Application of Functional Safety on Railways Part II: Software Development

Uğur Yıldırım
Istanbul Technical University
Istanbul, Turkey
yildirimu@itu.edu.tr

Mustafa Seçkin Durmuş
Istanbul Technical University
Istanbul, Turkey
durmusmu@itu.edu.tr

Mehmet Turan Söylemez
Istanbul Technical University
Istanbul, Turkey
soylemezm@itu.edu.tr

***Abstract*—*As well as modelling & designing part, software development process is another major process in development of railway signalization systems. Early mechanical railway interlocking systems do not need such software development processes where all train traffic is controlled directly from signalboxes via mechanical tools and signalmen. Along with rising train speeds and densities the need of reliable and safe signalization systems including safe software increased. In this study, software development process of a signalization system for a railway yard including related CENELEC (European Committee for Electrotechnical Standardization) software standards is explained.*

Keywords; Functional Safety, V-model, Railway Interlocking and Signalization Software Development.

I. INTRODUCTION

After the establishment of first mechanical interlocking system in 1843 in England [1] where signalization was operated by signalboxes via mechanical equipments, density on railways were not too much as it is today and mechanical interlocking systems were adequate to operate railway traffic. Instead of mechanical interlocking, electromechanical, relay-based, electronic interlocking and computerized interlocking systems also used with respect to the development of railway systems. Nowadays, programmable logic controllers (PLCs) and field programmable gate arrays (FPGAs) are on the scene.

SMILE [2] was an early computerized interlocking application where fail-safe microcomputer systems were used, microprocessors are also used in interlocking systems including accident probabilities [3]. In addition to these, STERNOL developed by ABB Signal was used to control a simple railway yard [4]. Besides formal verification of railway interlocking systems [5] and verification based on annotated logic [6], automatic verification of railway interlocking systems [7] and graph based railway interlocking applications [8], [9] are also studied. Studies on modelling of railway interlocking systems with statemate and state charts [10], [11] and [12], distributed interlocking systems with Petri Nets [13] and decentralized railway signalling [14] can also be found in the literature.

In addition to modelling, to achieve safety and reliability issues, designers have to consider the requirements of standards developed by *CENELEC* (European Committee for Electrotechnical Standardization) for railway applications. These are *EN 50126*, *EN 50128* and *EN 50129*. *EN 61508* is developed as an umbrella standard for functional safety requirements of all kinds of electronic and programmable devices. These standards also results with a very important definition known as Safety Integrity Level (SIL). SIL is the probability for the system to execute the safety functions required in all specified input conditions within a specified time interval [15].

For interlocking systems, safety integrity level has to be at SIL 3 level [16] which means that the range of failure per hour (is symbolized with λ) has to be $10^{-8} \leq \lambda < 10^{-7}$. In other words, the interlocking system has to work minimum 1000 years without falling into failure state. Both on hardware and software part, fail-safe solutions are also provided by different companies [17-19]. The easiest way of providing SIL3 level on hardware is to get COTS (Commercial off the shelf) products which are certified.

In this study, software development process for railway signalization systems are described by considering the related standards. In section 2, related standards and their requirements are described and code snippets of a railway switch and level crossing are also given as an example in section 3.

II. SOFTWARE DEVELOPMENT PROCESS

Related EN 61508-3 and EN 50128 standards recommends to use V-model (given in figure 1) in software development process.

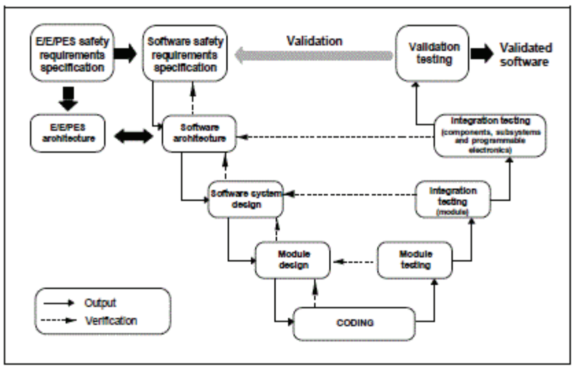
E/E/PES safety requirements specification
Software safety requirements specification
Validation
Validation testing
Validated software
E/E/PES architecture
Software architecture
Integration testing (components, subsystems and programmable electronics)
Software system design
Integration testing (module)
Module design
Module testing
CODING
Output
Verification

Figure 1. EN 61508-3 Software safety integrity and the development lifecycle (The V-model)

In this model, the whole system has to be divided into small pieces (module design) before coding. Some examples about the modules (models of railway components) can be found in [20]. The codes of these models will be given on next sections.

In addition to V-model, in order to develop safe software architecture different techniques are also suggested in EN 61508-3 and EN 50128. Among these techniques, Defensive Programming, Failure Assertion Programming and Diverse Programming techniques are preferred.

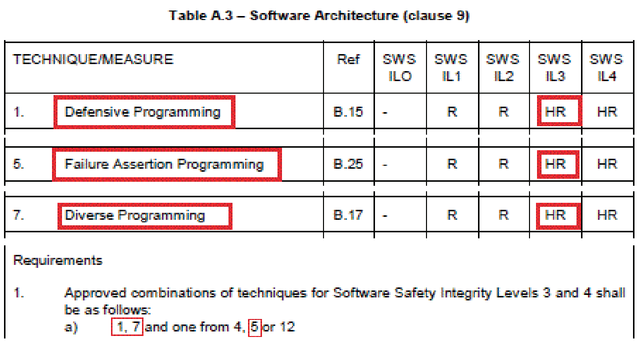

Table A.3 – Software Architecture (clause 9)

TECHNIQUE/MEASURE		Ref	SWS IL0	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1.	Defensive Programming	B.15	-	R	R	HR	HR
5.	Failure Assertion Programming	B.25	-	R	R	HR	HR
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

Figure 2. EN 50128 Recommendations for safe software architecture (HR – Highly Recommended, R – Recommended, NR – Not Recommended)

The aim of Defensive Programming is to produce programs which detect anomalous control flow, data flow or data values during their execution and react to these in a predetermined and acceptable manner. Many techniques can be used during programming to check for control or data anomalies. These can apply systematically throughout the programming of a system to decrease the likelihood of erroneous data processing. Error states can be detected by this technique, for example, if sensors of a switch both indicate reverse and normal position at the same time or a signal light flashes red even the interlocking software sends green command then these events indicates failure states.

The aim of Failure Assertion Programming is to detect residual software design faults during execution of a program, in order to prevent safety critical failures of the system and to continue operation for high reliability. The program checks a pre-condition (before a sequence of statements is executed, the initial conditions are checked for validity) and a post-condition (results are checked after the execution of a sequence of statements). If either the pre-condition or the post-condition is not fulfilled, the processing stops with an error. For example, if a train is moving on a switch area, the switch has to be locked until train finishes its route. In other words, a switch has to keep its position until a route reservation ends.

Lastly, in Diverse Programming a given program specification implemented N times in different ways. The same input values are given to the N versions, and the results produced by the N versions are compared. If the result is considered to be valid, the result is transmitted to the computer outputs. The N versions can run in parallel on separate computers, alternatively all versions can be run on the same computer and the results subjected to an internal vote. Different voting strategies can be used on the N versions depending on the application requirements. If the system has a safe state, then it is feasible to demand complete agreement (all N agree) otherwise a fail-safe output value is used. 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) or if it is a voting system, if one of the voters give different answer then the whole system has to go into a safe state.

To achieve this, different workgroups can work on the same problem in different places, by using different modelling and design techniques.

III. Conversion of Petri Net Models

As it is mentioned in EN 61508-3 and EN 50128 (Software Design and Implementation Table), in order to provide SIL 3 and SIL 4 safety integrity levels, use of semi-formal methods are Highly Recommended (HR). In this study, Petri Net models are used as semi-formal method. The Petri Net models of a switch and level crossing [20] are converted to fail-safe Programmable Logic Controller (fail-safe PLC) code. Definitons of Petri Net models can also be found in [20].

SILworX$^{\copyright}$ which is developed by HIMA [17] is used as an interface for this purpose. SILworX$^{\copyright}$ allows users to have SIL3 and SIL4 programming environment. Instead of widely used techniques like Ladder Diagrams (LD) or Function Block Diagrams (FBD), SILworX$^{\copyright}$ also allows designers to write their code as Sequential Function Charts (SFC) which is directly related with Petri Nets. Related code snippets with switch models are given in figure 4-9 and level crossing code snippets are given in figure 10 and 11.

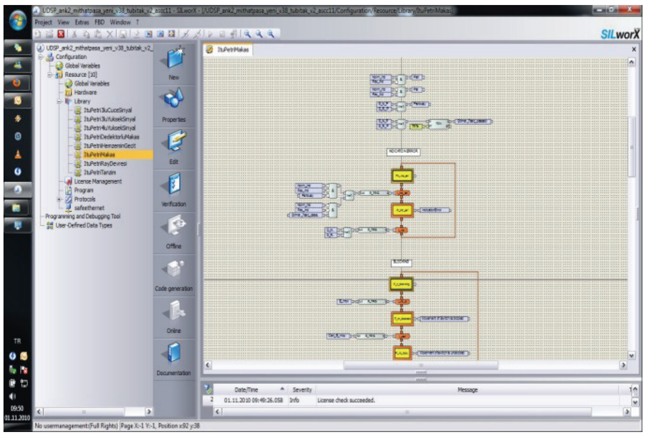

Figure 3. SILworX$^{\copyright}$ interface

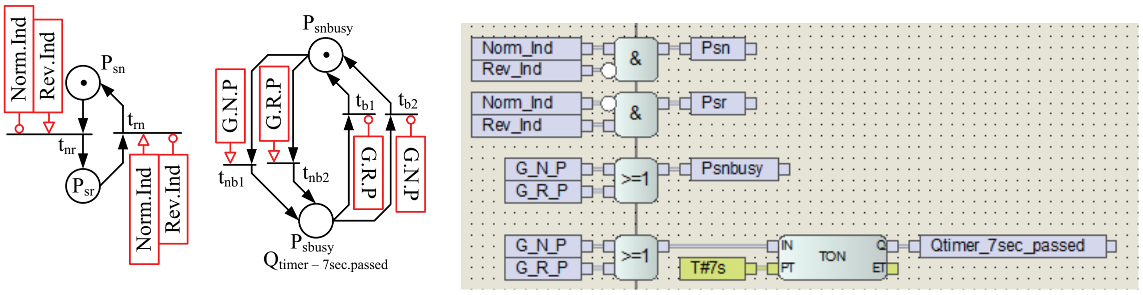

Figure 4. Switch can be on normal or on reverse position (left) and switch can be busy or not (right).

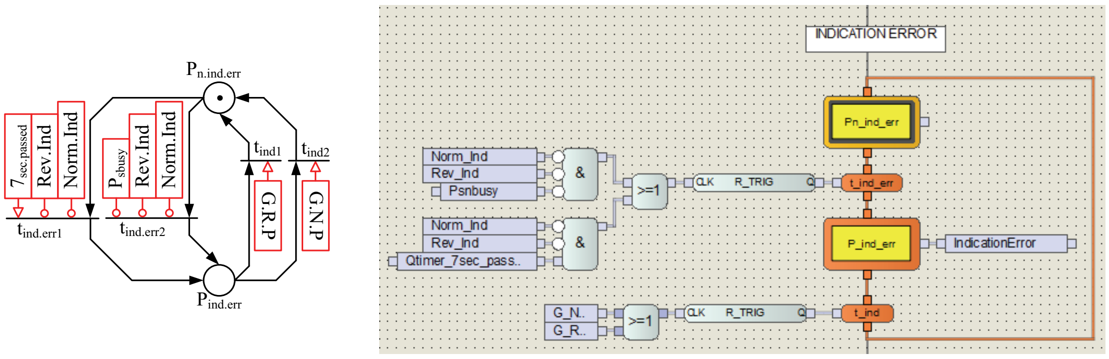

Figure 5. Indication error of a switch.

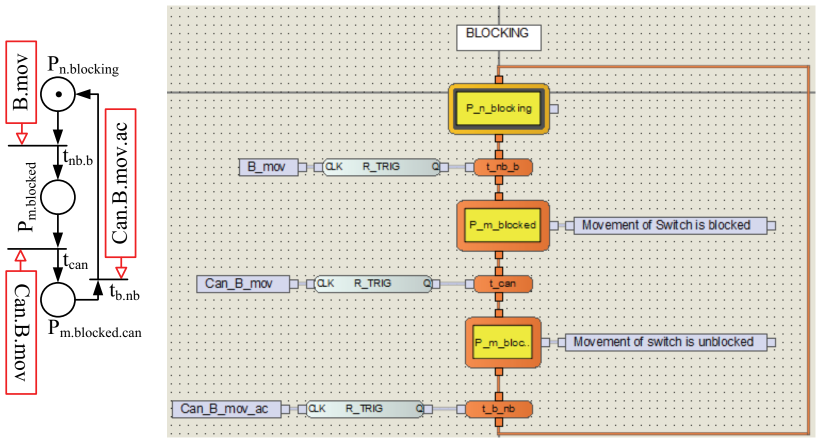

Figure 6. Switch movement can be blocked.

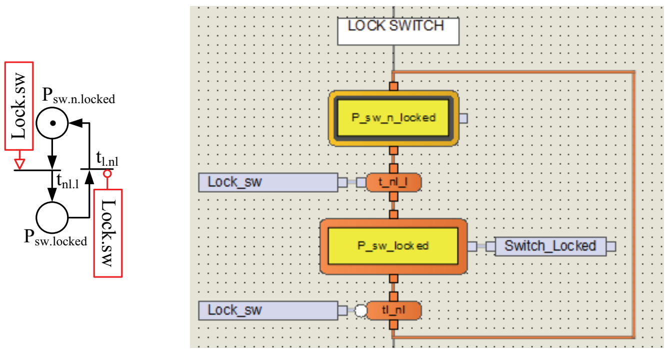

Figure 7. Switch have to be locked if it is reserved.

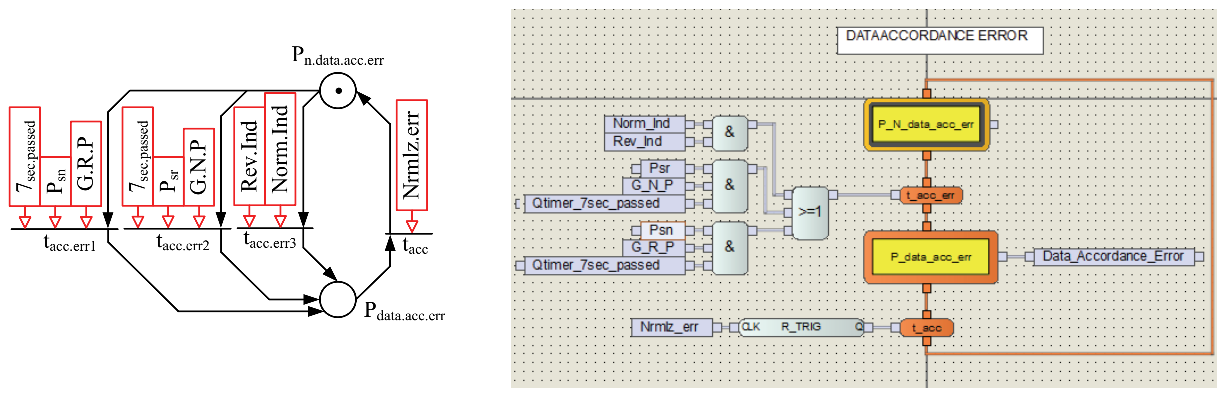

Figure 8. Data accordance error of a switch.

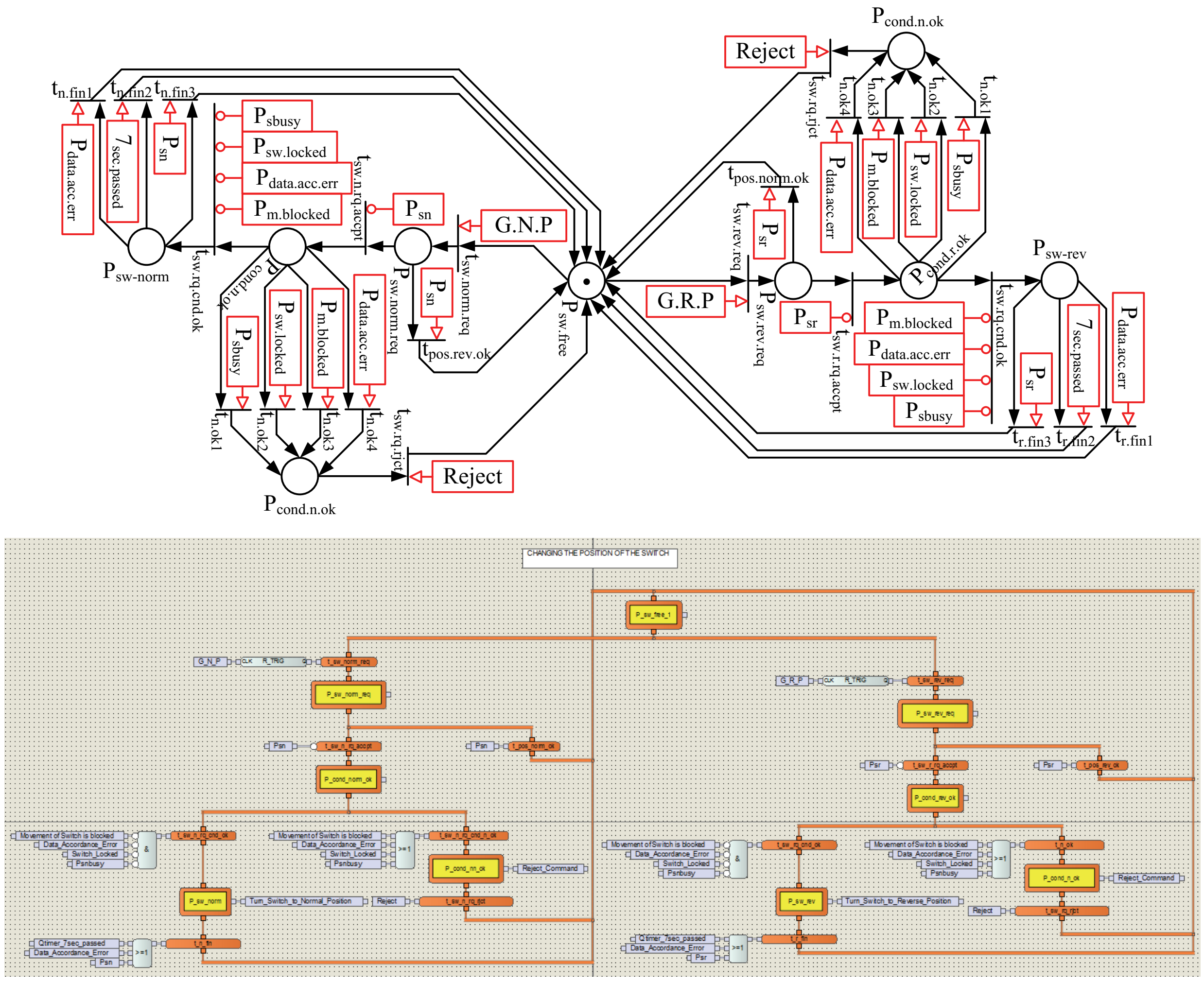

Figure 9. Movement of switch from normal to reverse and vice versa.

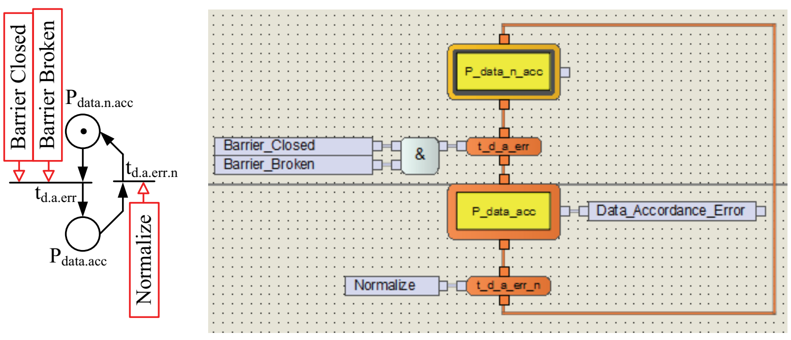

Figure 10. Data accordance error of level crossing.

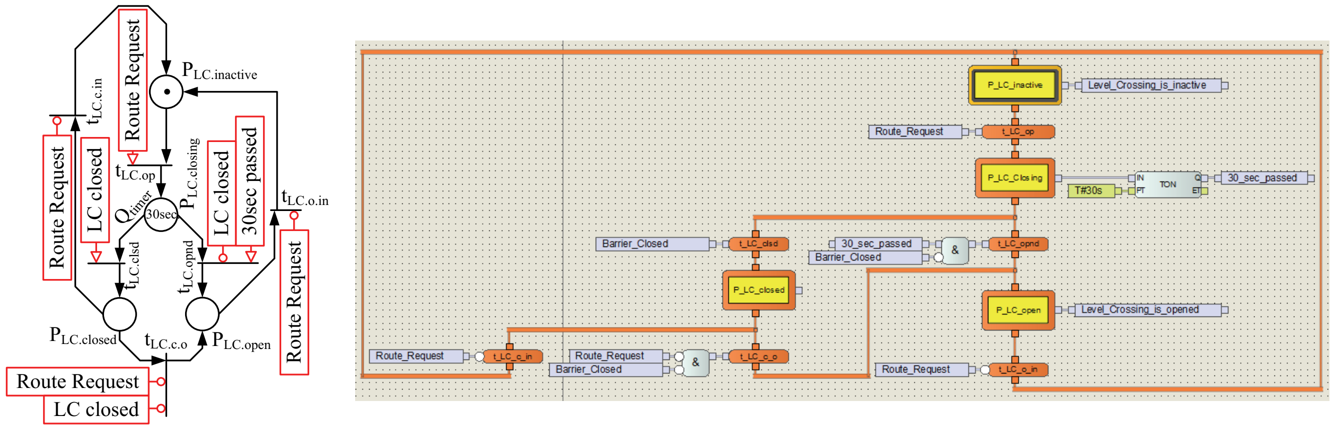

Figure 11. Opening the barrier of level crossing (LC).

In the given figures, rectangles represent the places, transitions are also represented by thin rectangles and the connection lines represent arcs. An obtained PN model can easily convert to a fail-safe PLC program by using SFC on SILworX©.

Results

In this study, software development processes (from models to codes) for railway signalization components are described. Standards related with the railway systems (EN 61508-3 and EN 50128) are considered to achieve related SIL level for railway applications.

Instead of heuristic approaches, in order to decrease the possible logical errors and provide visuality on programming, related standards *highly recommend* (HR symbols in figure 2) to model the system by dividing the whole into sub components and use semi-formal methods like Petri Nets on the modelling part. These models can then easily converted to PLC codes by using different techniques like Token Passing Logic (TPL) in the literature.

In this study, because of its easiness, easy error-tracking and safety requirements of the programming interface, Sequential Function Charts (SFC) are preferred to achieve SIL3 level. These SFC code snippets will be then used in a voting system (this program will run on a PLC which is a part of a voting system) as described in Diverse Programming after tests for various scenarios.

Acknowledgment

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

References

[1] S. Hall, “Modern Signalling Handbook,” Ian Allan Publishing, England, 2001.

[2] K. Akita, T. Watanabe, H. Nakamura, I. Okumura, “Computerized Interlocking System for Railway Signalling Control: SMILE,” IEEE Transactions on Industry Applications, vol. IA-21, no. 4, pp. 826-834, 1985.

[3] V. P. Rao and P. A. Venkatachalam, “Microprocessor-Based Railway Interlocking Control with Low Accident Probability,” IEEE Trans. on Vehicular Technology, vol. VT-35, no. 3, pp. 141-147, 1987.

[4] J. L. Petersen, “Automatic Verification of Railway Interlocking Systems: A Case Study,” Formal Methods in Software Practice, Proc. of the 2nd. workshop on Formal methods in software practice, pp. 1-6, 1998.

[5] V. Hartonas-Garmhausen, S. Campos, A. Cimatti, E. Clarke, and F. Giunchiglia, “Verification of a Safety-Critical Railway Interlocking System with Real-time Constraints,” Science of Computer Programming, vol. 36, pp. 53-64, 2000.

[6] K. Nakamatsu, Y. Kiuchi, W. Y. Chen. and S. L. Chung, “Intelligent Railway Interlocking Safety Verification Based on Annotated Logic Program and its Simulator, Proc. of the IEEE Int. Conf. on Networking, Sensing & Control, pp. 694-700, 2004.

[7] G. Dipoppa, G. D’Alessandro, R. Semprini and E. Tronci, “Integrating Automatic Verification of Safety Requirements in Railway Interlocking System Design,” Proc. of 6th IEEE Int. Symp. on High Assurance Systems Engineering, pp. 209-219, 2001.

[8] E. Roanes-Lozano, E. Roanes-Macias, and L. M. Laita, “Railway Interlocking Systems and Gröbner bases,” Mathematics and Computers in Simulation, vol. 51, pp. 473-481, 2000.

[9] X. She, Y. Sha, Q. Chen and J. Yang, “The Application of Graph Theory on Railway Yard Interlocking Control System,” Proc. of the IEEE Intelligent Vehicles Symposium, pp. 883-888, 2007.

[10] M. Banci, A. Fantechi and S. Ginesi, “The Role of Formal Methods in Developing a Distributed Railway Interlocking System,” Proc. of the 5th Symp. on Formal Methods for Automation and Safety in Railway and Automotive Systems, Braunschweig, Germany, 2004.

[11] M. Banci, A. Fantechi and S. Ginesi, “Some Experiences on Formal Specification of Railway Interlocking Systems using Statecharts,” Train International Workshop at SEFM2005, Koblenz, Germany, 2005.

[12] J. Bohn, W. Damm, J. Klose, A. Moik and H. Wittke, “Modeling and Validating Train System Applications Using Statemate and Live Sequence Charts,” Proc. of the Conf. on Integrated Design and Process Technology (IDPT2002), 2002.

[13] X. Hei, S. Takahashi and H. Nakamura, “Distributed Interlocking System and Its Safety Verification,” Proc. of the 6th World Congress on Intelligent Control and Automation, pp. 8612-8615. Dalian, China, 2006.

[14] X. Hei, S. Takahashi and H. Nakamura, “Toward Developing a Decentralized Railway Signalling System Using Petri Nets,” Proc. of the IEEE Conf. on Robotics, Automation and Mechatronics, pp. 851-855. Chengdu, China, 2008.

[15] M. Spellemaeker, L. Witrant, “How to Determine the Safety Integrity Level (SIL) of a Safety System,” www.oldhamgas.com.

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

[17] http://www.hima.com/default.php

[18] http://www.sea.siemens.com/us/Products/Automation/Programmable-Controllers/Pages/Programmable-Controllers.aspx

[19] http://www.mitsubishi-automation.com/

[20] M. S. Durmuş, U. Yıldırım, M. T. Söylemez, “Application of Functional Safety on Railways Part I: Modelling & Design,” (*accepted for the 8th Asian Control Conference 2011*).