



# Using synchronous language for signalling

G. Legoff, P. Sainton

*CS Transport, 4, avenue du Canada,*

*BP-243 Les Ulis, 91944 Courtaboeuf cedex, France*

## Abstract

Today, Signalling engineers must no longer merely program, but also specify, validate and implement, to attain the end goal of a proven fault-free system.

With the revolution in microprocessing power, designers of critical real time systems have experienced a gradual evolution in this field. Most of these designers have been forced to abandon their traditional way of thinking, a product of their electronics engineering culture, and to take on the culture of software engineers, who first believed they would come up with "miracle" tools and languages. The problem, of course, is that the programming languages proposed were not appropriate to the needs of signalling engineers. In particular, the sequential concept was not appropriate for the parallel structure intrinsic to the problems in question.

Synchronous Data Flow languages could provide a satisfactory solution for designers of critical real time signalling systems. The data flow model is, in fact, directly based on a description in the form of block diagrams preferred by automation engineers. Moreover, the synchronous model perfectly matches the execution mode of the systems described in terms of input sampling, calculation, output transmission and return in an infinite loop.

## 1 Introduction

The TVM 430 System, manufactured by CS transport, is now fully operational since 1993 on TGV North line (PARIS-LILLE), on the TransManche Link (Eurotunnel) and in progress in Korea. The TVM 430 System is a vital ATP/ATC system implemented on safety related computerized equipment distributed between trains and the equipment which is placed on the side of the

## 140 Computers in Railways

track every 12 km. The wayside equipment is configurable equipment using the same hardware and the same generic software application components. A specific tool has been developed for data generation according to the "system parameter" and the "geographical configuration" of each wayside equipment. The main characteristics of this tools are :

- graphical input (track layout and equations)
- verification of all boolean equations by signalling people
- automatic generation of Ada Code which is linked to the generic application program.

CS Transport is now implementing a new generation of ATP/ATC ground equipment. The vital computer hardware is similar to the TVM 430 but the software application program is new. It includes not only the headway management and train antifacing as in TVM 430, but also interlocking functions and route management. According to these new functions, an improvement of the parameter tool is in progress. The LUSTRE language is being tested to support the description of the functionality which is specific to each equipment according to its geographical configuration.

In our paper, we propose to develop the following topics :

- a short presentation of the TVM 430 parameter tool
- necessary enhancements according to the interlocking functions
- properties of the LUSTRE language
- application to some examples.

## 2 TVM 430 equipment parameterisation

### 2.1 General principles of TVM 430 signalling system

The general signalling principle retained is to divide the track into fixed block sections. Signalling data is associated with each block section. Deceleration or stopping sequences cover one or several sections.

The TVM 430 system chosen for the signalling of the TGV North line Transmanche link and the Korea line, makes possible the safety display of signalling data in the driver's cab, via track to train transmission and it enables the suppression of the wayside signalling.

In addition to supervision by the driver, a continuous speed control unit compares at all times the train speed with the maximum permissible speed. The supervision takes into account the target speed transmitted by the ground equipment and the actual position of the train within the block section, and therefore follows a parabolic deceleration curve. When the train speed exceeds the curve value, the speed control unit automatically triggers the emergency braking.

The following figures explain, the stopping sequence according to different types of trains.

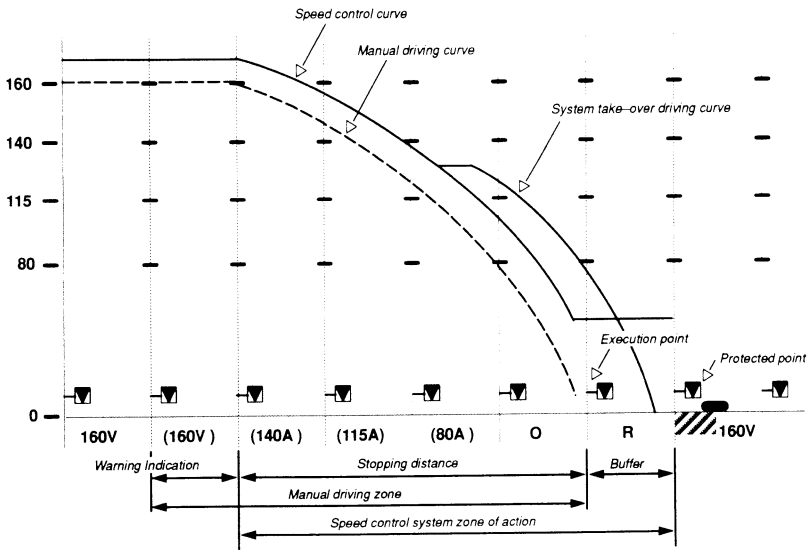


Figure 1 : Stopping sequence of a shuttle

## 2.2 TVM 430 Wayside equipment

The main functions achieved by the wayside subassembly include train detection by means of track circuits, computation of spacing conditions, coding and transmission to the train of data relating to speed, target distance, gradient and, finally, the transmission of this data to the neighbouring centres in order to ensure the continuity of the line.

The TVM 430 wayside subassembly consists of two types of cabinets : the processing rack (BTR) and the input/output rack (BES). The processing rack (BTR) includes wayside safety computers (PTR) in redundancy to execute functional processing, and a switching rack (PCO) ensuring switching in the event of failure of a wayside computer (PTR).

The input/output racks (BES) include block section racks (PCA) where the input/output relevant to one or several sections is gathered.

## 2.3 Aspect of parameterisation

Taking into account, the different applications of the TVM 430, particular care was taken to ensure that it can be adapted to different application and different geographical configuration.

For wayside equipment, a sophisticated parameterisation tool was developed enabling management of :

- the system parameter (as cab display sequence, blocking length, code for transmission) ; there is one system parameter table for each application (North TGV Line, Transmanche Link, Korean Line).
- the configuration parameter that is related to the geographical configuration of each equipment on the line (number of track circuits, I/O interfaces, boolean

## 142 Computers in Railways

equations for signalling implementation).

To illustrate the problem, we can give some facts and figures about the Channel Tunnel signalling application :

- 29 sets of TVM 430 track side equipment (each of them specific)
- 539 tracks circuits
- 4000 safety inputs
- 370 safety outputs
- 23000 boolean equations (some of them involving 100 terms)
- 8000 checking forms and 200000 elementary scenario on site test validation.

The parameterisation process is shown in figure 2.

- **2.3.1 Parameter tool** Each set of TVM 430 track side equipment is designed around a safety computer (PTR) and up to twelve input/output racks (PCAs) that can be added according to the geographical configuration of the railway.

The PTR supports generic safety software with specific parameters for each equipment. The generic software has been assessed by the SNCF for the TGV North line. It is linked to the safety application software which includes boolean equations specific to the equipment and is used to define the relationship between the inputs and the outputs.

For application definition, the parameter team uses a parameter tool directly with graphic and textual screens. This tool is developed around an ORACLE database and automatically generates the application software.

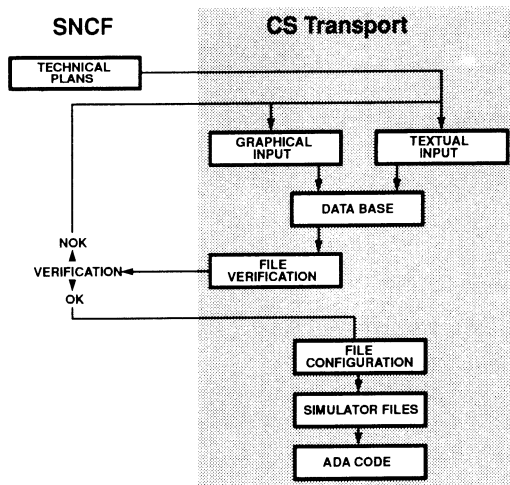


Figure 2 : Parametering process

- **2.3.2 Global safety policy** In accordance with SNCF experience, acquired on the TGV North line, we have used the following process for the Channel Tunnel project with SYSTRA :

In case of modification, the safety process implies an independence between the parameter team and the test team. The parameter team defines the application parameters to be modified and the test team modifies the checking forms that are to be used on site. This is generally a small subset of all the checking forms.

Using the parameter tool, two sets of PROMs are generated. The first set is sent for on site assessment. The second set is used in the laboratory for the non regression test.

All the checking forms previously assessed are executed on the laboratory simulator. It takes less than one month for one team to simulate all 29 sets of track side TVM 430 equipment, against six months with several teams for on site tests.

For each scenario, the simulator directly furnishes the OK/NOK agreement. At the end of the simulation, the NOTOK responses are checked by the test team against the expected results.

The figure 3 illustrate the safety policy.

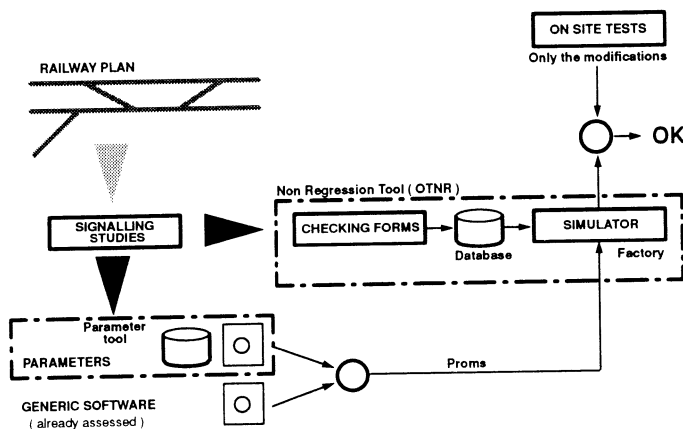


Figure 3 : Safety policy, modification process

### 3 A new language for signalling

CS Transport is now working on a new generation of ATP/ATC system. The ground equipment includes not only headway management but also interlocking functions an improvement of the parametering tools is in progress. In this chapter, we describe the main argument that influences our choice for a general

## 144 Computers in Railways

purpose synchronous language : LUSTRE.

### 3.1 Boolean equations and parallelism

Above all, signalling deals with the programming of industrial automatic control systems. In such systems, programs are non-terminating loops and interact with physical environments.

These environments have their own dynamics and cannot be synchronized with programs. Then, the timing constraints appear as the only means of synchronization between the programs and their environment. Moreover, these systems present another characteristic which has played a great role on language design choices : these systems are intrinsically parallel. They are generally specified and designed in terms of boolean equations, gates or switch networks, block diagrams or transfer functions. In addition, former discrete and analog technologies, or relays implementation was effectively parallel. The computerisation of these systems has led their designers to sequentialize these parallel behaviours, so as to adapt them to traditional programming languages.

This is a surprising tendency, since at the same time engineers are evolving towards better means of explaining and setting up parallelism.

### 3.2 General requirements

The designers' primary needs, therefore, remain a formal structure adapted to their culture enabling them to guarantee that their designs will function correctly, and to discover design errors as early in the process as possible. For other aspects, faced with the increasing complexity of software, the signalling engineers share most concerns of the software designers :

- specify and simulate a system ;
- automatically derive target code based on a high-level description ;
- enhance productivity, i.e. reuse rather than rewrite, and detect any inconsistencies as early as possible ;
- deploy a development methodology based on their own experience and the company culture ;
- improve software reliability and safety by simple means (readability of product code, reuse), or more sophisticated methods (formal verification, certification of tools used) ;
- be capable of automatically generating a complete set of documents based on what they develop ;
- if necessary, prove to the safety authorities that the software developed is fault-free ;
- guaranteed long-term viability of the tools they use.

### 3.3 Other domains experience

It seems very useful to use to have a look at other automation application domains. The experience of using synchronous graphical language for the Airbus A320 and A340 Jetliners seems particularly adapted to our needs.

Several articles have already spotlighted the specific advantages :

A very significant decrease in the number of encoding errors (from several hundred to just a few) due to the extensive use of automatic code generation based on detailed specifications. For the Airbus A 340, automatically-generated code accounted for 70% of the total. Progress in successive Airbus aircraft is summarized in the following table.

| AIRCRAFT                                 | A310      | A320     | A340   |
|------------------------------------------|-----------|----------|--------|
| Number of digital units                  | 77        | 102      | 115    |
| Volume of onboard software in M.bytes    | 4         | 10       | 20     |
| Encoding errors for 100 K.bytes software | $\pm 100$ | $\pm 12$ | $< 10$ |

Total control over the software life cycle, more specifically, the specification changes required during the development stage are under perfect control. Those involved pointed out the ability to very quickly and reliably deploy the modified software after a request for change.

### 3.4 Experiment decision

According to our experience with the TVM 430 parametering tool, it seems to us that the selection of synchronous language as LUSTRE was more in accordance with the signalling culture than using formal language such as B or Z.

So we have decided to experiment with using the general purpose synchronous language : LUSTRE, because :

- it has a formal base
- a graphical or textual representation is available
- it offers a hierarchical top down approach
- a library of elementary function can be built.

Based on the experience acquired by Schneider Electric and Aerospatiale in designing critical real time systems, Verilog decided to use the Lustre language for its SAO+SAGA (computer-aided specifications) environment. Working in close harmony with manufacturers and research teams, Verilog was able to construct this environment based on simple principles, with a constant focus on meeting user requirements, while at the same time taking advantage of advanced research into Lustre language applications.

SAO+SAGA is an open environment comprising two main tools.

- The first is a multiview graphics editor, for which parameters can be set according to the methodological rules established by the user. This editor can be connected to a documenter to produce structured documents in various formats (PostScript, Interleaf, SGML, etc ...)
- The second is an automatic code generator, that produces optimized, readable ANSI C code. It can also be connected, via an intermediate format, to a Petri net format property verifier.

An experiment of SAO+SAGA began in CS Transport January 95.

## 4 The lustre language and sao+saga tool

### 4.1 Lustre a synchronous Data Flow language

The data flow model is based on a block diagram description. A system is made up of a network of operators acting in parallel and in time with their input rate. Such formalism possesses several of the advantages required for a

## 146 Computers in Railways

high level language :

- it is a functional model, and, therefore has its mathematical cleanness. It clearly fits formal verification allowing a natural expression of properties,
- reusability is made easier by incremental construction of operator, networks becoming new operators,
- a maximal use of parallelism is ensured (the only constraints are dependencies between data),
- the data flow model has a natural graphical expression as well as a textual representation using mathematical equations. Consequently, it fits the different cultural backgrounds of the end-users.

The synchronous data flow approach consists of adding a time dimension to the data flow model. A natural way of doing this is to associate time with the rate of data flow.

The entities manipulated can naturally be interpreted as functions of time. A basic entity (or a flow) is a couple made of :

- a set of values of a given type,
- a clock representing a set of graduations (on the discrete time scale).

A flow takes the value in its set at the  $t^{\text{th}}$  instant of its clock.

Lustre is a synchronous language based on the data flow model. The synchronous aspect introduces constraints on the type of input/output relations that can be expressed ; the output of a program at a given instant cannot depend on the future inputs (causality), and can depend only on a limited number of inputs. The operators allowed to express the operators networks are the classical arithmetic and logic operators. Moreover, each cycle can memorize the value of the previous input using a specific operator called *pre*. Another specific operator can force a value of a data at the first cycle. Finally, operator networks can be computed at different rhythm manipulating the clocks of the data flows.

### 4.2 SAO + SAGA Tool

- **Graphic Editor** The SAO+SAGA environment editor enables the graphics manipulation of Lustre descriptions. It is a multiview editor, meeting OFS Motif recommendations, that supports advanced publishing functions and offers a number of high-level services :
  - Multiview publishing, which enables the creation, modification and incremental loading of block diagrams. This service proposes several views of the same description.
  - The hierarchical view allows manipulating the structure of a description, which facilitates the top-down design method. A network view allows description of the block diagrams, while a text view enables displaying and entering block diagrams in the form of equations (Lustre language). There is also a "type" view, that allows defining complex types that will be used to manipulate structured data, through an operator network.
- **Automatic code generator** The automatic code generator tool generates 100% of the C code that implements a description either directly in the Lustre test language, or using the SAO+SAGA environment editor. The code generated meets several requirements resulting from the needs expressed by the users :

- The C code generated is ANSI compatible and can be optimized on request in terms of memory size and processing speed.
- The code generated is readable : its structure matches the hierarchy of the Lustre description and the variables generated are named so as to ensure traceability with the data circulating in the Lustre description.
- The code generated can be simulated without being instrumented, thanks to the production of a main program. The obvious advantage here is that what is simulated is also what is installed on the target machine.

## 5 In progress experimentation

## 5.1 Code generation

For the Hong Kong Projects, it is necessary to replace old relays by new computerized algorithms, we decided to use the SAO+SAGA for automatic code generation :

- The relays schemas are translated to SAO language in textual. This work was very easy and must be verified very carefully because it is the same semantical level.
- Then, we have an automatic code generation in C.

The main point of this experiment was to check the textual language and the code generator.

No major problem occurred during this experiment, we detected only some syntactic problems due to input/output terms that are syntactically very close and can be interexchanged during manual translation in Lustre.

To illustrate the height of this experimentation some numbers :

- 16 stations
- 254 modes SAO+SAGA
- 19505 lines of Lustre language
- 1,5 Mo code C
- 2705 I/O on the main mode.

## 5.2 Signalling experiment

For this experiment, we simulate the signalling of a simple TGV station, and we adopt the geographical principle.

This principle can be split up into several functions (20). For each function, we define library unit which is described by a diagram *chaw* with SAO+SAGA units. In this paper, we give the sectional release route locking function in example.

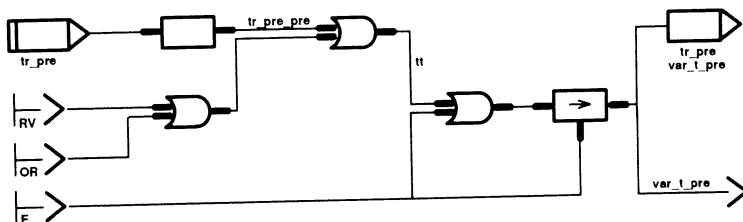


Figure 4 : Sectional release route locking unit



## 148 Computers in Railways

This diagram is drawn with SAO+SAGA unit. Each element of this diagram has a specific signification (and - or - initialize ...). Each library unit is tested with the SAO+SAGA simulator. This SAO+SAGA simulator give us the possibility to check the outputs in relation with the inputs. The tests are described by a script, if we modify the library unit functionality we can retest this one.

With the library units, when they are completely verified, we draw a diagram like the geographical of trach plan.

In result, the synchronous language is very adapt for a simular application, so the experiment, on a real station, is in progress by CS Transport and right now, we think to get quickly one's way.

## 6 Conclusion

For many years signaling function have been described by Relay representation.

New ATP and ATC functions using computerised equipments have been described using data flow diagram and specific software Tools.

In the wayside computerised equipments that will be implend by CS Transport for Mediteraen TGV.

Interlocking and ATC functions will be implend in the same safety computer.

We needs new tools for describy the relation ships between output, input and route setting. The Lustre language may be a possible evolution.

## 7 Reference

1. Pilaud, D. & Bergerand, J.L. Design and qualification of software for Protection and Control System.
2. Haibwachs, N., Pilaud, D., Ouabdesselam, F. & Glory, A.C. Specifying, Programming and Verifying Real-time Systems using a synchronous Declarative Language.
3. Haibwachs, N., Caspi, P., Raymond, P. & Pilaud, D. The synchronous Data FLOW Programming Language Lustre.
4. Aerospatiale Synchronous data flow languages for on board software.

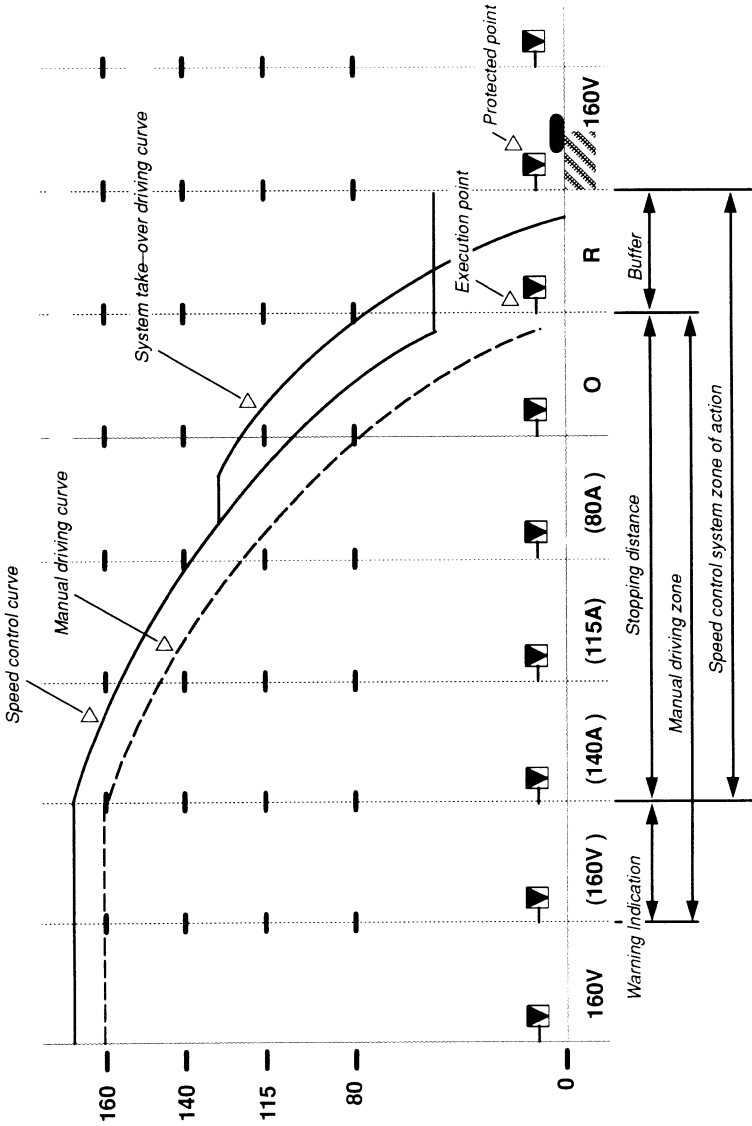


Figure 1 : Stopping sequence of a shuttle

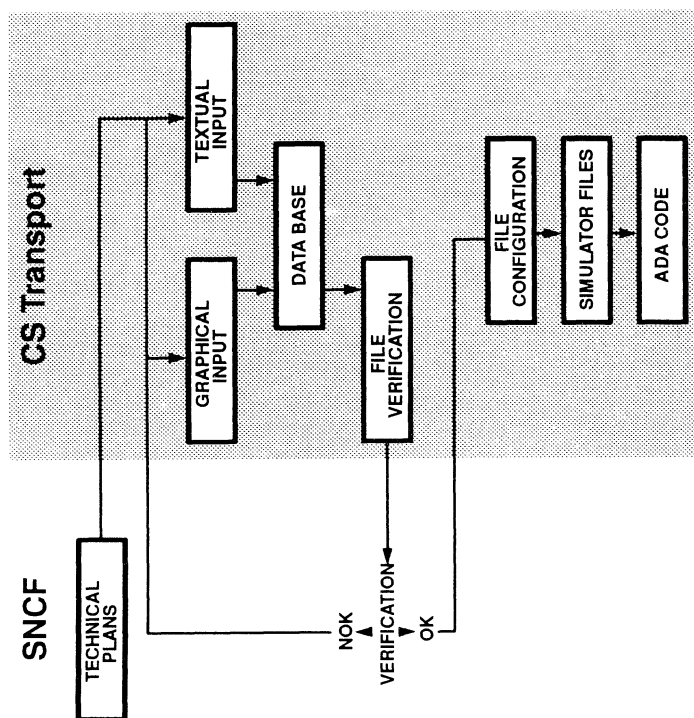


Figure 2 : Parametering process

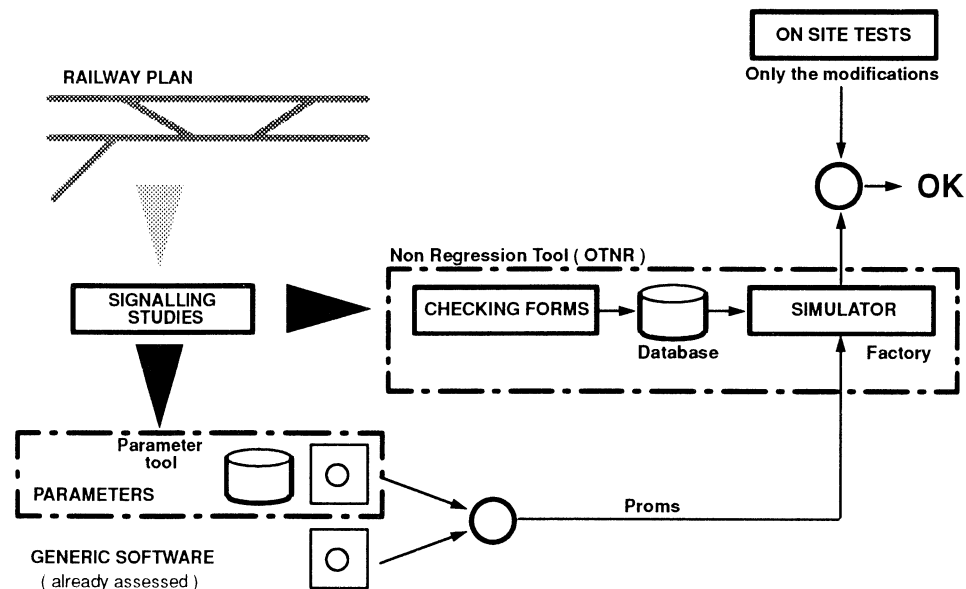


Figure 3 : Safety policy, modification process



## 152 Computers in Railways

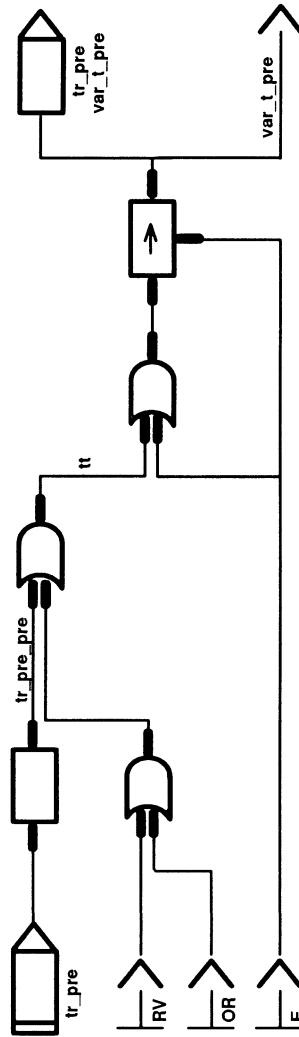


Figure 4 : Sectional release route locking unit