# Some experiences on Formal specification of Railway Interlocking Systems using Statecharts

Michele Banci
ISTI - CNR,
Formal Methods and Tools Group
Pisa, Italy
michele.banci@isti.cnr.it

Alessandro Fantechi
Università degli Studi di Firenze,
Dipartimento di Sistemi e Informatica
Firenze, Italy
fantechi@dsi.unifi.it

Stefania Gnesi
ISTI - CNR,
Formal Methods and Tools Group
Pisa, Italy
stefania.gnesi@isti.cnr.it

## Abstract

*The introduction on the market of computerized Railway Interlocking Systems has pushed 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 and equipments for trains through a railway yard. Recently, a trend has emerged about the use of the graphical language statecharts as a standard formalism to produce precise specifications of these systems.*
*This paper resumes our experiences in modeling railway interlocking systems using this formal language. Our studies have addressed the design problem from different points of view: we have modeled the specifications using two different approaches that we call functional description and geographical description. The names indicate that the first approach is focused on the modeling of the logical function of the Interlocking Systems, while the latter focuses on the geographical distributions of the element of the controlled station or yard.*
*The geographical approach has also inspired a proposal for a distributed interlocking system, where the logical rules are embedded into distributed equipments communicating by means of safe buses.*

## 1 Introduction

The case for Formal Methods in the development of RISs (Railway Interlocking Systems) has stimulated a considerable literature about formalization of interlocking systems (see, for example, [4, 6, 19]). And indeed European guidelines and standardization groups recommend the use of formal methods in the development of safety critical computer-controlled systems for railways applications [9, 10, 11].
Some European railway companies have constituted a consortium to define a standard interlocking system at a European level: the Eurointerlocking project. Inside this project a trend has developed towards the use of statecharts for modeling interlocking rules [16]. Statecharts have been considered suitable to express the sequences of checks and actions typical of an interlocking system.
In this paper we resume our experiences in modeling RISs using statecharts, following two different approaches to the formalization of Interlocking we have investigated: the functional approach and the geographical one.
The functional approach, discussed in section 3 faces the design of a RIS using logical objects not related to physical objects to the layout. Examples of logical objects are the reservation and the setting of a safe route through the controlled yard. On the other side a switch (or *point*) is a physical yard object, so defining a logical object associated to a switch can be considered as a geographical association. Inside the functional approach, we have in particular addressed the aspect of developing the statechart description of an Interlocking System for a given station or yard by instantiating generic statecharts, referring to general signalling principles, on the parameters coming from the actual layout of the controlled yard (see section 4).
The same problem has then been addressed in section 5 by proposing a geographical style of formal description, in which the model is obtained by the composition of objects

modeling the physical entities, following the geography of the controlled yard. We have then studied the impact of this style on the possibility of incremental validation.
Furthermore, the same approach has then inspired a proposal for developing a distributed RIS (see section 6), whose safety validation is strongly based on formal verification and simulation of the formal description.

## 2 Introducing formal specification in computer-based interlocking

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 operators failures;
- 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. 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 [12]. 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.

The main activity of the RIS, which is an embedded system, is to ensure the safe operation of the devices in a railway station. Such a system controls an arrangement of equipments (signals, switches, track circuits, automatic blocks) so interconnected that their functions shall be performed 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 have used in our experiences as a case study, refers to the layout shown in the figure 1, and is based on a real Italian Interlocking System [7]. Line segments represent track segments in the station; some of them have track circuits, that is, sensors of the presence of a train, which are numbered inside circles, joints between segments represent switches. Lollypop-like drawings represent signals of various type. Numbered labels are attached to each important part of a route. This example (a single track line station) is constituted by eight allowed routes, two switches, eight signals, six track circuits and two automatic blocks.
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 trains. 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 switches 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. A route

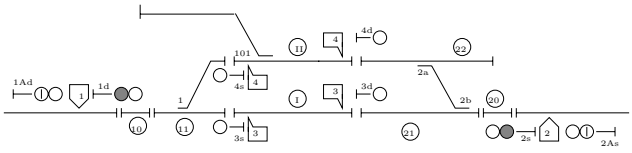



**Figure 1. The simplest railway yard layout.**

can be set free only if all switches on the route are being in the correct position, and no train is present. The signals can be set to green only if the route in front is set to free. These sentences express two examples of generic principles that hold for every interlocking systems. Actually, the precise and complete set of such rules depends on the kind of station or railway yard, and also on national policies traditionally established by railway companies or regulatory boards.

Obviously, a RIS being a safety-critical system, the formalization of such rules is a top requirement, and this is the reason of the wide interest generated by RISs in the formal method community. Note however that the generic rules expressed above need to be conjugated on every specific layout; for instance, the rules should be set for the route 1-3 in figure 1 taking into account switch 1, track circuit 10, 11, I, and so on.
In the traditional process adopted by many railway companies to develop relay based interlocking system since several decades, the generic principles were encoded into relay circuit templates.
When a new interlocking plant had to be installed these general principles had only to be adapted to the particular layout of the station in use. The adaptation process was also guided by some more or less formalized rules. At the end of the process there was a diagram containing the command and control circuits for each logical or physical object of the station.

An example of this kind of diagram is shown in figure 2; this diagram, taken from an Italian interlocking, represents a circuit, for the route 1-3, which is generated basing on templates supplied to help the engineers in the design of new stations. In this example, the permission to set the route 1-3 as free is given by the relay CD1_3, which is energized following the shown combination of various (normally-open and normally-closed) contacts activated by other relays, where command rules are given in other schemata. The template will be associated to layout objects and replicated for each object. In this circuit template there are all the contacts needed to manage that kind of object; the only action to perform on it is the substitution of these generic contacts with the ones dictated by the layout. The structure of all diagrams related to routes is always the same but the number and the name of some contacts (serial or parallel) is different from a route to another.
In the example (figure 2) the suffix of the variables are generated using the layout data, but furthermore the contacts (in this route there is only one) M1N, which are related to the switches used by the route, could be more then one (in series): this series of contacts means that in order to enable this route all the switches used by it have to be in the correct position.
The safety of old times, relay based, interlocking systems was based on single fail-safe concepts exploiting the intrinsic characteristics of relay technology. The introduction of computer in the control and command chain has subverted this approach to safety, since failure modes of computer controlled equipment may be much more diverse and difficult to predict.
The first approach followed by some railway companies was to maintain the traditional and well-established relay-based principle diagram as the trusted source of information for computer-based interlocking developers, looking for conformance of the installed systems to such source, by means of costly and tedious, but possibly not exhaustive, testing.

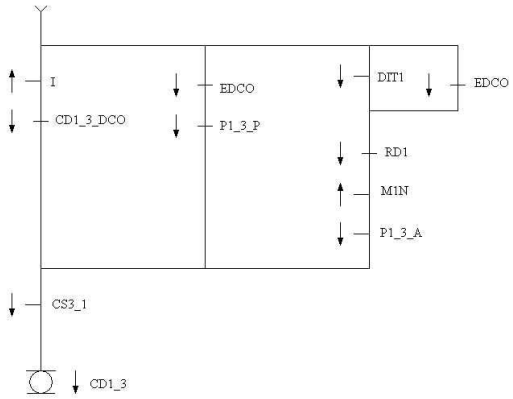



**Figure 2. An instantiated relay schema referred to route 1-3**

This approach has put on manufacturers' shoulders the burden of conceiving a family of interlocking products, together with means to instantiate the generic product by taking into account some proprietary, formalized version of principle diagrams suitable to be (more or less automatically) interpreted or compiled into running code, that has to be shown conformant to the trusted source. Actual approaches have varied from manufacturer to manufacturer (see [5, 12]).
But computer technology permits more flexibility: new features are very interesting but require an innovative approach and a complete formalization of the whole system [9]. Domain specific languages have been proposed for the formalization of RISs [14], the most prominent one being EURIS [4, 8].
More recently, the possibility of using commercial support tools has pushed the use of "general-purpose" languages [11], and a recent trend has indicated Statecharts as a means for defining a standard formalization [16]. In our experiences described in the following, we have adopted the Statemate statecharts style [20].

## 3 Functional approach to RIS specification

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 ap-

proach 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 [12]. This is why we call this approach functional. These functions are designed basing on a *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 [15]. 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.
A specification or implementation following this approach is a melting pot of functionalities not easy to separate. The geographic information of the yard is not available any more after the RIS functional description. An example of functional approach, implemented by relay technology, is shown in figure 2. We can define by statecharts the behavior of these relay circuits. This kind of description is very apretiated by railway company in fact relay technology is well consolidated and then safety engineers are more confident on this kind of technology instead a different one where relay schema are not direct referred.
This approach is similar to a translation of relay circuits to statecharts. It may be a starting point to introduce new formalisms into specification activity and then pass to more powerful kind of models.

The yard shown in figure 1 has been modeled using functional approach (both related to relay schemata and not), specific modules have been 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 *control table* given as input.

# 4 Automatic instantiation of statecharts

A problem that arises in the practical application of such formalization is that each produced interlocking system is dependent on the physical layout of the controlled yard. This has strong effects on development costs and especially on validation, which has to be repeated for each product.

To face the problem of redesign of the whole system for a new station, we have adopted an approach, in which generic principles of the logic of single entities are first modeled and validated.
Basing on data of the modeled station or yard, the generic statecharts are then instantiated, that is, replicated for all layout entities and expanded to cover the proper logic conditions.
The instantiation rules have been then coded into a tool, capable of taking as input a generic statechart and a description of a station layout, and to produce a statechart model of the interlocking system targeted to the considered layout.

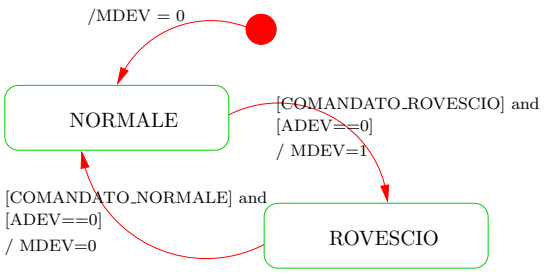
## 4.1 Generic charts

The idea of using generic specifications permits to create the whole specification in a compositional way, simply parameterizing control rules with respect to the layout and then instantiating them in relation to the station layout.

We actually use statecharts in a "generic way" to reach a final specification supplied by instantiated statecharts. This "generic way" includes two definitions:

- Definition of generic single statechart templates describing the generic interlocking principles
- Definition of the instantiation process

The generic statecharts that we have used are constituted by a subset of the Statecharts language, in particular they are constituted by the following entities: states, transitions, variables, events and parameters. Figure 3 shows an example of a generic statechart for a switch. We have defined




"NORMALE" = Normal position;
"ROVESCIO" = Reverse position;
"COMANDATO_NORMALE" = The switch is required (by a route setting) to move to normal position;
"COMANDATO_ROVESCIO" = The switch is required (by a route setting) to move to reverse position;
"ADEV" = Global variable saying whether the switch motor is on;
"MDEV" = Commanding the switch to a position (0,1).

**Figure 3. A generic statechart**

similar (more or less complex) generic statecharts for all physical entities constituting a railway yard layout, such as: switches, signals, track circuits and for all logical entities, such as a route, a starting point and so on. An example of parametrization is shown in figure 3: looking at the labels

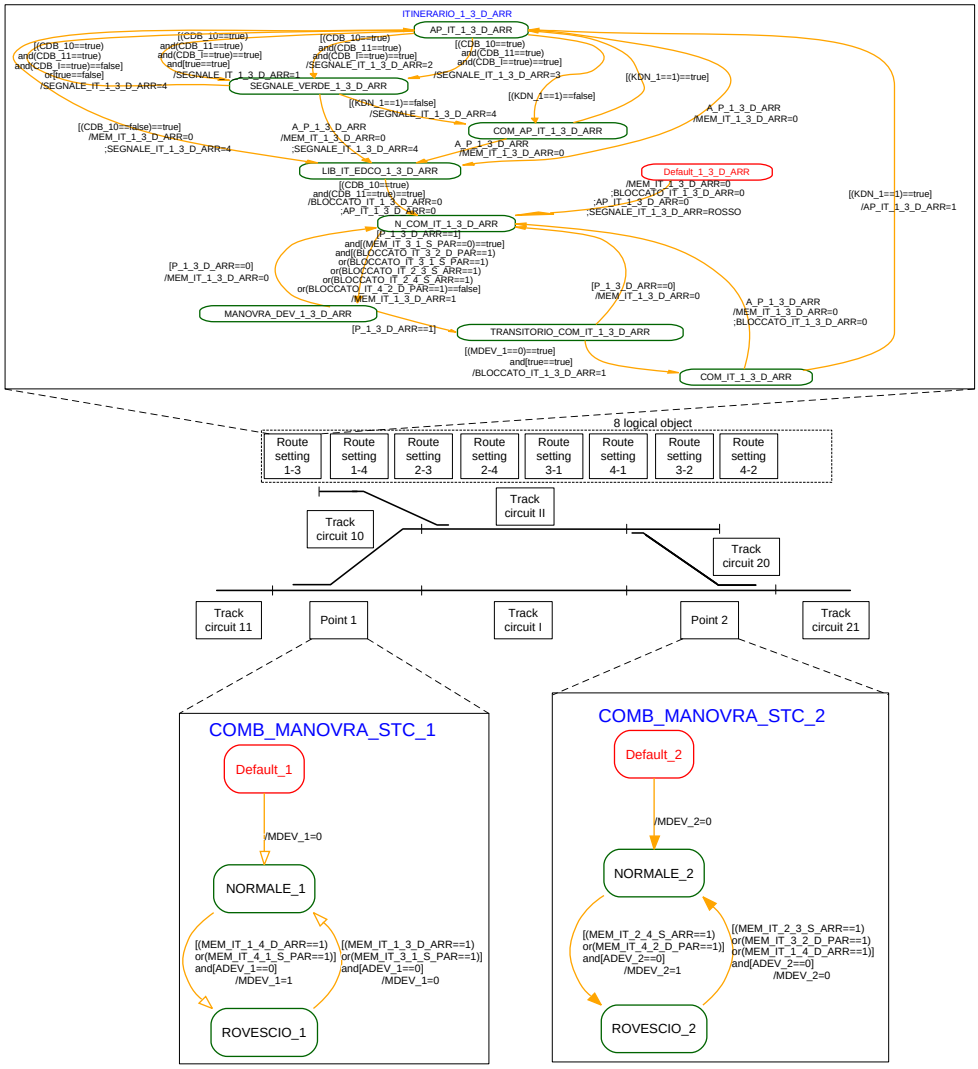



**Figure 4. The instantiated model as a collection of statecharts**

on the transitions and in the states, it is evident that no references to particular switches of the layout is done. The association between this generic chart and the final instantiated chart needs to be done basing on the layout information.
The instantiation is guided by some rule of instantiation and parameters association.

## 4.2 Instantiation methodology

The instantiation methodology consists in first replicating each generic chart for each physical or logical entity of the layout, and then in adding, in each replicated chart, a suffix to the name of charts, states and variables, suffix related to the particular physical object which is referred. The instantiation process is guided by the *control table* (*conditions table*), which is given as input. This table collects all the instantiated conditions (not abstracted) that have to be satisfied by the specific interlocking installation. The control table is a tool to provide abstract signalling rules (e.g. before a specific signal can be switched from red to green to admit a train into the track section beyond) but associated (instantiated) to specific yard layout (see Table 1 for layout in figure 1). In general it is the control table that embeds the knowledge about the specific control rules for the railway yard.
Figure 4 shows some objects that are instantiated for the layout of Figure 1, according to the control table of the example. Two statecharts related, therefore instantiated, to the two switches are shown in detail. To perform the command and control of a particular logical or physical station object,

| ROUTE | ROUTE CONV | NORMAL | REVERSE | TRACK C. | PT CDB |
|---|---|---|---|---|---|
| 1-3 | 2-4, 4-2 | 1 | | 11, I | 10 |
| 1-4 | 2-3 | 101 | 1 | 11, II | 10 |
| 3-1 | 2-4 | 1 | | 11 | 10 |
| 4-1 | 2-3 | 101 | 1 | 11 | 10 |
| 3-2 | | 2 | | 21 | 20 |
| 4-2 | 1-3 | 101 | 2 | 21,22 | 20 |
| 2-3 | 1-4, 4-1 | 2 | | 21, I | 20 |
| 2-4 | 1-3, 3-1 | 101 | 2 | 21, 22, II | 20 |

**Table 1. An example of Control Table**

such as a switch, we may use generic charts as templates without modifying states and transitions but simply substituting some parameters and names of variables and states. Therefore the generic chart will be used as a template in which the steps performed by the controller are coded, but not yet usable on a real station because lack of information deriving from layout. Information about the layout will be inserted using the parameters. Parameters constitute another dimension of genericity of generic statecharts, that allow the expressions on the transitions to be modified in relation to the specific layout. More detailed information about the instantiation process can be found in [2].

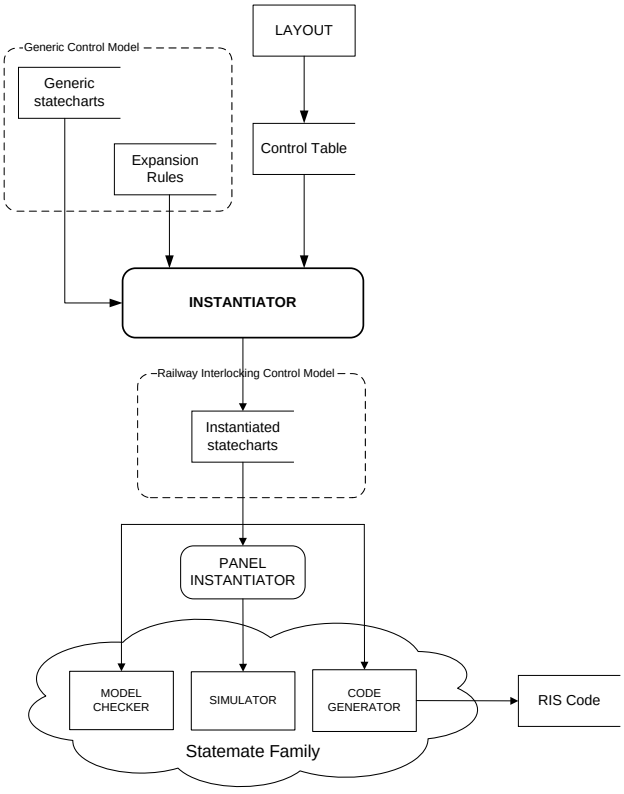



**Figure 5. The lifecycle**

In figure 5 the flow of information that is managed by our approach is shown: at first place there are obviously data related to the specific yard. This information is supplied, using the so called control table, in which there are specified relations between both logical and physical objects belonging to the specific interlocking system. The other input information is given by the generic model, that is a set of generic statecharts.

The output is a complete model composed by all needed statecharts. This model can be directly simulated by reimporting it into the Statemate tool. After this reimport of the complete model all the features of Statemate may be used. At first we are able to simulate the model in order to see the behaviour, interactively stimulating the model, but the most interesting option is the use of the model checker to perform a complete verification and validation of it. The final stage of the lifecycle is the automatic generation of code which can be directly used for running in a control unit.
In this case we have a so called closed lifecycle which, using an initial effort in designing and validating the generic model and another effort at the end to verify the instantiated model, permits to reduce human error.

## 5 The approach of geographical specification

### 5.1 A geographical approach

A different approach can consider distributing the knowledge of the interlocking rules to objects modeling the geographical placement of physical elements, in order to develop the specification of a particular interlocking product by simply composing objects, following the geography of the controlled yard. 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 [4, 8]. 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 com-

ponents. Actually it consists of a set of standardized railway control components [8], 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.

The EURIS language inspired our study on this direction but differently to it we have not used a domain specific language but a general purpose.

### 5.2 Statecharts geographical model

We have suggested a way to design an interlocking system starting from its layout and ending in its operational specification. In the work we focused 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, as has been shown in section 3 about the functional approach. 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 geographical experience, differently that the previous analysis of functional approach, shows how each module is dedicated to the management of each track circuit that has the responsibility to check its compatibility with commanded 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. Each object of the model implements the rules that interest only that object. If, for example, we consider the three signals of Figure 6, 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.

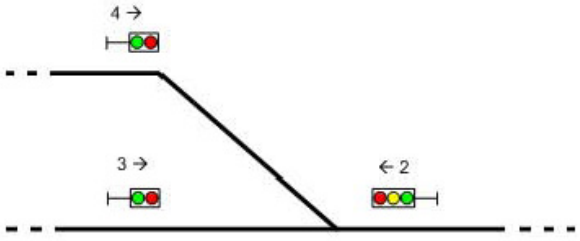



**Figure 6. An example of routing.**

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 control table that embeds the knowledge about the specific rules for the railway yard (see section 4). 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 interdependency 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.

#### 5.2.1 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 7). The first

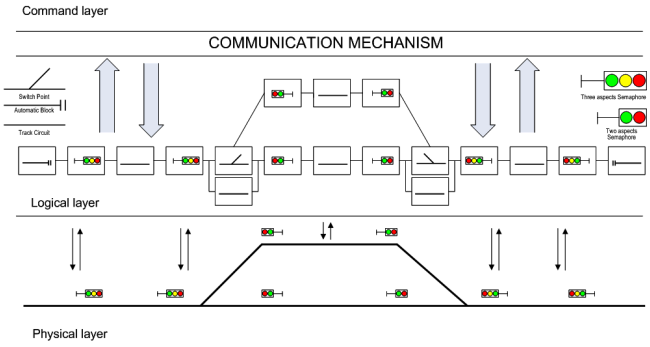



**Figure 7. Illustration of the layered interlocking architecture.**

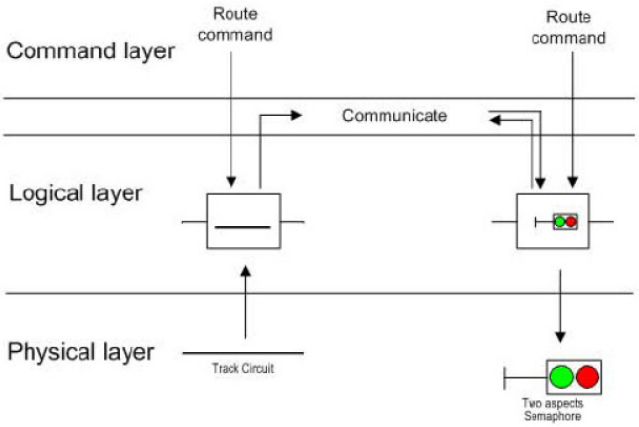



**Figure 8. Illustration of inter-object communication.**

layer (Command) is dedicated to the interaction with 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 8 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.

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

### 5.2.2 The statecharts model

The system is specified combining the geographical elements as it is illustrated in figure 7 and 8. Each geographical element is defined by an activity chart specified using the Statemate statechart formalism.

As shown in Figure 9, the top 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).

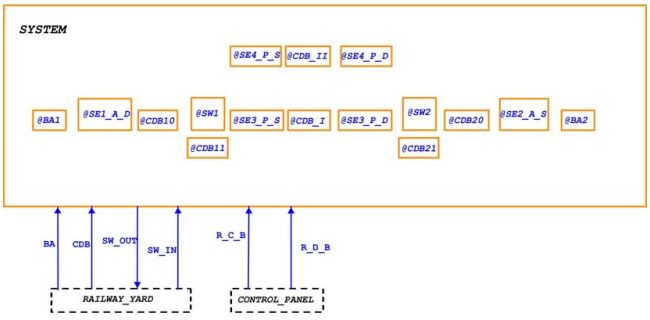



**Figure 9. The first level of the statechart model and the distribution of the internal activity charts.**

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 10 where a green light 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 track circuits 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 10 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 each chart works in parallel with the others, they are strictly interrelated by this massive usage of shared variables.

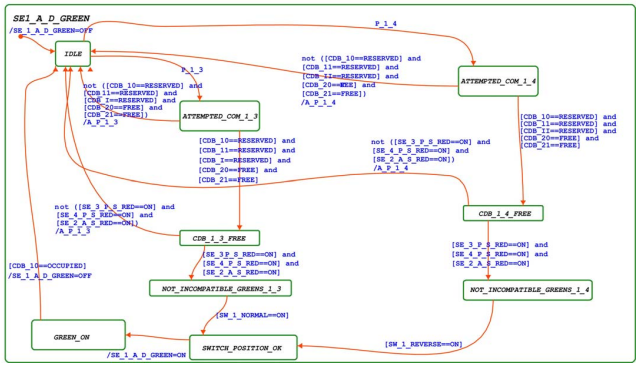



**Figure 10. The statechart controlling the green light of a semaphore.**

## 5.3 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.

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?"

The geographical approach appears interesting in the reduction of revalidation effort as well. In fact we can assume that software modification related to a geographical object affects only a closed set of other objects, so reducing the needed revalidation effort of the whole system.
Using 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. Since each object performs its controls independently and the system does not use global summarizing variables, we can think 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. More detailed information about the reduction of revalidation effort using a geographical approach can be found in [1].

# 6 A proposal for a Distributed Interlocking System (DRIS)

## 6.1 The Distributed System Architecture

We have observed that the geographic approach used to model the RIS keeps the original topology of the system, and this fact has inspired our proposal to physically distribute the control by deploying each activity in a separate controller embedded to the controlled entity [3]. The distribution consists in generating slices of the geographic model, described in section 5, as is illustrated by the Figure 11. An obstacle to physical code distribution is however represented by the shared variables used in the model for communication between the separate activities. The semantics of Statecharts requires that every activity is able to read and write shared variables at any step. In a distributed implementation, variables need either to be associated to an activity, which should provide for safe reading and writing by other activities, or to be replicated among the interested activities, and in this case consistency of the replicas should be guaranteed.

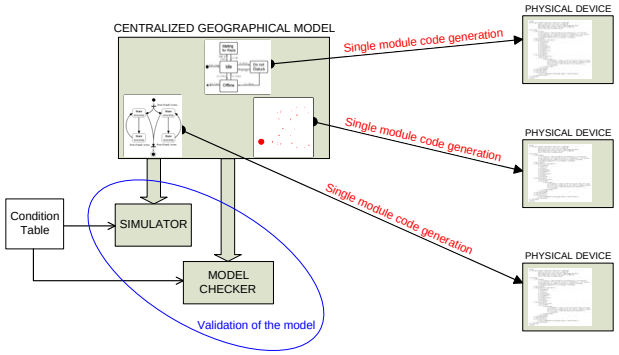



**Figure 11. Development cycle.**

The synchronous nature of the operation of Statecharts considers variables values to be read at the beginning of a step. Only when the evaluation of variables has dictated the live transitions that can be fired, one of them is fired and the associated actions, including writing on variables, are performed. It is possible to perform automatically checks that guarantee that no conflict is raised about simultaneously writing of a variable by two activities, in order to avoid race conditions.
This operational semantics allows to consider a distributed implementation based on the adoption of a field bus, by which variable values are broadcasted at the beginning of a new operation step, and by which writing commands issued by (one and only one) activity are conveyed to the owner of the variable at the end of the step.
Indeed, our idea is based on the rapid development of safe field bus area: we think that the market is now mature to accept this kind of approach in the railway area as well, given the large number of applications of field buses in different safety-concerned industrial areas: from factory automation to fly-by-wire and drive-by-wire, in avionics and automotive areas respectively.

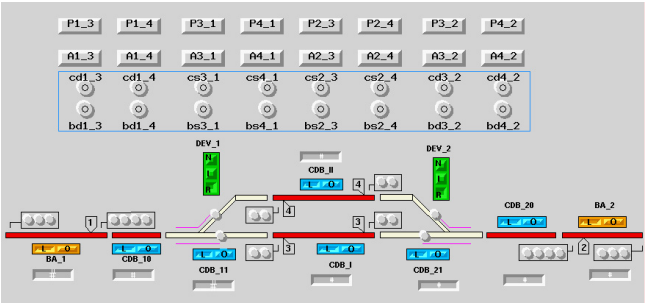



**Figure 12. The control panel.**

Due to the synchronous operation typical of Statecharts, a good candidate to act as the basic platform on which our approach is based is the architecture named TTA (Time Triggered Architecture) [17].

This architecture has been created for the implementation of dependable distributed embedded systems, and permits to decompose a large real-time application into nodes: obviously the main critical feature consists of the communication mechanism and the synchronization one. In the TTA, the system maintains a fault tolerant global time at every node. This global time permits to reduce the communication complexity allowing the use of shared variables for communication purposes; events that happens in the distributed system at different nodes at the same global clock-tick have to be considered simultaneous. The TTP (Time Triggered Protocol) is in charge of guaranteeing the consistency of different views of the same variable at any given clock tick.

Another issue that should be taken into account is given by the safety requirements in case of a fault. In our proposals, faults can occur in any distributed controller. The basic safety requirements, to be achieved both by exploiting the fault tolerant features of the bus protocol, and by properly designing the distributed components, are that:

- any failure of a component is reduced to a crash of the component itself, so that a failsilent policy is enforced;
- the presence of a silent component does not undermine the safe operation of the interlocking; that is, no route for which the failed component is needed can be set and acknowledged.

For increased availability, we can add the requirement:

- any failed (fail-silent) component does not affect the correct setting of a route which is totally (geographically) independent from the component interested by the failure.

Note that the distributed system is formed from redundant controls, which are located at every devices (Figure 13). The redundancy of the controls exhibited by the geographic 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 and independent, hence they constitute a safeguard against software faults. Note that this safety measure is not due to the exploitation of the TTA architecture (or of any other suitably fault tolerant bus), but is intrinsic in the model.
The system architecture is completed by a monitoring computer (or more than one computer) attached to the bus, able

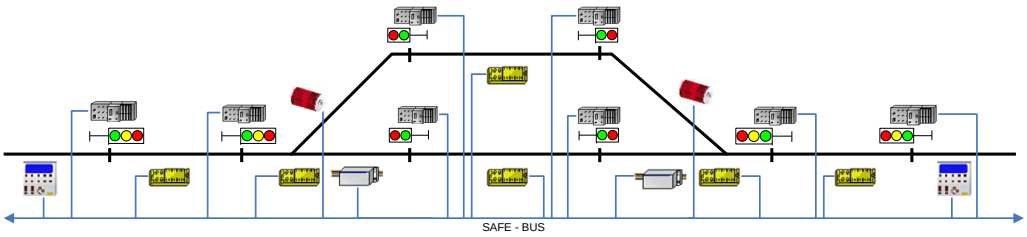



**Figure 13. The network devices deployment.**

to read the variables values that are exchanged on the field bus, which uses such data for diagnostics, for displaying to the humans the state of the yard and of the interlocking system, and for logging data about the system. The monitoring computer can also be used as an added safety measures by taking in charge the forcing of the system in a safe state in case it detects anomalous variable values.

### 6.2 Development cycle

What is needed to implement a distributed RIS can be summarized in the following development cycle, which includes validation activities as well, and is fully supported by the functionalities of the Ilogix Statemate tool:

1. Control table:
   This table is taken as the contractual input for the process, and fully describes the interlocking rules according to the given yard topology.

2. Statechart design using geographic approach:
   As described in section 5, a geographic model using activity and statecharts is developed. Furthermore, in order to verify the correctness of the geographical model a functional model, using a parallel design, should be developed as well.

3. Validation of distributed design:
   The geographic model is validated by means of two different alternative methods (which actually should be both applied for increased safety):

   (a) tests played by simulation:
   We are able to simulate the whole model with the Statemate simulator tool, that permits to interact directly by using a panel appropriately created as well (Figure 12). Extensive tests should be carried on defining suitable test scenarios, on the basis of the information given by the control table.
   An interesting way to achieve even more confidence is parallel co-simulation of the same scenarios applied both to the geographical model and to the functional model; this also because functional modeling is more consolidated in railway companies' practice.

   (b) Control table based safety properties proved by model checking:
   Safety properties described by the control table should be defined so that they cover the overall safety requirements. Typically, properties of the kind: "two conflicting routes can never be set simultaneously" should be expressed and verified by the Statemate model checker. Model checking is able to guarantee that such properties are always satisfied by the model, while simulation may leave some dangerous execution paths unexploited.

4. Fault injection:
   Extensive verification, again by simulation and/or model checking, should be done in order to validate the fail-safe behaviour of the model. Typically, faults should be injected in the model (e.g. by forcing a fail silent behaviour of some objects) in order to test the overall safety of the system in presence of faults.

5. Automatic code generation

   (a) Statemate code generation:
   From the statechart geographic centralized 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, for every single device. Because of the detailed nature of the model, the code generated is immediately usable without need of any other translation into lower level languages, except for the communication interface which is not part of the requirements specification. The resulting code shares with the geographic model the correspondence between software modules and yard devices. For this reason there is the possibility to generate code for each module, to be targeted and deployed on a local controller.

(b) Shared variables implemented through field bus protocol: The variables which are shared between the obtained software components should instead be implemented basing on the safe protocol established to this purpose over the adopted field bus.

6. Physical deployment and integration:
   Deployment of the various modules over the distributed controllers connected to the field bus is now possible.

7. System in field testing:
   Though the extensive validation effort carried on the model and the automatic generation of code are enough to guarantee the safety of the system, in field testing is necessary to guarantee that any possible interference from the physical world does not undermine the safety and functionality of the system.

The parallel co-simulation of the geographical and functional models is aimed to achieve a sufficient confidence on the absence of errors in the final DRIS. Even more convincing would be a proof of equivalence of the two models. This is a challenging task, due to the current unavailability of tools, and also to the inherent complexity of this approach, which is left as a subject for our future work.

## 7 Conclusions

This paper resumes our experiences in modeling railway interlocking systems using this formal language. Our studies have addressed the design problem from different points of view: we have modeled the specifications using two different approaches that we call functional description and geographical description.

All the experiences have been applied to some simple Italian interlocking examples. Instead, the instantiation process was inspired by a French experience [18] carried out by SNCF-RFF inside the EuroInterlocking project. To support this process a specific tool has been developed to instantiate generic charts into specific models, taking into account information about the station layout.
In this direction we need to verify the scalability to larger designs. In particular, we need to assess the suitability of the generated statecharts (which may become quite complex for a large station) to formal verification on one side (by means of model checkers such as the one integrated in Statemate, or such as other commercial or academic tools), and, on the other side, to automatic code generation.

Even more challenging is the experience made on developing a geographical approach aimed at developing a distributed RIS. In this experience we have pushed to the point that the interlocking logic can be entirely distributed on ”in the field” local controllers, following a trend consolidated in automotive and avionics applications, based on the use of robust field-buses. We have argued that in this approach formal verification gets the role of the primary method to asses the safety of the system. All the steps of the proposed developed cycle need to be carefully experimented and assessed on real designs.

## References


[1] M. Banci, A. Fantechi, Geographical vs. Functional Modeling by Statecharts of Interlocking Systems. FMICS Ninth International Workshop on Formal Methods for Industrial Critical Systems, Linz, September 20-21, 2004. Electronic Notes in Computer Science (Elsevier).

[2] M. Banci, A. Fantechi, Instantiating Generic Charts for Railway Interlocking Systems. FMICS 05, International Workshop on Formal Methods for Industrial Critical Systems, Lisbon, September 5-6, 2005.

[3] M. Banci, A. Fantechi, S. Gnesi, The role of Formal methods in developing a distributed railway interlocking system. FORMS/FORMAT 2004, Braunschweig, December 2-3, 2004.

[4] 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 7082, 1993

[5] C. Bernardeschi, A. Fantechi, S. Gnesi, S. Larosa, G. Mongardi and D. Romano. A Formal Verification Environment for Railway Signaling System Design, Formal Methods in System Design, Vol. 12, 2, pp. 139-161, 1998.

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

[7] P. E. Debarbieri, F. Valdambrini and E. Antonelli. A.C.E.I. Telecomandati per linee a semplice binario, schemi I0/19. CIFI Collana di testi per la preparazione agli esami di abilitazione, Quaderno 12, 1987.

[8] F. J. van Dijk, W. J. Fokkink, G. P. Kolk, P. H. J. van de Ven and S. F. M. van Vlijmen, EURIS, a specification method for distributed interlockings, in (W. Ehrenberger, ed) Proc. 17th Conference on Computer Safety, Reliability and Security - SAFECOMP’98, Heidelberg, Lecture Notes in Computer Science 1516, pp. 296-305, Springer, October 1998.

[9] L. H. Eriksson, G. Finnie, I. Herrtua and N. König; Formal Methods Strategy Study Report, Report of a study carried out on behalf of the International Union of Railways (UIC), project EURO-INTERLOCKING, Zürich 2000.

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

[11] U. Foschi, M. Giuliani, A. Morzenti, M. Pradella and 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.

[12] B. Fringuelli, E. Lamma, P. Mello and G. Santocchia. Knowledge-Based Technology for Controlling Railway Stations. IEEE Intelligent Systems, Vol. 7, 6, 45-52, Dec. 1992.

[13] A. E. Haxthausen, J. Peleska, Formal Development and Verification of a Distributed Railway Control System, IEEE Transactions on Software Engineering, Vol. 26, No. 8, pp. 687-701, 2000.

[14] A. E. Haxthausen, J. Peleska, Generation of Executable Railway Control Components from Domain-Specific Descriptions. In G. Tarnai, E. Schnieder (eds): FORMS 2003, Budapest/Hungary, May 15-16, 2003, pp.83-90.

[15] 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.

[16] N. König, S. 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.

[17] H. Kopetz, G. Bauer, The Time-Triggered Architecture, Proceeding of the IEEE Special Issue on Modeling and Design of Embedded Software. Vol. 91, Issue: 1 Jan 2003, 112- 126.

[18] P. Le Bouar, Interlocking SNCF Functional Requirements description, Eurointerlocking Project, Paris, 28 May 2003.

[19] M. J. Morley, Safety in Railway Signalling Data: A Behavioural Analysis. In proceedings of the 6th annual Workshop on Higher Order Logic Theorem Proving and its Applications, Vancouver, 4–6 August, 1993, LNCS Vol. 740, Springer-Verlag

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