Modelling a railway signalling system using SDL

Stefano Bacherini Simone Bianchi Leonardo Capecchi

Carlo Becheri Alessandro Felleca

General Electric Transportation Systems

Alessandro Fantechi Emilio Spinicci

Università degli Studi di Firenze

Dipartimento di Sistemi e Informatica

1. INTRODUCTION

Formal methods have been recognized as a mean for preventing the introduction of software faults in a safety critical system. This is particularly true in the field of railway signalling systems, so that even CENELEC guidelines [4] for software development recommend the use of formal methods.

This paper describes a pilot project aimed at the evaluation of the introduction of the SDL technology in the development cycle of General Electric Transportation Systems. Through limited in the complexity of the system and in the simulation capabilities exploited, the project has shown the potential of these technology for improving the software development cycle in terms of costs reduction and of quality of the software product.

The paper is structured as follows: section 2 describes the role of SDL related to the CENELEC standards; section 3 describes the system that has been considered in the project; section 4 presents the main features of the SDL specification; section 5 discusses the results obtained by simulating the specification; section 6 concludes the paper giving some direction in which the project can evolve in the future.

2. CENELEC EN 50128 AND FORMAL METHODS

The CENELEC EN 50128 Standard, "Railway Applications: Software for Railway Control and Protection Systems" is part of a group of related Standards: the EN 50126, "Railway Applications – Dependability for Guided Transport Systems", and the EN 50129, "Railway Applications – Safety-related electronic railway control and protection systems"; the first addresses system issues on the widest scale, while EN 50129 addresses the approval process for individual systems which may exist within the overall railway control and protection system. Particularly, the function of EN 50126 and EN 50129 is also to specify the safety functions allocated to software. The EN 50128 Standard concentrates on the methods which need to be used in order to provide software which meets the demands for safety; the key concept of this European Norm is that of software safety integrity levels (SILs): the more dangerous the consequences of a software failure, the higher the software safety integrity level will be.

So, following the guidelines of the related standards, a System Safety Requirements Specification, which identifies all safety functions allocated to software and determines the system safety integrity level, is needed to define the Software Requirement Specification and the Software Architecture.

The EN 50128 Standard recommends the use of formal methods in the Software Requirements Specification as stated in Table A2, Software Requirements Specification (clause 8): "*The Software Requirements Specification will always require a description of the problem in natural language and any necessary mathematical notation that reflects the application*".

The techniques listed are:

1. Formal Methods including for example CCS, CSP, HOL, LOTOS, OBJ, Temporal Logic, VDM, Z and B
2. Semi-Formal Methods
3. Structured Methodologies including for example JSD, MASCOT, SADT, **SDL**, SSADM, and Yourdon.

The classification of SDL outside the Formal Methods category may however be debated: actually, SDL has evolved in a fully equipped formal language.

SDL, due to its standardization by ITU-T [1], has been widely applied in the telecommunication industry, and commercial tools are available to support writing specifications in SDL and verifying them.

The availability of sophisticated support tools has allowed SDL to pass the borders of the telecommunication industry and to gain attention also in the railway industry, as witnessed by some previous experience [10].

General Electric Transportation Systems as well has chosen SDL for a first evaluation of the adoption of formal methods in its development cycle. Such methods are recommended by EN50128 for SIL 3 and SIL 4 systems: the SCA system described in the following, selected for this evaluation, is a SIL 4 system.

3. DESCRIPTION OF THE SCA SYSTEM

The SCA system (Sistema Conta Assi, wheelset counter system), developed by General Electric Transportation Systems, GETS, is a device which counts the number of wheelsets, belonging to trains which are travelling in a given railway section. This operation is made using suitable sensors, which are placed aside the track. The consecutive calculation of the number of wheelsets getting out of the railway section allows to establish if the section itself is free (if the outcoming number of wheelsets is equal to the income number of wheelsets) or is still occupied by a train. This information is used to enable or to forbid the next train to enter the section, by means of a suitable semaphore signalling.

The SCA system is composed by several control and acquisition units, named UCA (Unità Conta Assi, wheelset counter unit), placed along the tracks, and by detection points placed aside the track, named PRA (Punti di Rilevazione Assi, wheelset detection points). The UCA-PRA connection is shown in figure 1.

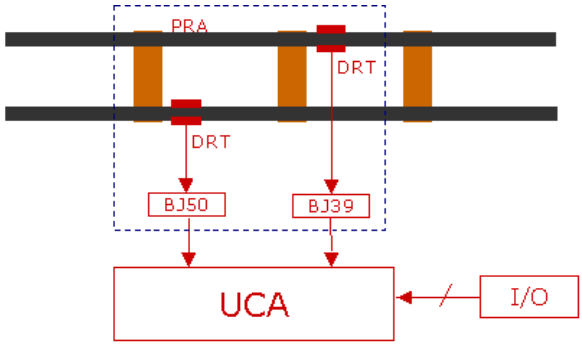

Figure 1 - UCA-PRA connection.

A PRA is formed by two DRTs (Detettore Ruota Treno, train wheel detector), each of them constituted by a couple of magnetic sensors, allocated at the sides of the rail. The sensors are able to reveal the wheels of the train running on the track.

Since the SCA architecture is safety-critical, DRTs are two for every PRA in order to implement hardware redundancy. The two magnetic detectors, (in figure 1, the Junction Blocks BJ50 and BJ39) must give the same signal consecutively. If only one of them sends the signal of a passing wheel, the UCA goes in a "Fail-safe" state; this is made by disabling a suitable automatic block relay.

UCA units are located at the beginning of a section; they receive and process the signals modulated by the DRTs, in order to determine the number of wheelsets running on the track; these data are sent to remote UCAs via modem. Using remote data, UCA can determinate if the section is still occupied or not, and eventually stop the train by disabling the automatic block relay. Since a track can be generally run in two directions, UCAs are able to check both the trains entering the railway section and the leaving ones.

The SCA type depends on the number of UCAs constituting the system, and on the configuration data of UCAs themselves. Among these we mention the most important configuration parameters:

1. Information for managing a standing train in the checked railway section;
2. UCA role during modem communication (master, slave);
3. Number of checked railway sections;
4. Number of connected PRAs;
5. Presence of GPS system for SOS emergency signalling;
6. Type of the SCA system (single-section, multiple-section);

The main UCA functionalities are the following:

- *Core function and Input/Output management*

  This is the core wheel counting functionality, but extends to read all the input signals (from DRTs or from other UCAs), handled by means of proper filters, and to produce according output, e.g the enabling/disabling of the automatic block relay.

- *Error management*

  This functionality analyzes all the errors occurred in the system units. If a *vital error* occurs, the section is declared busy, disabling the automatic block relay. All the error data are saved in a NOVRAM,

together with the time and date of occurrence. In this way, error data can be downloaded from a diagnostic PC.

- *Diagnostics*

  This function allows a runtime hardware check to be performed. It is possibile to get the diagnostic results, error and configuration data by connecting a PC to the UCA RS232 serial port.

4. SCA SDL SPECIFICATION

In order to evaluate the introduction of the SDL technology in the traditional GETS development cycle, a SDL specification has been developed for SCA (figure 2), consistently to the Software Requirements Specification and Architecture Specification documents. The main feature of this SDL description is the implementation of the multiple-configuration, which is performed taking advantage of the SDL semantics: the Configuration block stores the configuration parameters, described in the previous clause, and generates the number of istances of PRA and diagnostic processes, connected to the block UCA, that are allocated to simulate the given system configuration at the startup. The configuration parameters also identify the role (master or slave) of the UCA block during modem communication, depending on the type of the SCA system to be simulated (single-section, multiple section): for example, if the UCA block represents an intermediate UCA in a multiple-section configuration, the two blocks Remoto1 and Remoto2 simulate the remote UCAs which the UCA block is connected to. Diagnostic communications are specified by means of

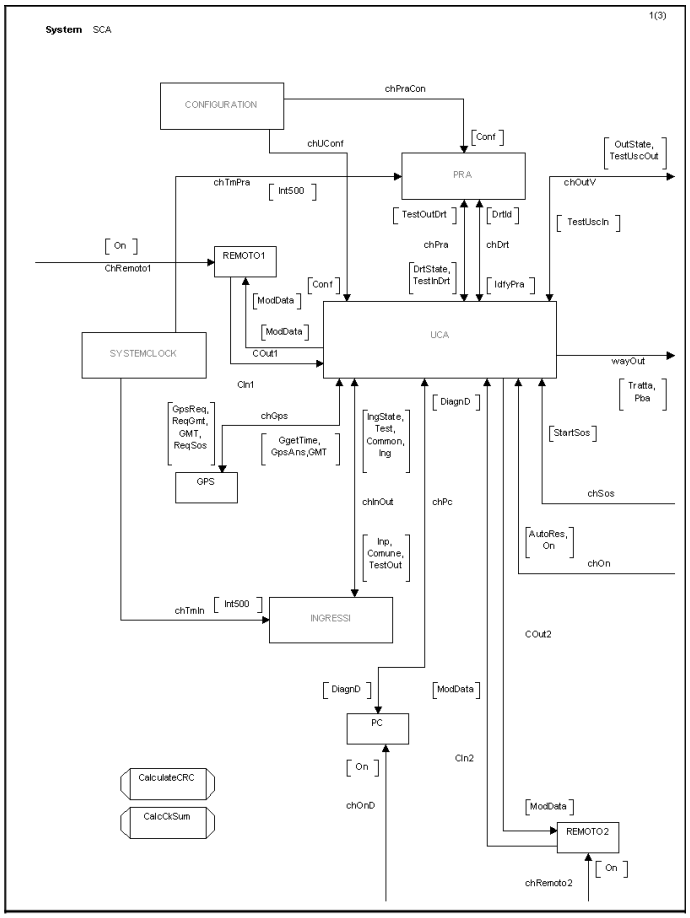

Figure 2. SCA SDL Formal Specification.

the PC block, that simulates the external RS232-connected PC.

The main unit of this SDL system is the UCA substructure (Figure 3), in which the input/output and error management functionalities are modeled. The MANAGE_DRT and MANAGE_INOUT blocks handle the signals coming from DRTs and the generation of errors, if the input signals are not the expected ones. The DIAGNOSTICA block is concerned with the diagnostic requests from the external PC. Communications to remote UCAs are modeled by the two istances MODEMSLOT1 and MODEMSLOT2 of the MODEM block type, which implements the diagnostic communication protocol. Accordingly to safety requirements, system tests are performed by a main loop inside the block LOOP_MAIN.

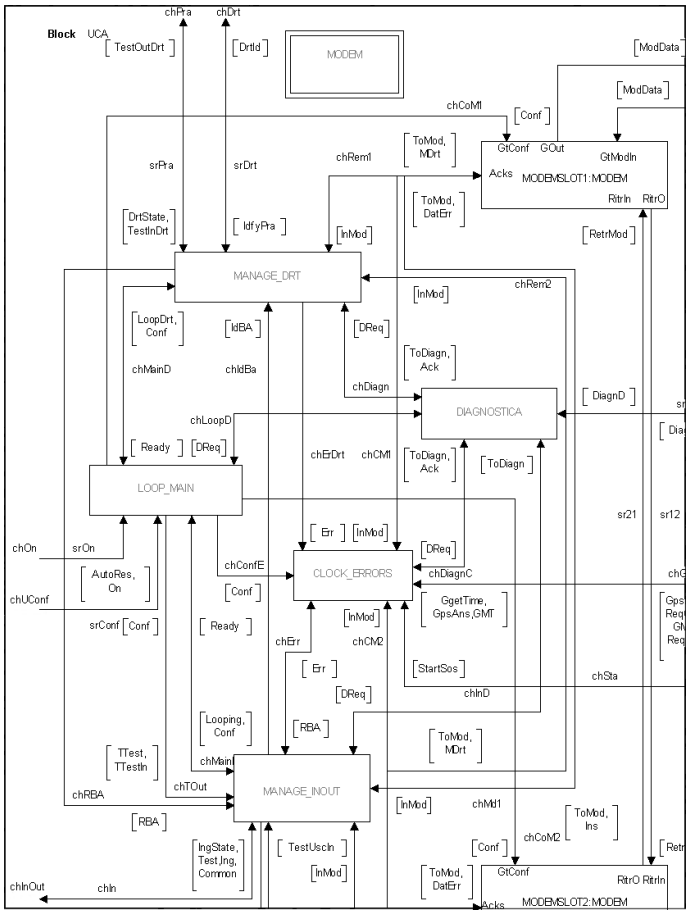

Figure 3. Substructure UCA in SCA system.

5. SIMULATION TESTS

Once the SDL specification has been obtained, the formal design of the system and its requirements have been validated by simulating the SCA SDL model, by means of the Cinerella SDL tool. During the simulation activity, the system is handled as a black-box, where, towards suitable input stimuli, the system must send the expected output signal(s) to the system environment. If any unwanted behavior is revealed, a possible flaw in the system design can be located searching the parts of the model which processed the input signals. Such processing can be analyzed in detail using the Message Sequence Chart formalism [2]: figure 4 shows the MSC revealing an anomaly occurred both in the SDL model and in the SCA code, and solved during the code implementation.

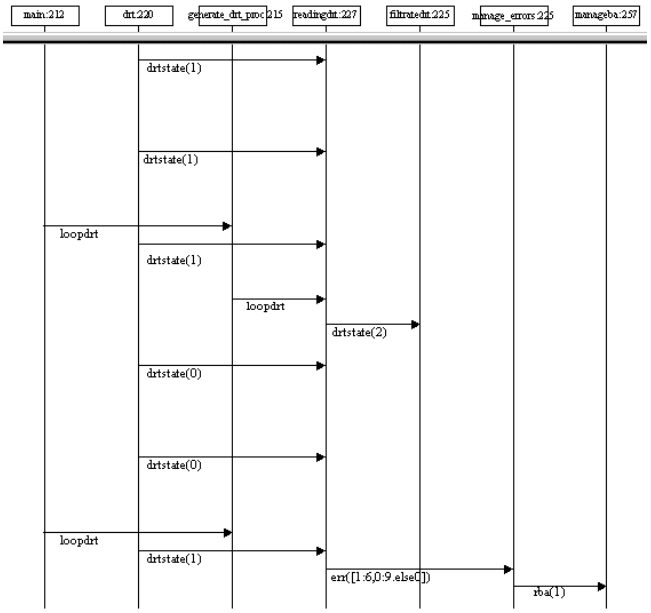

Figure 4. DRT Buffer reading and writing.

The problem deals with the buffer in which DRT states are registered. Such buffer is read while executing the software Main Loop, but written every 500 μsec; the system goes in a fail-safe state if a buffer cell is written twice without having read its value before, with the risk to lose important data. The MSC shows how the frequency of buffer writing operations (sending of *drtstate* signal to the istance of the *reading* process) is much greater than that of the reading ones (sending of *loopdrt* signal from process *main*). As a result, a message to the process *manage_errors* is sent, including the buffer overwrite error code. To fix this problem, the reading frequency of the buffer has been increased.

We want to point out that, during the functional testing of the actual code of SCA, the GETS engineers indipendently discovered this problem, analyzing data inside the EPROMs, and they adopted a similar solution: the data taken from the DRT are written on a circular buffer and read cyclically from a filtering process. The writing speed of the data on the buffer was too high w.r.t. the reading speed; though this event could not drive the system into an unsafe state, the reliability of the system was anyway degraded. Therefore the solution has been to slacken the time of acquisition of the data from the DRT.

The simulation of the SDL specification would have permitted to reveal the problem without executing any functional tests and thus proving potentially worth of great benefit for the system security and for the reduction of costs.

Security has an impact in this kind of product mainly for synchronization (communication) problems more than design complexity problems. The use of SDL allows to explore the behaviour of the different processes and their communication problems. This is a big task without using an overview support tool.

From the point of view of development costs, SDL lowers the costs of the testing phase resolving most of the problems during the early modeling of the system, without the need of building a prototype of the system to execute system functional tests.

6. CONCLUSIONS AND FUTURE WORK

We have described the experience made at GETS on the application of SDL and of the Cinderella SDL support tool, to a typical product of the company. The experience, through limited in the complexity of the system and in the simulation capabilities exploited, has shown the potential of these techniques and tools to improve the software development cycle.The next step in the prosecution of this evaluation activity will be the adoption of more sophisticated support tools; in particular, Telelogic Tau SDT allows for a thorough validation of a specification, as well as for the automatic code generation. However, although SDL is standardized, the porting of the SCA specification from Cinderella to SDT has proved more costly than expected, due to unforeseen incompatibilities of the format produced by the former and the one accepted by the latter. We are nevertheless interested to evaluate other formal methods as well: among these, the Statechart formalism and the companion Statemate tool by iLogix appear to be gaining a wider industrial acceptance also in the railway industry [9].

7. BIBLIOGRAPHY

[1] ITU-T, *Recommendation Z.100 - Specification and Description Language (SDL).* Geneva, 1992

[2] ITU-T, *Recommendation Z.120 - Message Sequence Charts (MSC).* Geneva, 1999.

[3] CENELEC EN 50126, *Railway Applications – Dependability for Guided Transport Systems.* European Committee for Electrotechnical Standardization, June 1997.

[4] CENELEC EN 50128, *Railway Applications – Software for Railway Control and Protection Systems.* European Committee for Electrotechnical Standardization, June 1997.

[5] CENELEC EN 50129, *Railway Applications – Safety-related electronic railway control and protection systems.*European Committee for Electrotechnical Standardization, December 1999.

[6] SS-PROG-SCA-001 Rev. 00, *SCA System Specification.* General Electric Trasportation System, Florence, April 2001.

[7] SRS-PROG-SCA-001 Rev. 00, *SCA System Software Requirement Specification.* General Electric Trasportation System, Florence, April 2001.

[8] SAS-PROG-SCA-001 Rev. 00, *SCA System Software Architecture Specification.* General Electric Trasportation System, Florence, May 2001.

[9] J. Bohn, W. Damm, H. Wittke, J. Klose, A. Moik, *Modeling and Validating Train System Applications Using Statemate and Live Sequence Charts.* Integrated Design and Process Technology, IDPT-2002, United States of America, June 2002.

[10] A. Cimatti, P. Pieraccini, R. Sebastiani, P. Traverso, A. Villafiorita, *Formal specification and validation of a vital protocol.* FM'99 - World Congress on Formal Methods. Toulouse, France, September 1999. LNCS 1709, Springer-Verlag.