13th IFAC Symposium on Control in Transportation Systems
The International Federation of Automatic Control
September 12-14, 2012. Sofia, Bulgaria

Safety-Critical Interlocking Software Development Process for Fixed-Block Signalization Systems

Mustafa S. Durmuş*, Uğur Yıldırım, Oytun Eriş**, Mehmet T. Söylemez****

**Control Engineering Department, Istanbul Technical University, Istanbul, TURKEY, (Tel: +90-505-2102113; e-mail: durmusmu@itu.edu.tr)*
***Control Engineering Department, Istanbul Technical University, Istanbul, TURKEY, (e-mail: {yildirimu, erisoy, soylemezm}@itu.edu.tr)}*

Abstract: *Reliability* is defined as "ability of an item to perform a required function under given conditions for a given period of time" in CENELEC (European Committee for Electrotechnical Standardization) EN 50128 document where software development requirements for railway applications are defined. Similarly, *Software Reliability* is defined as the probability of failure-free software operation for a specified period of time in a specified environment which is also important factor affecting system reliability (Pan, 2012). Systems such as railway signalization systems, where small errors can result in fatal accidents and death of several people or possibility of unwanted risks (like component malfunction) is high, also need software with high reliability. In this study, some parts of the safety-critical interlocking software development process for the Turkish National Railway Signalization Project (TNRSP) executed in cooperation with Istanbul Technical University (ITU) and The Scientific and Technological Research Council of Turkey (TUBITAK) for Turkish State Railways is defined.

Keywords: Railway Signalization Systems, Safety-Critical Software, Interlocking.

1. INTRODUCTION

In 25th February, 1991, during the Gulf War, a patriot missile system failed to destroy an incoming Scud missile which hit into US army barracks. 28 soldiers were killed and 98 were injured (Patra, 2007). The main reason was a software bug in patriot's radar that caused deviation from the target. Deviation was 137m from target after 20 hours of run time whereas it was 687m when 100 hours passed (Lum, 1992).

Early interlocking systems were mainly based on vital relays when the train density and the speeds of trains were not as high as today (Hall, 2001). Electronic (or computer) based interlocking systems are more flexible and easy to design but need much more software safety analysis from modeling to design. In addition to these, worst-case scenario software tests have to be realized to expose such software errors (or bugs) since error is inevitable, when there is a human involved in the process.

To cope with these kinds of errors, both human-induced and hardware-induced, communities like CENELEC, UIC (International Union of Railways) (UIC, 2012) and ERA (European Railway Agency) (ERA, 2012) developed standards for railway-related hardware and software components. The main aspect of the safety standards is to provide high levels of reliability and safety for all kinds of applications. As a result, a very important concept named as Safety Integrity Level (SIL) is described in these standards.

SIL is the probability of a system to execute the safety functions required in all specified input conditions within a specified time interval (Spellemaeker and Witrant, 2007). In other words, SIL is the safety target for each piece of equipment on the system depending upon its relative contribution to the hazard in question (Smith and Simpson, 2004).

SIL is considered in two parts namely, software SIL and hardware SIL. Probabilistic calculations are done based on the failure rates of components and the architecture of the design in order to determine the SIL level of a given hardware. However, acquiring required SIL on software needs appropriate selection of related methods and techniques for system modeling, module designing and software testing steps defined in safety-related standards developed by CENELEC.

SIL of an interlocking software that controls a passenger station has to be at least SIL 3 (Söylemez et. Al. 2011) depending on different parameters such as consequence of the risk (C), the frequency and the exposure time of risk (F), the possibility of failing to avoid hazard risk (P) and the probability of the unwanted occurrence of risk (W) those explained in figure D.1, figure D.2 and Table D.1 in EN 61508-5 standard (IEC 61508-5, 1997).

In this study, an interlocking software development process for Turkish State Railways from modeling to testing for fixed-block signalization systems is explained. The paper is structured as follows: In section 2, safety-critical software development process is explained, and in section 3 an example is given for module design using a Petri Net model. Finally, the conclusion is given on section 4.

2. SAFETY-CRITICAL SOFTWARE DEVELOPMENT

As a result of a recent change in the related standards, it is possible to use certified COTS (commercial off the shelf) products for the hardware part of railway interlocking systems. In addition to this, designers are obliged to use highly recommended (HR) methods and techniques like semi-formal modeling methods (Petri Nets or Automatons) (EN 50128, 2001, Durmuş et. al., 2011a, Yıldırım et. al., 2011) and software architectures like diverse programming (or n-version programming), failure assertion programming or defensive programming in order to satisfy the requirements of CENELEC for developing SIL 3 software. Development from modeling to code generation and test process is given in Fig. 1.

In fixed-block signalization systems, trains can move from one direction to another by route reservations. These reservations are all realized by *dispatchers* (officer who is in charge of all railway traffic in his area of responsibility) from *traffic control center*. All dispatcher commands including route reservations, switch movements etc. are sent to interlocking software for evaluation. Interlocking software compares the commands of dispatcher with the actual situation of the railway field and accepts or rejects these commands.

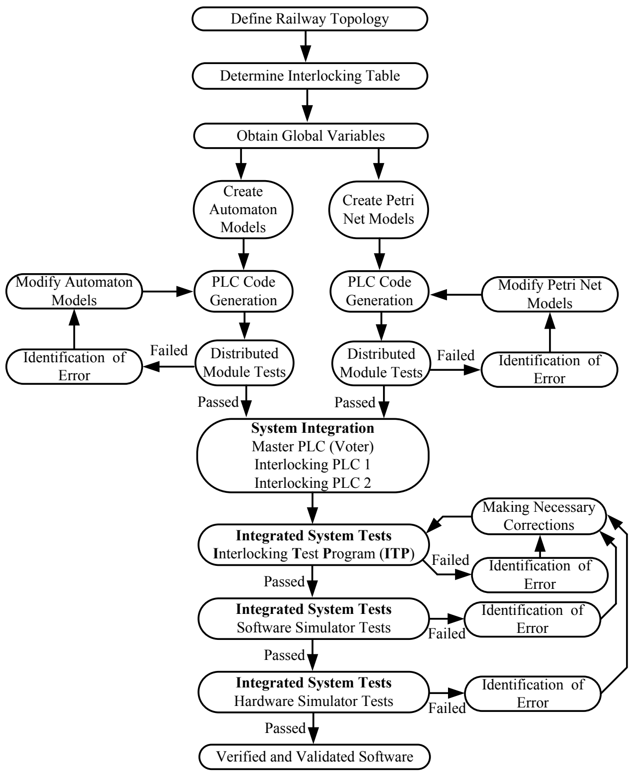

Fig. 1. Safety-critical software development process.

Since the main and the most important task of interlocking software is to provide safety, the first step of development process after defining *railway topology* is chosen as determining *interlocking table* (Interlocking table is railway station-dependent because the routes and related components are not the same for different railway areas). More definitions on interlocking tables can also be found in (Ferrari et. al., 2011, Vanit-Anunchai, 2010, Kuzu et. al., 2011, Yıldırım et. al., 2012).

After determination of the interlocking table the next step is to obtain *global variables*. Global variables are obtained jointly by different independent workgroups. As it is given in Fig. 1., there are at least two independent workgroups that means two independent interlocking software with the same input-output specifications. This kind of design is also known as N-version programming (or diverse programming) (Avizienis, 1976, Avizienis, 1985), which is a highly recommended (HR) technique in railway related CENELEC standards for a SIL 3 program (IEC 61508-5, 1997, EN 50128, 2001).

The workgroups can use Automatons (Cassandras and Lafortune, 1999, Dincel and Kurtulan, 2012) and Petri Nets (Murata, 1989) in order to obtain models of fixed-block signalization components. These two modeling techniques are referred as semi-formal methods in (EN 50128, 2001) and also HR for SIL 3 software. Since these topics are not in the scope of this paper they are not explained in details here. More definitions on fixed-block railway signalization components such as track circuits, colored signal lights, switches and level crossings can be found in (Hall, 2001, Durmuş et. al., 2010, Söylemez et. al. 2011).

After obtaining a model for each component, the model is converted to fail-safe programmable logic controller (PLC) codes using one of the programming languages defined in IEC 61131-3 standard such as function block diagrams (FBD) or sequential function charts (SFC). In this study, SFC is used as an example. Related component models are then connected to each other suitably by looking at the interlocking table.

Finally, distributed module tests are applied before combining two independently developed interlocking programs. Such tests have to be executed by persons other than the development group. Therefore, cross tests are preferred in TNRSP. After passing all test steps, two interlocking programs are connected to each other over a voting (Latif-Shabgahi et. al., 2004, Lorczak et. al., 1989, Gersting et al., 1991, Parhami, 1994) mechanism known as voter. Voter (master controller) is another fail-safe PLC which sends and receives commands between railway field, traffic control center and each interlocking PLC (voter can be classified as a kind of communication and decision making component) (Durmuş et. al., 2011b). A modeling example on route reservation with Petri Nets and description of test procedures are given in the next section.

3. AN EXAMPLE

In order to provide easy error tracking and flexibility in programming, all components in fixed block signalization systems are modeled as modules. Main components can be listed as signal lights, switches, level crossings and track circuits. All these components are modeled by both automatons and petri nets. Since the train movement is achieved by reserving routes in fixed-block signalization *route reservation* can also classified as a component and has to be added into the software model. The route reservation module is explained as an example in this section. The main module for route reservation and its sub-modules are given in Fig. 2. and in Fig. 3., respectively. A part of the route reservation procedure (Petri Net model) is given in Fig. 4. Descriptions related with the PN model in Fig. 4. are given in table 1.

As can be observed from Fig. 4, the behavior of the system can be completely described using a Petri Net (PN). Consider the process of handling a route request as described in Fig. 3. When a route request is received (t_2), the token, which is normally in P_2, moves to P_3 according to PN described in Fig.4. If all necessary components are available for the route request in question (t_5) then the token moves to P_4, otherwise it moves to P_1, which is the place for denying a route request. When all the switches are in required positions and there is no contradictory condition for locking the route (t_8) then the token moves to P_5, where an indication to the TCC is sent to inform the TCC that the route is ready to be locked. After receiving a final confirmation from the TCC (t_{11}), the token moves to P_6, where the route is reserved locally (by the current interlocking algorithm). When all algorithms reserve the route (t_{14}) the token moves to P_7, which indicates global reservation of the route. After this point the starting signal of the route gives the necessary color indication. If any problems are detected during this process then the route is denied, and if a route cancellation request is received from TCC the procedure is cancelled.

Every component has its own block (module) and they are linked to each other depending on conditions. For instance, a route reservation module contains all switches, signal lights and track circuits in that route.

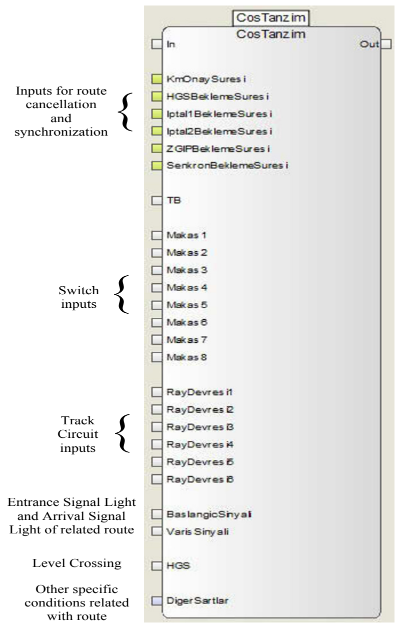

Fig. 2. Main module of route reservation.

A route reservation module as given in Fig. 2. should be used for each route on the railway yard with correct connections. Similarly, function blocks created for other components of the system are used for every component of the system and need to be connected together using the specifications of the interlocking table. If the railway topology changes (and so the number of components and possible routes) interlocking software can be updated easily by just adding related modules and connections. The railway station where the developed interlocking software is used contains 10 track circuits, 14 signal lights, 10 switches and 38 routes.

The next step after software development is to realize distributed software tests as described in Fig. 1. In this step, workgroup tests each other's interlocking software. If all tests are successful two independently developed programs combined together to work in a parallel manner with a voting mechanism (another fail-safe PLC named as Master PLC) (Eriş et. al., 2012). A photograph of the integrated system can be seen in Fig. 5.

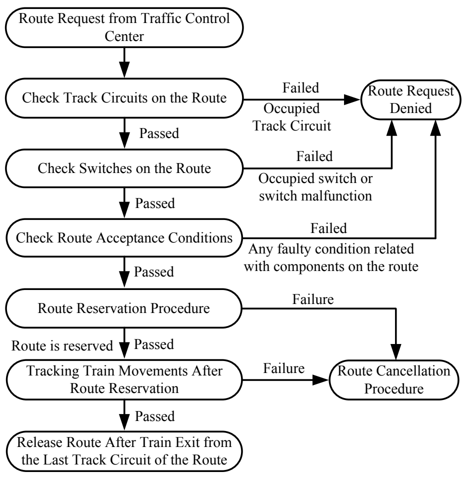
Route Request from Traffic Control Center
Check Track Circuits on the Route
Failed
Occupied Track Circuit
Route Request Denied
Passed
Check Switches on the Route
Failed
Occupied switch or switch malfunction
Passed
Check Route Acceptance Conditions
Failed
Any faulty condition related with components on the route
Passed
Route Reservation Procedure
Failure
Route is reserved
Passed
Tracking Train Movements After Route Reservation
Failure
Route Cancellation Procedure
Passed
Release Route After Train Exit from the Last Track Circuit of the Route

Fig. 3. Sub-module description of route reservation module.

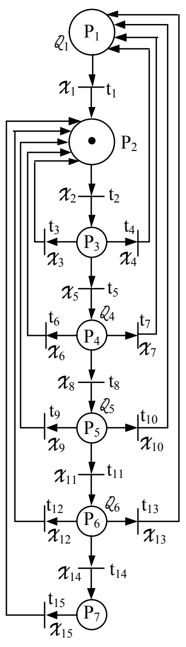

Fig. 4. Part of petri net model of route reservation procedure block in Fig. 3.

Table 1. Description of PN model in Fig. 4.

Places	**Description**	**Firing Conditions (Transition)**	**Description**
P_1	Route request denied	$\mathcal{X}_1$ (t_1)	Route request denied acknowledge
P_2	Route idle	$\mathcal{X}_2$ (t_2)	Incoming route request from control center
P_3	Route request received	$\mathcal{X}_3$ (t_3)	Route is cancelled (global signal)
P_4	Route request accepted	$\mathcal{X}_4$ (t_4)	Global deny signal from master PLC
P_5	Route is ready	$\mathcal{X}_5$ (t_5)	Check all components and conditions
P_6	Route is reserved	$\mathcal{X}_6$ (t_6)	Route is cancelled (global signal)
P_7	Route is reserved (globally)	$\mathcal{X}_7$ (t_7)	Global deny signal from master PLC
Outputs	**Description**	$\mathcal{X}_8$ (t_8)	Switches are arranged
Q_1	Route is denied	$\mathcal{X}_9$ (t_9)	Route is cancelled (global signal)
Q_4	Route is accepted	$\mathcal{X}_{10}$ (t_{10})	Global deny signal from master PLC
Q_5	Route is ready for reservation	$\mathcal{X}_{11}$ (t_{11})	Confirmation signal from control center
Q_6	Route is reserved	$\mathcal{X}_{12}$ (t_{12})	Route is cancelled (global signal)
		$\mathcal{X}_{13}$ (t_{13})	Global deny signal from master PLC
		$\mathcal{X}_{14}$ (t_{14})	Route is reserved by the other PLC
		$\mathcal{X}_{15}$ (t_{15})	Train exits from route or Route is cancelled (global signal)

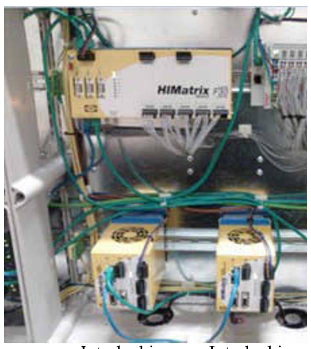

Fig. 5. Integrated system.

Testing the developed software (integrated system) is another time-consuming process and needs to be examined very carefully. After system integration the software is tested in three stages by three independent workgroups, respectively. The first test is achieved by using a program named interlocking test program (ITP), which is developed in ITU Control Engineering Department. ITP, which is the main tool used for verification, allows testing the system for all important scenarios that may occur on the railway yard in question both manually and automatically. Automatic tests can easily take a few weeks to accomplish all verification process.

After passing the functional verification tests done by ITP successfully, a separate workgroup tests the integrated system with a software simulator (Mutlu et. al., 2011), where all performance and stress tests are realized (Mutlu et. al., 2012). Finally, another independent workgroup takes the integrated system into hardware simulator tests which is 1:87 scaled model of the real railway station. Software and hardware simulator test procedures take approximately two weeks. A snapshot of the hardware simulator is given in Fig. 6.

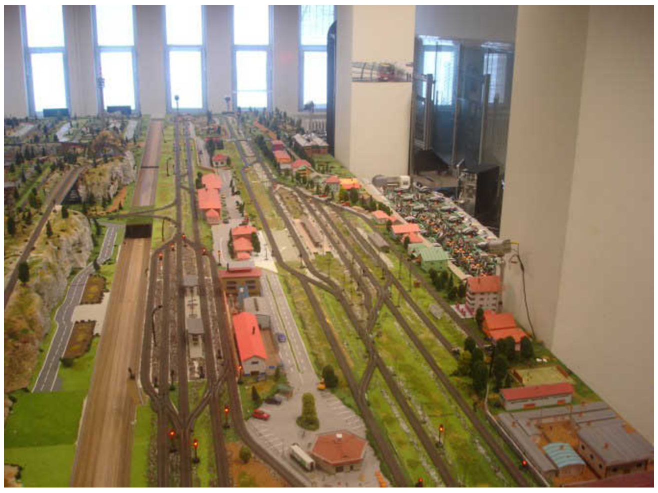

Fig. 6. Hardware simulator.

If an error is detected during the tests, the test procedure is stopped and software development groups identify and fix the error. After that, all the tests are performed from the very beginning.

4. CONCLUSIONS

In this study, development of safety-critical software (interlocking software) for fixed-block signalization systems is explained by considering safety criteria by railway-related CENELEC safety standards. An important advantage of the proposed design is its flexibility as it is quite easy to apply it even if the railway topology changes. The developed interlocking software is actually tested on Mithatpasa railway station in Sakarya, Turkey by Turkish State Railways. This study can be considered as an application of semi-formal based design methods (Automata and Petri Net based controllers) on a real-world transportation problem. Design of interlocking software for European Rail Traffic Management System (ERTMS) (level 1-level 3) can be considered as future work.

ACKNOWLEDGMENT

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

REFERENCES

Avizienis, A. (1976). Fault-Tolerant Systems, *IEEE Transactions on Computers*, Vol (C-25), pp. 1304-1311.

Avizienis, A. (1985). The N-version Approach to Fault-Tolerant Softwar(SE-11), pp. 1491-1501.

Cassandras, C.G. and Lafortune, S. (1999). *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, 1999.

Dincel, E. and Kurtulan S. (2012). Interlocking and Automatic Operating System Design with Automaton Method, submitted to *The 13th IFAC Symposium on Control in Transportation Systems, CTS 2012*.

Durmuş, M.S., Yıldırım, U., Kurşun, A. and Söylemez, M.T. (2010). Fail-safe signalization design for a railway yard: A level crossing case, *The 10th International Workshop on Discrete Event Systems*, *WODES'10*, Berlin, Germany.

Durmuş, M.S., Yıldırım, U. and Söylemez, M. T. (2011a). Application of Functional Safety on Railways Part I: Modelling & Design, *The 8th Asian Control Conference, ASCC'11*, Kaohsiung, Taiwan, pp. 1090-1095.

Durmuş, M.S., Eriş, O., Yıldırım, U. and Söylemez, M. T. (2011b). A New Voting Strategy in Diverse Programming for Railway Interlocking Systems, In *IEEE International Conference on Transportation and Mechanical & Electrical Engineering, TMEE'11*, Changchun, China, pp. 1175-1178.

EN 50128. (2001). Railway Applications, Communications, signalling and processing systems, Software for railway control and protection systems.

ERA. (2012). http://www.era.europa.eu/Pages/Home.aspx. (18.02.2012).

Eriş, O., Yildirim, U., Durmuş, M.S., Söylemez, M.T. and Kurtulan, S. (2012). N-version Programming for Railway Interlocking Systems: Synchronization and Voting Strategy, submitted to *The 13th IFAC Symposium on Control in Transportation Systems, CTS 2012*.

Ferrari, A., Magnani, G., Grasso, D. and Fantechi, A. (2011). Model Checking Interlocking Control Tables, Book Chapter, *Springer Berlin Heidelberg*, pp. 107-115.

Gersting, J.L., Nist, R.L., Roberts D.B. and Van Valkenburg, R.L. (1991). A Comparison of Voting Algorithms for N-version Programming, In *24th Annual Hawaii International Conference on System Sciences*, pp. 253-262.

Hall, S. (2001). *Modern Signalling Handbook*, Ian Allan Publishing, England.

IEC 61508-3. (1997). Functional Safety of Electrical/Electronic/Programmable electronic safety-related systems, Part 3: Software requirements.

IEC 61508-5. (1997). Functional Safety of Electrical/Electronic/Programmable electronic safety-related systems, Part 5: Examples of methods fort he determination of safety integrity levels.

Kuzu, A, Songuler, O., Sonat A., Turk, S., Birol, B. and Dogruguven, E. (2011). Automatic Interlocking Table Generation from Railway Topology, In *IEEE International Conference on Mechatronics*, Istanbul, Turkey, pp. 64-70.

Latif-Shabgahi, G., Bass, J.M. and Bennett, S. (2004). A Taxonomy for Software Voting Algorithms Used in Safety-Critical Systems, Vol (53), *IEEE Transactions on Reliability*, pp. 319-328.

Lorczak, P.R., Çağlayan, A.K. and Eckhardt, D.E. (1989). A Theoretical Investigation of Generalized Voters, In *19th International Symposium on Fault-Tolerant Computing*, pp. 444-451.

Lum, Andrew. (1992). Patriot Missile Software Problem, http://www.cs.utexas.edu/~downing/papers/PatriotA1992.pdf. (13.02.2012).

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

Mutlu, İ., Ovatman, T., Söylemez, M.T. and Gören Sümer, L. (2011). A New Test Environment for PLC Based Interlocking Systems, *In International Conference on Transportation, Mechanical and Electrical Engineering*, Changcun, China, pp. 1123-1127.

Mutlu, İ., Ergenç, A.F., Ovatman, T. and Söylemez, M.T. (2012). Design of a Hardware and Software based Test Bed for Railway Signalization System, submitted to *The 13th IFAC Symposium on Control in Transportation Systems, CTS 2012*.

Smith, D.J. and Simpson K.G.L. (2004). *Functional Safety – A straightforward guide to applying IEC 61508 standard and related standards*, 2nd ed. Elsevier Butterworth-Heinemann, 200 Wheeler Road, Burlington.

Söylemez, M.T., Durmuş, M.S. and Yıldırım, U. (2011). Functional Safety Application on Railway Systems: Turkish National Railway Signalization Project, In *24th Int. Cong. on Condition Monitoring and Diagnostics Engineering Management, COMADEM'11*, Stavanger, Norway, pp. 1683-1692.

Spellemaeker, M. and Witrant, L. (2007). How to Determine the Safety Integrity Level (SIL) of a Safety System, Available online: http://www.petro-online.com/articles/safety/15/michel_spellemaeker_lionel_witrant/how_to_determine_the_safety_integrity_level_sil_of_a_safety_system/348/. (12.02.2012).

Pan, Jiantao. (2012). http://www.ece.cmu.edu/~koopman/des_s99/sw_reliability/. (15.02.2012).

Parhami, B. (1994). Voting Algorithms, Vol (43), *IEEE Transactaions on Reliability*, pp. 617-629.

Patra, S. (2007). Worst-Case Software Safety Level for Braking Distance Algorithm of a Train, In *2nd International Conference on System Safety*, London, UK, pp. 206-210.

UIC. (2012). http://www.uic.org/. (18.02.2012).

Vanit-Anunchai, S. (2010). Modelling Railway Interlocking Tables Using Coloured Petri Nets, *Lecture Notes in Computer Science*, pp. 137-151.

Yıldırım, U., Durmuş, M.S. and Söylemez, M. T. (2011). Application of Functional Safety on Railways Part II:

Software Development, *The 8th Asian Control Conference, ASCC'11*, Kaohsiung, Taiwan, pp. 1096-1101.

Yıldırım, U., Durmuş, M.S. and Söylemez, M. T. (2012). Automatic Interlocking Table Generation for Railway Stations and Code Design, submitted to *The 13th IFAC Symposium on Control in Transportation Systems, CTS 2012*.