Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Electronic Notes in Theoretical Computer Science 133 (2005) 3–19
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Geographical Versus Functional Modelling by Statecharts of Interlocking Systems

Michele Banci[1]

Formal Methods and Tools Group
ISTI - CNR
Pisa, Italy

Alessandro Fantechi[2]

Dipartimento di Sistemi e Informatica
Università degli Studi di Firenze
Firenze, Italy

Abstract

The development of computer controlled Railway Interlocking Systems (RIS) has seen an increasing interest in the use of Formal Methods, due to their ability to precisely specify the logical rules that guarantee the safe establishment of routes for trains through a railway yard.
Recently, a trend has emerged about the use of statecharts as a standard formalism to produce precise specifications of RIS. This paper describes an experience in modelling a railway interlocking system using statecharts. Our study has addressed the problem from a "geographical", distributed, point of view: that is, our model is composed by models of single physical entities (points, signals, etc..) that collectively implement the interlocking rules, without any centralized database of rules, which is on the other hand a typical way of implementing such a system (what we call "functional" approach).
One of the main aims of our approach, is to verify its ability to reduce revalidation efforts in the case of physical modifications to the yard; we show how the geographical approach may reduce this effort by requiring only the revalidation of those software modules that are actually affected by the changes.

Keywords: Railway signalling, interlocking, safety-critical systems, formal specification, Statecharts, validation.

[1] Email: michele.banci@isti.cnr.it
[2] Email: fantechi@dsi.unifi.it

1571-0661 © 2005 Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.entcs.2004.08.055

1 Introduction

A Railway Interlocking System (RIS) is a complex installation for the safe establishment of routes for trains through a railway yard. Electronic signalling systems have replaced (and actually are still replacing) old mechanical interlocking systems due to their advantages in terms of:

- less dependency on operator's failure;
- self-diagnostic system checks to improve reliability;
- possibility of integrated and centralized systems.

Typically, a RIS receives a route request; the answer to this request has to go through a series of checks and actions of the kind:

- a route can not be locked if a track section element is occupied;
- a route is available if no part of the route is already locked;
- if the route is available it will be locked;
- locking a route implies that all its track elements become locked;
- when a route is locked signals for the route can be switched to green, and then the original route request is positively acknowledged.

The aim of these checks and actions is to guarantee that no train can be driven into a route occupied by another train.

The case for Formal Methods in the development of RISs is evident from this very list of checks and actions, and there is a considerable literature about formalization of interlocking systems (see for example [1,3,7]). And indeed CENELEC EN50128 guidelines for software development in the railway industry [5] recommend the use of formal methods in the development of safety critical computer-controlled systems.

We can observe however a new trend of the last years in the area of interlocking systems. The recent opening of markets inside Europe has seen the end of the traditional strict collaboration between national railway companies and national railway industries, in which in a sense there was a limited need of precise specifications, since every misunderstanding ended up to be resolved by phone. In the case of RIS, the principle schemata, a sort of ladder-like, relay based language, widely known by railway engineers, constituted the common reference language by which to describe interlocking systems, since the times interlockings were relay based [8]. In the introduction of computer-based RIS some industries had defined their own approach to produce computer based interlockings, more or less formalized, often based on some logic formalism or language, but they were anyway constrained to relate their own development/specification method/language to principle schemata, understood by the

national company signalling engineers.

On the other hand, the open, less protected markets have brought to the reorganization and merging of the previous national industries into a few multinational companies, which need now to merge different know-hows about interlocking production.

In an open market, greater reliance has to be put on precise contractual specifications to define the responsibility of each involved party; moreover, the multinational dimensions of railway industries asks for a standardization of the interlocking systems, and of their contractual specifications, in order to propose the same product to several companies Europe-wide. The Eurointerlocking project has been launched by a consortium of the main European railway companies, with the aim of developing a standard interlocking system at a European level, in order to decrease investment and maintenance costs for companies, thanks to standardization.

Inside this project a trend has developed towards the use of finite state machines for modelling interlocking rules [15]. In particular, a richer form of machines, namely the *Statecharts* [9,11], has been considered suitable to express the sequences of checks and actions typical of an interlocking system. Both the UML dialect [16] and the Statemate dialect [12] of Statecharts have been considered.

This paper follows this trend, describing an experience of modelling a railway interlocking system using Statecharts. Our study has addressed the problem from a "geographical", distributed, point of view, rather than a "functional", centralized, point of view: that is, our model is made up by models of single physical entities (points, signals, etc..) that collectively implement the interlocking rules, without any centralized database of rules, which is a typical way of approaching the problem, that we call "functional" approach.

The main aim of this study is to investigate the feasibility of the geographical approach, and to compare it to the functional one; one of the most interesting issues we are interested in evaluating, and which has actually motivated our work, is to verify the ability of the geographical approach to reduce revalidation efforts in the case of physical modifications to the yard: indeed often a small change of the physical configuration of the yard requires, in order to comply to the strict guidelines for safety, a complete new validation effort, e.g. by an extensive testing program, due to the changes done to a single central structure used by all the software modules. We show how the geographical approach may reduce this effort by allowing the re-validation only of those software modules that are actually affected by the changes.

2 Geographical and functional modelling

A RIS is an embedded system that ensures the safe operation of the devices in a railway station. Such a system controls an arrangement of signals and track points so interconnected that their functions shall succeed each other in proper sequence and for which interlocking rules are defined in order to guarantee safe operations. A simple (the simplest) example of RIS, that we will use in the paper as a case study, refers to the layout shown in the figure 1, and is based on a real Italian Interlocking System [4].

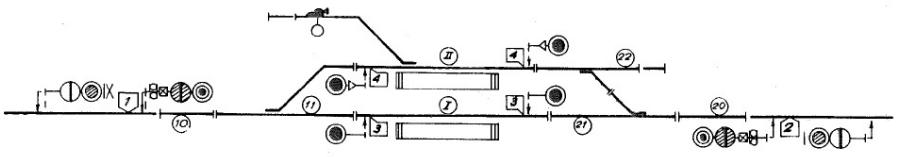

Fig. 1. The simplest railway yard layout.

Interlocking rules are obviously the core of the system, so their correctness is the main objective to be addressed by a formal specification. The rules aim at allowing only the safe combinations of switches positions, signals and trains in a station in order to avoid collisions. The signal indications, handled by the interlocking system, govern the correct use of the routes, authorizing the movement of train within interlocking limits.
The rules usually enforce a predefined sequence of actions: issuing a route request command usually first triggers a check that all the track elements involved in the route are free; in the case, commands are issued for the positioning of points for that route and for locking the track elements. This phase may be followed by a global centralized control over the correct state of the commanded elements, after which the route is locked and signal indications for the route are set.

2.1 A functional approach

Most of computer based interlocking systems use (in their implementation and/or formal specification) some form of centralized data base where the rules of the interlocking logic are stored. The main feature of this approach consists in generating the rules by adopting a design methodology focused on functions, such as the switch points checking function, the routes setting function or the routes verification [8]. This is why we call this approach "functional". These functions are designed basing on a "condition table" (*control table*), which

indicates all of the conditions that have to be satisfied before a signal can be switched from red to green to admit a train into the track section beyond [14]. The placement of the yard equipments is ignored in this design methodology, it does not exist a direct correlation between the geographical position on the yard of a device and the function that controls it implemented in the RIS. For this reason it is a hard task to identify the parts of the system stimulated by external events.

2.2 A geographical approach

A different approach can consider distributing the knowledge of the interlocking rules to objects modelling the geographical placement of physical elements.

An example of geographical specification is the EURIS language (EUropean Railway Interlocking Specification), which is a visual and graphical specification language for railway control systems [2,6]. Using this language it is possible to build in a component-based way the control system from the layout of the station to model. The EURIS specification language, proposed in 1992 and used at Siemens, is used to specify different railway yards using the same generical specification components. Actually it consists of a set of standardized railway control components [18], which should be composed together easily, increasing the speed of development and permitting the reuse of components. Each element includes a set of rules inside it, which are able to adjust to different layout configurations. Because EURIS is a graphical language another advantage is that specifications are easy to read, since they immediately represent the physical position of elements in the yard.

2.3 Revalidation

At first sight it could seem that the layout of a railway yard is fixed and cannot be changed. Actually, the layout can be changed during construction works to enable partial operation of the yard, or after for maintenance works, or for extensions. This may happen several times in the lifetime of an interlocking system, which anyway spans several decades. Computer based RIS have the obvious advantage over relay-based ones that any change can be addressed by a change in the software. However, the strict guidelines followed in the development of this piece of safety critical software require a costly validation activity: any change to the software requires a revalidation activity as well.
The adoption of methods and techniques that can reduce such efforts and costs is therefore an important industrial objective.
In this paper we study the impact that a geographical approach to the specification of a RIS can have on this particular aspect: the idea is that changing

a part of the layout affects only those software objects that are geographically close to the changed ones. Only the affected ones need therefore to be revalidated.

3 Statecharts

The Statecharts formalism [9] is an extension of the classic formalism of Finite State Machines (FSM), to allow hierarchical parallel interacting state machines to be specified. Transition from a state to another of a single machine (a *statechart*) is driven by trigger events, which can refer to the state of other machines or to global variables; therefore, the communication activity between statecharts is performed using broadcasting: every event is sent to the whole system, and can be received from any other part of the system. During a transition the actions generate events, which are triggered by conditions on other transitions or on global variables. Chains of internal events generated by only one external event are possible.

The hierarchy features of Statecharts permit to slice a system into well defined subsystems, so reducing the complexity, and permitting to build a structured model with concurrent parts. The system can be decomposed using AND-states, which evolve singularly and in a parallel way.

Two main dialects of Statecharts are actually used: UML (Unified Modelling Language) State Diagrams [16] and Statemate Statecharts [10].

3.1 The Statemate tool

In this paper, we follow the style of Statemate Statecharts. The I-Logix Statemate tool [17,13] supports the editing of the graphical Statecharts notation, but more importantly allows the complete system specification to be executed and graphically simulated, permitting to explore any scenarios determining the system correctness, and evaluating whether the specification meets the requirements.

The Statemate simulator allows to execute the model, permitting to verify the behavior examining the animation of the system. Furthermore it allows to animate panels to have an evidence of the model behavior, so that we can generate easily "what-if" scenarios. Statemate provides also the automatic generation of a C-based or Ada-based prototyping source code based on the model. Another important recent add-on to Statemate (which we have not used in this experience so far) is a powerful model checker, which is obviously an advantage in terms of the confidence that can be acquired on the specification correctness.

These features of Statemate make it currently superior w.r.t. UML-based tools which allows only editing of State Diagrams [7].

The Statemate tool supports also another sort of charts, useful to build a structured system: the *Activity Chart*. Activity charts can be viewed as multi-level (hierarchical) data-flow diagrams (DFD). An activity chart describes the functional decomposition of system's capabilities into functions, or activities, organized into hierarchies. This hierarchy details the functional components, or activities, that the system is capable of carrying out, and how these components communicate through information flow among them. The behavior of each activity is described using statecharts. In this study the activities are used to represent the distributed devices on the yard and therefore a controlling statechart is associated to each device.

These charts are more flexible than the statecharts since they can be activated and deactivated dynamically, which is not possible with statecharts. Each activity is a subsystem with a proper function, expressed in its turn by statecharts.

4 Statecharts geographical model

We suggest a way to design an interlocking system starting from its layout and ending in its operational specification. This approach is similar to the one used in EURIS language [2]. In this work we focus on a methodology that does not use any sort of global summarizing variables, which is usual instead for a functional approach.

With the term *summarizing variable* we mean a variable whose values depend from the values of a set of other single variables, each related to a physical entity of the layout. As an example we can consider a variable associated to a route, that is true if and only if at least one of the variables recording the occupancy of the track circuits belonging to the route is set to true. The use of summarizing variables, though useful for abstracting certain global aspects of the system, makes the model more distant from the physical topology.

The experience discussed in this paper has been actually preceded by a modelling by Statemate statecharts of the same interlocking system, using a classical functional approach: specific modules were dedicated to record commanded routes; these modules have been given the responsibility of checking the occupancy of the interested track circuits, following what prescribed by the *condition table* given as input. On the contrary, we will see how in our presented approach it is each module dedicated to the management of each track circuit that has the responsibility to check its compatibility with com-

manded routes. In the same way, all the activities performed by functional objects have been distributed to these geographical objects. Hence, the control for example of the correct position of a particular switch point is performed from all the interested elements and not by a single object dedicated to the management of all the switch points.

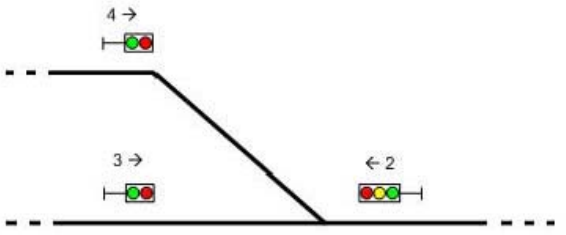

Fig. 2. An example of routing.

Each object of the model implements the rules that interest only that object. If, for example, we consider the three signals of Figure 2, the model will include an object for each semaphore. The object related to the semaphore 4, which permits the movement of a train in the right direction, should control that the red lights of semaphores *3 right* and *2 left* are fired and also that the green lights are switched off. This control is done not looking at a global summarizing variable which in the example would show that the route is free, but communicating with the objects that control the other semaphores and devices.

In this way, we obtain a model whose structure reflects the layout of the railway yard. This has positive effects on the readability of the model, and on the possibility of isolating in the model only those objects that are interested by a change in the physical layout.

On the other hand, we loose on generality: in the functional approach the module handling all switch points can be generic, and it is the condition table that embeds the knowledge about the specific rules for the railway yard. Our objects have not a generic behavior usable in any different geographical layout (like EURIS): we have to redesign them for any different station, though following expected patterns with predetermined rules.

The consequences of this approach are:

- The structure of the model should reflect the geographical topology of the yard.
- The elements of the model should replicate the behavior of the physical

components of the yard.

- The elements of the model should embed all the logical rules interesting the corresponding physical components (in order to avoid the usage of summarizing variables).

This kind of model is able to minimize the inter-dependency between objects.

Objects cannot be completely independent; in fact an interlocking system is by definition a system that have to control the interrelations between the objects. However, a geographical model maximizes independency: where there is independency in the yard, we want that it is reflected in the model.

4.1 Structure of the layered model

The model is built following a layered architecture for the RIS, which consists of three layers: *Command (human) layer*, *Logical layer* and *Physical layer* (Figure 3). The first layer (Command) is dedicated to the interaction with

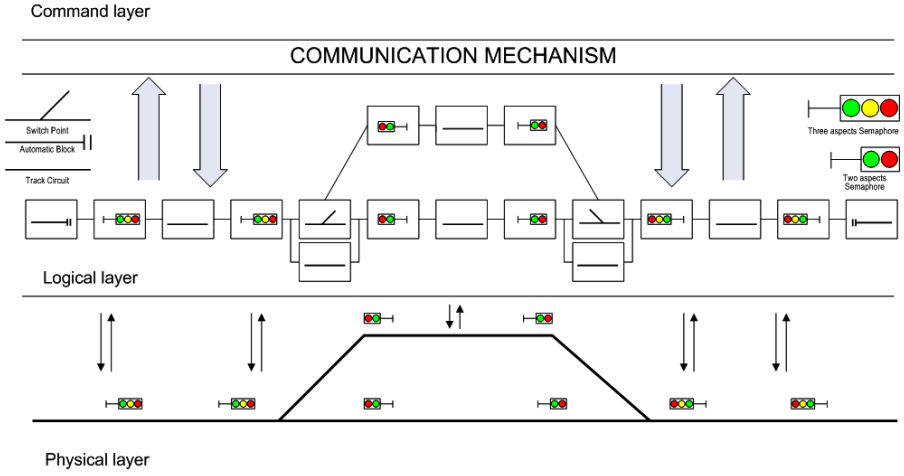

Fig. 3. Illustration of the layered interlocking architecture.

operators or other systems, which send commands to the RIS.

At the lowest level (Physical), there are the yard devices and equipments, which have to be commanded and controlled by the RIS. This level is constituted of the actual device interfaces, with actual variables used to control the yard.

The middle level is the core of the RIS, where the interlocking rules are specified. It is formed by a separate object for each physical device.

Figure 4 shows how the objects are interconnected with the command and the physical layer. The state of any object is one to one related with the actual state of physical device.

Every object related to a particular route is able to receive the command requesting that route, in which case it performs the proper checks about the physical layer and the other objects of the logical layer. When a route reservation command is sent by an external system (also human), this message is sent to all the objects related to that route. Then all the objects evaluate their rules interacting each other to confirm the received command.

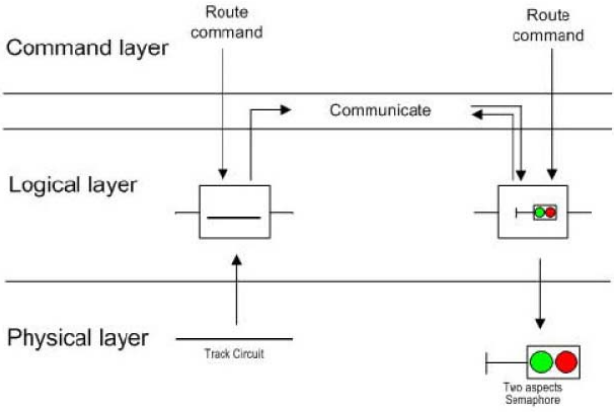

Fig. 4. Illustration of inter-object communication.

Inside each object it is therefore distributed the logic that is usually centralized in a classical functional approach: exists no coordinator object.

4.2 The statecharts model

The system is specified combining the geographical elements as it is illustrated in figure 3 and 4. Each geographical element is defined by an activity chart specified using the Statemate statechart formalism.

As shown in Figure 5, the main level of the model consists of an activity chart, composed by several activities, which are strictly related to physical objects placed on the yard. At a lower level each block is formed with a set of nested subactivities and statecharts that implement the interlocking rules. We can note that the topology of this level is exactly corresponding to the geographical layout of the yard (refer to Figure 1).

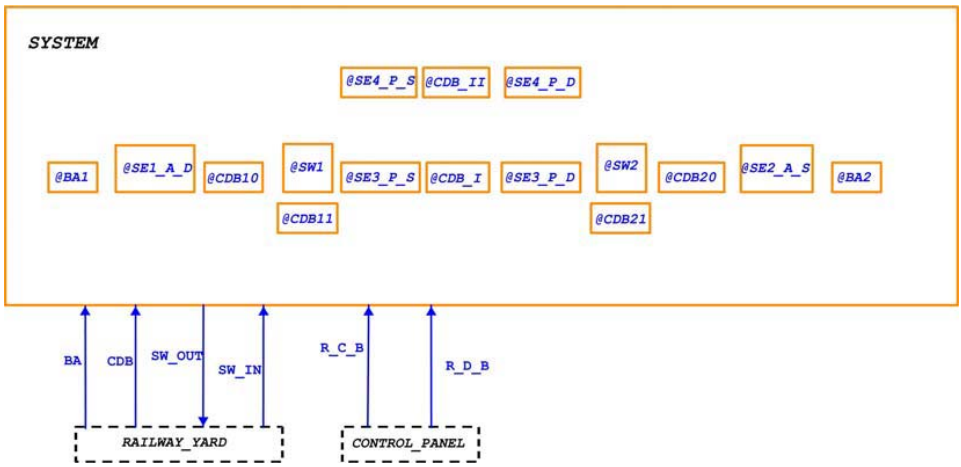

Fig. 5. The first level of the statechart model and the distribution of the internal activity charts.

The interaction between activities can occur by signalling an event (including variables value changes), either by direct addressing, i.e. the target is specified, or by broadcasting. In our model, every element (variables, events and so on) has a scope in which it is visible. The scope of an element can be explicitly defined by the user: every change in the value of an element is broadcast to all activities and statecharts in the element's scope, and is thus seen by all the charts within the scope.

Shared variables are therefore used to implement the communication between objects: every block checks which is the state of other nearby objects sniffing some of their state variables.

An example of the statechart describing the behavior of an activity (control activity chart) is shown in Figure 6 where a track circuit manager is illustrated: the figure shows the use of a logical state such as that used to reserve the object. These local states are needed because we do not have any global object that records which elements are in use, so the control logic has been decentralized. We can note that the chart communicates with plenty of other objects, such as semaphores and other distinct track circuits, by looking at shared variables. Indeed, it is evident how the interlocking rules are distributed over the conditions for the transitions in each activity chart.

Figure 7 represents the statechart controlling a green light: when an operator (either human or system) gives a command, this statechart and all the other statecharts controls that the track circuits related to it are reserved and the track circuits incompatible with it are free.

Also this example makes evident the large usage of shared variables. Though

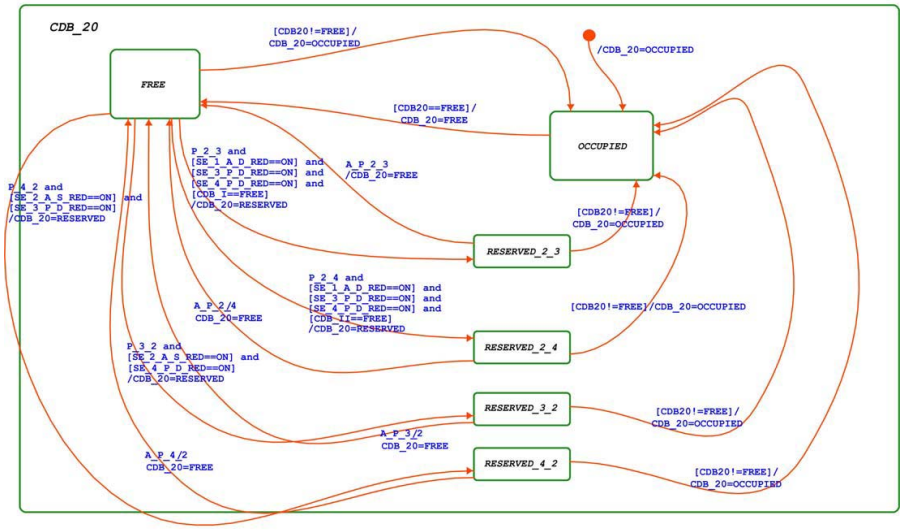

Fig. 6. A track circuit statechart.

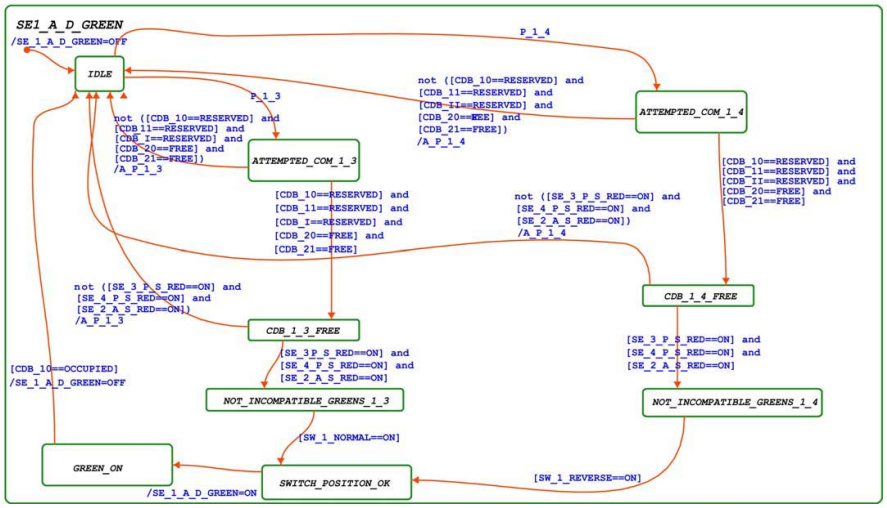

Fig. 7. The statechart controlling the green light of a semaphore.

each chart works in parallel with the others, they are strictly interrelated by this massive usage of shared variables.

5 A comparison between functional and geographical approach

At the end of the modelling activity we have been able to simulate the whole system with the Statemate simulator tool, that permits to interact directly by using a panel appropriately created as well (figure 8). We have used the same simulation scenarios used for the companion functional model we had built for the same RIS, establishing the relative conformance of the geographical and functional approaches.

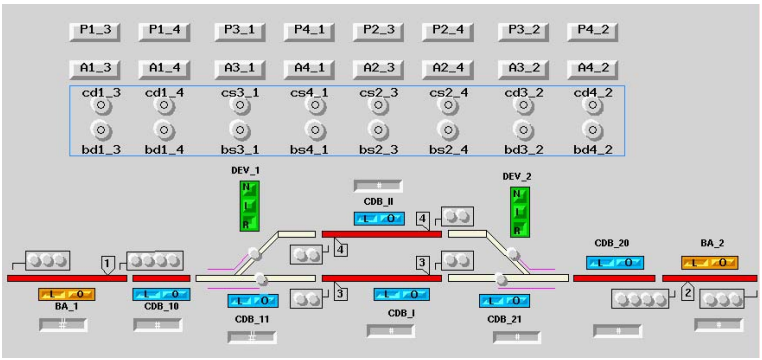

Fig. 8. The control panel.

From the statechart model it is also possible to generate C or ADA code by using the automatic code generator tool, which is part of the Statemate tool. Because of the detailed nature of the model, the code generated is immediately usable without need of any other translation into lower level languages. The resulting code shares with the geographical model the correspondence between software modules and yard devices.

Comparing the two specifications, the functional design approach results to be more compact than the geographical one, which requires more redundant controls, but also to be less readable, because of the melting of variables into global summarizing variables, which maintain information about objects not necessarily related each other.

On the other hand, the redundancy of the controls exhibited by the geographical approach can be considered as a positive safety measure: a decision about the establishing of a route is taken only if all the controls have been successful; the controls are redundant, but diverse, hence they constitute a safeguard against software faults.

5.1 The revalidation activity

Our main interest for the geographical approach lies in the possibility to reduce the revalidation effort needed when the yard is modified.

The main question is: "Do we have to revalidate the whole system? Or there is the possibility to understand what is actually changed, so to limit the revalidation effort?"

Actually, in the geographical approach, in case of a change to some parts of the interlocking system there may be some modules that do not need to be changed because they are not affected by the modification done, neither directly nor by propagation. Since each object performs its controls independently and the system does not use global summarizing variables, we can assert that those objects do not have to be revalidated. The objects affected by the modification need instead to be revalidated, e.g. using new test scenarios or modifying the already existing ones.

5.2 The interaction table

In order to select the objects that need or need not to be revalidated in face of a change, we should follow the interrelations between the objects, which propagate the change effects from the components directly associated with the physical devices affected by the change. These interactions between the objects are easily illustrated in an interrelation table, which is exemplified for the considered case study in Table 1.

Such a table shows the relation between routes (columns) and devices (rows). For each route there is a set of devices and equipments used to establish the route command and, viceversa, a device could be used by one or more routes.

The interrelation table is in general an useful tool that allows to relate an object to another. It is however useful only because we have implemented the model in the geographical way. If we define a similar interaction table for a functionally designed model, we will find out that there will be the 100% of correlation between objects, due to the dependence by a single central element (the condition table).

The interrelation table can be extracted from the geographical model: considering for example the chart in figure 7 we can immediately notice the interrelation between the semaphore SE1-A-D-GREEN (SE1-G in Table 1) and the routes 1-3 and 1-4, which in their turn are interrelated with a set of other objects such as CDB-10, CDB-11, etc.

	ROUTES							
DEVICES	**1-3**	**1-4**	**2-3**	**2-4**	**3-1**	**4-1**	**3-2**	**4-2**
CDB-10	X	X	X	X	X	X		
CDB-11	X	X	X	X	X	X		
CDB-20	X	X	X	X			X	X
CDB-21	X	X	X	X			X	X
CDB-I	X		X					
CDB-II		X		X				
BA-1	X	X			X	X		
BA-2			X	X			X	X
SW1-R		X				X		
SW1-N	X				X			
SW2-R				X				X
SW2-N			X				X	
SE1-G	X	X						
SE1-R	X	X	X	X	X			
SE2-G			X	X				
SE2-R	X	X	X	X			X	
SE3L-G					X			
SE3L-R	X	X			X	X		
SE3R-G							X	
SE3R-R			X	X			X	X
SE4L-G						X		
SE4L-R			X	X			X	X
SE4R-G								X
SE4R-R			X	X			X	X
CDB-IIbis		X		X				

Table 1
Interrelation table of the system.

Let us consider the following example of change made to the controlled rail yard: physical element CDB-II is split into two distinct elements named CDB-II and CDB-IIbis. The new element (CDB-IIbis) is added at the end of the table: following the routes related to the new added device, we can easily identify which are the devices interested by them (gray areas in Table 1).

As shown by Table 1, adding the CDB-IIbis element concerns the 1-4 and 2-4 routes, so we have to modify only those objects interested by these routes; in this example, 17 out of 24 devices: only the 70.83% of the system is interested by the modification: this indicates the potential for a 30% saving in the revalidation effort.

The advantages from this point of view are greater when the change affects only a marginal part of the layout, that is some device is actually used only by few routes.

About scalability of these considerations to larger designs, though we have not made further experiments, we can observe that in practice is not convenient to have in a railway yard a single device which is common to many routes, since this limits the capacity of the yard in term of the number of safe simultaneous set routes: independency among routes indeed sought as a primary objective in the design of the yard itself.

6 Conclusions

The experience we have presented is part of a wider research project aiming at investigating the design of RIS by means of different Statechart dialects and different commercial tools that support Statecharts, such as for example Stateflow (MathWorks), Telelogic TAU Generation 2 (Telelogic), Real-Time Studio (Artisan Software), visualSTATE (IAR Systems). We have concentrated in this paper on the possibility of adopting a geographical approach, with the main aim of investigating the reduction of revalidation effort in case of change. In this direction we need to verify the scalability to larger designs. Moreover, we need to assess the suitability of this approach to formal verification on one side (by means of model checkers such as the one integrated in Statemate, or such as other academic tools), and, on the other side, to automatic generation of the statechart model from a specification of the physical layout.

References

[1] S. Bacherini, S. Bianchi, L. Capecchi, A. Felleca, A. Fantechi, E. Spinicci. *Modelling a railway signalling system using SDL*, Symposium on Formal Methods for Railway Operation and Control Systems (FORMS 2003), Budapest, Hungary, 15-16 May 2003.

[2] J. Berger, P. Middelraad, and A.J. Smith. *EURIS, The European railway interlocking specification.* UIC, Commission 7A/16, 1992. In IRSE Proceedings 1992/93, pages 70–82, 1993.

[3] A. Cimatti, F. Giunchiglia, G. Mongardi, D. Romano, F. Torielli and P. Traverso, 1998, *Formal Verification of a Railway Interlocking System using Model Checking*, Formal Aspects of Computing, Vol 10, 361-380.

[4] P. E. Debarbieri, F. Valdambrini, E. Antonelli. *A.C.E.I. Telecomandati per linee a semplice binario, schemi I-0/19.* CIFI Collana di testi per la preparazione agli esami di abilitazione, Quaderno 12, 1987.

[5] European Committee for Electrotechnical Standardization, 2001, EN 50128, *Railway applications Communications, signaling and processing systems Software for railway control and protection systems.*

[6] W. J. Fokkink, P. R. Hollingshead, 1998, *Verification of Interlockings: from Control Tables to Ladder Logic Diagrams*, Proceedings of the 3rd Workshop on Formal Methods for Industrial Critical Systems - FMICS'98.

[7] U. Foschi, M. Giuliani, A. Morzenti, M. Pradella, P. San Pietro. *The role of formal methods in software procurement for the railway transportation industry*, Symposium on Formal Methods for Railway Operation and Control Systems (FORMS 2003), Budapest, Hungary, 15-16 May 2003.

[8] B. Fringuelli, E. Lamma, P. Mello, G. Santocchia. *Knowledge-Based Technology for Controlling Railway Stations*. In IEEE Intelligent Systems, Volume: 7, Issue: 6, Dec. 1992, Pages: 45-52.

[9] D. Harel. *Statecharts: A Visual Formalism for Complex Systems*. Sci. Comput. Programming 8 (1987), 231-274.

[10] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, A. Shtull-Trauring, M. Trakhtenbrot, *STATEMATE: A Working Environment for the Development of Complex Reactive Systems*, IEEE Transactions on Software Engineering, Vol. 16, N. 4, April 1990, pp. 403-414.

[11] D. Harel, A. Pnueli, J. Schmidt and R. Sherman. *On the Formal Semantics of Statecharts*, Proc. 2nd IEEE Symp. on Logic in Computer Science, Ithaca, NY, 1987, pp. 54-64.

[12] D. Harel and M. Politi, *Modeling Reactive Systems with Statecharts: The STATEMATE Approach*, McGraw-Hill, 1998. (Early version titled: The Languages of STATEMATE, I-Logix, Inc., Andover, MA, 1991.)

[13] J. Klose, W. Damm, 2001, *Verification of a Radio-Based Signaling System Using the STATEMATE Verification Environment* , Formal Methods in System Design, 19(2).

[14] G. Kolk. *Formal methods: Possibilities and difficulties in a railway environment from a user perspective*. In Proceedings of the Third International Workshop on Formal Methods for Industrial Critical Systems, May 25-26, 1998, 1998.

[15] Niklaus H. Köenig, Stefan Einer *The Euro-Interlocking Formalized Functional Requirements Approach (EIFFRA)*, Symposium on Formal Methods for Railway Operation and Control Systems (FORMS 2003), Budapest, Hungary, 15-16 May 2003.

[16] Object Management Group, 1999, *Unified Modeling Language Specification, Version 1.5* http://www.omg.org/technology/documents/formal/uml.htm

[17] *Statemate Magnum Simulation Reference Manual*. I-Logix Inc. Burlington, MA USA, 2003.

[18] Fokko van Dijk, Wan Fokkink, Gea Kolk, Paul van de Ven and Bas van Vlijmen, 1998 *EURIS, a Specification Method for Distributed Interlockings*, Lecture Notes in Computer Science, vol. 1516.