

**Étude Système:  
méthode et exemple**

par J.-R. Abrial

Octobre 1998



# Étude Système: méthode et exemple

J.-R. Abrial

Consultant

## Introduction

Alors que la problématique du *processus de développement* des logiciels a déjà fait l'objet de nombreuses études et semble avoir acquis maintenant une certaine stabilité dans la forme, on ne peut pas en dire autant de ce que l'on appelle les "Études Système".

Sous ce vocable, on a l'habitude de regrouper, de façon relativement vague, toutes les réflexions et décisions situées *en amont* de la rédaction du Cahier des Charges d'un système. Ces études ont, en principe, pour but de conduire à une rationalisation de l'écriture de ce Cahier des Charges et, plus généralement, à l'élaboration de l'architecture d'un système. Elles ont aussi pour but de montrer que le système que l'on a en tête est viable. Elles pourront à ce sujet déboucher d'ailleurs sur la détermination d'un certain nombre de paramètres fondamentaux dont elles fixeront les plages de valeurs.

Ce texte n'a pas du tout la prétention d'épuiser un aussi vaste sujet. Il a seulement l'ambition plus modeste d'exposer, sur un exemple précis, l'application d'une approche nouvelle en la matière. Cette approche fait appel aux techniques des Méthodes Formelles avec Preuves, qui sont de plus en plus utilisées aujourd'hui avec succès pour la réalisation de logiciels industriels.

Le document est organisé de la façon suivante. Nous ferons d'abord une présentation succincte de l'étude de cas. Ceci nous permettra de nous situer immédiatement dans un cadre concret. Nous présenterons ensuite les objectifs de l'Étude Système en termes assez généraux mais néanmoins illustrés par les éléments caractéristiques de l'étude de cas. Puis nous développerons l'analyse complète du système proposé de façon à obtenir une architecture finale tant matérielle que logicielle. Enfin nous dresserons un certain bilan.

Nous avons rejeté dans une annexe l'exposé succinct du cadre méthodologique formel dans lequel nous allons évoluer. Le lecteur peut soit s'y référer d'emblée (elle ne fait que quelques pages), soit s'y consacrer plus tard (la lecture de cette annexe n'est pas strictement indispensable à la compréhension du développement de l'étude de cas)

## Présentation de l'étude de cas

Nous rentrons maintenant dans l'étude de cas qui va nous permettre d'illustrer notre propos. On présente d'abord les principales fonctionnalités de ce projet. On justifie ensuite l'intérêt de l'Étude Système que l'on va entreprendre. À quoi sert-elle ? Que peut-on en attendre ? Puis l'on mène le développement jusqu'à son terme, c'est-à-dire jusqu'à l'obtention d'une architecture complète.

Cette étude de cas a été proposée par différents chercheurs [5] comme banc d'essai de diverses méthodes formelles.

### *But du système*

On désire construire un système chargé de contrôler l'accès de certaines personnes aux divers bâtiments d'un "lieu de travail": campus universitaire, site industriel, enceinte militaire, centre commercial, etc.

### *Autorisation*

Le contrôle s'effectue sur la base de l'autorisation que chaque personne concernée est censée posséder. Cette autorisation doit lui permettre, sous le contrôle du système, de pouvoir pénétrer dans certains bâtiments et pas dans d'autres. Par exemple une certaine personne  $p_1$  est autorisée à pénétrer dans le bâtiment  $b_1$  et pas dans le bâtiment  $b_2$ ; par contre, une autre personne  $p_2$  a le droit de pénétrer dans ces deux bâtiments. Ces autorisations sont données de façon "permanente": autrement dit, elles ne changent pas durant le fonctionnement normal du système.

Lorsqu'une personne se trouve à l'intérieur d'un bâtiment, sa sortie doit également être contrôlé par le système de façon à ce qu'il soit possible de savoir à chaque instant qui se trouve dans un bâtiment donné.

### *Cartes magnétiques*

Chaque personne reçoit une carte magnétique qui lui est assignée en propre au moyen d'un identificateur unique gravé sur celle-ci. Des lecteurs de cartes sont installés à chaque entrée et à chaque sortie de bâtiment. À proximité de chaque lecteur, on trouve deux voyants: un voyant rouge et un voyant vert. Chacun de ces voyants peut être allumé ou éteint.

### *Tourniquets*

Le transfert des personnes d'un bâtiment à l'autre s'effectuent grâce à des "tourniquets" qui sont normalement bloqués: personne ne peut les franchir sans le contrôle du système. Lorsqu'un tourniquet est débloqué par le système (voir le paragraphe ci-dessous), le passage éventuel d'une personne est détectée par un capteur. Chaque tourniquet n'est affecté qu'à une seule des deux tâches, entrer ou sortir, il n'existe pas de tourniquet "double sens".

### *Protocole d'accès*

L'entrée dans un bâtiment ou la sortie d'un bâtiment obéit à une procédure systématique composée d'une suite d'événements qui sont les suivants:

- Une personne souhaitant entrer ou sortir d'un bâtiment introduit sa carte dans le lecteur du tourniquet concerné. On se trouve alors devant l'alternative suivante:
  - Si la personne est autorisée à pénétrer dans le bâtiment en question (elle est toujours autorisée à sortir), le voyant vert s'allume et le tourniquet se débloque pour 30 secondes. On se trouve alors en présence de la nouvelle alternative suivante:
    - \* Dès que quelqu'un franchit effectivement le tourniquet avant la fin du laps de temps de 30 secondes le voyant vert s'éteint aussitôt et le tourniquet se bloque.

- \* Si, par contre, 30 secondes passent sans que personne ne franchisse le tourniquet, le voyant vert s'éteint alors et le tourniquet se bloque également.
- Si la personne n'est pas autorisée à pénétrer dans le bâtiment, le voyant rouge s'allume pour deux secondes et, bien sûr, le tourniquet reste bloqué.

## Objectifs de l'Étude Système

L'exposé ci-dessus n'a pas la prétention d'être complet ni d'avoir levé toutes les options techniques. Il ne saurait donc, en l'état, constituer le point de départ de la réalisation du logiciel de contrôle: il reste encore trop d'inconnues concernant, entre autres, le matériel et sa liaison avec le logiciel. Nous précisons ci-dessous quels sont, en l'occurrence, les rôles de l'Étude Système.

### *Répartition du contrôle*

Une importante question que l'on doit se poser au début des réflexions menées au sujet d'un tel système concerne la distribution, plus ou moins importante, du contrôle entre le logiciel et les différents périphériques (tourniquets, lecteurs).

Par exemple, on peut envisager d'avoir un micro-ordinateur sur chaque tourniquet: à ce moment-là, le contrôle est complètement décentralisé. On peut, à l'inverse, n'avoir qu'un seul micro-ordinateur qui centralise entièrement le contrôle. Il existe, bien entendu, des situations intermédiaires où chaque tourniquet possède une certaine autonomie: un exemple typique consiste à équiper chaque tourniquet d'une horloge qui conditionne une partie de son comportement.

### *Construction d'un modèle fermé*

En tout état de cause, l'argumentation technique qui peut conduire à telle ou telle décision en la matière ne peut s'effectuer qu'en analysant le système *dans son ensemble*. À noter que cette argumentation technique doit aussi se nourrir des informations dont on peut disposer concernant les matériels offerts sur le marché (on peut aussi décider de faire réaliser un matériel nouveau).

L'Étude Système va donc consister essentiellement à construire un *modèle fermé* de notre futur système et à *prouver* sur ce modèle que ses *propriétés caractéristiques* (qu'il va donc falloir expliciter avec précision) sont bien assurées.

### *Comportement des matériels*

Un important résultat de l'Étude Système concerne les spécifications comportementales des différents matériels. Pour mener à bien cette étude, on va être amené à introduire dans le modèle un certain nombre d'hypothèses concernant ces matériels. Par exemple, est-ce que le tourniquet se rebloque tout seul après le passage d'une personne, ou bien est-ce que ce blocage ne peut être obtenu qu'après réception par le tourniquet d'un ordre venant du logiciel? Bien entendu, l'option choisie conditionne l'organisation du logiciel. Cette option constitue une hypothèse sous laquelle le modèle du logiciel peut être prouvé fonctionner correctement.

À l'issue de l'Étude Système on aura ainsi effectué un certain nombre de choix de comportement du matériel. Ces choix devront être consignés dans un *Cahier des Charges Matériel* comportant, entre autres, une procédure de recette visant à vérifier que le matériel effectivement mis en place possède bien les propriétés que l'on attend de lui. Si ce n'était pas le cas en effet, il est clair que le mariage du logiciel et du matériel n'aurait aucune chance d'aboutir à un système fonctionnant correctement comme cela aurait été démontré sur le modèle (sous les hypothèses concernées). Cette démonstration n'aurait donc servi à rien.

#### *Généralisation éventuelle du problème*

On peut aussi se poser la question de savoir si le Cahier des Charges ci-dessus n'est pas trop orienté vers un seul type de système de contrôle, à savoir celui de l'accès à des bâtiments *depuis l'extérieur*. Plus généralement, le réalisateur de tels systèmes aurait peut-être intérêt à étudier un dispositif plus général qui saurait aussi s'adapter à des contrôles *entre bâtiments*. On voudrait donc savoir, grâce à l'Étude Système, si une telle généralisation est intéressante. Ce choix sera dicté essentiellement par la facilité de construction du modèle.

#### *Aborder les questions de sécurité*

Une importante question qui n'est pas du tout abordée dans le Cahier des Charges est celle de la sécurité des personnes impliquées dans un tel système. On voudrait que l'Étude Système nous renseigne à ce sujet, ou tout au moins pose un certain nombre de questions pertinentes. Par exemple, est-il possible que des personnes restent bloquées dans un bâtiment? Comment garantir le contraire?

#### *Les problèmes de synchronisation*

Plus techniquement, le Cahier des Charges est silencieux en ce qui concerne le timing fin de l'opération de transfert. Par exemple, quel décalage existe-t-il entre l'allumage du voyant vert, le déblocage du tourniquet et le démarrage du décompte de 30 secondes. Serait-il possible que le feu vert s'allume alors que le tourniquet est encore bloqué, ou que le voyant vert s'éteigne alors que le tourniquet est toujours déblocqué, etc ?

La question n'est pas tant de savoir si les comportements précédents sont bons ou mauvais, elle est plutôt de reconnaître l'existence de ces comportements et de savoir à l'avance s'ils vont pouvoir apparaître (même de façon très fugitive) dans le système final qui sera mis en service.

#### *Fonctionnement à la marge*

Une autre question qui n'est pas non plus abordée dans ce Cahier des Charges est celle du fonctionnement "à la marge" du système. Par exemple, quel est la réaction du système lorsqu'une carte est introduite dans le lecteur alors que les voyants verts ou rouges sont encore allumés (autrement dit, alors que la précédente *transaction* n'est pas terminée) ? Plus généralement, on voudrait pouvoir comprendre et prévoir comment le système réagit devant un comportement "hostile" de certains utilisateurs. Dans ce genre de système, on ne doit pas, en effet, compter sur des hypothèses trop fortes relatives au "bon" comportement des utilisateurs. Certains utilisateurs se comporteront certainement de façon "bizarre".

## Conventions méthodologiques

Nous observerons dans la suite les conventions de style suivantes. Le texte français sera parsemé de différentes “boîtes” qui pourront contenir: (1) soit l'énoncé de certaines *propriétés* décrites au moyen d'une ou deux phrases très courtes, ces boîtes sont numérotées avec le préfixe "P", (2) soit l'énoncé, lui aussi très court, d'une *décision*, ces boîtes sont numérotées avec le préfixe "D", (3) soit la formalisation mathématique d'un *invariant*, (4) soit enfin la description formelle d'un *événement*.

Pour améliorer la lisibilité des modèles, nous suivrons un certain nombre de conventions lexicales simples concernant l'écriture des identificateurs. Nous écrivons les variables ou les constantes correspondant à des éléments physiques (par exemple celles caractérisant les lecteurs de cartes) en n'utilisant que des lettres majuscules. Les variables qui relèvent plutôt du logiciel seront écrites en n'utilisant que des lettres minuscules. Enfin les variables qui caractérisent les messages circulant sur le réseau contiendront aussi bien des lettres majuscules que des lettres minuscules et, de plus, elles commenceront toutes par la lettre “m”.

## Spécification initiale du système

Nous allons maintenant procéder à la spécification initiale de notre système en construisant un premier modèle très abstrait dans lequel la séparation entre le logiciel et le matériel n'est pas encore d'actualité.

### *Spécification informelle initiale*

Nous veillons tout d'abord à déterminer quels sont les acteurs de ce système, acteurs qui vont rentrer dans le modèle sous la forme d'un certain nombre d'ensembles dont les caractéristiques seront précisées petit à petit.

P1: Le modèle comprend des personnes et des bâtiments.

Le système est chargé de veiller à contrôler les autorisations permanentes de rentrée des personnes dans les bâtiments.

P2: Chaque personne est autorisée à pénétrer dans certains bâtiments (et pas dans d'autres). Les bâtiments non consignés dans cette autorisation sont implicitement interdits. Il s'agit d'une affectation permanente.

Une propriété implicite, qu'il est important de mettre en évidence dès le début de la construction du modèle, concerne l'impossibilité pour une même personne de se trouver simultanément dans deux bâtiments distincts.

P3: À un instant donné, une personne se trouve dans un bâtiment au plus.

Une simplification (et une généralisation) du modèle consiste à identifier “l’extérieur” à un bâtiment (dans lequel toute personne a le droit de se trouver!). Ceci se concrétise par la décision suivante, qui donne donc une précision supplémentaire par rapport au Cahier des Charges initial:

D1: Le système gère le passage des personnes d'un bâtiment à l'autre.

Il vient alors la propriété complémentaire suivante:

P4: À un instant donné, une personne se trouve dans un bâtiment au moins.

Nous présentons enfin la propriété principale du système, qui manifeste que le contrôle est correctement réalisé, à savoir que toute personne est bien, à chaque instant, autorisée à se trouver dans le bâtiment dans lequel elle se trouve.

P5: Toute personne se trouvant dans un bâtiment est bien autorisée à y être.

*Exemple*

À titre d’illustration, supposons que nous ayons quatre bâtiments b1,b2,b3,b4 et trois personnes p1,p2,p3 munies des autorisations suivantes:

p1	b2, b4
p2	b1, b3, b4
p3	b2, b3, b4

**Autorisation**

Les situations suivantes représentent alors une évolution satisfaisante du système puisque, comme on peut le constater, les propriétés P1 à P5 sont bien toujours respectées.

p1	b4
p2	b4
p3	b4

Situation 1

p1	b2
p2	b4
p3	b4

Situation 2

p1	b2
p2	b1
p3	b4

Situation 3

p1	b4
p2	b1
p3	b4

Situation 4

p1	b4
p2	b1
p3	b3

Situation 5

*Premier modèle du système: les ensembles de base*

Nous sommes maintenant à même de construire notre premier modèle. Nous appelons *prs* l'ensemble des personnes et *bat* l'ensemble des bâtiments. On suppose, bien entendu, que ces ensembles sont finis et non vides.

$prs \neq \emptyset$
$bat \neq \emptyset$

*Les constantes et les variables*

Les autorisations sont représentées par un ensemble de couples “personne-bâtiment” dont chacun relie une personne à un bâtiment où elle est autorisée à se trouver. Lorsqu'une personne *p* est autorisée à aller dans plusieurs bâtiments, on trouve pour cette personne autant de couples qu'il y a de bâtiments concernés. Cet ensemble de couples, *aut*, est donc une relation binaire entre *prs* et *bat*.

La situation dynamique des gens dans les bâtiments est représentée par une certaine fonction *sit*. Comme la relation *aut* envisagée ci-dessus, cette fonction contient des couples “personne-bâtiment” qui relient chaque personne au bâtiment dans lequel elle se trouve. Le fait qu'il s'agisse d'une fonction correspond à la formalisation de la propriété P3 (une personne dans un bâtiment au plus). Le fait que cette fonction soit totale correspond à la propriété P4 (une personne dans un bâtiment au moins).

$aut \in prs \leftrightarrow bat$
$sit \in prs \rightarrow bat$

*L'invariant du système*

La propriété caractéristique P5 (les personnes présentes dans un bâtiments à un moment donné ont bien l'autorisation de s'y trouver) se formalise en stipulant que la fonction *sit* est incluse dans la relation *aut*. Autrement dit, si le couple  $(p, b)$  appartient à la fonction *sit*

(la personne  $p$  se trouve alors dans un bâtiment  $b$ ) alors ce couple doit aussi appartenir à la relation *aut* pour que  $p$  soit bien autorisée à être dans  $b$ . Il en résulte finalement l'invariant suivant qui doit donc être toujours vérifié en permanence (en dehors des transitions considérées comme intemporelle):

$$\boxed{sit \subseteq aut}$$

### Événement

À ce niveau d'abstraction, nous ne pouvons *observer* qu'une seule vraie transition, qui se manifeste lors de l'arrivée de l'événement **passer**, et qui provoque la rentrée d'une personne  $p$  dans un bâtiment  $b$  (c'est-à-dire  $sit(p) := b$ ). Cet événement ne doit pouvoir se produire (condition nécessaire) que si  $p$  est bien autorisée à se trouver dans  $b$  (c'est-à-dire si l'on a  $(p, b) \in aut$ ) et si, bien entendu, il n'est pas déjà dans  $b$  (c'est-à-dire si l'on a  $sit(p) \neq b$ ).

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

Il faut bien noter que lorsque nous disons que l'événement **passer** peut se produire, cela ne signifie pas nécessairement qu'il va effectivement se produire dans les faits. Nous disons seulement qu'il est déclenchable et donc qu'il *pourrait être observé*.

### Conclusion

Il est facile de *prouver* que l'invariant est bien préservé par la transition ci-dessus lorsqu'elle se produit. On notera que l'événement **passer** est très abstrait: on ne sait pas par quel processus la personne  $p$  rentre dans le bâtiment  $b$ . On ne sait pas non plus si le bâtiment dans lequel se trouve la personne  $p$  communique bien avec ce bâtiment  $b$  dans lequel elle désire se rendre. En fait, nous ne pouvons pas exprimer cette propriété car nous n'avons pas encore formalisé la "géométrie" des bâtiments.

Le seul élément important du présent modèle est l'expression des règles fondamentales du système (dans l'invariant) et la preuve que l'unique événement observable et intéressant à ce niveau (**passer**) maintient bien ces propriétés. Nous sommes donc d'ores et déjà sûr que les modèles qui vont suivre respecteront bien cette propriété si, bien entendu, nous *prouvons* qu'ils constituent des *raffinements corrects* du présent modèle.

## Premier Raffinement

### Extension de la spécification informelle

Nous allons maintenant procéder à notre premier raffinement qui va consister à introduire dans le modèle la notion de *communication possible* entre bâtiments.

P6: La géométrie des bâtiments sert à définir quels bâtiments peuvent communiquer entre eux et dans quel sens.

Il existe une contrainte supplémentaire évidente stipulant que deux bâtiments qui communiquent entre eux sont nécessairement distincts.

P7: Un bâtiment ne communique pas avec lui-même.

La règle suivante indique qu'une personne ne peut pas passer du bâtiment où elle se trouve à n'importe quel autre bâtiment où elle est autorisée à se trouver (comme s'était le cas dans l'abstraction précédente), encore faut-il que ces deux bâtiments communiquent entre eux:

P8: Une personne ne peut se déplacer d'un bâtiment où elle se trouve à un autre où elle désire aller que si ces deux bâtiments communiquent bien entre eux.

### Construction du deuxième modèle: constantes

Dans ce deuxième modèle, on formalise la communication possible entre bâtiments par un ensemble de couples "bâtiment-bâtiment" qui relie chaque bâtiment  $b1$  à un bâtiment  $b2$  avec lequel le premier est censé communiquer. Cet ensemble de couples définit une relation binaire,  $com$ , relation qui est nécessairement non-réflexive pour respecter la propriété P7 (un bâtiment ne communique pas avec lui-même). Les invariants qui en résultent peuvent donc se formaliser comme suit:

$$\begin{aligned} com &\in bat \leftrightarrow bat \\ com \cap id(bat) &= \emptyset \end{aligned}$$

### Événement

À ce niveau, nous ne pouvons toujours observer que le seul événement intéressant `passer`. On notera que la deuxième garde,  $(sit(p), b) \in com$ , de cet événement garantit la satisfaction de la propriété P8 (les bâtiments concernés par le passage d'une personne doivent communiquer entre eux):

```
passer ≐
  ANY p, b WHERE
    (p, b) ∈ aut ∧
    (sit(p), b) ∈ com
  THEN
    sit(p) := b
  END
```

### Preuve de raffinement

Cet événement est bien trivialement un raffinement de la précédente version car l'action est identique dans les deux cas ( $sit(p) := b$ ) et la garde du second, à savoir:

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

est manifestement plus forte que celle du premier, qui est la suivante:

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

Il est clair, en effet, que si la condition  $(sit(p), b) \in com$  est vérifiée (si la personne  $p$  se trouve dans un bâtiment qui communique avec le bâtiment  $b$ ) alors le bâtiment où se trouve cette personne est bien distinct de  $b$  (puisque un bâtiment ne communique pas avec lui-même). On notera donc que la non-réflexivité de la relation  $com$  a son importance dans cette preuve.

### Preuve de non-bloquage

Il nous faut prouver maintenant que, en l'absence d'événements nouveaux dans ce raffinement, l'événement concret **passer** n'arrive pas moins souvent que son homologue abstrait (on rappelle que ceci est une condition nécessaire du raffinement).

En fait, nous nous nous trouvons ici de toute évidence devant une difficulté. Il n'est pas possible, en effet, de prouver que l'événement raffiné **passer** ne se produit pas moins souvent que son homologue plus abstrait. En effet, pour cela il faudrait montrer que la garde de l'événement abstrait implique celle de l'événement concret, soit:

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

Il est clair que cette condition ne peut pas être vérifiée en général. Il n'y a aucune raison, en effet, pour que la condition  $sit(p) \neq b$ , stipulant que le bâtiment où se trouve une certaine personne  $p$  est distinct du bâtiment elle désire aller, n'implique nécessairement l'existence d'une (éventuellement) autre personne  $p$  et d'un (éventuellement autre) bâtiment  $b$  tels que la condition  $(sit(p), b) \in com$  soit remplie, c'est-à-dire que le bâtiment où se trouve cette personne communique avec le bâtiment où elle désire aller.

### Apparition d'un problème lié à la sécurité: nécessité d'une exigence complémentaire

L'impossibilité de prouver la condition ci-dessus indique qu'il se peut que *des personnes restent indéfiniment bloquées dans des bâtiments*. En effet, si une personne se trouve dans un bâtiment  $b$  et qu'elle n'a aucune autorisation lui permettant de se trouver dans un bâtiment qui communique avec  $b$ , elle est bien bloquée dans  $b$ . L'impossibilité de faire la preuve ci-dessus a fait apparaître un problème de sécurité, que l'on peut énoncer sous la forme d'une nouvelle propriété que le système doit satisfaire:

p9: 

Aucune personne ne doit pouvoir rester bloquer dans un bâtiment.
--

### Tentative de solution

Il nous faut donc trouver une contrainte suffisante pour que cette propriété soit pratiquement satisfaite. L'obligation de preuve précédente va nous servir à découvrir cette contrainte. Cette obligation de preuve peut se ré-écrire comme suit:

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

Il nous *suffit* alors de prouver

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

soit, de façon équivalente

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

Pour prouver cette condition, il *suffit* de prouver ce qui suit (puisque la condition  $prs \neq \emptyset$  implique que la fonction totale  $sit$  n'est pas vide):

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

Que dit cette condition ? Si nous la développons, il vient:

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

Autrement dit, le bâtiment où est situé chaque personne  $p$  à un moment donné est en communication avec au moins un autre bâtiment  $b$  où  $p$  a bien l'autorisation d'aller: la personne  $p$  peut donc sortir par  $b$  du bâtiment où elle se trouve.

Cette condition pourrait être imposée comme un nouvel invariant du système: elle nécessiterait cependant de renforcer la garde de l'événement **passer** de façon à n'admettre dans un certain bâtiment que les personnes autorisées à y rentrer (c'est déjà le cas), dont on aurait aussi la garantie qu'elles pourraient bien en ressortir. Dans la négative, l'autorisation de rentrer dans ce bâtiment, autorisation que détiendrait la personne concernée, ne lui servirait pas à grand'chose puisqu'on lui en refuserait l'accès pour la raison qu'elle ne pourrait pas en ressortir. On préférerais donc avoir une condition (suffisante) *indépendante de la situation des gens*. En fait, c'est possible car nous avons déjà la propriété invariante  $sit \subseteq aut$ . Pour prouver la condition  $sit \subseteq (aut; com^{-1})$  ci-dessus, il *suffit* donc, par transitivité de l'inclusion, de prouver:

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

Cette condition constitue donc un invariant supplémentaire, qui peut se traduire ainsi:

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

La lecture de cet invariant est très instructive: on y voit que quelque soit le couple  $(p, b)$  appartenant à  $aut$  (la personne  $p$  pourrait donc se trouver dans le bâtiment  $b$  puisqu'elle est autorisée à y être), il existe un bâtiment  $c$  tel que  $(p, c)$  appartienne à  $aut$  (la même personne  $p$  est donc aussi autorisée à aller dans le bâtiment  $c$ ) et tel que, de plus, le couple  $(b, c)$  appartienne à  $com$  (donc les deux bâtiments  $b$  et  $c$  communiquent). En fin de compte, la personne  $p$ , qui pourrait se trouver dans  $b$ , n'y resterait pas indéfiniment bloquée puisqu'elle est aussi autorisée à aller dans le bâtiment  $c$  qui communique avec  $b$ . Elle peut donc sortir de  $b$  par  $c$ . On satisfait donc la propriété P9 au moyen d'une exigence plus forte dont l'énoncé aura été déterminé (calculé) par l'étude mathématique avant d'être exprimé en français comme suit:

P10:

Toute personne autorisée à se trouver dans un bâtiment doit aussi être autorisée à aller dans un autre bâtiment qui communique avec le premier.

À noter que le problème que nous venons d'évoquer devrait être généralisé à celui de la garantie que toute personne se trouvant dans un bâtiment  $a$ , non seulement la possibilité d'en sortir mais, plus généralement, celle de toujours pouvoir rejoindre un "bâtiment" particulier appelé "dehors", où tout le monde  $a$ , bien entendu, l'autorisation de se trouver. Nous n'abordons pas ce problème ici. Ceci se traduit donc par la décision suivante.

D2: Le système que nous allons construire ne garantit pas que les personnes peuvent toujours rejoindre l'"extérieur". Il garantit seulement que les personnes peuvent toujours passer d'un bâtiment à un autre.

*Retour sur la preuve de non-bloquage*

Sommes-nous maintenant certain que le blocage détecté ci-dessus est bien éliminé? Rappelons que nous devons prouver la propriété suivante:

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

Cette condition est maintenant bien vérifiée car si  $p$  est autorisée à aller dans  $b$  (ceci représente une partie de la condition de gauche) alors il existe bien une personne  $p'$  (la même, en l'occurrence), autorisée à aller dans un certain bâtiment  $b'$  (pas forcément le même que  $b$ ) qui communique avec celui où se trouve  $p'$ , disons  $c$ . Ceci est du au fait que la personne  $p'$  est certainement autorisée à être dans  $c$  puisqu'elle s'y trouve (propriété P5) et que toute personne autorisée à être dans un bâtiment est aussi autorisée à aller dans un autre bâtiment  $b'$  qui est en communication avec lui (propriété P10).

*Exemple*

À titre d'illustration, supposons que les relations  $aut$ ,  $com$  et donc  $com^{-1}$  et  $aut; com^{-1}$  soient définies comme indiqué sur la figure ci-dessous. Comme on peut le voir immédiatement, les couples  $(p1, b2)$ ,  $(p3, b2)$  et  $(p2, b3)$  appartiennent bien à  $aut$  mais pas à  $aut; com^{-1}$ . Cela veut dire que les personnes  $p1$  et  $p3$  sont bien autorisées à aller dans le bâtiment  $b2$  mais qu'elles ne pourront plus en sortir (en fait, ce bâtiment est un cul-de-sac comme on peut le voir dans la relation  $com$ ). De même, la personne  $p2$  a-t-elle l'autorisation d'aller dans le bâtiment  $b3$ , mais elle ne pourra plus en sortir (cette fois-ci, ce bâtiment communique bien avec le seul bâtiment  $b2$ , mais malheureusement,  $p2$  n'a pas l'autorisation d'y aller).

p1	b2	p2	b4
p1	b4	p3	b2
p2	b1	p3	b3
p2	b3	p3	b4

*aut*

b1	b3
b1	b4
b3	b2
b4	b1
b4	b2
b4	b3

*com*

b1	b4
b2	b3
b2	b4
b3	b1
b3	b4
b4	b1

*com*<sup>-1</sup>

p1	b1	p2	b4
p1	b3	p3	b1
p1	b4	p3	b3
p2	b1	p3	b4

*aut; com*<sup>-1</sup>

Une solution consiste donc à ouvrir une porte entre **b2** et **b4** (de cette façon lorsque **p1** ou **p3** sont dans **b2** ils peuvent en sortir par **b4** où ils ont bien l'autorisation d'aller) et à autoriser **p2** à aller dans **b2** (d'où il pourra sortir par la porte que l'on vient juste d'ouvrir vers **b4** où il a aussi l'autorisation d'aller).

p1	b2	p2	b4
p1	b4	p3	b2
p2	b1	p3	b3
p2	b1	p3	b4
p2	b3		

*aut*

b1	b3
b1	b4
b2	b4
b3	b2
b4	b1
b4	b2
b4	b3

*com*

b1	b4
b2	b3
b2	b4
b3	b1
b3	b4
b4	b1
b4	b2

*com*<sup>-1</sup>

p1	b1	p2	b3
p1	b2	p2	b4
p1	b3	p3	b1
p1	b4	p3	b2
p2	b1	p3	b3
p2	b2	p3	b4

*aut; com*<sup>-1</sup>

On rajoute donc les couples (b2,b4) et (p3,b2) à *com* et *aut* respectivement. Il vient alors la configuration précédente qui, comme on peut le constater, donne satisfaction.

## Deuxième Raffinement

### *Extension de la spécification informelle*

Au cours de ce deuxième raffinement, nous allons introduire les portes à sens unique faisant communiquer les bâtiments entre eux:

P11: Les bâtiments communiquent entre eux au moyen de portes, qui sont à sens unique. On peut donc parler des bâtiments origine et destination de chaque porte.

Les portes peuvent être physiquement bloquées. C'est ce qui assure matériellement qu'on ne puisse pas franchir une porte sans contrôle. La propriété suivante précise les modalités de déblocage d'une porte:

P12: Une porte ne peut être franchie que si elle est déblocuée. Une porte ne peut être déblocuée que pour une seule personne à la fois. Inversement toute personne impliquée dans le déblocage d'une porte ne peut pas l'être dans celui d'une autre.

La propriété suivante précise les conditions de déblocage d'une porte pour une personne donnée. *Cette condition constitue l'essence du contrôle exercé par le système.* En effet, une fois qu'une porte est déblocuée pour une personne, son franchissement ne peut plus être physiquement empêché.

P13: Lorsqu'une porte est déblocuée pour une certaine personne, celle-ci se trouve dans le bâtiment origine de la porte en question. Par ailleurs, cette personne est bien autorisée à aller dans le bâtiment destination de cette même porte.

Un voyant vert est associé à chaque porte. Il est allumé et éteint dans les conditions suivantes:

P14: Le voyant vert d'une porte est allumée tant que celle-ci est déblocuée. Dès qu'une personne est passée, la porte se rebloque. Au bout de 30 secondes, si aucune personne ne passe une porte déblocuée, celle-ci se rebloque toute seule. Dans les deux cas, le voyant vert s'éteint.

De même un voyant rouge est-il associé à chaque porte. Il est allumé et éteint dans les conditions suivantes:

P15:

Le voyant rouge d'une porte dont l'accès vient d'être refusé s'allume pour une période de 2 secondes, la porte restant évidemment bloquée.

Des deux conditions précédentes, on peut déduire la propriété suivante:

P16:

Les voyants rouge et vert d'une même porte ne peuvent pas être allumés simultanément.

### *Nouvelles constantes et invariant*

La formalisation passe par l'introduction de l'ensemble fini non vide  $prt$  qui modélise les portes. À chaque porte est associée un bâtiment origine, représenté par la fonction totale  $org$ , et un bâtiment destination, représenté par la fonction totale  $dst$ . Pour toutes ces portes, les bâtiments origine et destination représentent exactement les couples de bâtiments impliqués dans la relation  $com$  introduite au cours du raffinement précédent. Ceci est formalisé comme suit:

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

### *Exemple*

Dans le dernier exemple du précédent raffinement, la relation  $com$  comprenait 7 couples. Nous allons donc avoir 7 portes  $q1$ ,  $q2$ ,  $q3$ ,  $q4$ ,  $q5$ ,  $q6$  et  $q7$  avec les fonctions  $org$ ,  $org^{-1}$  et  $dst$  suivantes (on forme aussi  $org^{-1}; dst$  pour constater son égalité avec  $com$ )

q1	b1
q2	b1
q3	b2
q4	b3
q5	b4
q6	b4
q7	b4

*org*

b1	q1
b1	q2
b2	q3
b3	q4
b4	q5
b4	q6
b4	q7

*org<sup>-1</sup>*

q1	b3
q2	b4
q3	b4
q4	b2
q5	b1
q6	b2
q7	b3

*dst*

b1	b3
b1	b4
b2	b4
b3	b2
b4	b1
b4	b2
b4	b3

*org<sup>-1</sup>; dst*

b1	b3
b1	b4
b2	b4
b3	b2
b4	b1
b4	b2
b4	b3

*com*

*Variables et invariant*

La formalisation de la propriété P12 (débloquage exclusif d'une porte pour une seule personne) s'effectue à l'aide d'une fonction injective partielle,  $pdb$ , qui relie des personnes à des portes. Plus précisément, cette fonction correspond à l'ensemble des couples reliant une personne à la porte qui est débloquée pour elle. Les deux prédicats qui suivent formalisent la propriété P13 (condition de débloquage d'une porte).

$$pdb \in prs \rightsquigarrow prt$$

$$(pdb; org) \subseteq sit$$

$$(pdb; dst) \subseteq aut$$

Le co-domaine de la fonction  $pdb$ , c'est-à-dire l'ensemble  $ran(pdb)$ , comprend toutes les portes qui sont engagées dans le processus éventuel de passage d'une certaine personne qui, cependant, n'est pas encore passée. Les voyants verts des portes en question sont donc allumés. On introduit la définition suivante de l'ensemble  $vrt$  des portes dont le voyant vert est allumé. Cette définition correspond partiellement à la propriété P14 (allumage du voyant vert dès que la porte est débloquée).

$$vrt \hat{=} ran(pdb)$$

Par symétrie, on introduit l'ensemble *rge* des portes dont le voyant rouge est allumé. On notera que les voyants rouge et vert d'une même porte ne peuvent pas être allumés simultanément (propriété P15). Il vient alors le nouvel invariant suivant:

$$\begin{aligned} rge &\subseteq prt \\ vrt \cap rge &= \emptyset \end{aligned}$$

### Événements

On introduit maintenant plusieurs événements nouveaux. D'abord les événements **débloquer** et **refuser** qui correspondent à la découverte d'une personne désirant passer dans un autre bâtiment. Dans le premier cas, cette personne est admise, dans le second elle ne l'est pas. La condition d'admission peut s'exprimer comme suit:

P17: Toute personne est admise à franchir une porte faisant communiquer le bâtiment où elle se trouve à un bâtiment où elle est autorisée à aller. De plus cette personne ne doit pas être déjà engagée avec une porte.

Formellement, il vient donc la définition suivante du prédicat  $admis(p, q)$ :

$$\begin{aligned} admis(p, q) &\hat{=} \\ &org(q) = sit(p) \wedge \\ &(p, dst(q)) \in aut \wedge \\ &p \notin \text{dom}(pdb) \end{aligned}$$

On peut maintenant présenter les deux événements **débloquer** et **refuser**. Les gardes de ces événements ont une "racine" commune, à savoir une personne  $p$  et une porte  $q$ , dont les feux vert ou rouge sont éteints (on verra comment ceci peut être assuré avec certitude lors d'un raffinement ultérieur). Ensuite les gardes divergent, celle de **débloquer** correspondant au prédicat  $admis(p, q)$  et celle de **refuser** au prédicat  $\neg admis(p, q)$ .

débloquer  $\hat{=}$   
 ANY  $p, q$  WHERE  
 $p \in prs \wedge$   
 $q \in prt \wedge$   
 $q \notin vrt \cup rge \wedge$   
 $admis(p, q)$   
 THEN  
 $pdb(p) := q$   
 END

refuser  $\hat{=}$   
 ANY  $p, q$  WHERE  
 $p \in prs \wedge$   
 $q \in prt \wedge$   
 $q \notin vrt \cup rge \wedge$   
 $\neg admis(p, q)$   
 THEN  
 $rge := rge \cup \{q\}$   
 END

On trouve ensuite l'ancien événement **passer** qui peut se déclencher pour une porte  $q$  dont le feu vert est allumé et qui provoque le passage de la personne engagée avec cette porte ainsi que son désengagement. Ceci a pour effet d'éteindre le feu vert.

```

passer ≐
  ANY q WHERE
    q ∈ vrt
  THEN
    sit(pdb-1(q)) := dst(q) ||
    pdb := pdb ▷ {q}
  END

```

On trouve enfin les deux nouveaux événements **rebloquer** et **libérer** qui ont pour effet d'éteindre de façon asynchrone (pour l'instant) les feux respectivement vert ou rouge d'une certaine porte. Le premier d'entre eux provoque aussi le désengagement de la personne concernée d'avec la porte en question.

```

rebloquer ≐
  ANY q WHERE
    q ∈ vrt
  THEN
    pdb := pdb ▷ {q}
  END

```

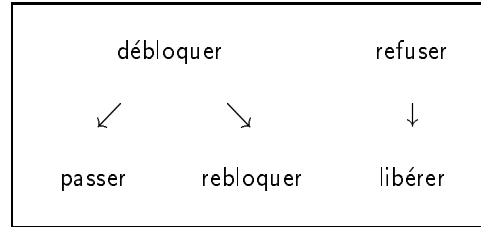
```

libérer ≐
  ANY q WHERE
    q ∈ rge
  THEN
    rge := rge - {q}
  END

```

### Synchronisation

Comme on peut le voir, ces différents événements sont synchronisés de façon relativement lâche pour l'instant comme indiqué dans le diagramme ci-dessous:



### Preuves

Il est simple de prouver que la nouvelle version de l'événement **passer** raffine la précédente. De même est-il facile de prouver que les nouveaux événements raffinent tous l'événement qui ne fait rien, **skip**.

Il nous reste à prouver deux choses: (1) que l'événement **passer** n'arrive pas moins souvent que son abstraction, compte tenu, bien évidemment, des événements nouveaux, et (2) que les nouveaux événements ne peuvent prendre le contrôle indéfiniment, empêchant ainsi l'événement **passer** de se produire.

En fait, la preuve de (1) est relativement aisée, par contre celle de (2) est tout simplement *impossible*. On a là une difficulté nouvelle qu'il nous faut éclaircir et éventuellement corriger par des exigences additionnelles.

Compte tenu du Cahier des Charges initial, nous savons que l'événement **passer** envisagé ci-dessus n'est pas le seul événement "de base" qui puisse être observé. Nous savons que l'entrée d'une personne peut être refusée et aussi qu'une personne qui désirait rentrer dans un bâtiment peut changer d'avis "en cours de route": dans ce dernier cas la porte est "re-bloquée" automatiquement au bout de 30 secondes. Ceci correspond aux événements **refuser** et **rebloquer** envisagé ci-dessus.

Cependant, lors de leur introduction, on doit alors prouver que ces deux événements ne peuvent pas empêcher indéfiniment l'événement **passer** de se produire (c'est-à-dire que leurs gardes ne sont pas indéfiniment vraies en même temps que celle de l'événement **passer**), ce qui, en théorie sinon en pratique, n'est pas absolument impossible. On peut en effet imaginer observer un comportement bizarre du système où personne ne pourrait jamais rentrer dans les bâtiments, soit parce que des gens n'ayant pas les autorisations requises essaient indéfiniment de rentrer, soit parce que d'autres gens, qui ont bien ces autorisations glissent leurs cartes dans un lecteur mais changent toujours d'avis au dernier moment. Pour pouvoir prouver que ces comportements ne sont pas indéfiniment possibles, il nous faut proposer de quoi les prévenir.

#### *Propositions pour empêcher l'obstruction permanente*

Une première façon de faire serait de formaliser le mécanisme par lequel le système forcerait, d'une façon ou d'une autre, les gens qui n'ont pas droit d'accéder à un bâtiment à ne pas tenter *indéfiniment* de le faire (essayer indéfiniment un refus). De même, faudrait-il que le système force, d'une façon ou d'une autre, ceux qui y ont droit à ne pas indéfiniment renoncer "au dernier moment" à rentrer dans un bâtiment (provoquer *indefinement* un re-bloquage). Ce type de comportement drastique ne fait pas partie, manifestement, de la spécification du système que nous analysons.

Une deuxième façon de faire, plus douce mais tout aussi efficace, consisterait à ce que le système "élimine" les personnes qui tendent à se conduire de cette manière-là trop souvent. On leur retirerait tout simplement l'autorisation de pénétrer dans quelque bâtiment que ce soit. Ces personnes seraient ainsi "confinées" dans le bâtiment où elles se trouvent, par exemple "dehors". Ceci est très facile à formaliser puis à réaliser: c'est d'ailleurs ce que l'on trouve dans la plupart des systèmes de contrôle par carte à puces. Par exemple, après trois essais successifs infructueux d'une personne avec un Distributeur Automatique de Billets, la carte est "mangée", ce qui est une façon très efficace d'empêcher la personne en question de bloquer indéfiniment l'accès au distributeur.

#### *Décision finale*

À la suite de cette discussion nous décidons cependant de ne pas envisager non plus cette dernière possibilité qui compliquerait par trop les lecteurs de carte (question de coût). Autrement dit, nous acceptons, pour une raison financière, un risque d'obstruction indéfini, qui a, malgré tout, peu de chances de jamais se produire dans la pratique. Il vient alors la décision suivante:

D3:

Le système que l'on va construire n'empêchera pas que des personnes bloquent indéfiniment les portes soit en essayant indéfiniment de rentrer dans des bâtiments auxquels elles n'ont pas droit, soit en abandonnant indéfiniment "en cours de route" leur intention de rentrer dans des bâtiments dans lesquels elles sont bien autorisées à rentrer.

Clairement, le système que nous allons construire n'est donc pas totalement correct par rapport à nos critères théoriques. Nous l'acceptons mais, et ce qui est très important, nous avons pris soin de le (faire) savoir, et de le signaler explicitement par une décision nettement exprimée.

### Troisième Raffinement

#### *Extension de la spécification informelle*

On introduit maintenant les lecteurs de cartes dans le modèle. C'est la première fois que nous tenons compte explicitement de tels éléments matériels. Ces appareils sont caractérisés par: (i) la capture de l'information qui est lue sur la carte introduite par l'utilisateur et (ii) l'envoi de cette information au micro-ordinateur de contrôle au moyen d'un message véhiculé par un réseau. Par ailleurs, lorsqu'une carte est lue, on suppose que le lecteur se bloque physiquement (la fente en est obstruée) jusqu'à ce qu'il reçoive un message d'acquiescement venant du système de contrôle. Tout ceci correspond à la décision comportementale suivante des lecteurs de cartes:

D4:

Chaque lecteur de cartes est supposé rester bloqué (fente obstruée) entre le moment où le contenu d'une carte est envoyé au système et la réception par ce lecteur de l'acquiescement correspondant. Cet acquiescement vient lorsque le protocole de passage est entièrement achevé (avec succès ou non)

Par cette décision, nous faisons en sorte que l'on ne puisse pas introduire une carte dans un lecteur de façon intempestive. À noter que nous devons payer un certain prix pour cela, celui de l'installation de lecteurs avec fentes obstruables (ils n'appartiennent pas tous à cette catégorie).

#### *Hypothèses concernant le réseau de communication*

Nous ne ferons que des hypothèses minimales concernant l'acheminement des messages sur le réseau. Par exemple, celui-ci ne garantit pas que les messages soient reçus dans l'ordre dans lequel ils ont été émis. Par contre, on suppose que les messages envoyés sur le réseau ne sont ni perdus, ni modifiés, ni dupliqués. On pourrait, bien sûr, tenir compte de ces contraintes particulières mais le modèle serait alors plus compliqué. En tout état de cause, de telles contraintes ne seraient introduites que lors de raffinements ultérieurs.

## Variables et invariant

Nous identifierons dans le modèle chaque lecteur physique avec la porte auquel il est associé. Il n'y a donc pas lieu d'introduire un ensemble particulier pour les lecteurs. L'ensemble des lecteurs bloqués est représenté par un sous-ensemble des portes que nous appelons  $LBL$ .

Les messages qui sont envoyés des lecteurs vers le système de contrôle sont des couples "porte-personne" représentés collectivement par la variable  $mLe$  (chacun de ces couples  $(q, p)$  représente ce que le lecteur associé à la porte  $q$  a lu sur la carte de la personne  $p$ ). Ils forment une fonction partielle des portes vers les personnes (il s'agit d'un invariant du système qu'il s'agira de démontrer: intuitivement, cela vient du fait que chaque lecteur ne peut pas mettre en cause plus d'une personne dans les messages qu'il envoie car sa fente est obstruée comme on l'a vu ci-dessus). Par contre, il ne s'agit pas d'une fonction injective car rien n'empêche une personne de glisser sa carte dans un autre lecteur alors qu'elle n'a pas franchie la porte associée au premier et que les 30 secondes correspondantes ne sont pas non plus écoulées (il s'agit là d'un comportement bizarre que l'on ne saurait éliminer).

Enfin, l'ensemble des messages d'acquiescement est représenté par l'ensemble  $mLa$ , qui est donc un sous-ensemble des portes. Ces divers éléments sont formellement définies comme suit:

$$\begin{array}{l} LBL \subseteq prt \\ mLe \in prt \rightarrow prs \\ mL a \subseteq prt \end{array}$$

Pendant qu'un lecteur est obstrué, la porte en question est dans l'une des quatre situations *exclusives* suivantes: (1) elle est consignée dans un message non encore traité par le système de contrôle, (2) son feu vert est allumé, (3) son feu rouge est allumé, (4) elle est consignée dans un message d'acquiescement non encore traité par le lecteur. Ces différents états caractérisent la *progression de l'information*. Ils correspondent à l'invariant supplémentaire suivant:

$$\begin{array}{l} \text{dom}(mLe) \cup vrt \cup rge \cup mL a = LBL \\ \text{dom}(mLe) \cap (vrt \cup rge \cup mL a) = \emptyset \\ mL a \cap (vrt \cup rge) = \emptyset \end{array}$$

## Événements

Le nouvel événement que l'on introduit à ce niveau est celui qui correspond à la lecture d'une carte. Il s'agit d'un événement "physique":

```

LECTURE  $\hat{=}$ 
  ANY  $p, q$  WHERE
     $p \in prs$ 
     $q \in prt - LBL$ 
  THEN
     $LBL := LBL \cup \{q\} \parallel$ 
     $mLe := mLe \cup \{q \mapsto p\}$ 
  END

```

On trouve maintenant les raffinements des deux événements **débloquer** et **refuser**. Ils sont presque identiques à leurs versions précédentes sauf que maintenant les éléments  $p$  et  $q$  impliqués sont ceux lus sur un message venant d'un lecteur de cartes:

```

débloquer  $\hat{=}$ 
  ANY  $p, q$  WHERE
     $(q, p) \in mLe \wedge$ 
     $admis(p, q)$ 
  THEN
     $pdb(p) := q \parallel$ 
     $mLe := mLe - \{q \mapsto p\}$ 
  END

```

```

refuser  $\hat{=}$ 
  ANY  $p, q$  WHERE
     $(q, p) \in mLe \wedge$ 
     $\neg admis(p, q)$ 
  THEN
     $rge := rge \cup \{q\} \parallel$ 
     $mLe := mLe - \{q \mapsto p\}$ 
  END

```

L'événement **passer** est presque identique à sa version précédente: on acquitte en plus la lecture de la carte par l'envoi du message correspondant vers le lecteur correspondant.

```

passer  $\hat{=}$ 
  ANY  $q$  WHERE
     $q \in vrt$ 
  THEN
     $sit(pdb^{-1}(q)) := dst(q) \parallel$ 
     $pdb := pdb \triangleright \{q\} \parallel$ 
     $mLa := mLa \cup \{q\}$ 
  END

```

De même les deux événements **rebloquer** et **libérer** contiennent-ils l'envoi d'un message d'acquiescement au lecteur de cartes.

```

rebloquer  $\hat{=}$ 
  ANY  $q$  WHERE
     $q \in vrt$ 
  THEN
     $pdb := pdb \triangleright \{q\} \parallel$ 
     $mLa := mLa \cup \{q\}$ 
  END

```

```

libérer  $\hat{=}$ 
  ANY  $q$  WHERE
     $q \in rge$ 
  THEN
     $rge := rge - \{q\} \parallel$ 
     $mLa := mLa \cup \{q\}$ 
  END

```

On trouve enfin un nouvel événement physique qui clôt le protocole:

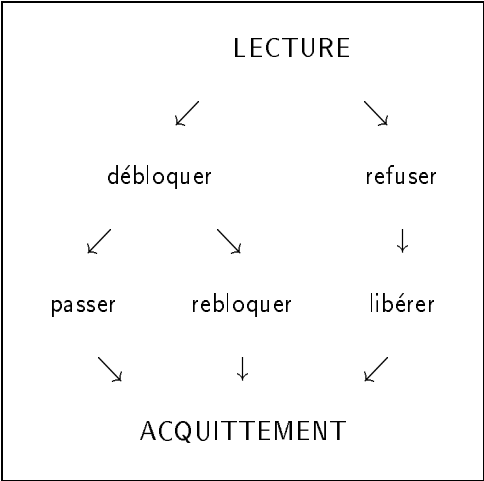
```

ACQUITTEMENT ≐
  ANY q WHERE
    q ∈ mLa
  THEN
    LBL := LBL - {q} ||
    mLa := mLa - {q}
  END

```

*Synchronisation*

Les divers événements de ce raffinement sont maintenant synchronisés comme suit:



*Preuves*

La preuve de ce raffinement n'introduit pas de difficultés particulières comme celles que nous avons rencontrées précédemment.

**Quatrième Raffinement**

*Extension de la spécification informelle*

On introduit maintenant la commande physique de la porte (bloquage et débloquage) ainsi que la détection du franchissement. On prend aussi la décision suivante concernant le comportement "local" des portes.

D5: Lorsqu'une porte est franchie, elle se rebloque localement sans intervention du système de contrôle.

On prend maintenant la nouvelle décision suivante

D6:

On suppose que chaque porte comporte une horloge locale qui assure le reboilage temporisé au bout de 30 secondes et l'extinction du feu vert, ou l'extinction du feu rouge au bout de 2 secondes.

*Variables et invariant: la chaine verte*

La formalisation est effectuée à l'aide d'un certain nombre de variables nouvelles. D'abord l'ensemble  $mDe$  des messages envoyés par le système de contrôle pour débloquent des portes. Comme précédemment avec les lecteurs de cartes, on introduit l'ensemble des portes débloquentées, que l'on appelle  $VRT$  puisqu'il correspond aux portes dont le feu vert est physiquement allumé. On a ensuite l'ensemble  $mPe$  des messages envoyés par chaque porte après détection du franchissement. On trouve enfin l'ensemble  $mBa$  des messages envoyés par les portes pour signaler le reboilage automatique après les 30 secondes. On obtient l'invariant suivant:

$$mDe \subseteq prt$$

$$VRT \subseteq prt$$

$$mPe \subseteq prt$$

$$mBa \subseteq prt$$

Ces quatre ensembles sont exclusifs et leur union est égale à l'ensemble  $vrt$  des portes dont le voyant vert est logiquement allumé. Comme précédemment, ces propriétés manifestent la progression de l'information.

$$mDe \cup VRT \cup mPe \cup mBa = vrt$$

$$mDe \cap (VRT \cup mPe \cup mBa) = \emptyset$$

$$VRT \cap (mPe \cup mBa) = \emptyset$$

$$mPe \cap mBa = \emptyset$$

*Variables et invariant: la chaine rouge*

De façon complètement symétrique à la "chaine verte", nous étudions maintenant la "chaine rouge". On trouve les variables suivantes. D'abord l'ensemble  $mRe$  des messages utilisés pour envoyer à une porte l'ordre d'allumer son feu rouge. Ensuite l'ensemble  $RGE$  des portes dont le feu rouge est physiquement allumé. On a enfin l'ensemble de messages  $mRa$  utilisés pour acquitter auprès du logiciel l'extinction du feu rouge. Ces derniers messages sont envoyés automatiquement par la porte 2 secondes après l'allumage du feu rouge. On obtient l'invariant suivant:

$$mRe \subseteq prt$$

$$RGE \subseteq prt$$

$$mRa \subseteq prt$$

Ces trois ensembles sont exclusifs et leur union est égale à l'ensemble *rge* des portes dont le voyant rouge est logiquement allumé. Comme précédemment, ces propriétés manifestent la progression de l'information.

$$mRe \cup RGE \cup mRa = rge$$

$$mRe \cap (RGE \cup mRa) = \emptyset$$

$$RGE \cap mRa = \emptyset$$

### Événements

Venons-en maintenant aux événements. Ceux qui correspondent aux lecteurs de cartes n'étant pas changés dans ce raffinement nous ne les recopions pas ici. Par contre l'événement **débloquer** est, lui, légèrement modifié. On y introduit l'envoi du message de déblocage physique des portes:

```
débloquer ≐
  ANY p, q WHERE
    (q, p) ∈ mLe ∧
    admis(p, q)
  THEN
    pdb(p) := q ||
    mLe := mLe - {q ↦ p} ||
    mDe := mDe ∪ {q}
  END
```

On trouve maintenant l'événement physique de déblocage d'une porte et d'allumage physique du feu vert:

```
DÉBLOQUAGE ≐
  ANY q WHERE
    q ∈ mDe
  THEN
    VRT := VRT ∪ {q} ||
    mDe := mDe - {q}
  END
```

Il est intéressant de constater l'écart entre le déblocage logique de la porte (événement **débloquer** du logiciel) et le déblocage physique (événement **DÉBLOQUAGE** du matériel).

Cet écart énonce le problème majeur des systèmes distribués qui est celui de la distinction entre l'intention (logicielle) et l'action effective (matérielle). On trouve maintenant l'événement physique correspondant au franchissement de la porte. On notera que la porte ne "sait" pas qui la franchit.

```

PASSAGE ≐
  ANY  $q$  WHERE
     $q \in VRT$ 
  THEN
     $VRT := VRT - \{q\} ||$ 
     $mPe := mPe \cup \{q\}$ 
  END

```

Ce passage physique est suivi d'un passage logique presque identique à sa version du raffinement précédent. La seule différence correspond au fait que le lancement de cet événement est maintenant du à la réception d'un message. On notera que ici, l'événement **passer** "sait" qui passe: c'est la personne impliquée dans le déblocage de la porte. Là encore on s'aperçoit du décalage entre la détection physique (événement **PASSAGE** du matériel) et son effet logique (événement **passer** du logiciel).

```

passer ≐
  ANY  $q$  WHERE
     $q \in mPe$ 
  THEN
     $sit(pdb^{-1}(q)) := dst(q) ||$ 
     $pdb := pdb \triangleright \{q\} ||$ 
     $mLa := mLa \cup \{q\} ||$ 
     $mPe := mPe - \{q\}$ 
  END

```

L'événement de re blocage physique de la porte (à partir d'une horloge supposée se trouver dans la porte et se déclenchant 30 secondes après son déblocage si personne ne l'a franchi entre temps) est le suivant. Le message d'acquiescement du re blocage est envoyé au logiciel.

```

REBLOQUAGE ≐
  ANY  $q$  WHERE
     $q \in VRT$ 
  THEN
     $VRT := VRT - \{q\} ||$ 
     $mBa := mBa \cup \{q\}$ 
  END

```

On trouve enfin la nouvelle version de l'événement **rebloquer**, qui se déclenche à la réception du message précédent.

```

rebloquer  $\hat{=}$ 
  ANY  $q$  WHERE
     $q \in mBa$ 
  THEN
     $pdb := pdb \triangleright \{q\} ||$ 
     $mLa := mLa \cup \{q\} ||$ 
     $mBa := mBa - \{q\}$ 
  END

```

L'événement `refuser` est légèrement modifié pour permettre l'envoi du message destiné à allumer le feu rouge.

```

refuser  $\hat{=}$ 
  ANY  $p, q$  WHERE
     $(q, p) \in mLe \wedge$ 
     $\neg admis(p, q)$ 
  THEN
     $rge := rge \cup \{q\} ||$ 
     $mLe := mLe - \{q \mapsto p\} ||$ 
     $mRe := mRe \cup \{q\}$ 
  END

```

Le premier événement matériel suivant correspond à la réception du message précédent et à l'allumage effectif du feu rouge. L'extinction automatique du feu rouge au bout de 2 secondes correspond au deuxième événement suivant qui envoie un message au logiciel pour l'en avertir

```

REFUS  $\hat{=}$ 
  ANY  $q$  WHERE
     $q \in mRe$ 
  THEN
     $RGE := RGE \cup \{q\} ||$ 
     $mRe := mRe - \{q\}$ 
  END

```

```

LIBÉRATION  $\hat{=}$ 
  ANY  $q$  WHERE
     $q \in RGE$ 
  THEN
     $RGE := RGE - \{q\} ||$ 
     $mRa := mRa \cup \{q\}$ 
  END

```

L'événement `libérer` est légèrement modifié par rapport à sa précédente version: il est maintenant déclenché par la réception du message précédent

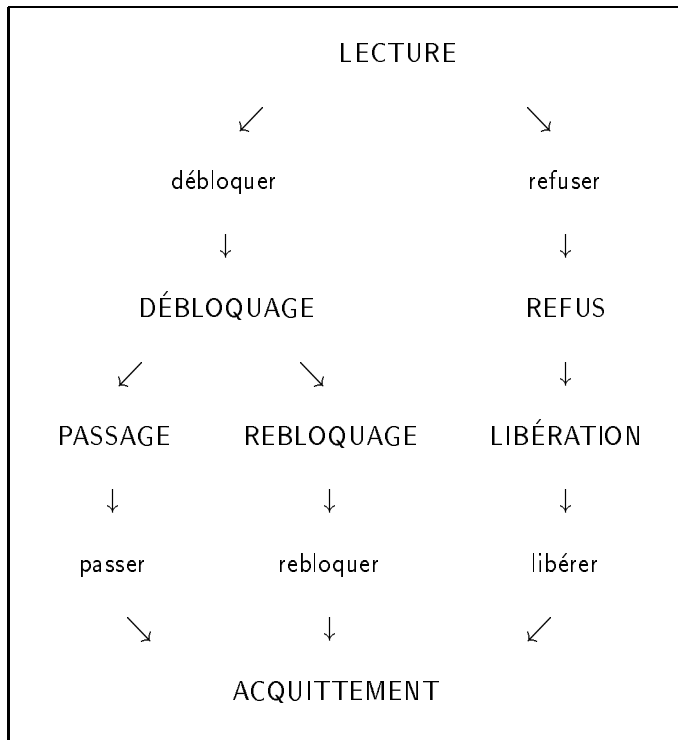
```

libérer  $\hat{=}$ 
  ANY  $q$  WHERE
     $q \in mRa$ 
  THEN
     $rge := rge - \{q\} ||$ 
     $mLa := mLa \cup \{q\} ||$ 
     $mRa := mRa - \{q\}$ 
  END

```

### *Synchronisation*

Nous obtenons maintenant la synchronisation complète suivante entre le logiciel et le matériel:



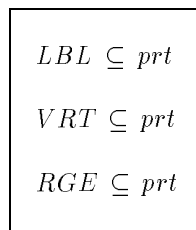
*Preuves*

La preuve de ce raffinement n'introduit pas de difficultés particulières.

### Cinquième Raffinement

*Décomposition*

Nous allons maintenant procéder à la décomposition de notre système en trois entités séparées: le logiciel, le réseau et le matériel. En fait, notre ensemble d'événements va s'implanter sur trois machines abstraites correspondant à ces diverses entités. Voici d'abord la répartition des données entre ces trois machines:



**Données Matérielles**

$$\begin{aligned}
& aut \in prs \leftrightarrow bat \\
& org \in prt \rightarrow bat \\
& dst \in prt \rightarrow bat \\
& aut \subseteq aut; dst^{-1}; org \\
& sit \in prs \rightarrow bat \\
& pdb \in prs \leftrightarrow prt \\
& rge \subseteq prt
\end{aligned}$$

**Données Logicielles**

$$\begin{aligned}
& mL_e \in prt \rightarrow prs \\
& mL_a \subseteq prt \\
& mD_e \subseteq prt \\
& mP_e \subseteq prt \\
& mB_a \subseteq prt \\
& mR_e \subseteq prt \\
& mR_a \subseteq prt
\end{aligned}$$

**Données Réseau**

On trouvera ci-dessous la liste des opérations des deux machines logiciel et matériel. On notera que seules les "opérations logicielles" se traduisent, dans les faits, par du code. Les "opérations matérielles" représentent *globalement* le comportement interne de tous les matériels: lecteurs de carte, portes, feux.

$$\begin{aligned}
& \text{tester\_soft}(p, q) \\
& \text{débloquer\_soft}(p, q) \\
& \text{refuser\_soft}(q) \\
& \text{passer\_soft}(q) \\
& \text{rebloquer\_soft}(q) \\
& \text{libérer\_soft}(q)
\end{aligned}$$

**Opérations Logicielles**

$$\begin{aligned}
& (p, q) \leftarrow \text{LECTURE\_HARD} \\
& \text{DEBLOQUAGE\_HARD}(q) \\
& \text{REFUS\_HARD}(q) \\
& q \leftarrow \text{PASSAGE\_HARD} \\
& q \leftarrow \text{REBLOQUAGE\_HARD} \\
& q \leftarrow \text{LIBERATION\_HARD} \\
& \text{ACQUITTEMENT\_HARD}(q)
\end{aligned}$$

**Opérations Matérielles**

Puis viennent les opérations de la machine *réseau*. On a distingué ci-dessous: (i) les opérations qui relient le logiciel au réseau, prefixées par *lire* ou *écrire*, et (ii) celles qui relient le matériel au réseau, prefixées par **RECEVOIR** et **ENVOYER**. Comme on peut le voir, ces opérations sont symétriques: à chaque opération logicielle *lire* correspond une opération matérielle **ENVOYER** et à chaque opération logicielle *écrire* correspond une opération matérielle **RECEVOIR**.

```

(p, q) ← lire_carte
crire_debloquage(q)
crire_refus(q)
q ← lire_passage
q ← lire_rebloquage
q ← lire_liberation
crire_acquittement(q)

```

### Opérations Réseau-Logicielles

```

ENVOYER_CARTE(p, q)
q ← RECEVOIR_DEBLOQUAGE
q ← RECEVOIR_REFUS
ENVOYER_PASSAGE(q)
ENVOYER_REBLOQUAGE(q)
ENVOYER_LIBERATION(q)
q ← RECEVOIR_ACQUITTEMENT

```

### Opérations Réseau-Matérielles

### Événements

Nous présentons ensuite les différents événements de ce cinquième raffinement, événements qui sont maintenant “implémentés” à l’aide des opérations que nous venons de présenter. On a groupé en un seul événement les deux événements *débloquer* et *refuser*.

```

LECTURE ≐
VAR p, q IN
  (p, q) ← LECTURE_HARD;
  ENVOYER_CARTE(p, q)
END

```

```

débloquer_refuser ≐
VAR p, q, b IN
  (p, q) ← lire_carte;
  b ← tester_soft(p, q);
  IF b = true THEN
    débloquer_soft(p, q);
    crire_debloquage(q)
  ELSE
    refuser_soft(q);
    crire_refus(q)
  END
END

```

```

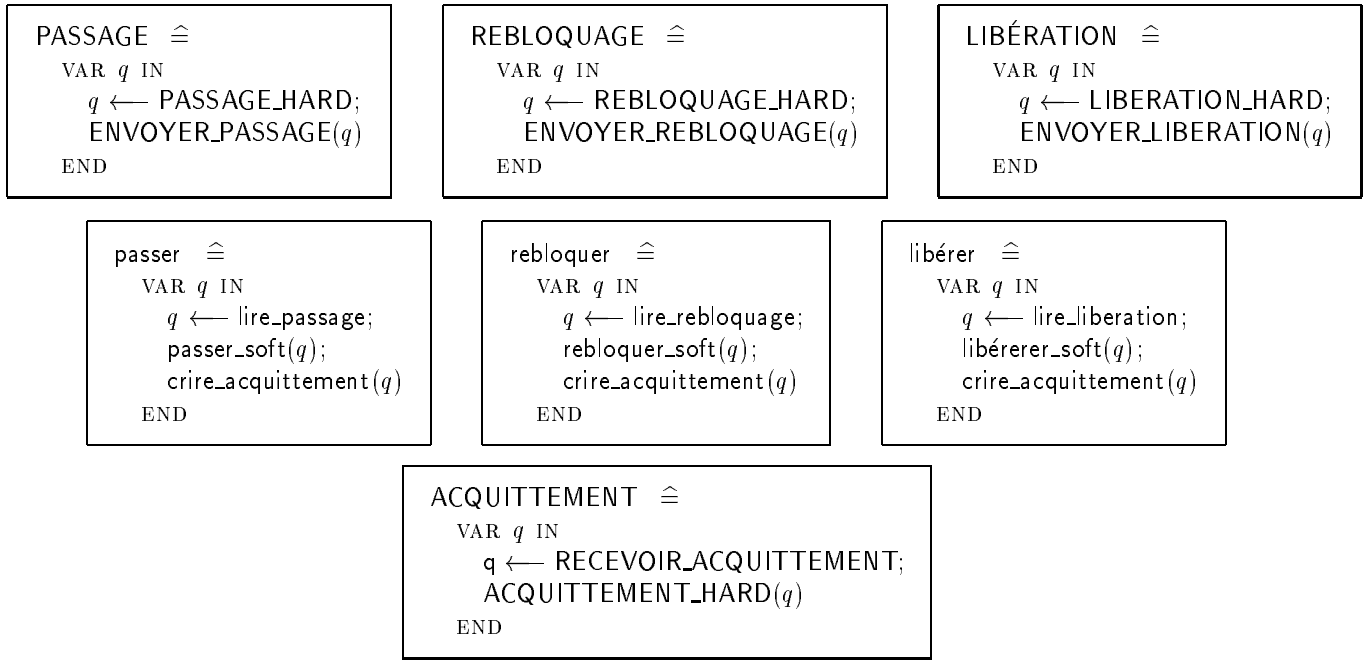
DEBLOQUAGE ≐
VAR q IN
  q ← RECEVOIR_DEBLOQUAGE;
  DEBLOQUAGE_HARD(q)
END

```

```

REFUS ≐
VAR q IN
  q ← RECEVOIR_REFUS;
  REFUS_HARD(q)
END

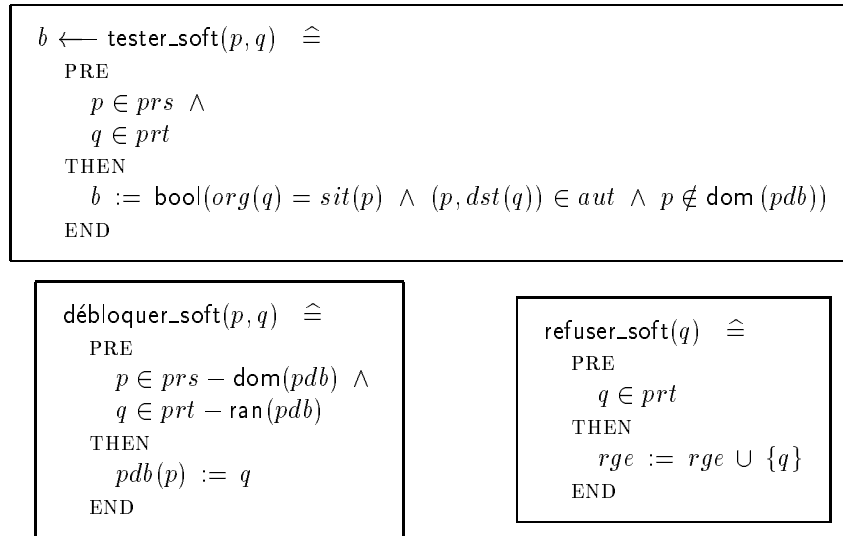
```



On notera que les événements logiciels débloquent\_refuser, passer, rebloquer et libérer constituent la *couche de communication* de notre logiciel. Ces événements commencent tous par une opération de lecture: ils sont, en fait, déclenchés par l'interruption correspondante venant du réseau.

#### Opérations de la machine "Logiciel"

Voici maintenant les opérations de la machine logiciel. On notera que nous avons désormais des opérations qui sont toutes *pré-conditionnées* (et non plus *gardées*). Ces opérations constituent la spécification formelle de la *couche interne* de notre logiciel. Elles pourront, par la suite, être raffinées. On doit, bien sûr, prouver qu'elles préservent l'invariant des données logicielles défini ci-dessus.



```

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

```

```

rebloquer_soft(q) ≐
PRE
  q ∈ ran(pdb)
THEN
  pdb := pdb ▷ {q}
END

```

```

libérer_soft(q) ≐
PRE
  q ∈ prt
THEN
  rge := rge - {q}
END

```

### Operations de la machine "Réseau"

Nous présentons ci-dessous les opérations de liaison entre le logiciel et le réseau. Évidemment, ces opérations seront codées à la main car elles dépendent trop des particularités du système d'exploitation sur lequel on se trouve. Néanmoins, c'est avec elles que l'on fait directement la preuve de ce cinquième raffinement.

```

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

```

```

écrire_debloquage(q) ≐
PRE
  q ∈ prt
THEN
  mDe := mDe ∪ {q}
END

```

```

écrire_refus(q) ≐
PRE
  q ∈ prt
THEN
  mRe := mRe ∪ {q}
END

```

```

q ← lire_passage ≐
ANY b WHERE
  b ∈ mPe
THEN
  mPe := mPe - {b} ||
  q := b
END

```

```

q ← lire_rebloquage ≐
ANY b WHERE
  b ∈ mBa
THEN
  mBa := mBa - {b} ||
  q := b
END

```

```

q ← lire_liberation ≐
ANY b WHERE
  b ∈ mRa
THEN
  mRa := mRa - {b} ||
  q := b
END

```

```

écriture_acquittement( $q$ )  $\hat{=}$ 
PRE
   $q \in prt$ 
THEN
   $mLa := mLa \cup \{q\}$ 
END

```

Nous en venons ci-dessous aux opérations réseau de liaison avec le matériel. Elles ne recevront pas d'implémentation sous forme de code. Elles servent uniquement à définir le modèle et à faire la preuve.

```

ENVOYER_CARTE( $p, q$ )  $\hat{=}$ 
PRE
   $p \in prs \wedge$ 
   $q \in prt$ 
THEN
   $mLe := mLe \cup \{q \mapsto p\}$ 
END

```

```

 $q \leftarrow$  RECEVOIR_DEBLOQUAGE  $\hat{=}$ 
ANY  $b$  WHERE
   $b \in mDe$ 
THEN
   $mDe := mDe - \{b\} ||$ 
   $q := b$ 
END

```

```

 $q \leftarrow$  RECEVOIR_REFUS  $\hat{=}$ 
ANY  $b$  WHERE
   $b \in mRe$ 
THEN
   $mRe := mRe - \{b\} ||$ 
   $q := b$ 
END

```

```

ENVOYER_PASSAGE( $q$ )  $\hat{=}$ 
PRE
   $q \in prt$ 
THEN
   $mPe := mPe \cup \{q\}$ 
END

```

```

ENVOYER_REBLOQUAGE( $q$ )  $\hat{=}$ 
PRE
   $q \in prt$ 
THEN
   $mBa := mBa \cup \{q\}$ 
END

```

```

ENVOYER_LIBERATION( $q$ )  $\hat{=}$ 
PRE
   $q \in prt$ 
THEN
   $mRa := mRa \cup \{q\}$ 
END

```

```

 $q \leftarrow$  RECEVOIR_ACQUITTEMENT  $\hat{=}$ 
ANY  $b$  WHERE
   $b \in mLa$ 
THEN
   $mLa := mLa - \{b\} ||$ 
   $q := b$ 
END

```

*Opérations de la machine "Matériel"*

Il ne nous reste plus finalement qu'à préciser les opérations correspondant à la machine matériel.

```

(p, q) ← LECTURE_HARD ≐
  ANY a, b WHERE
    a ∈ prs ∧
    b ∈ prt - LBL
  THEN
    LBL := LBL ∪ {b} ||
    p, q := a, b
  END

```

```

DEBLOQUAGE_HARD(q) ≐
  PRE
    q ∈ prt
  THEN
    VRT := VRT ∪ {q}
  END

```

```

REFUS_HARD(q) ≐
  PRE
    q ∈ prt
  THEN
    RGE := RGE ∪ {q}
  END

```

```

q ← PASSAGE_HARD ≐
  ANY b WHERE
    b ∈ VRT
  THEN
    VRT := VRT - {b} ||
    q := b
  END

```

```

q ← REBLOQUAGE_HARD ≐
  ANY b WHERE
    b ∈ VRT
  THEN
    VRT := VRT - {b} ||
    q := b
  END

```

```

q ← LIBERATION_HARD ≐
  ANY b WHERE
    b ∈ RGE
  THEN
    RGE := RGE - {b} ||
    q := b
  END

```

```

ACQUITTEMENT_HARD(q) ≐
  PRE
    q ∈ prt
  THEN
    LBL := LBL - {q}
  END

```

## Bilan et conclusion

Nous avons, en fin de compte, déterminé dix-sept propriétés et pris six décisions importantes dont une concerne une *généralisation du système* (accès entre bâtiments), deux concernent des choix négatifs ayant à voir avec des *usages restrictifs* du système (les gens ne peuvent pas nécessairement rejoindre l'“extérieur”, et des obstructions sur les lecteurs de cartes ne seraient pas éventuellement impossible), et trois concernent des *options portant sur le matériel* (bloquages automatiques des lecteurs et des portes, et mise en place d'horloges sur les portes). Bien sûr, ces décisions sont contestables: l'important, nous semble-t-il, est cependant de les avoir nettement identifiées.

Une autre architecture aurait pu être étudiée. Elle aurait correspondu à une *simplification du matériel* par annulation de la décision consistant à mettre des horloges sur les portes. On

aurait en fait centralisé tous les problèmes de timing dans le micro-processeur. On aurait alors constaté une *complexification du logiciel*, due à la difficulté de le synchroniser finement avec le matériel. La technique de développement aurait consisté à envisager d’abord de mettre des horloges dans les portes comme nous l’avons fait ici, puis à raffiner le système en déplaçant les horloges des portes vers le micro-processeur: les difficultés seraient apparues à ce moment-là.

## Annexe: Présentation du cadre formel

Dans cette première section, on rappelle succinctement l’organisation de l’appareil méthodologique formel, ainsi que de l’ensemble des énoncés, qui vont permettre d’effectuer et de valider les Études Système en général. Ce cadre a déjà été détaillé dans différents articles [1] [2] [3] [4], auxquels on renvoie le lecteur intéressé. Les descriptions qui suivent sont un peu arides. L’étude de cas proposée ensuite devrait les illustrer de plus ample façon.

### *Les modèles*

La partie formelle d’une Étude Système est constituée d’une suite de *modèles mathématiques*. Chaque modèle y est supposé *raffiner* le précédent. Cette suite manifeste la prise en compte progressive du *système global* que nous désirons étudier.

Un modèle se présente d’abord sous la forme de la définition d’un certain nombre de *constantes* et de *variables*, qui constituent ce que l’on appelle son *état*. Pratiquement, ces variables ou constantes dénoteront le plus souvent des objets mathématiques simples: ensembles, relations binaires, fonctions. Ces variables et ces constantes sont, par ailleurs, contraintes au moyen d’un certain nombre de conditions exprimant les *propriétés invariantes* du modèle.

Ces variables et ces constantes, liées par les propriétés invariantes en question, décrivent l’état du système dynamique que l’on désire analyser. Elles peuvent représenter des “données” de natures très diverses. Par exemple, elles pourront avoir une certaine contre-partie informatique dans de futurs logiciels, mais elles pourront aussi correspondre à la formalisation de certains éléments matériels, ou même à des messages transitant sur un réseau.

Comme on peut le voir, une Étude Système porte donc, a priori, sur beaucoup plus que la partie logicielle d’un système, même complexe. Elle porte essentiellement sur le couplage entre ce logiciel et son environnement, ou mieux sur la décomposition progressive d’un système à partir d’une vision globale qui donnera naissance à ces deux entités. En fait, l’Étude Système a l’ambition d’analyser l’environnement d’un logiciel de façon aussi détaillée que celui-ci lui-même. On peut d’ailleurs dire que la distinction entre logiciel et matériel n’a pas toujours grand sens à ce niveau.

### *Les événements*

Le modèle contient ensuite la définition d’un certain nombre d’*événements* qui manifestent la façon d’évoluer du système. Ces événements sont supposés être seulement *observables*, ils ne sont en aucun cas déclenchables explicitement par un quelconque mécanisme externe du genre “bouton”. En effet, on ne décrit pas ici la réalisation, même très abstraite, d’un système. On réalise plutôt une *simulation mathématique*, qui nous permet de raisonner au sujet du futur système que nous voulons construire. C’est précisément ce raisonnement qui

va nous permettre d'analyser très tôt le comportement de notre futur système et d'en tirer une éventuelle *architecture*.

Chaque événement est composé d'une *garde* et d'une *action*. La garde est la condition sous laquelle l'événement est *déclenchable*. L'action, comme son nom l'indique, détermine la façon dont les variables d'état vont pouvoir évoluer lorsque l'événement se déclenche.

Chaque événement se produit sous l'effet de l'arrivée "spontanée" d'un *élément déclencheur* que l'on ne connaît pas. Autrement dit, encore une fois, la cause effective du déclenchement d'un événement ne nous intéresse pas, seule l'observation de ce qui se passe lorsqu'il arrive retient notre attention. L'arrivée de cet élément déclencheur ne peut s'effectuer, bien entendu, que si la garde de l'événement le permet.

Il est possible que plusieurs événements puissent se déclencher simultanément (leurs gardes étant toutes vraies). Dans ce cas, cependant, un seul élément déclencheur va "gagner" et, donc, un seul des événements éligibles va pouvoir effectivement se produire, sans que l'on ait aucun moyen de savoir lequel. Nos modèles présentent, de ce point de vue, un certain *non-déterminisme externe*. Il faut aussi noter qu'il n'est pas impossible qu'un événement éligible puisse ne jamais se produire, nous reviendrons sur cette question plus bas.

Il faut enfin observer que les événements sont, a priori, *essentiellement asynchrones*. Le seul synchronisme indirect qui peut éventuellement jouer entre les événements est la conséquence des actions de certains d'entre eux sur les gardes de certains autres.

#### *Forme pratique d'un événement*

Pratiquement, un événement se présente sous la forme suivante:

<pre> xxx ≐   ANY x, y, ... WHERE     P(x, y, ..., v, w, ...)   THEN     S(x, y, ..., v, w, ...)   END </pre>
---

où les identificateurs ci-dessus ont la signification suivante:

- xxx est le nom de l'événement,
- $x, y, \dots$  désigne un certain nombre de variables locales à l'événement,
- $v, w, \dots$  désigne un certain nombre de variables ou de constante d'état du modèle,
- $P(x, y, \dots, v, w, \dots)$  est une condition,
- $S(x, y, \dots, v, w, \dots)$  est l'action associée à l'événement.

Dans ce cas, la garde de l'événement correspond au prédicat existentiel suivant:

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

Autrement dit, la condition nécessaire (mais pas suffisante) pour que l'événement xxx puisse se déclencher dans l'état courant des variables  $v, w, \dots$  du modèle, est que l'on puisse globalement affecter aux variables  $x, y, \dots$ , locales à l'événement xxx, des valeurs rendant le prédicat  $P(x, y, \dots, v, w, \dots)$  vrai. Comme on le voit, cet événement xxx présente un certaine latitude dans le choix des valeurs possible des variables locales  $x, y, \dots$ . On parle ici

de *non-déterminisme interne*.

Dans la très grande majorité des cas, l'action se présente sous la forme d'une simple affectation simultanée de valeurs à certaines variables d'état (à noter que celles qui ne sont pas concernées ne changent pas). Ces modifications dépendent, en général, des valeurs des variables d'états et de celles des variables locales qui sont, rappelons-le, contraintes par la condition  $P(x, y, \dots, v, w, \dots)$ .

#### *Ajouts d'événements nouveaux*

On peut, lors d'un raffinement, ajouter de nouveaux événements. Ils ont pour but de formaliser certaines fonctionnalités, qui n'étaient pas encore prises en charge par le modèle jusque là. Ils permettent aussi d'affiner le "grain" avec lequel on mesure le découpage temporel du système étudié: dans les modèles les plus abstraits, ce grain est très grossier, mais il devient de plus en plus fin au fur et à mesure de la progression dans la série des raffinements.

#### *Décomposition finale*

Le dernier raffinement de la suite envisagée ci-dessus est suivie par la *décomposition* du système en plusieurs entités complètement séparées qui, néanmoins, communiquent entre elles. Nous avons obtenu une architecture. On trouve généralement d'un côté le *logiciel*, de l'autre le *matériel* et, entre les deux le *réseau* qui leur permet de communiquer. Ces entités, en particulier celle correspondant au logiciel, pourront ensuite être développées indépendamment.

### **Les preuves**

Nous présentons maintenant les différentes preuves que l'on peut être amenées à effectuer sur la description de modèles que nous venons de proposer.

#### *Preuves de conservation d'invariants*

Une première série d'énoncés de preuves concerne la conservation des invariants par chacun des événements du modèle. On doit prouver que l'action associée à chacun d'entre eux modifie les variables concernées de telle manière que les propriétés invariantes soient vraies, et ce, sous l'hypothèse de ces mêmes propriétés et de la condition associée à la garde de l'événement.

#### *Preuves de raffinement correct*

Chaque modèle de la suite, excepté, bien sûr, le premier est, comme on l'a dit plus haut, un raffinement du précédent. Ce raffinement se concrétise, entre autres, par une condition reliant les variables du modèle abstrait (celui qui précède) à celles du modèle concret (celui qui nous occupe). La preuve du bon raffinement de chaque événement s'effectue sous l'hypothèse de cette condition.

On doit d'abord démontrer que la *garde concrète est plus forte que la garde abstraite*: ceci a pour conséquence qu'un événement concret se produit *moins souvent* que son homologue plus abstrait (nous reviendrons sur ce phénomène au paragraphe suivant). On doit aussi

prouver que l'action concrète a le "même" effet que l'action abstraite sur les variables d'état du modèle (avec, cependant, peut-être moins de non-déterminisme interne).

Les nouveaux événements, quant à eux, doivent raffiner l'"événement" qui ne fait rien (skip). Autrement dit, les nouveaux événements n'ont aucun effet sur les variables abstraites.

#### *Preuves de la limitation du renforcement des gardes des événements raffinés*

Nous avons vu plus haut que le raffinement d'un événement s'accompagnait du renforcement de sa garde. On pourrait donc renforcer tellement cette garde au cours d'un raffinement qu'elle devienne identiquement fausse: l'événement raffiné ne se produirait tout simplement plus du tout. Ce n'est manifestement pas ce que l'on désire.

En fait, le renforcement de la garde d'un événement concret par rapport à son abstraction doit être compensée par le déclenchement éventuel de nouveaux événements (c'est en cela qu'ils apparaissent comme la manifestation d'une granularité plus fine du temps). Pour cela, on doit prouver que la garde abstraite d'un événement (déjà plus faible que sa contre-partie concrète comme nous l'avons vu au paragraphe précédent) est, en fait, plus forte que la disjonction de la garde concrète et de celle des gardes des nouveaux événements. Autrement dit, un événement concret se produit moins souvent que l'événement abstrait associé parce que de nouveaux événements peuvent se déclencher à sa place.

Il en résulte que dans le cas d'un raffinement qui ne s'accompagne pas de nouveaux événements (c'est toujours possible), les gardes abstraites et concrètes de chaque événement sont identiques.

#### *Preuves de l'impossibilité du monopole des nouveaux événements*

Il est nécessaire enfin de prouver que les nouveaux événements ne peuvent, en aucun cas, empêcher indéfiniment les anciens d'arriver. Dans le cas contraire le modèle concret ne serait pas fidèle à son abstraction qui permettrait à ces événements d'arriver sous certaines conditions.

Il ne s'agit pas ici d'un problème lié au renforcement des gardes concrètes des événements anciens. Il s'agit plutôt du recouvrement des gardes de ces événements par celles des nouveaux. On veut seulement empêcher que ce recouvrement soit permanent. On veut supprimer la possibilité d'une éventuelle "mise en famine" des événements anciens.

## 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. J.-R. Abrial. *Extending B Without Changing it (for Developing Distributed Systems)*. First B Conference (H. Habrias editor). Nantes (1996)
3. J.-R. Abrial et 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)
4. J.-R. Abrial et 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)
5. Y. Ledru. et al. *Étude de cas: Système de contrôle d'accès (version 1.2)* IMAG Grenoble (1998)

