consists of two fail-safe PLCs synchronized with each other. The programs running on these PLCs are designed using Automata and PN methods. Decision unit is another fail-safe PLC where the votes of PLCs are compared and all the communications between the railway yard, Traffic Control Center and the voting system are handled.

2. Software Development Process

A software development model known as V-model is recommended in EN 50128 which is given in figure 2.

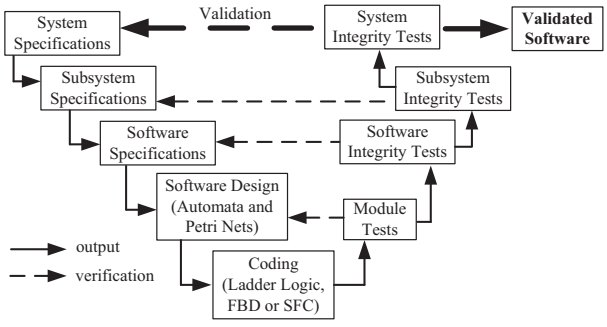

Fig. 2. V-model

In this paper, especially Software Design part is explained. The other specifications are derived from the interlocking table, railway system requirements and the needs of the customer (Turkish State Railways). Interlocking table consists of possible route reservations and related requirements and it is considered as the starting point of modeling of a signalization system for a railway yard [18]. An example railway yard and its interlocking table are given in figure 3 and table 1, respectively.

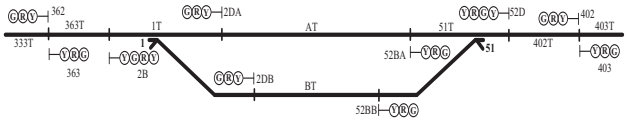

Fig. 3. An Example railway yard

Table 1. Interlocking table

Route Selection	Controlled Signal		Signal Lights	Switch 1 Position
363T – AT	2B	Green	52BA (**Y**) or (**G**)	Normal
		Yellow	52BA (**R**)	
363T – BT		Yellow-Green	52BB (**Y**) or (**G**)	Opposite
		Yellow-Yellow	52BB (**R**)	

Railway components need to be modeled separately in order to simplify error tracking and visuality. Railway yard given in figure 3 consist of Signal Lights (SLs), Switches and Track Circuits (TCs). The definition of railway components can also be found in [19].

A recommendation of EN 50128 for safe software architecture is Diverse Programming technique (can be seen in figure 4) which means that a specification of a given program have to be implemented N times in different ways. These N versions can run on separate controllers (in our study N is equal to two and these two versions is running on two separate fail-safe PLCs). The standard recommends that different versions to be developed by different groups in order to guarantee diversity in programming and reduce the probability of common cause failures. Different voting strategies can be used in combining the decisions of N different algorithms depending on the application requirements.

Table A.3 – Software Architecture (clause 9)

TECHNIQUE/MEASURE	Ref	SWS IL0	SWS IL1	SWS IL2	SWS IL3	SWS IL4
7. Diverse Programming	B.17	-	R	R	HR	HR

Requirements

1. Approved combinations of techniques for Software Safety Integrity Levels 3 and 4 shall be as follows:
   a) 1. 7 and one from 4, 5 or 12

Fig. 4. Part of EN 50128 recommendations for safe software architecture

For example, if a switch malfunction occurred after the entrance of a train on a route then all signal lights have to show red (safe state). Similarly, if one of the voters produces different output than the others, the whole system has to go into a safe (predetermined) state. To achieve this, two separate workgroups in Istanbul Technical University (ITU) worked on the same problem using different modeling (Automata and Petri Nets) and design techniques.

2.1. Petri Nets as a Modeling Tool

Petri Nets (PNs) [10] have some advantages over Automata both in their graphical and mathematical features [12]. PNs are defined in the literature by (1).

$$PN = (P, T, Pre, Post, M_0) \quad (1)$$

where

- **P** : $\{P_1, P_2, \ldots, P_n\}$, finite set of places.
- **T** : $\{t_1, t_2, \ldots, t_n\}$, finite set of transitions.
- **Pre** : $(PxT) \rightarrow N$, directed ordinary arcs from places to transitions.
- **Post** : $(TxP) \rightarrow N$, directed ordinary arcs from transitions to places.
- $\mathbf{M_0}$: $P \rightarrow N$, initial marking (N is a set of nonnegative numbers).

A sample switch model is given in figure 5, and related places and transitions are given on table 2.

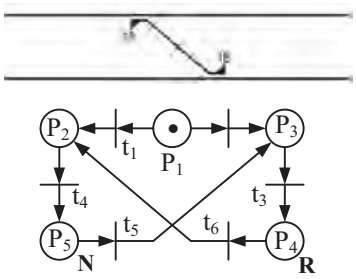

Fig. 5. Switch and its PN model

Table 2. Definitions of PN places and transitions

Event	Description
P_1	Switch starting position
P_2	Switch moves to normal position
P_3	Switch moves to reverse position
P_4	Switch is on reverse position
P_5	Switch is on normal position
t_1 (t_2)	Normal (Reverse) position request for switch
t_4 (t_3)	Switch arrives on normal (reverse) position
t_5 (t_6)	Switch position request from normal (reverse) position to reverse (normal) position

At the beginning switch is assumed to be on normal position (P_1). After an incoming position command, switch moves to that position (normal or reverse) and stays there until a new command is received.

Converting this model to a useful PLC code is simpler by using Sequential Function Charts (SFCs) which is one of the five languages defined by IEC 61131-3 standard. Besides, several formal conversion techniques are also available in the literature to convert PNs to PLC codes [20], [21]. The reader is referred to [19], [22] and [23] for more information on railway component PN models.

2.2. Automatons as a Modeling Tool

An automaton, denoted by G, is a six tuple [11]:

$$G = (Q, \Sigma, f, \Gamma, q_0, Q_m) \quad (2)$$

where

- **Q** : is the set of states,
- **Σ** : is the finite set of events
- **f** : QxΣ→ Q is the partial transition function on its domain.
- **Γ** : Q→2Σ is the active event function.
- **Γ(q)** : is the set defined for every state of G and represents the feasible events of q.
- $\mathbf{q_0}$: is the initial state.
- $\mathbf{Q_m}$: is the set of marked states representing a completion of a given task or operation.

In order to apply the method based on the Automaton model, first the events have to be identified and a state transition graph have to be obtained. Automata model of the switch given in figure 5 can be seen in figure 6 and related definitions of the model are given in Table 3.

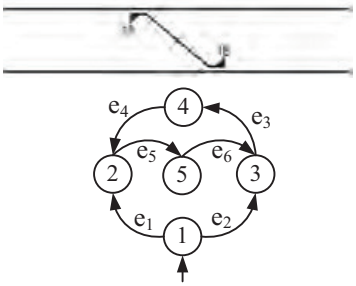

Fig. 6. Automaton (State transition graph)

Table 3. Events defined for automation

Event	Description
e_1	Normal position request
e_2	Reverse position request
e_3 (e_5)	Switch is on reverse (normal) position
e_4 (e_6)	Switch position request from reverse (normal)position to normal (reverse) position
State 1	Switch starting state
State 2	Switch moves to normal position
State 3	Switch moves to reverse position
State 4	Switch is on reverse position
State 5	Switch is on normal position

Converting of Automata models to PLC codes is similar to PNs [17], [24].

3. Synchronization of Automata and Petri Nets

As it is mentioned in section 2, two PLC programs are obtained using Automata and PN modeling techniques. These two interlocking PLCs (I-PLCs) are connected to the railway yard through a voting system and so they have to be synchronized with each other. This synchronization is achieved by another fail-safe PLC called as Communication and Decision Making Unit (CDMU). CDMU receives requests from dispatcher, who is an officer in the Traffic Control Center (TCC), where all railway traffic is monitored and logged. CDMU also receives signals (indications from sensors of the railway components) and sends commands to railway yard.

The architecture of the interlocking system is given in figure 7. When a route request is made by the dispatcher, CDMU sends this request to both I-PLCs where the comparison of this request with the current situation is achieved. After the comparison the request of the dispatcher is accepted or rejected depending on the situation of the railway yard. Situation of the railway yard is updated in every second by sending the sensor information of the railway components to the I-PLCs. After an incoming request from CDMU to I-PLCs, they send their decisions (votes) back to CDMU about that request. If it is accepted the necessary commands sends to railway yard and TCC.

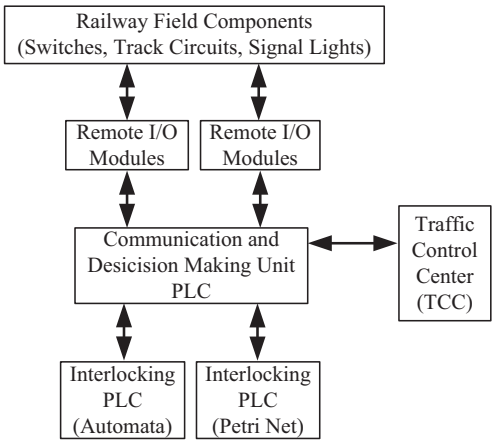

Fig. 7. The architecture of interlocking system

Whenever an I-PLC sends a signal to CDMU after an incoming request, I-PLC waits an answer from CDMU about its signal. If the answer does not come back in 10 sec. (10 sec. is also called as *synchronization time* of I-PLCs) then I-PLC rejects the request and so CDMU.

CDMU have to wait each PLC for its decision because the response times (depending on the cycle time of each I-PLC) of I-PLCs are not same. When the decision of an I-PLC arrives to CDMU, it waits for 2 sec (2 sec is also called as *adaptation time* of I-PLCs) for the other I-PLC to answer. If the other I-PLC does not answer in 2sec. CDMU doesn't answer back to both I-PLCs and the request will be rejected.

In briefly, these timings are related with the safe states of related signals. For example, red signal output is safe state for a signal light where other signal outputs (green or yellow) considered as unsafe state. Similarly for switches, moving a switch from one position to another is an unsafe state where keeping its current position is considered as safe state.

Sometimes the decisions of I-PLCs may not be same. For example, when a position changing request for a switch came from the TCC, assume that one I-PLC accepts but the other does not. In this situation, CDMU records this as an error (and informs dispatcher about this situation) and does not accept the request since changing position of a switch by the dispatcher is considered as an unsafe signal. After observing the fact that final switch movement signal has not been sent from CDMU after the adaptation time the I-PLC that decided to accept switch movement request changes its answer as reject.

Similarly, above expression can be expand to signal lights, route requests and other possible incompatible situations. In other words, CDMU also synchronizes I-PLCs in case of incompatible votes.

CDMU also contains some Safety Instrumented Functions (SIF), for example, if both I-PLCs accept a request which normally should not accepted (e.g. movement of a switch on a track occupied by a train), CDMU does not send this signal to

railway yard and records this as an error and give information about this error to TCC.

In addition to these, by the help of fail-safe feature of CDMU, communication failures can also be identified. If communication of CDMU with the remote I/O modules or with any of the I-PLCs is interrupt, this failure is also recorded by CDMU and TCC is informed about this situation. One more feature is that, if communication with an I-PLC is interrupted for a long time (more than 2 sec.), this is considered is as a fatal error and all fail-safe PLCs get into safe state (predetermined conditions).

To sum up briefly, CDMU;

- communicates I-PLCs, TCC and Railway yard with each other,
- sends commands to Railway yard,
- updates the sensor information of I-PLCs,
- receives request from TCC and sends back the answer of each request,
- records decision errors of I-PLCs,
- contains SIF (enforce if necessary),

synchronizes I-PLCs when the decisions are not same.

4. Results

In this study, a railway interlocking system which is working as a voting system is explained where railway related functional safety requirements of CENELEC standards, especially EN 61508 and EN 50128, are also considered. Voting system consists of two fail-safe PLCs with the running codes determined by using Automata and Petri Net methods regarded semi-formal methods by EN 50128. Communication between Traffic Control Center, Railway Yard and Interlocking PLCs are achieved by a master controller called as Communication and Decision Making Unit which is also a fail-safe PLC where all decisions are compared, considered and executed. Obtained interlocking software is tested and verified by a simulator called Interlocking Test Program (ITP) developed by ITU. Interface of ITP can be seen in figure 8 and the real interlocking system can be seen on figure 9.

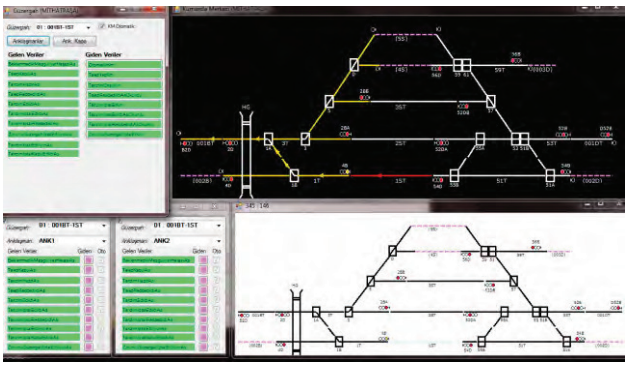

Fig. 8. Interlocking Test Program. CDMU window (left), TCC window (right bottom) and railway field (right up)

Fig. 9. The architecture of interlocking system

Acknowledgment

This work is supported by The Scientific and Technological Research Council of Turkey (TÜBİTAK) project number 108G186 – The National Railway Signalization Project.

5. References

[1] M. T. Söylemez, "Functional Safety Applications on Railway Systems: Turkish National Railway Signalization Project," in *2nd International Industrial Safety Systems Conference*, Istanbul, Turkey, 2010 (in Turkish).

[2] M. T. Söylemez, M. S. Durmuş, U. Yıldırım, S. Türk and A. Sonat, "The Application of Automation Theory to Railway Signalization Systems: The Case of Turkish National Railway Signalization Project," (*accepted for IFAC World Congress 2011*).

[3] IEC 61508-3, Functional Safety of Electrical/Electronic/Programmabel electronic safety-related systems, Part 3: Software requirements, 1997.

[4] EN 50128, Railway Applications, Communications, signalling and processing systems, Software for railway control and protection systems, 2001.

[5] M. Spellemaeker, L. Witrant, "How to Determine the Safety Integrity Level (SIL) of a Safety System," Available: www.oldhamgas.com (12.05.2011).

[6] Available: http://www.hima.com/default.php (14.06.2011)

[7] Available: http://www.sea.siemens.com/us/Products/Automation/Programmable-Controllers/Pages/Programmable-Controllers.aspx (14.06.2011)

[8] Available: http://www.mitsubishi-automation.com/ (14.06.2011)

[9] P. J. Ramadge and W. M. Wonham, "The Control of Discrete Event Systems," *Proc. of IEEE*, vol. *77*, no. *1*, pp. *81-98*, 1989.

[10] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proc. of IEEE*, vol. *77*, no. *4*, pp. *541-580*, 1989.

[11] C. G. Cassandras and S. Lafortune, "*Introduction to Discrete Event Systems*", Kluwer Academic Publishers, 1999.

[12] A. Giua, "Petri Net Techniques for Supervisory Control of Discrete Event Systems," in *1st Int. Workshop on Manufacturing and Petri Nets*, pp. *1-30*, Osaka, Japan, 1996.

[13] A. Giua and C. Seatzu, "Modeling and Supervisory Control of Railway Networks Using Petri Nets," *IEEE Trans. On*

Automation Science and Engineering, vol. *5*, no. 3, pp. *431-445*, July 2008.

[14] A. M. Hagalisletto, J. Bjork, I. C. Yu and P. Enger, "Constructing and Refining Large-Scale Railway Models Represented by Petri Nets," *IEEE Trans. On System, Man and Cybernetics-Part C: Applications and Reviews*, vol. *37*, no. *4*, pp. *444-460*, July, 2007.

[15] R. Zurawski and M. C. Zhou, "Petri Nets and Industrial Applications: A Tutorial," IEEE Trans. on Industrial Electronics, vol. 41, no. 6, pp. 567-583, 1994.

[16] A. D. Febbraro, G. Porta and N. Sacco, "A Petri Net modelling approach of intermodal terminals based on Metrocargo© system", in *Proc. of the IEEE Intelligent Transportation Systems Conf*., pp. *1442-1447*, 2006

[17] İ. T. Hasdemir, S. Kurtulan, L. Gören, "An implementation methodology for supervisory control theory", *International Journal of Advanced Manufacturing Technology*, vol. *36*, no. *3-4*, pp. *373-385*, 2008.

[18] M. S. Durmuş, K. Akın, M. T. Söylemez, "Supervisory Control Approach by Inhibitor Arcs for Signalization and Interlocking Design of a Railway Yard,", International Symposium on INnovations in Intelligent SysTems and Applications, INISTA'10, Kayseri & Cappadocia, TURKEY, 21 - 24 June, 2010.

[19] M. S. Durmuş, U. Yıldırım, A. Kursun, M. T. Söylemez, "Fail-Safe Signalization Design for a Railway Yard: A Level Crossing Case", WODES'10, International Workshop on Discrete Event Systems, 30 August - 01 September, Berlin, Germany, 2010.

[20] M. Uzam, "Petri-net-based Supervisory Control of Discrete Event Systems and Their Ladder Logic Diagram Implementations," PhD. Thesis, University of Salford, SALFORD, M5 4WT, UK, 1998.

[21] . Thapa, S. Dangol and G.-N. Wang, "Transformation from Petri Nets Model to Programmable Logic Controller using One-to-One Mapping Technique," Proc. of the 2005 Int. Conf. on Computational Intelligence for Modelling, Control and Automation and Int. Conf. on Intelligent Agents, Web Technologies and Internet Commerce, vol. 2, pp. 228-233, 2005.

[22] M. S. Durmuş, U. Yıldırım, M. T. Söylemez, "Signalization and Interlocking Design for a Railway Yard: A Supervisory Control Approach by Enabling Arcs", The in *7th International Symposium on Intelligent and Manufacturing Systems, IMS 2010*, 15-17 September, Sarajevo, Bosnia Herzegovina, 2010.

[23] U. Yıldırım, M. S. Durmuş, M. T. Söylemez, "Fail-Safe Signalization and Interlocking Design for a Railway Yard: An Automation Petri Net Approach", in *7th International Symposium on Intelligent and Manufacturing Systems, IMS 2010*, 15-17 September, Sarajevo, Bosnia Herzegovina, 2010.

[24] O. Eriş and İ. Mutlu, "Design of Signal Control Structures Using Formal Methods for Railway Interlocking Systems", in *11th International Conference on Control, Automation, Robotics and Vision*, 7-10 December, Singapore, 2010.