

# **Event Driven System Construction**

by J.-R. Abrial

(April 1999)

# Event Driven System Construction

## Introduction

While the problematics of the *development process* for software has already been treated in many papers, and now seems to have acquired a certain stability of form, the same thing cannot be affirmed about “System Studies”.

Under this term we usually set, in a relatively casual way, all the reflexions and decisions situated *before* the drawing up of the Requirements Document of a system. Such studies are mainly aimed at rationalising the drawing up of that document, and, more generally, the elaboration of the architecture of a system. They are also aimed at showing that the system being considered is viable. Besides, they can end up in the determination of a certain number of fundamental parameters for which they will fix the range of values.

This text does not in any way claim to exhaust such a vast subject. It has only the more modest ambition of exposing, through a precise example, the application of a new approach to the matter. This approach appeals to the techniques of the Formal Methods with Proofs, which are being used successfully more and more nowadays in the realisation of industrial software.

This document is organised in the following way. We will first make a brief presentation of the case study. As a result, we shall immediately be placed within a concrete framework. Then we will present the aims of the System Study in quite general terms, even if they will be illustrated by characteristic elements from the case study. Then we will develop the complete analysis of the system so as to obtain a final architecture, hardware as well as software. Finally, we will establish a review of the situation.

We have pushed the brief account of the formal methodological framework in which we are going to evolve out to an Appendix. The reader may either refer to it immediately (it is only composed by a few stages), or consult it later (the reading of the Appendix is not necessary for the comprehension of the development of this case study)

## Informal Presentation of the Case Study

We are now tackling the case study which is going to allow us to illustrate what we are talking about. The main points of the project will be presented. This case study has been proposed by several researchers [6] as a test bed for diverse formal methods.

### *The purpose of the System*

The idea is to elaborate a system which will be able to control the access of certain persons to different buildings of a “workplace” : for example, a university campus, an industrial site, a military compound, a shopping mall, etc.

### *Authorization*

The control takes place on the basis of the authorization that each person who is concerned is supposed to possess. This authorization should allow him, controlled by the system, to penetrate into certain buildings, and not into others. For example, a certain person  $p_1$  is authorized to enter the building  $b_1$  and not the building  $b_2$ ; however, another person  $p_2$  is allowed to enter both buildings. These authorizations are given on a “permanent” basis: in other words, they will not change during a normal functioning of the system.

When someone is inside a building, his eventual exit must also be controlled by the system, so as to be able to know at any moment who is inside a given building.

### *Magnetic Cards*

Each person involved receives a magnetic card with a unique identifying sign, which is engraved on the card itself. Card readers are installed at each entrance and at each exit of concerned buildings. Near to each reader, two control lights can be found: a red one and a green one. Each one of these lights can be on or off.

### *Turnstiles*

The transfer of people from one building to another takes place thanks to “turnstiles” which are normally blocked: nobody can get through them without being controlled by the system, any person getting through is detected by a sensor. Each turnstile is only affected to a single task, either entry or exit, there are no “two-way” turnstiles.

### *Access Protocol*

The entry or the exit of a building follows a systematic procedure composed of a suite of events which follow:

A person wishing to enter or exit a building puts his card into the card reader on the appropriate turnstile. One is then faced with the following alternative:

- If the person is authorised to pass through the turnstile, the green control light is lit and the turnstile is opened during 30 seconds. We are now faced with the following alternative situation:
  - As soon as the individual gets through the turnstile within the 30 seconds limit, the green control light goes out immediately and the turnstile blocks.
  - If, however, 30 seconds go by without anybody going through the turnstile, the control light goes out and the turnstile is also blocked.
- If the person is not authorised to pass through the turnstile, the red control light goes on for two seconds and, of course, the turnstile remains blocked.

## **Aims of the System Study**

The above informal presentation of the system does not pretend to be complete, or to have raised all the technical options. Indeed, it is merely the minimal starting point for the realization of the future control software, but clearly there remain many unknowns concerning,

among others, the hardware and its links with the software. We give precisions hereafter as to the role of the System Study.

#### *Sharing out of the Control*

An important question which should be asked at the outset about such a system concerns the distribution, which is more or less important, of control between the software and the diverse peripherals (turnstiles, readers).

For example, there can be a computer at each turnstile: in this case, control is entirely decentralised. Inversely, a single computer can entirely centralise the control. Of course, there are intermediate situations in which each turnstile has a certain form of autonomy: an example consists in equipping each turnstile with a clock which conditions part of its behaviour.

#### *Construction of a Closed Model*

In any case, technical argumentation which can lead to such and such a decision can only be realized *by analyzing the system as a whole*. It is to be noted that this technical argumentation must also be nourished by information concerning the equipment available on the market (it can also be decided to realize new equipment).

The System Study will therefore essentially consist in building a *closed model* of our future system and in *proving* on this same model that its *characteristic properties* (which will have to be precisely explained) are in fact assured.

#### *Behaviour of the Equipment*

An important result of the System Study concerns the behavioural specification of different equipment. So as to conduct this study correctly, we will be obliged to introduce into the model a certain amount of supposition concerning this equipment. For example, does the turnstile block of its own accord after the pass of a single person, or does it block only after the reception of an order from the software? Of course, the chosen option conditions the organisation of the software. This option makes up a hypothesis under which the software can function correctly.

After the System Study, it will be necessary to make a certain number of choices concerning the behaviour of the equipment. These choices must be stated in the Requirements Document for Equipment composed of, among other things, a receipt procedure aimed at verifying that the equipment which has been put in place has in fact the expected qualities. If in fact this were not the case, it is clear that the union of the software and the hardware would have no chance of working correctly as had been demonstrated on the model (under the relative hypotheses). This demonstration will therefore have been useless.

#### *Possible Generalization of the Problem*

One can also ask oneself the question if the above informal presentation is not too oriented towards a single control system, that is to say, that of the access to buildings *from the outside*. More generally, the designer of such systems should perhaps study a more general disposition which could also be adapted to controls *between buildings*. We would therefore

like to know if, thanks to the System Study, such generalization is interesting. This choice will mainly be dictated by the facility of construction of the model.

### *Tackling Safety Questions*

An important question which is not tackled at all in the informal presentation concerns the safety of persons implicated in this system. We would like the System Study to inform us on this point, or at least that it asks a certain number of pertinent questions. For example, can people be blocked for ever in a building ? How can we guarantee the contrary ?

### *Synchronisation Problems*

On a more technical level, the informal presentation says nothing about the details of the timing of the transfer operation. For example, what time-lag is there between the lighting up of the green control light, the acceptance of the turnstile and the start of the 30 second countdown ? Could the green light go on while the turnstile is still blocked, or could it go out while the turnstile is still accepted, etc ?

The question is not so much to find out if the preceding behaviour is good or bad, it is rather to recognise the existence of a certain behaviour and to know in advance if it can occur (even in a fugitive form) in the final system which will be installed.

### *Functioning at the Limits*

Another question which is not treated in the informal presentation is what happened when the system functions “at the limits”. For example, what is the reaction of the system when a card is introduced into the reader whilst the green or red lights are still lit (that is to say, before the preceding *action* is finished) ? More generally, we would like to be able to understand and predict how the system reacts when faced with “hostile” behaviour of certain users. In this sort of system, one must not count upon hypotheses which rely too much upon the “good” behaviour of users. Some users do behave in a “strange” way.

## **Methodological Conventions**

In the next part we will observe the following conventions of style. The English text will be strewn with different “boxes” which can contain (1) either the exposition of certain *properties* described by means of one or two very short sentences, these boxes are identified by the prefix ”P”, (2) or the exposition, also very short, of a *decision*, these boxes are identified by the prefix ”D”, (3) or the mathematical formalization of an *invariant*, (4) or lastly the formal description of an *event*.

So as to make these models more legible, we will follow a certain number of simple lexical conventions concerning the way identifiers are written. We will write the variables or the constants corresponding to physical elements (for example those characterising the card readers) solely in capital letters. The variables which are rather the concern of the software will only be written in small letters. Finally, the variables which characterize the messages circulating on the network will contain capital and small letters and, moreover, they will all begin by the letter “m”.

## Initial Model of the System

We are now going to proceed to the initial formal description of our system by constructing a first, very abstract, model in which the separation between the software and the hardware is not the current issue.

### *Informal Initial Specification*

We must be careful, first of all, to determine who are actors of this system, actors who will enter the model in the form of a certain number of sets which will be made clear little by little.

P1: The model is composed of people and buildings.

The system must control the permanent authorizations to enter or leave the buildings.

P2: Each person is authorized to enter certain buildings (and not others). Access to buildings not consigned in this authorization is strictly forbidden. This is a permanent assignment.

An implicit property, which it is important to make conspicuous from the outset of the construction of the model, concerns the impossibility for a same person to be in two different buildings at the same time. This means that buildings in no way overlap.

P3: At any one moment, a person can only be in one building.

A simplification (and a generalization) of the model consists in identifying “the exterior” to a building (in which anyone has the right to be!). This is given concrete expression by the following decision, which gives further precisions to the initial informal presentation:

D1: The system manages the pass of persons from one building to another.

The following complementary property then arrives:

P4: At any one moment, a person must be in at least one building.

Lastly, we present the principal property of the system, which shows that the control is being done correctly, that is to say that each person is, at each instant, authorized to be in the building in which he finds himself.

P5: Any person in a given building is authorized to be there.

*Example*

To illustrate this, let us suppose that we have four buildings  $b_1, b_2, b_3, b_4$  and three persons  $p_1, p_2, p_3$  equipped with the following authorizations:

$p_1$	$b_2, b_4$
$p_2$	$b_1, b_3, b_4$
$p_3$	$b_2, b_3, b_4$

**Authorization**

The following situations can therefore represent a satisfying evolution of the system since, as can be noted, the properties P1 to P5 are always respected.

$p_1$	$b_4$
$p_2$	$b_4$
$p_3$	$b_4$

**Situation 1**

$p_1$	$b_2$
$p_2$	$b_4$
$p_3$	$b_4$

**Situation 2**

$p_1$	$b_2$
$p_2$	$b_1$
$p_3$	$b_4$

**Situation 3**

$p_1$	$b_4$
$p_2$	$b_1$
$p_3$	$b_4$

**Situation 4**

$p_1$	$b_4$
$p_2$	$b_1$
$p_3$	$b_3$

**Situation 5**

*First Formal Model of the System: the Basic Elements*

We are now in a position to build our first model. We name  $prs$  the set of persons and  $bld$  the set of buildings. It is supposed, of course, that these sets are finite, and are not empty.

$prs \neq \emptyset$ $bld \neq \emptyset$
---

### Constants and Variables

The authorizations are represented by a set of “person-building” pairs, each of which linking a person to a building in which he has the right to be. When a person  $p$  is authorized to enter several buildings, there are as many pairs for that person as there are buildings concerned. This set of pairs,  $aut$ , is therefore a binary relation between  $prs$  and  $bld$ .

The dynamic situation of persons in buildings is represented by a certain total function  $sit$ . As the relation  $aut$  envisaged above, this function contains “person-building” pairs, each of which linking a given person to the building inside which he is. The fact that it is a function corresponds to the formalization of the property P3 (a person in one building at the most). The fact that this function is total corresponds to the property P4 (a person must be in at least one building).

$$aut \in prs \leftrightarrow bld$$

$$sit \in prs \rightarrow bld$$

### The Invariant of the System

The characteristic property P5 (the persons present in buildings at a given moment do have the right to be there) is formalized by stipulating that the function  $sit$  is included in the relation  $aut$ . In other words, if the pair  $(p, b)$  belongs to the function  $sit$  (the person  $p$  is then in a building  $b$ ) then this pair must also belong to the relation  $aut$  so that  $p$  is really authorized to be in  $b$ . This ends up with the following invariant which should therefore always be verified:

$$sit \subseteq aut$$

### Event

At this level of abstraction, we can only *observe a single transition*, which manifests itself on the arrival of the event **pass**, and which provokes the entrance of a person  $p$  in a building  $b$  (that is to say  $sit(p) := b$ ). This event should only be able to happen (necessary condition) if  $p$  is authorized to be in  $b$  (that is to say if we have  $(p, b) \in aut$ ) and if, of course, he is not already in  $b$  (that is to say if we have  $sit(p) \neq b$ ).

```
pass  $\hat{=}$ 
  ANY  $p, b$  WHERE
     $(p, b) \in aut \wedge$ 
     $sit(p) \neq b$ 
  THEN
     $sit(p) := b$ 
  END
```

It must be remarked that when we say the event **pass** can happen, that does not necessarily mean that it is in fact going to happen. We are only saying that it is triggerable and

therefore that it *could be observed*.

### *Conclusion*

It is easy to *prove* that the invariant is well preserved by the above transition when it happens. It is to be remarked that the event **pass** is very abstract; we do not know by which process the person  $p$  enters the building  $b$ . Neither do we know if the building in which the person  $p$  is, communicates with this building  $b$  in which he wishes to go. In fact, we cannot express this property because we have not yet formalized the “geometry” of the buildings.

The only important element of the present model is the expression of fundamental rules of the system (in the invariant) and the proof that the unique observable and interesting event at this level (**pass**) maintains these properties. We are already sure that the following models will respect this property if, of course, we can *prove* that they constitute *correct refinements* of the present model.

### **First Refinement**

#### *Extension of the Informal Specification*

We are now going to proceed to our first refinement which will consist in introducing into the model the notion of possible communication between two buildings.

P6: 

The geometry of buildings serves to define which buildings can communicate between each other, and in which direction.
--

There is another evident constraint stipulating that two buildings which communicate between each other must necessarily be distinct. This reinforces in a sense the non-overlapping of the buildings.

P7: 

A building does not communicate with itself.
--

The following rule indicates that a person cannot pass from the building in which he finds himself to any other building where he is authorized to be (as was the case in the previous abstraction) unless the two buildings communicate with each other:

P8: 

A person can only go from one building in which he is to another where he desires to go if these two buildings communicate with each other.
---

#### *Construction of the Second Model : Constants*

In this second model, the possible communication between the two buildings is formalized by a set of pairs “building- building” which link each building  $b_1$  to a building  $b_2$  with which

it is supposed to communicate (in one way only, that is from  $b1$  to  $b2$ ). This set of pairs defines a binary relation,  $com$ , relation which is necessarily non-reflexive so as to respect the property P7 (a building does not communicate with itself). The invariants which result from this can be formalized as follows:

$$\begin{array}{l} com \in bld \leftrightarrow bld \\ com \cap id(bld) = \emptyset \end{array}$$

*Event*

At this level, we can only again observe the single event **pass**. It is to be remarked that the second guard,  $(sit(p), b) \in com$ , of this event guarantees the satisfaction of the property P8 (the buildings concerned by the pass of a person must communicate with each other):

$$\begin{array}{l} \mathbf{pass} \hat{=} \\ \text{ANY } p, b \text{ WHERE} \\ \quad (p, b) \in aut \wedge \\ \quad (sit(p), b) \in com \\ \text{THEN} \\ \quad sit(p) := b \\ \text{END} \end{array}$$

*Proof of Refinement*

This event is quite simply a refinement of the preceding version because the action is identical in both cases ( $sit(p) := b$ ) and the guard of the second, that is to say

$$\exists (p, b) \cdot ((p, b) \in aut \wedge (sit(p), b) \in com)$$

is clearly stronger than the first, which is the following:

$$\exists (p, b) \cdot ((p, b) \in aut \wedge sit(p) \neq b)$$

Roughly speaking, if the condition  $(sit(p), b) \in com$  is verified (if the person  $p$  is in a building which communicates with the building  $b$ ) then the building where this person finds himself is certainly distinct from  $b$  (since a building can't communicate with itself), therefore the condition  $(sit(p) \neq b)$  holds.

*Proof of freedom from deadlock* It must now be proved that, in the absence of new events in this refinement, the concrete event **pass** does not happen less often than its abstract homologue (we remind you that this is a necessary condition of the refinement).

In fact, we are obviously faced with a difficulty here: it is not possible to prove that the refined event **pass** does not happen less often than its more abstract homologue. For demonstrating this, we would have to prove that the guard of the abstract event implies that of the concrete event, that is:

$$\exists (p, b) \cdot ((p, b) \in aut \wedge sit(p) \neq b) \Rightarrow \exists (p', b') \cdot ((p', b') \in aut \wedge (sit(p'), b') \in com)$$

It is clear that this condition can not be verified in general. Indeed, there is no reason for the condition  $(sit(p) \neq b)$ , stipulating that the building where a certain person  $p$  finds himself is distinct from the building that person wishes to enter, there is no reason for this condition to imply the existence of a person  $p'$  and of a building  $b'$  so that the condition  $sit(p'), b' \in com$  be filled, that is to say that the building where this person  $p'$  is, communicates with the building  $b'$  where he wishes to go: a counter-example is easy to exhibit.

The failure to prove the above condition indicates that *there are possibilities that some persons could stay permanently blocked in buildings*. And to be more precise, this could be the case even if this possibility does not exist in the abstraction, that is to say even if the authorizations are well defined to begin with so that it cannot happen. In fact, the geometry of communication between buildings clearly adds an extra constraint limiting the way people can move.

#### *Appearance of a Problem Linked to Safety: Need for Supplementary Requirements*

Indeed, if a person is in a building  $b$  and has no authorization allowing him to be in any of the buildings which communicate with  $b$ , then that person is blocked in  $b$ . The impossibility to make the above proof has brought up a safety problem, which can be set out in the form of a new property which the system must satisfy:

P9: No persons must remain blocked in a building

Notice that, what is stated here is stronger than what would have been needed, strictly speaking, to allow us to perform the proof that failed. That weaker statement would have been the following: “the geometry of buildings does not introduce extra possibilities of blockage beyond those already present without this constraint”. As a matter of fact, the mathematics of our System Study has revealed a problem that gives us the idea of a wider safety question that was totally ignored in the informal presentation.

#### *Attempted Solution*

So we must find a sufficient constraint so that this property P9 is practically satisfied. The obligation of the preceding proof will serve as a model for this constraint. This obligation can be re-written as follows:

$$(aut \cap (sit; \overline{id(bld)})) \neq \emptyset \Rightarrow (aut \cap (sit; com)) \neq \emptyset$$

For this, it is sufficient to prove

$$(aut \cap (sit; com)) \neq \emptyset$$

Or, in an equivalent way

$$((aut; com^{-1}) \cap sit) \neq \emptyset$$

So as to prove this condition, it is sufficient to prove the following (since the condition  $prs \neq \emptyset$  implies that the total function  $sit$  is not empty):

$$sit \subseteq (aut; com^{-1})$$

What does this condition say? If we develop it, we obtain:

$$\forall p \cdot \exists b \cdot ((p, b) \in aut \wedge (sit(p), b) \in com)$$

In other words, the building  $sit(p)$  in which each person  $p$  is situated at any moment is in communication with at least one other building  $b$  in which  $p$  is authorized to go: the person  $p$  can therefore go out of the the building in which he is via  $b$ .

This condition could be imposed as a new invariant of the system: however, it would be necessary to reinforce the guard for the **pass** event so as to admit in any one building only the persons authorized to enter it (this is already the case), and who would also be authorized to exit. If faced with an interdiction, the authorization to enter such a building, which the person concerned would hold, would not be of much use since access would be refused for him anyway. So, it would be preferable to have a (sufficient) condition *which would be independent of the situation of people*. Indeed, this is possible, because we already have the invariant property  $sit \subseteq aut$ . So as to prove the condition  $sit \subseteq (aut; com^{-1})$  above, it is sufficient to prove the following, by transitivity of the inclusion:

$$aut \subseteq aut; com^{-1}$$

This condition makes up a further invariant, which can be translated as follows:

$$\forall (p, b) \cdot ((p, b) \in aut \Rightarrow \exists c \cdot ((p, c) \in aut \wedge (b, c) \in com))$$

The interpretation of this invariant is quite instructive : it can be remarked that whenever the pair  $(p, b)$  belonging to  $aut$  (the person  $p$  could therefore be in the building  $b$  since he is authorized to be there), there is a building  $c$  such as  $(p, c)$  belongs to  $aut$  (the same person  $p$  is therefore authorized to enter the building  $c$ ) and, moreover, such that the pair  $(b, c)$  belongs to  $com$  (so the two buildings  $b$  and  $c$  do communicate). When all is said and done, the person  $p$ , who could be in the building  $b$ , would not remain blocked indefinitely since he is also authorized to enter the building  $c$  which communicates with  $b$ . So this person can leave  $b$  via  $c$ . As can be seen, property P9 has now been satisfied thanks to a stronger requirement whose expression has been determined (calculated) by mathematical study before being expressed in English as follows:

P10:

Any person authorized to be in a building must also be authorized to go in another building which communicates with the first one.

It is to be noted that the problem we have just evoked should be widened to guarantee that any person in a building, can not only go out of it but, more generally, can also get into a “building” which can be named “outside”, where, of course, everyone has the right to be. This problem will not be evoked here. So this can be translated by the following decision.

D2:

The system that we are going to construct does not guarantee that persons can go “outside”. It only guarantees that persons can get from one building to another.

## Second Refinement

### *Extension of the Informal Specification*

During this second refinement, we are going to introduce one-way doors to communicate from one building to another:

P11:

The buildings communicate with each other via doors, which are one-way. We can therefore refer to buildings of origin and destination for each door.

The doors can be physically blocked. This is what insures in a material way that it is impossible to get through a door without being controlled. The following property defines the modalities for acceptance to a door:

P12:

A person can only get through a door if it is accepted. A door can only be accepted for one person at a time. Conversely, any person implied in the acceptance of a door cannot do the same for another.

The following property makes clear the conditions for acceptance to a door for a given person. *This condition makes up the essence of the control made by the system.* Indeed, once a door is accepted for a person, its clearing can not be physically stopped (at least for a certain time, as we shall see below).

P13:

For a door to be accepted for a certain person, this person should be inside the building of origin of that door. Moreover, this person should be authorized to enter the destination building of that same door.

A green light is associated to each door. It is lit or off in the following conditions:

P14:

The green light of a door is lit as long as the latter is accepted. As soon as a person has got through, the door blocks itself again. After 30 seconds, if no-one goes through an accepted door, the latter will block automatically. In both cases, the green light goes off.

In the same way, a red light is associated to each door. It is on or off in the following conditions:

P15:

The red light of a door whose access has just been refused stays on for a period of 2 seconds, the door stays blocked of course.

In the two preceding conditions, the following property can be deduced:

P16:

The red and green lights of a same door can not be lit simultaneously.

#### *New Constants and Invariant*

The formalization goes through the introduction of a finite non empty set  $dor$  which makes models of the doors. Each door is associated with a building of origin, represented by the total function  $org$  and a destination building, represented by the total function  $dst$ . For all these doors, the buildings of origin and destination represent exactly the pairs of buildings implied in the relation  $com$  introduced during the previous refinement. This is formalized as follows:

$$\begin{aligned} dor &\neq \emptyset \\ org &\in dor \rightarrow bld \\ dst &\in dor \rightarrow bld \\ com &= (org^{-1}; dst) \end{aligned}$$

#### *Variables and Invariant*

The formalization of the property P12 (exclusive acceptance of a door for a single person) comes about with the help of a partial injective function,  $dap$  (it stands for door accepted for a person), which links persons to doors. More precisely, this function corresponds to the set of pairs linking a person to the door which is accepted for him. The following two predicates formalize the property P13 (condition for acceptance to a door).

$$\begin{aligned} dap &\in prs \rightsquigarrow dor \\ (dap; org) &\subseteq sit \\ (dap; dst) &\subseteq aut \end{aligned}$$

The range of the function  $dap$ , that is to say the set  $\text{ran}(dap)$ , is made up of all the doors which are engaged in the eventual process of the pass of a certain person who, however, has not passed yet. The green lights of the doors in question are therefore on. The following definition of the subset  $grn$  of the doors whose green light is on is introduced. This definition corresponds to the property P14 (lighting of the green light as soon as the door is accepted).

$$grn \hat{=} \text{ran}(dap)$$

By symmetry, the set *red* of doors whose red light is on is introduced too. It is to be noted that the red and green lights can not both be on at the same time (property P15). The following new invariant presents itself:

$$\begin{array}{l} red \subseteq dor \\ grn \cap red = \emptyset \end{array}$$

### Events

Now several new events are introduced. First the events **accept** and **refuse** which correspond to the discovery of a person who wishes to go from one building to another. In the first case, this person is admitted, in the second he is not. The condition of admission can be expressed as follows:

P17: Any person is allowed to go through a door which communicates from the building in which he is to a building in which he is authorized to go. Moreover this person must not be already engaged with another door.

Formally, the following definition of the predicate *admitted*(*p*, *q*) follows:

$$\begin{array}{l} admitted(p, q) \hat{=} \\ org(q) = sit(p) \wedge \\ (p, dst(q)) \in aut \wedge \\ p \notin dom(dap) \end{array}$$

We can now present the two events **accept** and **refuse**. The guards of these events have a common “root”, that is to say a person *p* and a door *q*, whose red or green lights are off (we will see during a following refinement how this can become a certitude). Then the guards diverge, the **accept** one corresponding to the predicate *admitted*(*p*, *q*) and **refuse** to the predicate  $\neg admitted(p, q)$ .

```

accept  $\hat{=}$ 
  ANY p, q WHERE
    p  $\in$  prs  $\wedge$ 
    q  $\in$  dor  $\wedge$ 
    q  $\notin$  grn  $\cup$  red  $\wedge$ 
    admitted(p, q)
  THEN
    dap(p) := q
  END

```

```

refuse  $\hat{=}$ 
  ANY p, q WHERE
    p  $\in$  prs  $\wedge$ 
    q  $\in$  dor  $\wedge$ 
    q  $\notin$  grn  $\cup$  red  $\wedge$ 
     $\neg admitted(p, q)$ 
  THEN
    red := red  $\cup$  {q}
  END

```

The former event **pass** can now be found which can be triggered for a door *q* whose green light is on. It provokes the pass of a person. This results in the turning off of the green light (that is, *q* is removed from the range of *dap*).

```

pass ≐
  ANY q WHERE
    q ∈ grn
  THEN
    sit(dap-1(q)) := dst(q) ||
    dap := dap ▷ {q}
  END

```

Finally, we can find the two new events `off_grn` and `off_red` which result in the putting out in an asynchronous way (for the present moment) of the respective green or red lights of a given door. The first also provokes the disengagement of the person concerned with the door in question.

```

off_grn ≐
  ANY q WHERE
    q ∈ grn
  THEN
    dap := dap ▷ {q}
  END

```

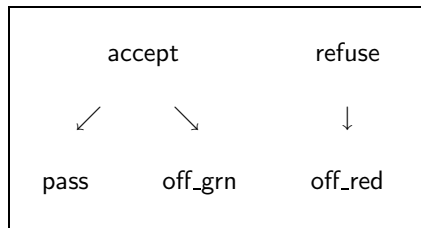
```

off_red ≐
  ANY q WHERE
    q ∈ red
  THEN
    red := red - {q}
  END

```

### Synchronisation

As illustrated in the following diagram, the synchronisation between the different events is quite weak for the moment.



### Proofs

It is easy to prove that the new version of the event `pass` refines the preceding one. It is also easy to prove that the new events all refine the event which does nothing, `skip`.

Two things remain to be proved: (1) that the event `pass` does not happen less often than its abstraction, taking into account, of course, the new events, and (2) that the new events cannot take control indefinitely, thus preventing the event `pass` from taking place.

In fact, the proof of (1) is relatively easy, however that of (2) is quite simply *impossible*. We have here a new difficulty which we will have to investigate and probably correct by additional requirements.

### *Risk of Permanent Obstruction of Card Readers*

Taking into account the initial informal presentation, we now know that the event `pass` envisaged above is not the only “basic” event which can be observed. We know that the entrance of a person can be refused and also that a person who wishes to enter a building can change his mind just before passing: in this last case the door is automatically blocked again after 30 seconds. This corresponds to the events `refuse` and `off_grn` envisaged above.

However, during their introduction, it must be proved that these two events cannot prevent indefinitely the event `pass` from happening (that is to say that their guards are not indefinitely true at the same time as those of the event `pass`) which, in theory rather than in practice, is not impossible<sup>1</sup>.

Indeed, we can imagine observing strange behaviour of the system where nobody could ever enter the buildings, either because people without the necessary authorizations keep trying to get in, or because other people, who are authorized put their cards in the reader but change their minds at the last minute. So as to prove that this behaviour is not indefinitely possible, we must propose something to prevent them.

### *Propositions for Preventing Permanent Obstruction*

A first way of proceeding would be to formalize a mechanism whereby the “system” (at large) would force, in one way or another, people who are not allowed to access a building not to keep on *indefinitely* trying to get in (meeting indefinitely with refuse). In the same way, should the system force, in one way or another, those who have the right, not to give up at the last minute (thus provoking *indefinitely* a new blockage). This type of drastic behaviour, in all evidence, is not part of the specification of the system we are analysing.

A second way of proceeding, more gentle but also more efficient, would consist in eliminating from the system all the persons who tend to behave in this way too frequently. They would simply no longer be allowed to enter any building at all. Their authorization to enter any building at all would be taken away. These persons would therefore be confined to the building in which they find themselves, for example “outside”. This is very easy to formalize and then to realise: moreover, this is the situation to be found in most smart card systems. For example, after three successive unfruitful tries with a cash dispenser, the card is “snatched”, which is a very efficient way of preventing the person in question from blocking indefinitely the access to the dispenser. Note however that such a drastic solution has its drawback in our case: the persons whose card has been removed simply cannot leave the building where this has happened, causing yet another safety problem.

---

<sup>1</sup> We must in fact remember a construction rule for this type of formalization, in which new events which occur during the development, are “fair” compared to more abstract events. This can be rationalized in the following way : if the more abstract system proposes a possible observation of certain events, we must make sure that any refinement really guarantees the conservation of this abstract observability. In other words, we must be able to prove that the new events which can appear during the refinements, cannot prevent the (refinements of) more abstract events from happening.

### *Final Decision*

At the issue of this discussion we decide not to envisage this last possibility which would make the card readers too complicated (too expensive) and would introduce an extra safety problem. In other words, we accept, for financial (and safety) reasons, a risk of indefinite obstruction, which, despite everything, will probably never happen in reality. The following decision ensures:

D3:

The system we are going to construct will not prevent people from blocking doors indefinitely either by trying indefinitely to enter buildings into which they are not authorized to enter, or by abandoning "on the way" their intention to enter the buildings in which they are in fact authorized to enter.

Clearly, the system we are going to construct is thus not totally correct according to our theoretical criteria. We accept this, but, and this is very important, we have taken great care to make it known, and to point it out explicitly by a clearly expressed decision.

### **Third Refinement**

#### *Extension of the Informal Specification*

We now introduce the card readers into the model. This is the first time we will have been taking into account such a material element. This device can be characterized by: (i) the capture of information which is read on the card introduced by the user and (ii) the expedition of this information towards the controlling computer by the means of a networked message. Moreover, when a card is read, it can be supposed that the reader is physically blocked (the slot is obstructed) until it receives an acknowledgement message coming from the control system. All this corresponds to the following behavioural decision for card readers:

D4:

Each card reader is supposed to stay physically blocked (slot obstructed) between the moment when the contents of a card is sent to the system and the reception by this reader of the corresponding acknowledgement. This comes when the pass protocol has been entirely completed (successfully or not)

By this decision, we are making sure that no-one will be able to introduce a card into a reader at random. It is to be noted that we must pay a certain price for this, that of the installation of readers with obstructable slots (they do not all belong to this category).

#### *Assumptions Concerning the Communication Network*

We will only make minimal assumptions concerning the forwarding of messages through the network. For example, we suppose that the network does not guarantee that the messages

be received in the order in which they have been sent. However, it can be supposed that the messages sent along the network are neither lost, nor modified, nor duplicated. Of course, we could take into account these particular constraints but then the model would be more complicated. In any case, such constraints would only be introduced during ulterior refinements.

#### *Variables and Invariant*

We will identify in the model each physical reader with the door it is associated to. Therefore, it is not necessary to have a particular set for the readers. The set of blocked readers is represented by a subset of doors which we will name *BLR* (for blocked readers).

The messages which are sent from the readers towards the control system are “door-person” pairs represented collectively by the variable *mCard* (each one of these pairs  $(q, p)$  represents what the reader associated with the door  $q$  has read on the card of the person  $p$ ). They make up a partial function of the doors towards the persons (it is in fact an invariant of the system which will have to be demonstrated: intuitively, this comes from the fact that no reader can implicate more than one person in the messages it sends because its slot is obstructed as seen above). However, this is not an injective function because nothing prevents a person from sliding his card into another reader when he has not gone through the door associated to the first one and the 30 seconds corresponding to this are not over (this is a case of strange behaviour that cannot be eliminated).

Lastly, the set of acknowledgement messages is represented by the set *mAckn*, which is therefore a subset of doors. These elements are formally defined as follows:

$$\begin{array}{l} BLR \subseteq dor \\ mCard \in dor \leftrightarrow prs \\ mAckn \subseteq dor \end{array}$$

While a reader is obstructed, the door in question is in one of the four following *exclusive* situations : (1) it is consigned in an input message as yet untreated by the system, (2) its green light is on, (3) its red light is on, (4) it is consigned in an acknowledgement message as yet untreated by the reader. These different states characterize the progression of the information through the system. They correspond to following supplementary invariant:

$$\begin{array}{l} \text{dom}(mCard) \cup grn \cup red \cup mAckn = BLR \\ \text{dom}(mCard) \cap (grn \cup red \cup mAckn) = \emptyset \\ mAckn \cap (grn \cup red) = \emptyset \end{array}$$

Since we already know that the sets *grn* and *red* are disjoint, we can say that the four sets  $\text{dom}(mCard)$ , *grn*, *red*, and *mAckn* form a partition of the set *BLR*.

## Events

The new event we are introducing at this stage is the one which corresponds to the reading of a card. It is a “physical” event:

```

CARD ≐
  ANY  $p, q$  WHERE
     $p \in prs$ 
     $q \in dor - BLR$ 
  THEN
     $BLR := BLR \cup \{q\} \parallel$ 
     $mCard := mCard \cup \{q \mapsto p\}$ 
  END

```

Note the the guard  $q \in dor - BLR$  indicates that no card can be introduced in the reader while ist is blocked. This is a “physical” guard.

We can now find the refinements of two events **accept** and **refuse**. They are almost identical in their previous versions except that now the implied elements  $p$  and  $q$  are those read on a message coming from a card reader :

```

accept ≐
  ANY  $p, q$  WHERE
     $(q, p) \in mCard \wedge$ 
     $admitted(p, q)$ 
  THEN
     $dap(p) := q \parallel$ 
     $mCard := mCard - \{q \mapsto p\}$ 
  END

```

```

refuse ≐
  ANY  $p, q$  WHERE
     $(q, p) \in mCard \wedge$ 
     $\neg admitted(p, q)$ 
  THEN
     $red := red \cup \{q\} \parallel$ 
     $mCard := mCard - \{q \mapsto p\}$ 
  END

```

Note that the message  $(q, p)$ , once read, is removed from the “channel”  $mCard$ .

The event **pass** is almost identical to its previous versions. The reading of the card is only confirmed by the dispatching of the corresponding acknowledgement message towards the corresponding reader by means of the “channel”  $mAckn$ .

```

pass ≐
  ANY  $q$  WHERE
     $q \in grn$ 
  THEN
     $sit(dap^{-1}(q)) := dst(q) \parallel$ 
     $dap := dap \triangleright \{q\} \parallel$ 
     $mAckn := mAckn \cup \{q\}$ 
  END

```

Likewise the two events **off\_grn** and **off\_red** also contain the dispatching of an acknowledgement message to the card reader.

```

off_grn  $\hat{=}$ 
  ANY  $q$  WHERE
     $q \in grn$ 
  THEN
     $dap := dap \triangleright \{q\} \parallel$ 
     $mAckn := mAckn \cup \{q\}$ 
  END

```

```

off_red  $\hat{=}$ 
  ANY  $q$  WHERE
     $q \in red$ 
  THEN
     $red := red - \{q\} \parallel$ 
     $mAckn := mAckn \cup \{q\}$ 
  END

```

Lastly, a new physical event ACKN closes the protocol by unblocking the corresponding reader:

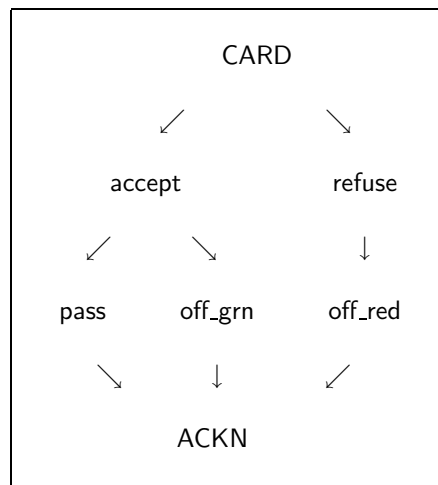
```

ACKN  $\hat{=}$ 
  ANY  $q$  WHERE
     $q \in mAckn$ 
  THEN
     $BLR := BLR - \{q\} \parallel$ 
     $mAckn := mAckn - \{q\}$ 
  END

```

### *Synchronisation*

The various events of this refinement are now synchronised as follows:



### *Proofs*

The proof of this refinement does not induce any particular problem.

## **Fourth Refinement**

### *Extension of the Informal Specification*

We are now introducing the physical controls of the door (blocking and acceptance) as well as those for the detection of pass. We are also taking the following decision concerning “local” behaviour of doors.

D5: When a door has been cleared, it blocks itself automatically without any intervention from the control system.

We are also making another important decision, which is the following:

D6: It is supposed that each door incorporates a local clock which assures temporized blocking after 30 seconds together with the extinction of the green light, or the extinction of the red light after 2 seconds.

*Variables and Invariant: the Green Chain*

Formalization takes place thanks to a certain number of new variables. First the set  $mAccept$  of messages sent by the control system to accept the doors. As we have seen above with the card readers, the set of accepted doors is introduced: it is called  $GRN$  since it corresponds to the doors whose green light is physically on. Then we have the set  $mPass$  of messages sent by each door after detection of clearing. Lastly, we have the set  $mOff-grn$  of messages sent by the doors to signal automatic reblocking after 30 seconds. The following invariant is obtained:

$$\begin{aligned} mAccept &\subseteq dor \\ GRN &\subseteq dor \\ mPass &\subseteq dor \\ mOff-grn &\subseteq dor \end{aligned}$$

These four sets are exclusive and their union is equal to the set  $grn$  of doors whose green light is logically on. As before, these properties show the progression of the information.

$$\begin{aligned} mAccept \cup GRN \cup mPass \cup mOff-grn &= grn \\ mAccept \cap (GRN \cup mPass \cup mOff-grn) &= \emptyset \\ GRN \cap (mPass \cup mOff-grn) &= \emptyset \\ mPass \cap mOff-grn &= \emptyset \end{aligned}$$

*Variables and Invariant: the Red Chain*

In a completely symmetrical way to the “green chain”, we are now going to study the “red chain”. The following variables are to be found. First the set  $mRefuse$  of messages used

to send to a door the order to put on its red light. Then the set  $RED$  of doors whose red light is physically on. Finally we have the set of messages  $mOff\_red$  used for sending the corresponding information to the part of the software concerned with the extinction of the red light. These last messages are sent automatically by the door 2 seconds after the red light goes on. The following invariant is obtained :

$$\begin{array}{l} mRefuse \subseteq dor \\ RED \subseteq dor \\ mOff\_red \subseteq dor \end{array}$$

These three sets are exclusive and their union is equal to the set  $red$  of doors whose red light is logically on. As before, these properties show the progression of information.

$$\begin{array}{l} mRefuse \cup RED \cup mOff\_red = red \\ mRefuse \cap (RED \cup mOff\_red) = \emptyset \\ RED \cap mOff\_red = \emptyset \end{array}$$

#### Events

Let us now consider the events. Those which correspond to card readers which have not been changed in this refinement will not be copied here. Inversely, the event `accept` has been slightly modified . The sending of a physical acceptance message to the doors has been added :

$$\begin{array}{l} \text{accept} \hat{=} \\ \text{ANY } p, q \text{ WHERE} \\ \quad (q, p) \in mCard \wedge \\ \quad \text{admitted}(p, q) \\ \text{THEN} \\ \quad dap(p) := q \parallel \\ \quad mCard := mCard - \{q \mapsto p\} \parallel \\ \quad mAccept := mAccept \cup \{q\} \\ \text{END} \end{array}$$

We now find the physical event of acceptance to a door and the physical lighting of a green light :

$$\begin{array}{l} \text{ACCEPT} \hat{=} \\ \text{ANY } q \text{ WHERE} \\ \quad q \in mAccept \\ \text{THEN} \\ \quad GRN := GRN \cup \{q\} \parallel \\ \quad mAccept := mAccept - \{q\} \\ \text{END} \end{array}$$

It is interesting to remark the gap between the logical acceptance of the door (**accept** event in the software) and the physical acceptance (event **ACCEPT** of the hardware). This gap evokes a major problem of distributed systems which is that of distinguishing between the intention (software) and the real action (hardware). We now find the physical event corresponding to the clearing of the door. It is to be noted that the door does not “know” who is clearing it.

```

PASS ≐
  ANY  $q$  WHERE
     $q \in GRN$ 
  THEN
     $GRN := GRN - \{q\} ||$ 
     $mPass := mPass \cup \{q\}$ 
  END

```

This physical pass is followed by a logical pass which is almost identical to its version during the previous refinement. The only difference corresponds to the fact that the launching of this event is now due to the reception of a message. It is to be noted here that the event **pass** “knows” who is passing: we are dealing with the person implied in acceptance of the door. Once again we can remark a gap between physical detection (**PASS** event of the hardware) and its logical effect (**pass** event of the software).

```

pass ≐
  ANY  $q$  WHERE
     $q \in mPass$ 
  THEN
     $sit(dap^{-1}(q)) := dst(q) ||$ 
     $dap := dap \triangleright \{q\} ||$ 
     $mAckn := mAckn \cup \{q\} ||$ 
     $mPass := mPass - \{q\}$ 
  END

```

The event which consists in physically blocking the door (from a clock supposedly inside the door which starts up 30 seconds after its acceptance if no-one has cleared it in the meantime) is the following. The message of reblocking is sent to the software.

```

OFF_GRN ≐
  ANY  $q$  WHERE
     $q \in GRN$ 
  THEN
     $GRN := GRN - \{q\} ||$ 
     $mOff-grn := mOff-grn \cup \{q\}$ 
  END

```

Finally, we can find a new version of the event **off\_grn**, which gets underway on reception of the previous message.

```

off_grn ≐
  ANY q WHERE
    q ∈ mOff_grn
  THEN
    dap := dap ▷ {q} ||
    mAckn := mAckn ∪ {q} ||
    mOff_grn := mOff_grn - {q}
  END

```

The event `refuse` is slightly modified so as to allow a message concerning the lighting of the red light to be sent.

```

refuse ≐
  ANY p, q WHERE
    (q, p) ∈ mCard ∧
    ¬admitted(p, q)
  THEN
    red := red ∪ {q} ||
    mCard := mCard - {q ↦ p} ||
    mRefuse := mRefuse ∪ {q}
  END

```

The first hardware event after this corresponds to the reception of the previous message and the effective lighting up of the red light. The automatic extinction of the red light after 2 seconds corresponds to the following second event which sends a message to the software so as to warn it.

```

REFUSE ≐
  ANY q WHERE
    q ∈ mRefuse
  THEN
    RED := RED ∪ {q} ||
    mRefuse := mRefuse - {q}
  END

```

```

OFF_RED ≐
  ANY q WHERE
    q ∈ RED
  THEN
    RED := RED - {q} ||
    mOff_red := mOff_red ∪ {q}
  END

```

The event `off_red` is slightly modified as regards its previous version: it is now set off by the reception of the previous message

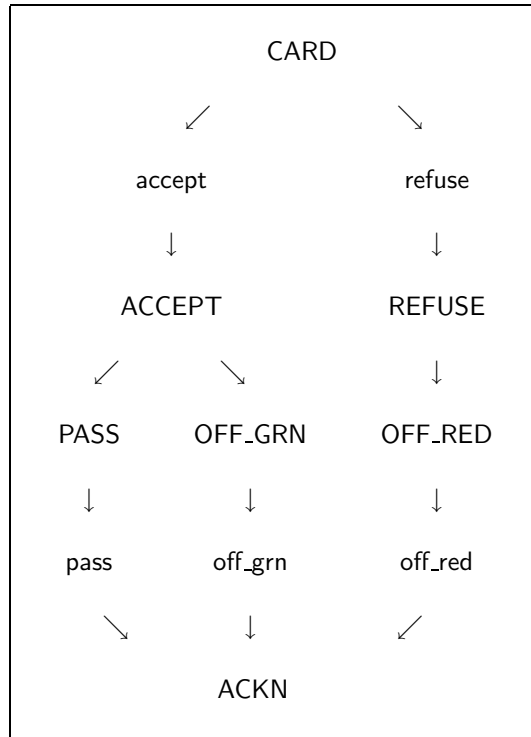
```

off_red ≐
  ANY q WHERE
    q ∈ mOff_red
  THEN
    red := red - {q} ||
    mAckn := mAckn ∪ {q} ||
    mOff_red := mOff_red - {q}
  END

```

### *Synchronisation*

We now obtain the following complete synchronisation between the software and the hardware :



### *Proofs*

The proof of this refinement does not introduce any particular difficulties.

### **Fifth Refinement**

#### *Decomposition*

We are now going to proceed to the decomposition of our system into three separate entities: the software, the network and the hardware. Indeed, our set of events is going to install itself on three abstract machines corresponding to diverse entities. First, here is the sharing out of data between the three machines:

$$\begin{aligned}
& aut \in prs \leftrightarrow bld \\
& org \in dor \rightarrow bld \\
& dst \in dor \rightarrow bld \\
& aut \subseteq aut; dst^{-1}; org \\
& sit \in prs \rightarrow bld \\
& dap \in prs \rightsquigarrow dor \\
& red \subseteq dor
\end{aligned}$$

**Software Data**

$$\begin{aligned}
& mCard \in dor \rightsquigarrow prs \\
& mAckn \subseteq dor \\
& mAccept \subseteq dor \\
& mPass \subseteq dor \\
& mOff_grn \subseteq dor \\
& mRefuse \subseteq dor \\
& mOff_red \subseteq dor
\end{aligned}$$

**Network Data**

$$\begin{aligned}
& BLR \subseteq dor \\
& GRN \subseteq dor \\
& RED \subseteq dor
\end{aligned}$$

**Hardware Data**

A list can be found below of the operations of the two machines **software** and **hardware**. It is to be remarked that only the “software operations” are translated, in fact, by the code. The “hardware operations” represent *globally* the internal behaviour of all the hardware: card readers, doors, lights.

$$\begin{aligned}
& test\_soft(p, q) \\
& accept\_soft(p, q) \\
& refuse\_soft(q) \\
& pass\_soft(q) \\
& off\_grn\_soft(q) \\
& off\_red\_soft(q)
\end{aligned}$$

**Software Operations**

$$\begin{aligned}
& (p, q) \leftarrow CARD\_HARD \\
& ACCEPT\_HARD(q) \\
& REFUSE\_HARD(q) \\
& q \leftarrow PASS\_HARD \\
& q \leftarrow OFF\_GRN\_HARD \\
& q \leftarrow OFF\_RED\_HARD \\
& ACKN\_HARD(q)
\end{aligned}$$

**Hardware Operations**

Then the operations of the **network** machine present themselves. We have distinguished below (i) the operations which link the software to the network, prefixed by **read** or by **write**, and (ii) those which link the hardware to the network, prefixed by **RCV** and **SND**. As can be seen, these operations are symmetrical: each software **read** operation corresponds to a hardware operation **SND** and each software operation **write** corresponds to a hardware operation **RCV**.

$(p, q) \leftarrow \text{read\_card}$	$\text{SND\_CARD}(p, q)$
$\text{write\_accept}(q)$	$q \leftarrow \text{RCV\_ACCEPT}$
$\text{write\_refuse}(q)$	$q \leftarrow \text{RCV\_REFUSE}$
$q \leftarrow \text{read\_pass}$	$\text{SND\_PASS}(q)$
$q \leftarrow \text{read\_off\_grn}$	$\text{SND\_OFF\_GRN}(q)$
$q \leftarrow \text{read\_off\_red}$	$\text{SND\_OFF\_RED}(q)$
$\text{write\_ackn}(q)$	$q \leftarrow \text{RCV\_ACKN}$

**Network-Software Operations**

**Network-Hardware Operations**

### *Events*

We are now going to present the different events of this fifth refinement, events which have now been “implemented” thanks to the operations we have just presented. The two events **accept** and **refuse** have been merged into a single one.

```

CARD ≐
  VAR p, q IN
    (p, q) ← READ_HARD;
    SND_CARD(p, q)
  END

```

```

accept_refuse ≐
  VAR p, q, b IN
    (p, q) ← read_card;
    b ← test_soft(p, q);
    IF b = true THEN
      accept_soft(p, q);
      write_accept(q)
    ELSE
      refuse_soft(q);
      write_refuse(q)
    END
  END

```

```

ACCEPT ≐
  VAR q IN
    q ← RCV_ACCEPT;
    ACCEPT_HARD(q)
  END

```

```

REFUSE ≐
  VAR q IN
    q ← RCV_REFUSE;
    REFUSE_HARD(q)
  END

```

```

PASS ≐
  VAR q IN
    q ← PASS_HARD;
    SND_PASS(q)
  END

```

```

OFF_GRN ≐
  VAR q IN
    q ← OFF_GRN_HARD;
    SND_OFF_GRN(q)
  END

```

```

OFF_RED ≐
  VAR q IN
    q ← OFF_RED_HARD;
    SND_OFF_RED(q)
  END

```

```

pass ≐
  VAR q IN
    q ← read_pass;
    pass_soft(q);
    write_ackn(q)
  END

```

```

off_grn ≐
  VAR q IN
    q ← read_off_grn;
    off_grn_soft(q);
    write_ackn(q)
  END

```

```

off_red ≐
  VAR q IN
    q ← read_off_red;
    off_red_soft(q);
    write_ackn(q)
  END

```

```

ACKN ≐
  VAR q IN
    q ← RCV_ACKN;
    ACKN_HARD(q)
  END

```

It is to be noted that the software events `accept_refuse`, `pass`, `off_grn` and `off_red` make up the *communication layer* of our software. These events all begin by a reading operation: they are, in fact, set off by a corresponding interruption coming from the network.

#### Operations of the “Software” Machine

Here now are the operations of the software machine. It is to be remarked that we are now dealing with operations which are all *pre-conditioned* (and no longer *guarded*). These operations constitute the formal specification of the *internal layer* of our software.

```

b ← test_soft(p, q) ≐
  PRE
    p ∈ prs ∧
    q ∈ dor
  THEN
    b := bool(org(q) = sit(p) ∧ (p, dst(q)) ∈ aut ∧ p ∉ dom(dap))
  END

```

```

accept_soft(p, q) ≐
PRE
  p ∈ prs - dom(dap) ∧
  q ∈ dor - ran(dap)
THEN
  dap(p) := q
END

```

```

refuse_soft(q) ≐
PRE
  q ∈ dor
THEN
  red := red ∪ {q}
END

```

```

pass_soft(q) ≐
PRE
  q ∈ ran(dap)
THEN
  sit(dap-1(q)) := dst(q) ||
  dap := dap ▷ {q}
END

```

```

off_grn_soft(q) ≐
PRE
  q ∈ ran(dap)
THEN
  dap := dap ▷ {q}
END

```

```

off_red_soft(q) ≐
PRE
  q ∈ dor
THEN
  red := red - {q}
END

```

### Operations of the "Network" Machine

You will find below the linking operations between the software and the network. Of course, these operations will be coded manually because they depend too much on particularities of the operating system in which it is functioning. Nevertheless, the proof of the fifth refinement has been made with them.

```

(p, q) ← read_card ≐
ANY a, b WHERE
  (b, a) ∈ mCard
THEN
  mCard := mCard - {b ↦ a} ||
  p, q := a, b
END

```

```

write_accept(q) ≐
PRE
  q ∈ dor
THEN
  mAccept := mAccept ∪ {q}
END

```

```

write_refuse(q) ≐
PRE
  q ∈ dor
THEN
  mRefuse := mRefuse ∪ {q}
END

```

```

q ← read_pass ≐
ANY b WHERE
  b ∈ mPass
THEN
  mPass := mPass - {b} ||
  q := b
END

```

```

q ← read_off_grn ≐
  ANY b WHERE
    b ∈ mOff_grn
  THEN
    mOff_grn := mOff_grn - {b} ||
    q := b
  END

```

```

q ← read_off_red ≐
  ANY b WHERE
    b ∈ mOff_red
  THEN
    mOff_red := mOff_red - {b} ||
    q := b
  END

```

```

write_ackn(q) ≐
  PRE
    q ∈ dor
  THEN
    mAckn := mAckn ∪ {q}
  END

```

We are now getting to the operations between the linking network and the hardware. They will not receive an implementation in the form of a code. They will only serve as models and for making proofs.

```

SND_CARD(p, q) ≐
  PRE
    p ∈ prs ∧
    q ∈ dor
  THEN
    mCard := mCard ∪ {q ↦ p}
  END

```

```

q ← RCV_ACCEPT ≐
  ANY b WHERE
    b ∈ mAccept
  THEN
    mAccept := mAccept - {b} ||
    q := b
  END

```

```

q ← RCV_REFUSE ≐
  ANY b WHERE
    b ∈ mRefuse
  THEN
    mRefuse := mRefuse - {b} ||
    q := b
  END

```

```

SND_PASS(q) ≐
  PRE
    q ∈ dor
  THEN
    mPass := mPass ∪ {q}
  END

```

```

SND_OFF_GRN(q) ≐
  PRE
    q ∈ dor
  THEN
    mOff_grn := mOff_grn ∪ {q}
  END

```

```

SND_OFF_RED(q) ≐
  PRE
    q ∈ dor
  THEN
    mOff_red := mOff_red ∪ {q}
  END

```

```

q ← RCV_ACKN ≐
  ANY b WHERE
    b ∈ mAckn
  THEN
    mAckn := mAckn - {b} ||
    q := b
  END

```

Operations of the “Hardware” Machine

Finally, we only need to make clear the operations corresponding to the hardware machine.

```

(p, q) ← READ_HARD ≐
  ANY a, b WHERE
    a ∈ prs ∧
    b ∈ dor - BLR
  THEN
    BLR := BLR ∪ {b} ||
    p, q := a, b
  END

```

```

ACCEPT_HARD(q) ≐
  PRE
    q ∈ dor
  THEN
    GRN := GRN ∪ {q}
  END

```

```

REFUSE_HARD(q) ≐
  PRE
    q ∈ dor
  THEN
    RED := RED ∪ {q}
  END

```

```

q ← PASS_HARD ≐
  ANY b WHERE
    b ∈ GRN
  THEN
    GRN := GRN - {b} ||
    q := b
  END

```

```

q ← OFF_GRN_HARD ≐
  ANY b WHERE
    b ∈ GRN
  THEN
    GRN := GRN - {b} ||
    q := b
  END

```

```

q ← OFF_RED_HARD ≐
  ANY b WHERE
    b ∈ RED
  THEN
    RED := RED - {b} ||
    q := b
  END

```

```

ACKN_HARD(q) ≐
  PRE
    q ∈ dor
  THEN
    BLR := BLR - {q}
  END

```

## Assessment and Conclusion

We have, in actual fact, determined seventeen properties and taken six important decisions, one concerning a *generalization of the system* (access between buildings), two concerning negative choices in relation with *restrictive use* of the system (people cannot necessarily get “out”, and obstructions for card readers are not to be completely struck off), and three concerning *options about the hardware* (automatic blocking of readers and doors, and setting up of clocks on doors). Of course, these decisions can be contested: the important point is that they have been clearly identified.

We could have studied another form of architecture. It could have corresponded to a *hardware simplification* by annulling the decision to put clocks on doors. In fact, all the timing problems could have been centralized to the main computer. We would then have remarked a complexification of the software, due to the difficulty of fine synchronising with the hardware. The development technique would have consisted in planning first to put clocks into the doors as we have done here, then in refining the system by moving the clocks from the doors to the central computer: difficulties would have appeared at that moment.

## Appendix: Presentation of the Formal Frame

In this first section, we will briefly remind you of the organisation of a formal methodological apparatus, as well as all of the statements, which will permit the setting up and validation of the System Studies in general. This frame has already been presented in detail in different articles [1] [2] [3] [4] [5], to which we refer the interested reader.

### *The Models*

The formal part of a System Study is made up of a suite of *mathematical models*. Each model is supposed to *refine* the previous one. This suite demonstrates the progressive taking into account of the *global system* which we want to study.

A model is first presented in the form of a definition of a certain number of *constants* and of *variables* which make up what we call its *state*. In practical terms, these variables or constants mainly show simple mathematical objects: sets, binary relations, functions. These variables and constants are, in other respects, constrained by a certain number of conditions expressing the *invariant properties* of the model.

These variables and constants, linked by the invariant properties in question, describe the state of the dynamic system which is to be analyzed. They can represent “data” of very diverse nature. For example, they can have a certain counterpart in future software, but they can also correspond to the formalization of certain hardware elements, or even to messages transiting via the network.

As can be seen, a System Study is, a priori, about much more than the software part of a system, even if it is complex. It is essentially about the coupling of the software and its environment, or even better, the progressive decomposition of a system from its global vision which will give birth to these two entities. Indeed, the System Studies aims at analyzing software environment in as detailed a way as it is itself. Moreover, it can be said that the

distinction between software and hardware does not have much meaning at this level.

### *The Events*

Then the model contains a certain number of *events* which show the way a system evolves. These events are only supposed to be *observable*, they are in no way to be detached by any kind of external mechanism such as a “knob”. Indeed, we are not describing the realization here, all be it abstract, of a system. We are rather making a *mathematical simulation*, which allows us to reason about the future system we are going to construct. This reasoning is precisely what is going to allow us to analyze very early on the behaviour of our future system and to draw up a possible *architecture*.

Each event is composed of a *guard* and an *action*. The guard is the condition under which the event can be *set off*. The action, as its name indicates, determines the way in which the state variables are going to evolve when the event is set off.

Each event happens upon the “spontaneous” arrival of a *setting off element* which is unknown. In other words, once again, the real cause of the setting off is of no interest for us, only the observation of what happens when it occurs retains our attention. The arrival of this setting off element can only occur, of course, if it is permitted by the guard of the event.

It is possible for several events to be set off simultaneously (their guards all being true). In this case, however, a single setting off element will “win” and, therefore, only one of these eligible events will really occur, without being able to predict which one it will be. From this point of view, our models present a certain *external non-determinism*. It is also to be noted that it is not impossible that an eligible event might never happen, we will come to this question further on.

Lastly, we must observe that the events are, a priori, *essentially asynchronous*. The only indirect synchronism which could possibly come into play between the events is the consequence of the actions of some of them on the guards of certain others.

### *Practical Form of an Event*

Practically speaking, an event is presented in the following form:

xxx $\hat{=}$ ANY $x, y, \dots$ WHERE $P(x, y, \dots, v, w, \dots)$ THEN $S(x, y, \dots, v, w, \dots)$ END
---

Where the identifiers below have the following signification:

- xxx is the name of an event,
- $x, y, \dots$  denotes a certain number of variables local to the event,
- $v, w, \dots$  denotes a certain number of variables or state constants of the model,
- $P(x, y, \dots, v, w, \dots)$  is a condition,
- $S(x, y, \dots, v, w, \dots)$  is the action associated to the event.

In this case, the guard of the event corresponds to the following existential predicate:

$$\exists(x, y, \dots) \cdot P(x, y, \dots, v, w, \dots)$$

In other words, the necessary (but insufficient) condition for an event xxx to take place with the current value of the state variables  $v, w, \dots$  of the model, is that it be possible to assign to the local variables  $x, y, \dots$ , of the event xxx, values making the predicate  $P(x, y, \dots, v, w, \dots)$  true. As can be seen xxx presents a certain latitude in the choice of possible values for the local variables  $x, y, \dots$ . We can speak here about *internal non-determinism*.

Usually, the action presents itself in the form of a simple simultaneous assignment of certain state variables (it is to be noted that those which are not concerned do not change). These modifications depend, generally, on the values of the state and local variables which are, let us remind ourselves, constrained by the condition  $P(x, y, \dots, v, w, \dots)$ .

#### *Addition of New Events*

During a refinement, it is possible to add new events. Their purpose is to formalize certain functionalities, which had not been taken care of by the model up to that point. They permit a fining down of the “grain” with which the temporal apportionment of the system being studied is measured: in the more abstract models, this grain is very rough, but it gets finer as the series of refinements progresses.

#### *Final Decomposition*

The last refinement in the suite envisaged above is followed by the *decomposition* of the system into several completely separate entities which, nevertheless, communicate with each other. We have obtained an architecture. The *software* is generally found on one side, and the *hardware* on the other, between them the *network* which permits them to communicate. These entities, in particular those corresponding to the software, will then be able to be developed independently.

## **The Proofs**

We now present the different proofs which it may be necessary to carry out on the description of the models we have just proposed.

#### *Proofs of Preservation of the Invariants*

A first series of statements of proofs concerns the preservation of invariants for each of the events of the model. It must be proved that the action associated to each one of them modifies the variables in question in such a way that the invariant properties are true, and that, in the hypothesis of these same properties and the condition associated to the guard of the event.

#### *Proofs of Correct Refinement*

Each model in the suite, except, of course, the first one, is, as was stated above, a refinement of the preceding one. This refinement finds its concrete expression, among others, via

a condition linking the variables of the abstract model (the previous one) to those of the concrete model (the one we are dealing with). The proof of a correct refinement for each event takes place under the hypothesis of this condition.

We must first demonstrate that the *concrete guard is stronger than the abstract one*: the consequence of this is that a concrete event happens *less often* than its more abstract homologue (we will come back to this in the next paragraph). We must also prove that the concrete action has the “same” effect as the abstract action on the state variables of the action (with, however, perhaps less internal non-determinism).

As for the new events, they must refine the “event” which does nothing (skip). In other words, the new events have no effect on the abstract variables.

#### *Proofs of the Limitation of the Reinforcement of the Guards of Refined Events*

We saw above that the refinement of an event was accompanied by the reinforcement of its guard. This guard could be reinforced so much during the course of a refinement that it would become identically false: the refined event would simply no longer take place. This is obviously not what we are looking for.

In fact, the refinement of the guard of a concrete event as regards its abstraction must be compensated for by a possible setting off of new events (that is how they appear as the manifestation of a finer granularity of time). To do that, it must be proved that the abstract guard of an event (which is already weaker than its concrete counterpart as we saw in the last paragraph) is, in fact, stronger than the disjunction of the concrete guard and that of the guards of the new events. In other words, a concrete event occurs less often than its associated abstract event because new events can be set off instead.

The result is that in the case of a refinement which is not accompanied by new events (this is always possible), the abstract and concrete guards of each event are identical.

#### *Proofs of the Impossibility of Monopoly of New Events*

Finally, it is necessary to prove that the new events can, in no way, prevent indefinitely former ones from occurring. In the opposite case the concrete model would not be faithful to its abstraction which allowed them to happen under certain conditions.

This is not a problem linked to the reinforcement of the concrete guards of former events. It is rather an overlapping of the guards of these events by the new ones. We wish simply to prevent this overlapping from being permanent. We wish to wipe out the possibility of a possible “starvation” of former events.

## References

1. R.J.R. Back and R. Kurki-Suonio. *Decentralization of Process Nets with Centralized Control*. 2nd ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing (1983)
2. M.J. Butler. *Stepwise Refinement of Communicating Systems*. Science of Computer programming (1996)
3. J.-R. Abrial. *Extending B Without Changing it (for Developing Distributed System)*. First B Conference (H. Habrias editor). Nantes (1996)

4. J.-R. Abrial and L. Mussat. *Specification and Design of a Transmission Protocol by Successive Refinements Using B.* in *Mathematical Methods in Program Development* Edited by M.Broy and B. Schieder. Springer-Verlag (1997)
5. J.-R. Abrial and L. Mussat.. *Introducing Dynamic Constraints in B.* in *B98' Recent Advances in the Development and Use of the B Method* Edited by D.Bert LNCS 1393 Springer (1998)
6. Y. Ledru. et al. *Etude de cas: Systeme de contrôle d'accs (version 1.2)* IMAG Grenoble (1998)