Proceedings of the 18th World Congress
The International Federation of Automatic Control
Milano (Italy) August 28 - September 2, 2011

The Application of Automation Theory to Railway Signalization Systems: The Case of Turkish National Railway Signalization Project

Mehmet T. Söylemez*, Mustafa S. Durmuş, Uğur Yıldırım**, Serhat Türk***, Arcan Sonat*****

**Control Engineering Department, Istanbul Technical University
Istanbul,Turkey (Tel: +90-212-2853570; e-mail: soylemezm@itu.edu.tr).
**Control Engineering Department, Istanbul Technical University
Istanbul,Turkey (e-mail: {durmusmu,yildirimu}@itu.edu.tr).
***UEKAE, TUBITAK, Kocaeli, Turkey
(e-mail: {serhat.turk,arcan.sonat@bte.tubitak.gov.tr).*

Abstract: The application of control and automation theory to practical areas can provide significant means for improvement in developing countries like Turkey, where human (brain) power is relatively inexpensive. A possible application area for the well established automation theory is railway signalization, where formal methods are required to be used in order to comply with the related safety standards. Turkish National Railway Signalization Project (TNRSP) is examined in this paper with this perspective as a case study. Development stages, architecture and design of the software produced in this project are briefly discussed.

Keywords: railway signaling, interlocking design, university-industry relations in developing countries.

1. INTRODUCTION

Railways play an important role in forming the infrastructure of a country as they usually offer more economic, energy efficient and environment friendly transportation in comparison to its alternatives (mainly road and air transportation). As an example, typical carbon dioxide emission and energy consumption values for a passenger travelling between Frankfurt, Germany and Lyon, France are given in Fig. 1. The advantages of railway transportation are obvious from this figure. Moreover, the death risk per passenger kilometre is around 17 for railways whereas 140 for road traffic, while the injury risks are 41 and between 8500 and 10000, respectively, according to a European study (Transportation Council 2009).

Due to their high (initial) building costs as well as political interests, however, railway networks are not advanced in many developing countries like Turkey in comparison to developed countries. Currently, the length of rail lines in Turkey is only 10984 km. Moreover, 83% of these are without signalization and 82% are without electrification (Turkish Railways 2010). Nevertheless, this situation is expected to change as investment on railways has become a state policy. It is planned that 2622 km high speed lines that are being build will be finished by 2012, and 6792 km high speed and 4707 km conventional lines (a total of 11499 km) will be added to current network by 2023 (Transportation Council 2009). Moreover most of the current network is planned to be electrified and signalized.

Signalization costs form an important proportion of expenditure for the establishment of railways. The cost of a signalization system can change depending on its complexity. However, the average cost is around a million US dollars per kilometre of signalized line. Most of this cost is due to software development and related know how information. As a result, building their own signalization systems is very valuable in the advancement of developing countries.

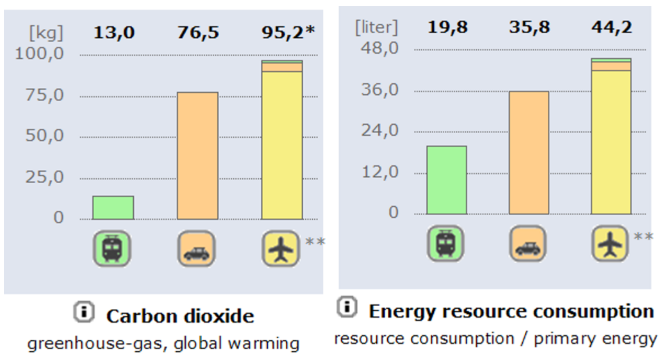

Fig. 1. Typical emission and energy consumption (converted into liter petrol) values for a person travelling between Frankfurt and Lyon. **Road and rail transportations are also assumed to be used during air transportation. Ecopassanger (2010).

Turkish National Railway Signalization Project (TNRSP) is a project funded by The Scientific and Technological Research Council of Turkey (TUBITAK) to develop a national railway signalization system for Turkish Railways (TCDD). The project, which is planned to be finished by July 2011, is carried out by Istanbul Technical University (ITU) and National Electronic Cryptology Research Institute (TUBITAK-UEKAE) with close collaboration of Turkish Railways. It is expected that Turkey's direct and indirect benefits from the project will sum up to at least one billion US dollars in the next ten years.

The interlocking, which is responsible from the safety of the system, has a very important role in building a fail-safe signalization system. Both hardware and software of interlocking must satisfy several standards to guarantee human safety. Specially, the use of formal methods in the development of interlocking software plays an important role in this perspective. Automation theory has now been well developed to provide solutions for such problems. This paper provides an overview of the use of automation theory in railway signalization systems. The development process, the architecture and design of interlocking software are considered in particular.

In the next section, components of railway signalization systems are introduced. The design of interlocking with an emphasis on interlocking software is discussed in Section 3. Conclusions and final remarks can be found in the last section.

2. COMPONENTS OF A RAILWAY SIGNALIZATION SYSTEM

2.1 Traffic Command Centre

All operations of a railway region can be managed from the Traffic Command Centre (TCC). All the field elements can be monitored and controlled from TCC using proper user interfaces by dispatchers.

2.2 Interlocking System

Interlocking provides a safe interface between the TCC and field elements. While proper commands of TCC are applied to the field, improper commands are rejected by the Interlocking System. Line reservations made by TCC are tracked to provide necessary electronic interlocking, and as the trains pass the reserved line blocks the corresponding interlockings are unlocked. Since the part of signalling system from interlocking to field elements carry vital importance, this part (including the interlocking) is constructed using a failsafe design methodology.

2.3 Switches

Unlike road vehicles trains change lines using switches. A switch can be in normal or reverse position. The position of a switch can be changed from TCC unless it is locked by interlocking. So as to provide fail-safe operation, *normal* and *reverse* indications also exist for a switch to show its current position.

2.4 Signals

Signals on railways inform the drivers on the traffic ahead. Similar to road traffic, red means that the next block is occupied (or reserved) so the train must stop. Yellow means that the next block is free but not the block after that (ie the next signal is red, so the train should proceed to stop at the next signal). Green signal means at least the next two blocks are free, so the train can proceed within speed limits. Unlike road signals, however, some railway signals can have four aspects especially near station areas. The fourth aspect, which is a bottom yellow light for the Turkish Railways, designate a line change ahead (so the train should proceed with reduced speed). On station exit locations where there is always a line change ahead, dwarf signals are used instead of four aspect signals.

2.5 Track Circuits and Axle Counters

Existence of a train in a railway block can be determined by the help of track circuits or axle counters. Track circuits are basically DC (or AC) electric circuits that exist at fixed blocks of rails. When a train enters into the block, its axles short circuit the rails. This is interpreted as the occupancy of the block by the track circuit. Axle counters are used at the ends of railway blocks to count the number of incoming and outgoing axles. When the total number of incoming axles is higher than that of outgoing axles, the block is assumed to be occupied.

3. INTERLOCKING DESIGN

In order to design a failsafe interlocking system, several international standards including EN 61508, EN 50126, EN 50128 and EN 50129 need to be utilized. EN 61508 is designed as an umbrella standard for all applications that include electric, electronic and programmable components with functional safety requirements. EN 50126/8/9 standards are developed by European Committee for Electrotechnical Standardization (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. In particular, determination of safety integrity level (SIL) for given hardware is described in EN 50129 (and EN 61508-2).

Until recently, it had been demanded that specialized hardware equipment is to be used for railway signalization, where the required SIL is usually 3 or 4. However, with the advance of off-the-shelf failsafe programmable devices (such as failsafe PLCs), it is now acceptable to use generic failsafe devices in such applications. Therefore it is possible to construct the interlocking hardware using certified failsafe components and concentrate on software where the major “know how” exist (Söylemez 2010). This approach brings important opportunities for developing countries where a deep understanding of automation theory exists and constructing failsafe hardware is difficult and time consuming. Therefore, the development process as well as architecture and design of interlocking software are discussed in the rest of this section.

3.1 Software Development Process

Both EN 61508 and EN 50128 suggests the use of V-model in the development of safety critical software (see Fig 2). According to this model the system specifications are determined first. The subsystem and software specifications are then derived from these. Using well defined software specifications, the software is designed using formal or semi-formal methods (such as automatons and Petri Nets). This is followed by the coding phase where ladder diagrams or function block diagrams (FBD) are used. Each phase is tested accordingly and if a problem is detected the phase and all the phases below that phase are reconsidered as illustrated in Fig 2.

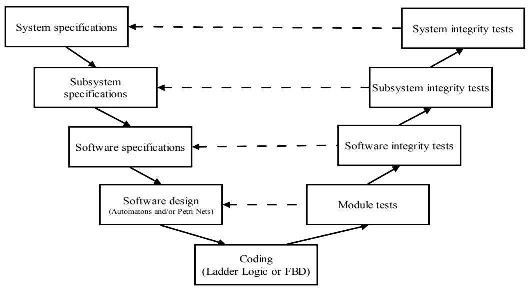

Fig. 2. Interlocking development lifecycle (the V-model)

3.2 System Architecture

EN 50128 proposes several techniques for forming the architecture of safety critical software such as interlocking software where the SIL requirement is more than 2. One such technique is the use of *diverse programming*. In this technique, more than one independent group develop the software with the same specifications but different techniques. The outputs of algorithms provided by separate groups are compared online to make a safety critical decision. In order for a safety critical signal (for instance a change of position for a particular switch) to be sent to the field, all algorithms must agree. If a disagreement occurs then the system proceeds to a safe state (all lights are set to red and all switches are locked for the region of interest).

Four independent groups (two from ITU, two from TUBITAK-UEKAE) have developed interlocking software using automatons and Petri Nets in TRSP. Three of these algorithms are to be used for diverse programming purposes as illustrated in Fig 3. Although it increases safety integrity level of a program, the main disadvantage with diverse programming is that the system can fall into safe state frequently (causing operation distractions), if the algorithms do not produce the same results at the same time. A possible way to overcome this shortcoming is documenting software specifications very clearly and using a careful synchronization in communication and decision making unit (CDMU). The synchronization code should be written to allow time discrepancies between interlocking algorithms.

Another technique recommended by EN 50128 standard for the architecture of safety critical software is the use of *defensive programming*. Defensive programming implies several extra control points in the program to check the validity of results and/or variables of the program. One possible approach in the context of defensive programming is checking the general (and some national) rules before making safety critical decisions in developing interlocking software (see Fig 4). For instance, no matter what a signal light cannot be yellow (or green) when the next block (or the block after) is occupied. This is a general rule and should be checked independently from software design.

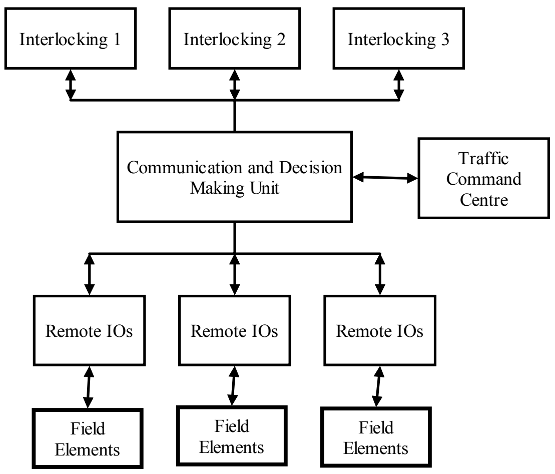

Fig. 3. Architecture of the interlocking system

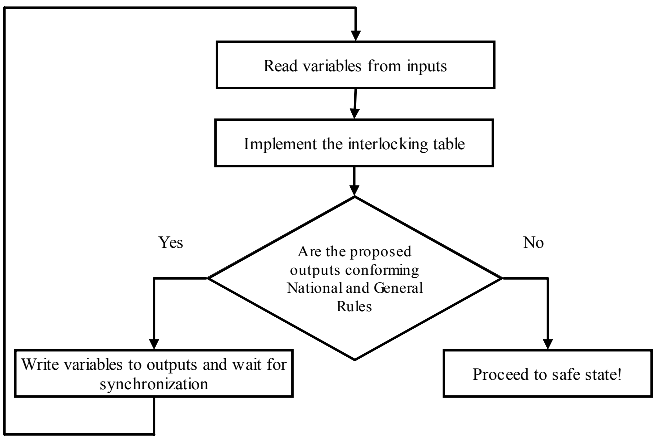
Read variables from inputs
Implement the interlocking table
Are the proposed outputs conforming National and General Rules
Yes
No
Write variables to outputs and wait for synchronization
Proceed to safe state!

Fig. 4. Structure of rules to be obeyed by interlocking software.

3.3 Software Design

The design of interlocking software usually starts by defining an interlocking table. The interlocking table basically describes possible routes and conditions to reserve these for a given railway yard. Colour combinations of entering signals to blocks are also described depending on those of the end signals in interlocking tables. An example interlocking table is given in Table 1 for a simple railway yard depicted in Fig 5. Two possible routes namely 1BT-1ST and 1BT-2ST exist for this sample railway yard. It can be read from Table 1, for instance, that switch one (Sw1) must be in reverse position and locked in order to lock the route 1BT-1ST. Moreover, the entering signal (2D) is set to yellow over green (YG), if the block 1BT-1ST is reserved and the next signal (52DB) is yellow (Y) or green (G). The signal 2D is set to yellow over yellow (YY), when the block 1BT-1ST is reserved and the signal 52DB is red (R). Interlocking table forms the main communication tool between signalling engineers and control and automation engineers in clarifying the system specifications.

Table 1 Interlocking Table

Route Selection	Signal Light 2D	Position of Switches	Next Signal Light
1BT-2ST	G	Sw1 – N	52DA – Y,G
	Y		52DA - R
1BT-1ST	YG	Sw1 – R	52DB – Y,G
	YY		52DB – R

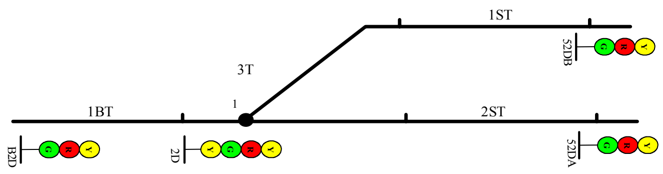

Fig. 5. Sample Railway yard (G-Green, Y-Yellow, R-Red).

Both the standards EN 61508-3 and EN 50128 highly recommend the use of formal and semi-formal methods in the design of safety critical software where SIL 3 or more is targeted. Automatons or Petri Nets can be used as formal techniques in the case of PLC programming.

An automaton is basically a finite state machine where states are represented by circles and transitions are represented by arrows. A system can be in one, and only one, state at any given time. A state transition can occur when an event related with the current state occurs. The names of events that result in state transitions are written on transition arrows in an automaton. A deterministic automaton G is described formally by a six-tuple

$$G = (X, E, f, \Gamma, x_0, X_m) \tag{1}$$

Where X is the set of states, E is the finite set of events, f is the transition function, Γ is the active event function, x_0 is the initial state, and X_m is the set of marked states. Detailed information on the theory of automatons can be found in Cassandras and Lafortune (2008).

Building an automaton to describe certain *behaviour* is actually a very intuitive and straightforward process. Furthermore, it is possible to easily produce PLC code that provides the behaviour described by a given automaton. An automaton with 3 states (0, 1, and 2) and 3 events (t1, t2 and t3) is given in Fig 6. The automaton changes from state 0 to state 1 when the event t1 occurs, and from state 1 to state 2 when the event t2 occurs. The automaton goes to state 0 or state 1 with the events t3 or t1, respectively, when it is in state 2.

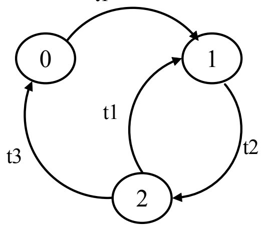

Fig. 6. An example automaton

In order to produce PLC code for a given automaton, the following rules can be used (see Hasdemir and Kurtulan 2006, Türk 2010, Sonat 2010).

$$q_n(k+1) = (S_n + q_n(k)\bar{R}_n)\prod_{j=1}^{n-1}\overline{q_j(k+1)} \tag{2}$$

for $n = 1, 2, \cdots, N-1$. Here, $q_n(k)$ represents the status of the n^{th} state at time step *k*, S_n is the set of conditions that move the system into state *n*, R_n is the set of conditions that move the system away from state *n*, and N is the number of states. State 0, which is usually considered as the initial state, is set when none of the other states are set. That is

$$q_0(k+1) = \prod_{j=1}^{N-1}\overline{q_j(k+1)} \tag{3}$$

An example pseudo code for generating a PLC program for the example automaton given in Fig 6 can be found in Table 2.

Table 2 Pseudo code for the example automaton

```
Q1 = q1 !t2 + (q0+q2) t1
Q2 = !Q1 (q2 !t3 + q1 t2)
Q0 = !Q1 !Q2
q0 = Q0
q1 = Q1
q2 = Q2
```

A well known problem related with automatons is the phenomenon called state explosion that happens in describing complex systems. The number of states can increase radically to prohibit any feasible implementation in such cases. It is possible to divide the system into subsystems to prevent state explosion in the design of interlocking systems. The automatons describing the behaviours of switches, signals, and route reservations are designed separately in this approach. Hence, the behaviour of the whole system is described by parallel automatons. An automaton for controlling switches is given as an example in Fig 7.

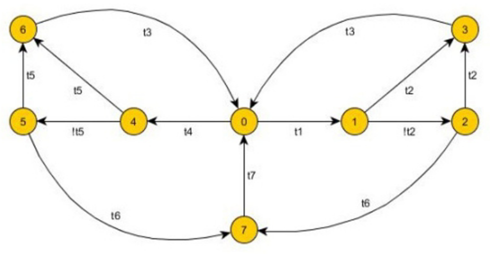

Fig. 7. An automaton for controlling switches.

Petri Nets (PNs) provide an alternative approach for describing behaviour of discrete event systems as stated in Cassandras and Lafortune (2008) and in Murata (1989). A Petri Net is defined by a five-tuple

$$PN = (P, T, A, w, x) \tag{4}$$

where P is the finite set of places, T is the finite set of transitions, $A \subseteq (P \times T) \cup (T \times P)$ is the set of arcs from places to transitions and from transitions to places, $w \in \mathbb{Z}^+$ is the weight function on arcs, and x is a marking of the set of places P. Usually, marking of a place is called as the number of tokens in that place. An example PN that has a similar behaviour with the automaton given in Fig. 6 is depicted in Fig. 8. There exist a token in place P0 for the PN given in Fig. 8. It is said that firing conditions for transition t1 are satisfied since all the places (P0 in this case) that are connected to this transition with incoming arcs have more than or equal tokens to weights on the arcs (1 in this case). Therefore when the event related with t1 happens, token in P0 passes to P1. Hence, the firing condition of t1 is disabled and firing condition of t2 is enabled. The PN given in Fig. 8 is a *safe* (or 1-bounded) PN since there exists one, and only one, token available in the PN at any given time (Murata 1998). It is possible to show that safe PNs can be used to describe the behaviour of interlocking. Hagalisletto et al (2007), for instance, examine the use of PNs for modelling and control of large scale railways.

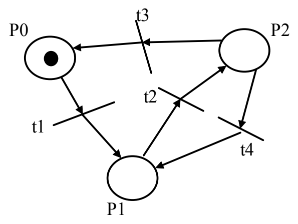

Fig. 8. An example Petri Net

Automated Petri Nets (APNs), where in addition to classical arcs, enabling and disabling arcs are defined, are given as an extension to classical PNs to simplify the notation (Uzam 1998). Enabling arcs in an APN define conditions to enable transitions, whereas disabling arcs define conditions for disabling transitions. The application of APNs in the design of interlocking software is described in Durmuş and Söylemez (2009), and Durmuş et al (2010a, 2010b, 2010c). It is, for example, possible to use supervisory control theory (Ramadge and Wonham (1987, 1989)) in the design of interlocking systems (Giua and Seatzu 2008, Durmuş et al 2010d).

An Automated Petri Net model that can be used to control switches is given in Fig. 9. The places SW1n and SW1r indicate the switch is in normal and reverse positions, respectively. The place SW1nr indicates the switch is moving from normal to reverse position, whereas SWrn indicates vice-versa. SW1f is a place to indicate a switch failure. The red hollow arrow from 1BT-1ST to transition t36 is an enabling arc. Hence, for example, switch 1 is only allowed to move from normal position to reverse position when the route 1BT-1ST is reserved according to APN given in Fig. 9.

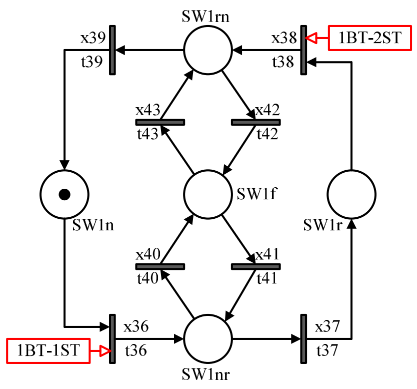

Fig. 9. An Automated Petri Net design for controlling switches (Durmuş et al 2010b)

Similar to automatons, it is possible to easily produce the corresponding PLC code for a given PN or APN (see for example Frey 2000, Thapa et al 2005, and Genter et al 2007). Therefore any design similar to that given in Fig. 8 can be translated into PLC code directly. Actually, it is possible to automatically derive the interlocking table from the topology of a given railway yard. It is also possible to automatically define all the necessary variables, the number of which can easily be expressed in terms of thousands in a medium sized interlocking (Sonat 2010, Türk 2010). It is then possible to generate automatons (Sonat 2010, Türk 2010) or PNs (Durmuş et al 2010d) automatically from the interlocking table. Finally, the corresponding PLC code can be generated automatically for the given set of variables and automatons (or PNs). Therefore most of the process of developing interlocking software (see Fig.10.) can be automated using well-defined procedures. This does not only reduce the possibility of human errors in the design process, but also reduces the time required for software development considerably, leaving more time for formal checking and testing of the system.

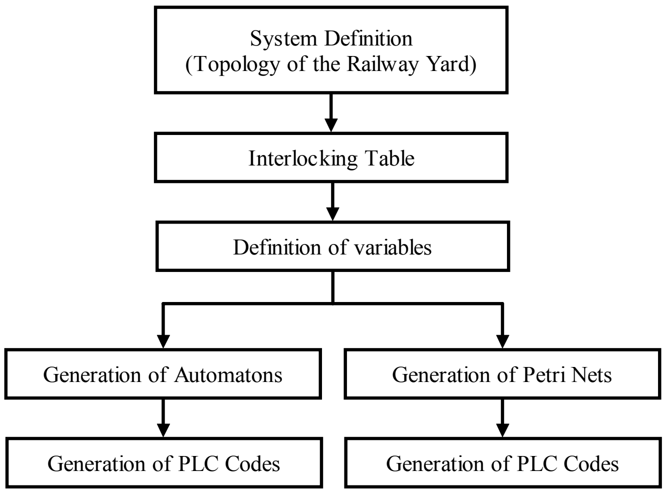

Fig. 10. Development steps for interlocking software.

4. CONCLUSION

In this study, application of automation theory to railway interlocking systems is examined. It has been discussed that both automatons and Petri Nets can be used in developing safe software for interlocking. Both techniques have their advantages and disadvantages. For instance, building automaton models is more intuitive and easier; whereas analysis of PNs has more advantages. In this paper, it has actually been suggested to use both techniques simultaneously using diverse programming techniques to increase safety integrity level of the interlocking software.

Future studies should include complete automation of the development of interlocking software and hence reducing the risk and cost in this important process. It is thoroughly believed that developing countries can benefit from such improvements considerably.

ACKNOWLEDGEMENT

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

REFERENCES

Cassandras, C.G. and Lafortune, S. (2008). *Introduction to Discrete Event Systems*, 2nd ed., Springer, New York, USA, ISBN: 978-0-387-33332-8.

Durmuş, M.S. and Söylemez, M.T. (2009). Railway Signalization and Interlocking Design via Automation Petri Nets. *The 7th Asian Control Conference*, Hong Kong, China.

Durmuş, M.S., Akın, K. And Söylemez, M.T. (2010a) Supervisory Control Approach by Inhibitor Arcs for Signalization and Interlocking Design of a Railway Yard. *Int. Symp. on INnovations in Intelligent SysTems and Applications*, Kayseri, Turkey.

Durmuş, M. S., Yıldırım, U., Kurşun, A. and Söylemez, M.T. (2010b). Fail-Safe Signalization Design for a Railway Yard: A Level Crossing Case. *The 10th International Workshop on Discrete Event Systems*, Technische Universität Berlin, Germany.

Durmuş, M. S., Yıldırım, U. and Söylemez, M.T. (2010c). Signalization and Interlocking Design for a Railway Yard: A Supervisory Control Approach by Enabling Arcs. *The 7th International Symposium on Intelligent and Manufacturing Systems*, Sarajevo, Bosnia Herzegovina.

Durmuş, M. S., Yıldırım, U. and Söylemez, M.T. (2010d). Automatic Generation of Petri Net Supervisors: Interlocking & Signalization Design for Railways. *Submitted to IFAC World Congress 2011, Milano, Italy.*

Ecopassanger (2010) http://www.ecopassenger.org/, (reached in September 2010)

Frey, G. (2000). Automatic Implementation of Petri Net Based Control Algorithms on PLC, *American Control Conference (ACC 2000)*, Chicago, USA, pp 2819-2823.

Genter, G., Bogdan, S. Kovacic, A. and Grubisic, I. (2007). Software tool for modeling, simulation and real-time implementation of Petri net-based supervisors, *16th IEEE International Conference on Control Applications*, Singapore, paper id: TuB01.3.

Giua, A. and Seatzu, C. (2008). Modeling and Supervisory Control of Railway Networks Using Petri Nets. *IEEE Trans. on Automation Science and Engineering*, Vol(5), pp. 431-445.

Hagalisletto, A.M., Bjork, J., Yu, I.C. and Enger P. (2007). 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), pp. 444-460.

Hasdemir, İ.T. and Kurtulan, S. (2006). Automatic PLC code Generation Using MATLAB, *3rd IFAC Workshop on Discrete Event Systems*, DOI: 10.3182/2000926-3-PL-4904.00022.

Murata, T. (1989). Petri Nets: Properties, Analysis and Applications, *Proc. of the IEEE*, vol(77), pp 541-580.

Ramadge, P.J. and Wonham, W. M. (1987). Supervisory Control of Discrete Event Processes. *SIAM Journal of Control and Optimization*, vol(25), pp. 206-230.

Ramadge, P.J. and Wonham, W. M. (1989). The Control of Discrete Event Systems. *Proc. of IEEE*, vol(77), pp. 81-98.

Sonat, A. (2010). Interlocking Algorithm Design in Railway Transportation Systems and Automated Code Generation with Automata Approach, *MSc Thesis*, *ITU Institute of Science and Technology*, Istanbul, Turkey.

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

Thapa, D., Dangol, S. and Wang, G.-N. (2005). Transformation from Petri Nets Model to Programmable Logic Controller using One-to-One Mapping Technique, International Conference on Modelling, Control and Automation, pp 228-233.

Transportation Council (2009), http://www.ulastirmasurasi.org, in Turkish, (reached in September 2010)

Turkish Railways (2010), http://www.tcdd.gov.tr/ (reached in September 2010)

Türk, S. (2010). Automatic Interlocking Algorithm and Code Generation Method for Railway Transportation Signalization Systems, *MSc Thesis*, *ITU Institute of Science and Technology*, Istanbul, Turkey.

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