MASTER EN SCIENCES INFORMATIQUES

Analyse et modélisation des exigences non-fonctionnelles de sécurité et de performance

Thiry, Eric

Award date:
1996

Awarding institution:
Universite de Namur

Facultés Universitaires
Notre-Dame de la Paix
Institut d' Informatique
Namur

Analyse et Modélisation des Exigences Non-fonctionnelles de Sécurité et de Performance

(TOME I)

Eric THIRY

Promoteur : M. Eric DUBOIS

Mémoire présenté en vue de l'obtention du diplôme de Maître en Informatique
Année Académique 1995 - 1996

LBS 685o610

3.7374

Facultés Universitaires Notre-Dame de la Paix
Institut d' Informatique

Eric THIRY
3ème Maîtrise en Informatique

RESUME

L'analyse et la modélisation des exigences non-fonctionnelles de sécurité et de performance est actuellement un problème qui soulève beaucoup d'attention. On va tenter d'apporter dans ces pages une méthode qui permet de pouvoir "traiter" une contrainte non-fonctionnelle de sécurité et de performance. C'est-à-dire, partant d'une exigences exprimée généralement de manière abstraite qui ne correspond par conséquent à aucun élément opérationnel d'un système à modéliser, on s'attachera à présenter une méthode formelle et un guide de raffinement qui permettront d'enrichir et de préciser ce qu'exprime la contrainte. Ce traitement a pour but final de permettre l'expression de la contrainte sous forme opérationnelle et par-là même d'autoriser son intégration dans les spécifications du système développé.

ABSTRACT

The security and performance non-functional requirements analysis and modelisation are at the present time a problem under concern. Consequently, we will try to bring in these pages a methodology who perform a treatment about security and performance non-functional requirements. That is to say that this type of requirement is generally expressed in an abstract form which is therefore not in relationship with an operational item of the system to realize. According to that, we need to present a formal methodology and a refinig guide who are able to enrich and to specify the meaning of the requirement. The final goal of this treatment is to express the requirement under an operational form and then to allow integration in the system's specification.

Table des matières

TOME I : Analyse et Modélisation des Exigences Non-fonctionnelles de Sécurité et de Performance.

Chapitre 1. Introduction

Chapitre 2. Etat de l'art

Chapitre 3. Présentation de la méthodologie

Chapitre 4. Etude des exigences non-fonctionnelles de sécurité et de performance

Chapitre 5. Etude de cas

Chapitre 6. Conclusion

Bibliographie

TOME II : Annexes.

Annexes

Table des figures

TOME I : Analyse et Modélisation des Exigences Non-fonctionnelles de Sécurité et de Performance.

Chapitre 1. Introduction

Chapitre 2. Etat de l'art

Chapitre 3. Présentation de la méthodologie

Chapitre 4. Etude des exigences non-fonctionnelles de sécurité et de performance

Chapitre 5. Etude de cas

TOME I : Analyse et Modélisation des Exigences Non-fonctionnelles de Sécurité et de Performance.

Chapitre 1

Introduction

De nos jours, lorsque l'on décide de développer un projet informatique, il devient indispensable de considérer un certain nombre d'exigences bien particulières qu'il était courant de laisser de côté jusqu'il y a peu. Ces différentes exigences sont regroupées sous le label d'*exigences non-fonctionnelles* par opposition aux exigences fonctionnelles qui touchent directement à la fonction que doit réaliser la partie logiciel du projet. Ces exigences non-fonctionnelles sont aussi nommées contraintes non-fonctionnelles ou contraintes de qualité.

Dans ce chapitre, une présentation des exigences non-fonctionnelles est tout d'abord établie de manière générale et non-exhaustive. Ensuite, reprenant les deux exigences qui sont plus particulièrement visées ici, c'est-à-dire les exigences de sécurité et les exigences de performance, il y aura lieu de préciser pourquoi l'importance de celles-ci se marque de plus en plus dans les projets informatiques. Cette présentation tient également lieu de motivation quant au choix de ce sujet pour la thèse développée dans ces pages. Il convient suite à cela de présenter quel sera l'objectif, la portée de cette thèse par rapport aux problèmes soulevés. Et finalement une présentation de la vue globale du contenu de la thèse et de sa structure est établie dans un dernier point.

1.1. Présentation générale

Derrière le terme d'exigence non-fonctionnelle se cache un ensemble de concepts hétéroclites qu'il serait fastidieux de vouloir énumérer complètement ici, mais dont les principaux représentants sont [13] :

- *la performance*,
- *la sécurité*,
- *la facilité d'utilisation* (du produit développé),
- *la vérification* (de la correspondance aux besoins),
- *la maintenabilité*,
- *la réutilisabilité*,
- *la portabilité*,
- *l'inter-opérabilité*.

Tous ces concepts présentent deux propriétés remarquables. Premièrement, ces exigences peuvent être liées entre elles par des liens d'interdépendances. Ces liens constituent le véritable problème des exigences non-fonctionnelles car ils déterminent une compatibilité ou une incompatibilité entre ces concepts. De sorte que le jeu du concepteur réside à leur égard à déterminer un compromis résultant sur une solution au problème des exigences non-fonctionnelles qui sera acceptable et non pas optimale. Ainsi si l'on considère la sécurité et la facilité d'utilisation, il va de soi que plus le mécanisme de sécurité est lourd et plus la facilité d'utilisation est précaire. Inversement, plus l'on veut assurer la facilité d'utilisation et plus les contraintes de mise en oeuvre de la sécurité sont faibles. Par conséquent, "traitement des exigences non-fonctionnelles" rime avec recherche de conciliation.

Deuxièmement et contrairement aux autres exigences, les concepts d'exigences non-fonctionnelles ne sont pas "complets" par eux-mêmes. C'est-à-dire que leur prise en charge dans un projet se matérialise par des contraintes qui sont exprimées à l'égard des autres éléments du système. Ainsi si l'on désire en toute généralité développer un système

informatique performant, cela est réalisé en prenant des dispositions au niveau de l'architecture informatique, des spécifications du logiciel, du modèle conceptuel des données du système d'information, etc...

A ce stade, on remarque déjà que les exigences non-fonctionnelles ne peuvent être traitées indépendamment les unes des autres et qu'il faut considérer toutes les étapes de développement du projet pour assurer la répercussion de celles-ci dans la solution développée. Il reste maintenant à préciser le positionnement de la prise en compte du traitement des exigences non-fonctionnelles dans la méthodologie de développement d'un système informatique, et de préciser tous les éléments de cette dernière afin de bien voir l'environnement avec lequel les exigences non-fonctionnelles interagissent (figure 1.1).

Dans la première phase de la méthodologie, il s'agit de prendre connaissance des souhaits du client. Toutes les exigences sont identifiées et l'on détermine déjà les types de fonctionnalités que devra couvrir le futur système ainsi que la première évaluation du contenu de son système d'information. Tous ces souhaits sont informels, c'est-à-dire que l'on est dans une phase de dialogue avec le client qui se fait en langage naturel assurant moins de précision quant à l'interprétation des exigences mais ne nécessitant pas de connaissance informatique pour le client (car ce n'est pas son rôle).

La deuxième phase a pour but de formaliser tous les souhaits exprimés par le client. C'est à ce niveau que le traitement des exigences non-fonctionnelles prend place. Et c'est aussi à partir de cette phase que les implications des solutions apportées aux contraintes non-fonctionnelles vont pouvoir être répercutées sur les autres exigences. Cette répercussion se fait soit au niveau de cette deuxième phase, soit dans les étapes ultérieures de développement des différentes exigences. Parmi ces exigences on identifie à la figure 1.1 :

1. *La spécification logiciel* reprenant toutes les exigences fonctionnelles touchant au développement des applications logiciels du futur système informatique. Ces spécifications sont le point de départ de l'ingénierie des logiciels qui regroupe la phase de spécification, de conception et d'implémentation.

2. *L'architecture informatique* qui reprend les souhaits au niveau physique du système informatique. Par exemple, pour un système d'ascenseur, il peut s'agir d'exigences impliquant un certains nombres de contraintes sur la cage de l'ascenseur.

3. *Les procédures organisationnelles*, qui identifient toutes les procédures qu'il faut créer au sein de l'organisation pour pouvoir s'adapter à la mise en oeuvre du système. C'est-à-dire la modification du batîment, l'adaptation des activités humaines, etc...

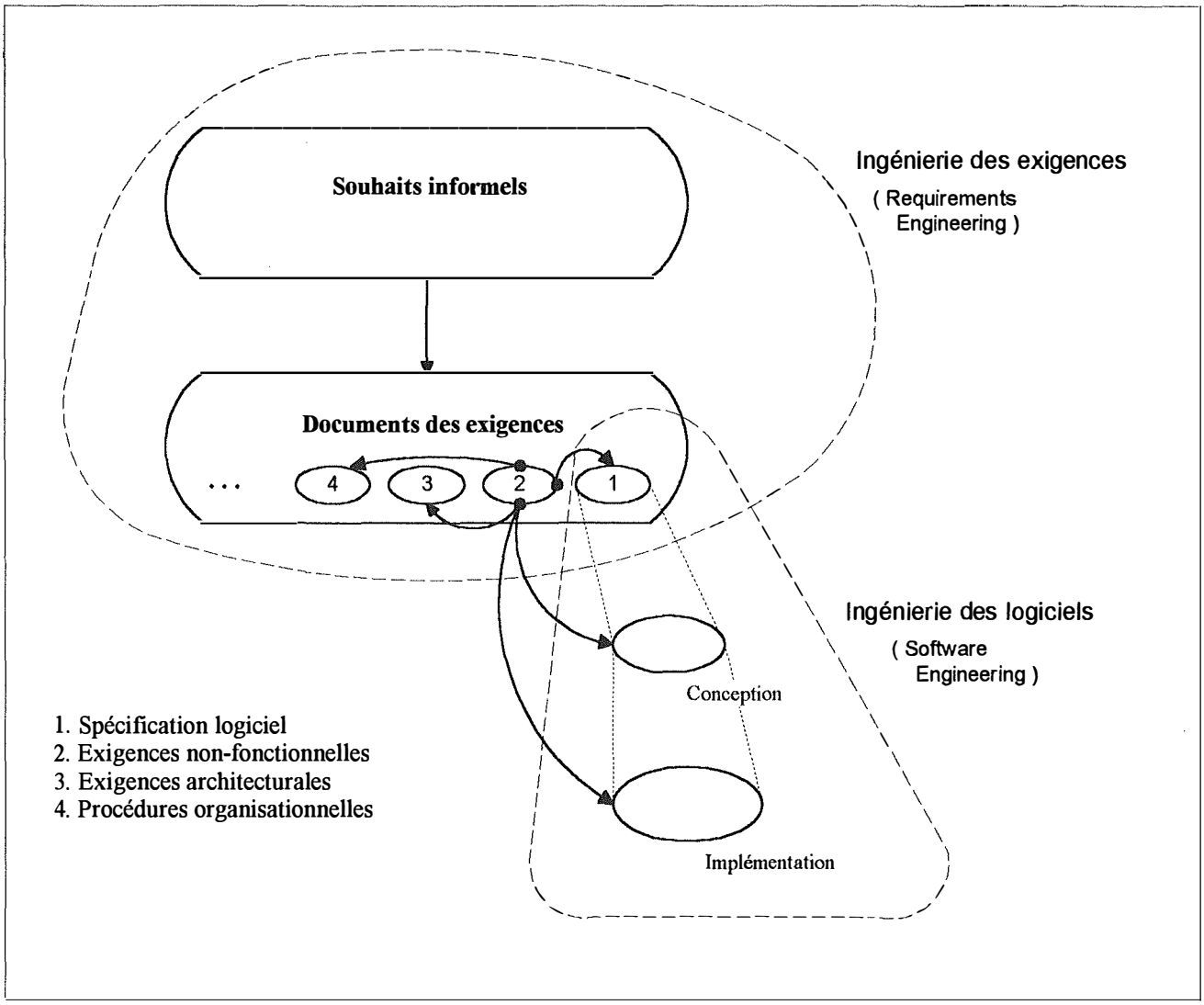


Figure 1.1. La méthode de développement du logiciel.

Par rapport à la figure 1.1, une question pourrait être posée. Pourquoi la répercussion des exigences non-fonctionnelles ne se marque pas toujours directement au niveau de la deuxième phase, mais peut se répercuter dans les niveaux ultérieurs du développement? Si l'on prend le cas de l'ingénierie des logiciels, on peut expliquer cela simplement par le fait que toutes les modifications logiciels qu'impliquent la prise en compte d'une exigence non-fonctionnelle ne se répercutent pas nécessairement au niveau de la spécification. Ainsi, si pour des raisons de performance, l'on demande un accès rapide à un attribut d'une base de données, cette exigence aura une répercussion sur le niveau de l'optimisation de la base de données se déroulant dans la phase de conception.

1.2. Ambition de la thèse

Pourquoi parler plus aujourd'hui qu'hier des problèmes de performance et de sécurité; pourquoi ces problèmes laissés de côté prennent-ils aujourd'hui de plus en plus d'importance? Et si l'engouement pour de tels problèmes est justifié par une réelle nécessité, alors que peut-on faire pour résoudre le manque de méthode déployée dans le traitement de telles exigences? Que doit-on prendre en considération pour répondre complètement aux besoins exprimés en matière de sécurité et de performance?

C'est à apporter une réponse à l'ensemble de ces questions qu'est dédié cette partie. Dans un premier temps, on relèvera les considérations qui montrent que les problèmes de sécurité et de performance ont de bonnes raisons d'être considérés comme primordiaux aujourd'hui. Ensuite une seconde réflexion destinée à présenter ce qui doit être partie intégrale d'une méthodologie de traitement des contraintes de performance et de sécurité détermine quelle est la portée de la thèse.

a. Intérêt du sujet

Dans les premiers pas de l'informatique comme outil mis à la disposition de la société, les organisations cherchaient avant tout à restructurer leur base opérationnelle. C'est-à-dire qu'il était surtout question de se servir de l'informatique comme substitut au travail dans ce qui constituait l'activité directe de production des organisations (figure 1.2).

Aujourd'hui, les préoccupations des organisations ont bien changées. Il est devenu important de réduire leur pyramide de coordination afin de pouvoir y gagner en souplesse et en

coût de fonctionnement. C'est à ce titre que l'on parle de plus en plus de système d'information, parce que tout le but de la coordination est ciblé sur le traitement et la circulation rapide de l'information. Ceci implique que l'informatique c'est lancée vers une nouvelle responsabilité, celle de gérer le traitement et la mobilité des INFORMATIONS STRATEGIQUES. A ce titre il est primordial de pouvoir assurer une sécurité des données stratégiques manipulées par les systèmes actuels. De sorte que si la sécurité était déjà considérée au niveau de la base opérationnelle, il y a une prise de conscience de ces besoins qui apparaît aujourd'hui au niveau hiérarchique. Par conséquent, un plus grand nombre d'activités sont concernées par les besoins de sécurités des systèmes informatiques. Et de plus, le domaine d'expertise de la sécurité s'accroît car les techniques disponibles au niveau de la base opérationnelle ne sont pas nécessairement celles de la coordination qui poursuit des objectifs différents.

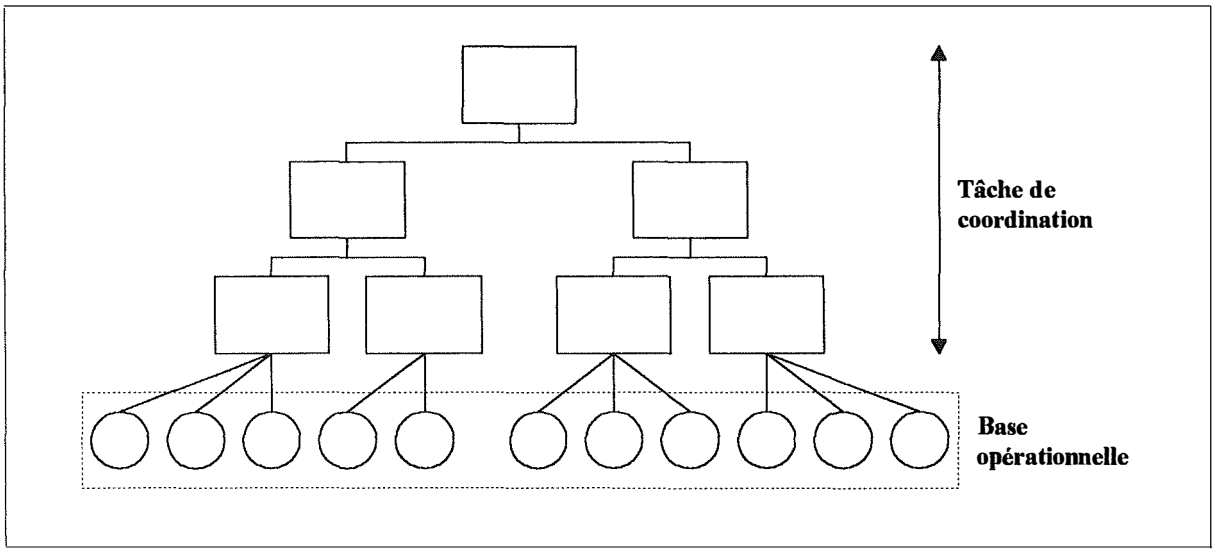


Figure 1.2. Structure d'une organisation.

Une des raisons de la prise de conscience de l'importance de la performance est toute autre. Il s'agit plus ici d'une répercussion de la manière dont l'évolution des techniques informatiques s'est opérée plutôt que d'une quelconque évolution historique du rôle de l'informatique. Afin de montrer le lien qui existe entre l'évolution des techniques informatiques et l'importance accrue du concepts de performance, il est nécessaire de considérer quatre paramètres caractérisant cette évolution : *la capacité des moyens de stockage d'information*; *la vitesse des CPU*, *la quantité d'information traitée par MIPS* et finalement *la performance des dispositifs d'accès aux données* (figure 1.3 : information StorageTek).

La capacité des supports c'est accrue avec la croissance des besoins de la capacité de stockage nécessaire pour les informations. Durant la même période, la vitesse de traitement des CPU s'est accrue dans une proportion globalement semblable. Mais si le nombre d'informations pouvant être traitées par MIPS reste plus ou moins constant, la performance d'accès au données reste également sensiblement égale. Ceci implique que les processeurs sont capables d'assumer la croissance du nombre d'informations à traiter, mais la où le bas blesse, c'est dans le temps qui est nécessaire afin d'avoir accès à une information. Ainsi, il y a beaucoup plus de chance à l'heure actuelle d'aboutir à une solution non performante si l'on ne considère pas dès le début un suivi du développement du système propre à assurer une bonne performance. En présence d'un goulot d'étranglement concernant la performance, il est donc nécessaire de considérer le problème le plus en amont possible afin de penser les astuces qui permettront de limiter les effets néfastes de ces performances d'accès (on en est en effet arrivé à implémenter des solutions gadgets pour passer outre le problème : accès parallèle, optimisation de la base de données, etc ...)

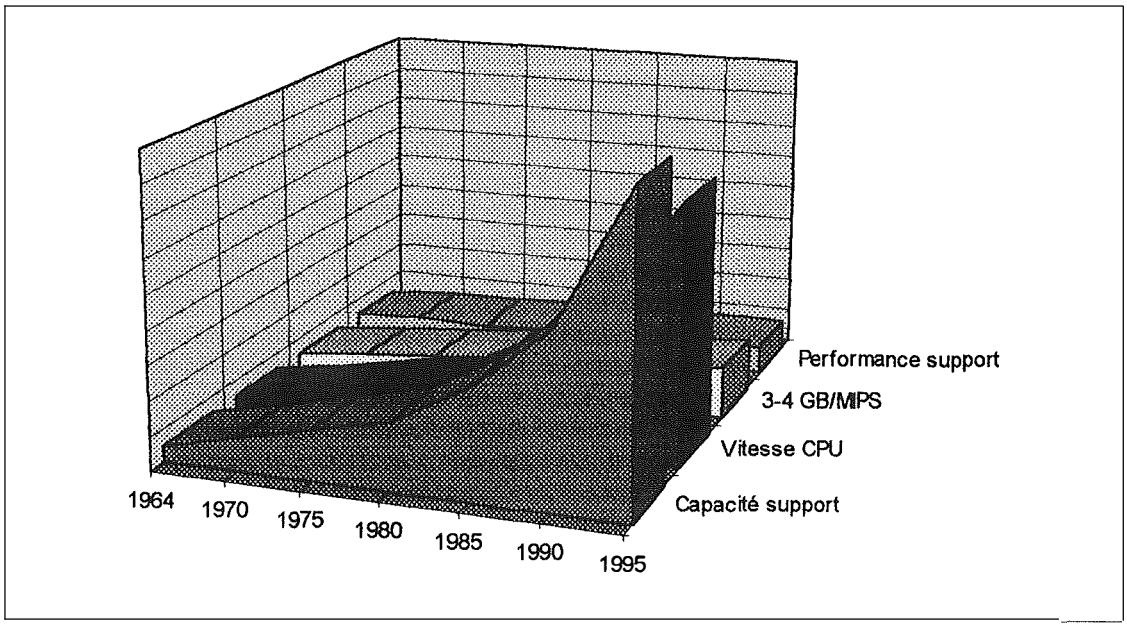


Figure 1.3. Evolution des performances informatiques.

Mais les deux propriétés remarquables des exigences non-fonctionnelles impliquent également un intérêt de recherche par rapport aux *techniques de génération automatique* de solutions et de *traçabilité des choix* de solutions (maintenance et évolution).

Le fait que les contraintes non-fonctionnelles peuvent être interdépendantes entre elles implique que le changement de spécification lors de la maintenance évolutive d'un système peut, par la modification d'une de ces exigences, permettre de poser des choix plus optimaux pour le reste des exigences non-fonctionnelles. Ce qui peut permettre de rendre l'ensemble du système globalement plus optimal.

La génération automatique rencontre également un obstacle avec les exigences non-fonctionnelles car leur mise en oeuvre dans le système développé dépend fortement du domaine d'application de ce projet, par conséquent il est difficile de proposer une solution optimale face au problème des contraintes non-fonctionnelles par une méthode de génération automatique.

b. La portée de la thèse

En présence d'un besoin certain quant à la prise en charge des exigences de sécurité et de performance, il y a lieu de développer une méthodologie qui soit apte à assurer le suivi de ces exigences tout au long du processus de développement d'un projet informatique. A cet effet, on peut donner les caractéristiques qui sont importantes à considérer et qui orienteront le choix quant à la construction de la méthode. Ce sont ces caractéristiques que l'on se devra d'assurer tout au long de la présentation d'une méthode dans ces pages.

Une des grandes évolutions actuels consiste à créer des outils informatiques d'assistance de développement et de maintenance de projet (CASE). On tente par là de simplifier la tâche du concepteur mais aussi de diminuer le temps nécessaire au développement. C'est pourquoi il est indispensable d'introduire dans la méthodologie ***un formalisme*** qui

autorise l'automatisation du processus de traitement des exigences de performance et de sécurité.

La sécurité et la performance sont des concepts complexes qui peuvent être décomposés en concepts plus atomiques et plus maîtrisables permettant de cibler le type de sécurité ou de performance que l'on entend couvrir. Cette décomposition permet d'aboutir à un ensemble de moyens de mise en oeuvre (pour la sécurité) ou de mesures de paramètres à respecter (pour la performance) qui sont adaptés à la situation traitée. A la vue de ceci, afin d'assister le plus possible le concepteur dans sa démarche, il est utile de définir *une taxonomie*, c'est-à-dire cette décomposition successive de concepts complexes en concepts de plus en plus précis et simples qui puisse être fournie par la méthodologie. Ainsi le concepteur éprouvera moins de difficulté à déterminer les moyens adaptés à la prise en compte des exigences de performance et de sécurité qu'il manipulera.

Cette taxonomie et la présence d'un formalisme assurent également la possibilité d'implémentation *d'un mécanisme de réutilisation des concepts* utilisés dans des projets antérieurs. Toute une base est ainsi fournie pour permettre à un outil d'assistance basé sur cette méthodologie de pouvoir mémoriser et présenter des possibilités de décomposition des concepts. Avec le temps, de nouvelles techniques assurant un certain type de sécurité peuvent voir le jour. Il est donc possible d'ajouter ces techniques dans la taxonomie afin d'ENRICHIR le modèle.

L'utilisation d'un formalisme de représentation est requis mais ne va pas dans le sens d'une lecture aisée du problème tant pour le concepteur qui le manipule que pour le client qui sera impliqué dans la validation des décisions prises. C'est pourquoi il est indispensable de munir la méthodologie *d'une représentation graphique* pouvant permettre une vision du problème plus directe et globale pour le concepteur. Mais pouvant permettre aussi au client une implication sans trop de contraintes. Le principe reste que le client n'a pas à faire l'effort pour permettre la compréhension dans la communication ou dans l'échange des informations avec les concepteurs.

Finalement, il a été dit qu'une des propriétés remarquables des exigences non-fonctionnelles se marque par la répercussion du résultat du traitement de ces exigences sur les autres niveaux du développement du système. En conséquence, il faut pouvoir garantir ***une gestion de la traçabilité*** des exigences non-fonctionnelles de performance et de sécurité sur les exigences des autres niveaux du développement.

1.3. Vue globale sur la thèse

L'élaboration d'une analyse et d'une modélisation des contraintes non-fonctionnelles de performance et de sécurité est réalisée en trois chapitres. Le chapitre 3 est destiné à présenter le modèle général sur lequel va s'appuyer toute l'élaboration de la méthodologie. Celle-ci repose sur toute les recherches effectuées par l'équipe de L. Chung à Toronto. Dans le chapitre 4, le modèle de Chung sera instancié au problème propre de la sécurité puis de la performance. Ensuite, une étude de cas est proposée dans le chapitre 5 afin de mettre en pratique toutes les techniques vues dans les chapitres antérieurs. Finalement, une conclusion synthétise le contenu de la thèse, rappelle la contribution apportée par celle-ci et enfin indique les directions à développer dans le futur. Mais tout d'abord, le chapitre 2 va présenter l'état de l'art de la discipline.

Chapitre 2

Etat de l'art

Ce chapitre est destiné à présenter les différentes approches considérées aujourd'hui afin de permettre une analyse et une modélisation des contraintes non-fonctionnelles. Dans un premier point, on s'attarde à présenter i* développé par E. Yu dans [30]. L'approche de A. van Lamsweerde développée dans le méta-modèle KAOS tel que décrit dans [9] est ensuite abordée. Dans un troisième temps, on détaille les résultats obtenus par le projet F^3 (From Fuzzy to Formal). F^3 est un projet qui a été développé dans le cadre de Esprit III et dont la partie concernant la prise en charge des contraintes non-fonctionnelles a été assumée par l'équipe de P. Loucopoulos. Le manuel de l'ingénierie des exigences sert de base à cet exposé [12]. Finalement nous terminons ce chapitre par un récapitulatif des spécificités propres à chacune de ces méthodes.

2.1. Présentation de i*

La méthode développée dans i* est présentée par E. Yu dans [29]. On trouvera également une illustration de la mise au service de i* pour la modélisation d'une contrainte de sécurité dans [30] toujours développée par E. Yu.

Après une brève introduction des caractéristiques de la méthode, on présente tout d'abord *le modèle de dépendance stratégique (Strategic Dependancy Model)*, suivi par la présentation du *modèle d'analyse stratégique raisonnée (Strategic Rationale Model)*. C'est deux modèles forment la structure générale de la méthode. Finalement, un exemple est présenté afin de montrer comment i* prend en considération les contraintes non-fonctionnelles.

La méthode que propose i* permet de modéliser les dépendances organisationnels en exprimant ceux-ci à travers les relations organisationnelles liant des éléments de l'organisation. Ce modèle est surtout intéressant parce que son utilisation permet de comprendre dans la globalité quels sont les buts poursuivis lorsque l'on rencontre des exigences à considérer dans un projet informatique. Il permet par conséquent d'aboutir à une meilleure compréhension des objectifs à réaliser dans le projet.

Les principaux concepts qui sont manipulés par i* sont les "goals", les "tasks", les "resources" et les "actors". De sorte que l'on exprime dans le modèle que les acteurs (actors) veulent atteindre un but (goal) qu'ils réalisent en exécutant des tâches (tasks) et en ayant à leur disposition des ressources (resources). La méthodologie est composée de deux modèles, le premier est appelé le ***modèle de dépendance stratégique*** et le second est appelé le ***modèle d'analyse stratégique raisonnée***. Ces deux modèles vont être maintenant détaillés, mais un

exemple qui fait l'objet d'une étude dans [29] sera également partiellement présenté dans l'exposé afin d'en faciliter la compréhension et de présenter un cas d'application aux contraintes non-fonctionnelles. Cet exemple montre un client qui dépend d'une banque pour effectuer un transfert. Il veut en outre que ce transfert soit sécurisé.

a. Le modèle de dépendance stratégique :

On a présenté le modèle comme étant caractérisé principalement par les concepts de ressource, de tâche, de but et d'acteur. Face à cela, le modèle de dépendance stratégique a pour but de modéliser la situation en exprimant les dépendances qui lient les acteurs les uns aux autres par rapport aux buts poursuivis, aux tâches à réaliser et aux ressources à fournir.

Pour pouvoir exprimer la dépendance entre acteurs, on a besoin de définir des liens de dépendance. Les quatre types de liens définis par la méthodologie sont :

1. *La dépendance de buts* (goal dependancy) : qui exprime qu'un agent dépend d'un autre agent afin de réaliser un objectif. Par exemple, un client a besoin d'un vendeur pour faire un achat.

2. *La dépendance de tâche* (task dependancy) : permet de spécifier la tâche à réaliser en spécifiant comment celle-ci doit être faite. Par exemple, un système du personnel explique à la secrétaire la manière dont elle doit mettre à jour un dossier.

3. *La dépendance de ressource* (resource dependancy) : permet d'exprimer que l'on dépend de la disponibilité d'une ressource pour effectuer une tâche. Par exemple, un programme de gestion de compte à besoin d'une armoire à disques pour effectuer son traitement.

4. *La dépendance de but abstrait* (softgoal dependancy) : qui est semblable à la dépendance de but à ceci près que l'objectif est assez mal défini, est abstrait. C'est ce type de lien auquel on s'intéresse dans la modélisation de contraintes non-fonctionnelles puisque ces contraintes sont en général définies de manière abstraite. Ce que l'on entend par abstrait, c'est que l'on

sait en quoi consiste l'exigence, mais on ne sait pas encore bien comment la matérialiser, la rendre fonctionnelle. Ceci parce qu'une exigence non-fonctionnelle peut être matérialisée par des dispositions couvrant fortement ou moins fortement celle-ci et que l'on ne sait pas encore très bien jusqu'où cette couverture doit aller.

Le cas d'étude développé pour le modèle de dépendance stratégique est présenté à la figure 2.1. Toutes les conventions de représentation graphique de la méthodologie sont exprimées dans cette même figure sous forme de légende. On retrouve l'acteur "Client" qui dépend de l'acteur "Système de gestion de comptes" afin de transférer un montant financier du compte c1 vers le compte c2. Ce client dépend également du système de gestion pour assurer que le transfert de fond soit exécuté en toute sécurité. Le système de gestion quant à lui dépend du caissier pour s'assurer de l'identité du client lorsqu'il doit effectuer un transfert. Le caissier pour s'assurer de l'identité du client doit pour sa part avoir celui-ci en face de lui (présence physique).

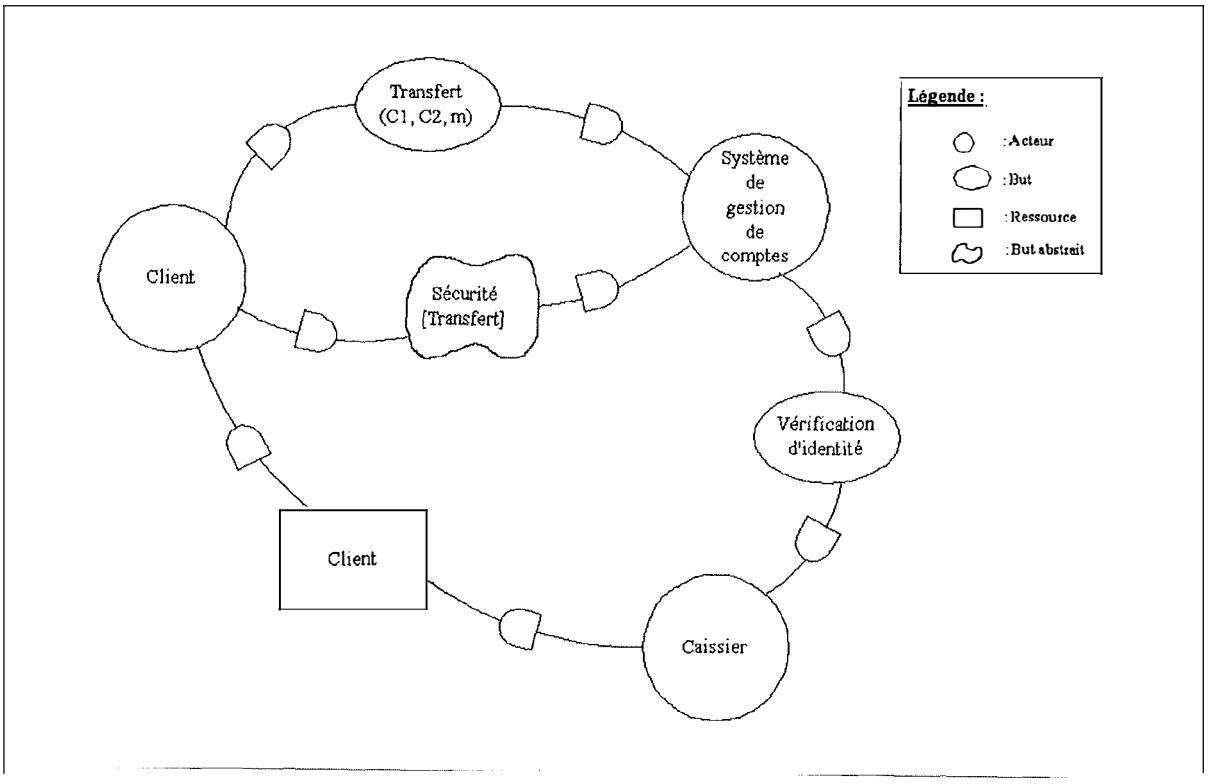


Figure 2.1. Modèle de dépendance stratégique de l'exemple.

b. Le modèle d'analyse stratégique raisonnée :

Ce modèle donne une description interne des acteurs qui étaient définis dans le modèle de dépendance stratégique. Les liens entre acteurs se définissent maintenant au regard des tâches à exécuter ou des objectifs à réaliser. Ce que l'on remarque plus particulièrement, c'est la définition de liens de contribution aux buts abstraits (contribution to softgoal) qui permettent d'exprimer la contribution d'une tâche à un but abstrait. Cette contribution peut être soit négative, soit positive. Ceci autorise la modélisation du fait qu'une tâche peut apparaître comme un moyen de garantir la prise en considération d'une contrainte non-fonctionnelle. On exprime alors cela par un lien de contribution positif indiquant qu'une tâche bien précise apporte une solution ou une contribution à une exigence non-fonctionnelle définie dans le modèle.

Si l'on en revient à l'exemple, le modèle d'analyse stratégique raisonnée est présenté à la figure 2.2. On précise cette fois la tâche que doit exécuter le client pour exécuter son transfert de fond. Cette tâche appelée "Demande de transfert à la banque" se décompose en une tâche "Visite à la banque" et "Demande en personne" qui expriment que le client doit, pour exécuter sont transfert, se rendre à la banque et faire la demande de transfert en personne. Du coté du système de gestion, la tâche de transfert est décomposée en "Manipulation de la demande" et "Enregistrement" de celle-ci. Le caissier quant à lui se voit attribuer la tâche de "Vérification d'identité" du client.

Ainsi, à travers le fait que le client fait sa demande de transfert en personne et que le caissier effectue une vérification d'identité, on trouve deux raisons qui permettent de contribuer à l'exécution du transfert en toute sécurité. Cette constatation se modélise en reliant par un lien de contribution positif "Demande en personne" ainsi que "Vérification d'identité" au but abstrait "Sécurité [transfert]".

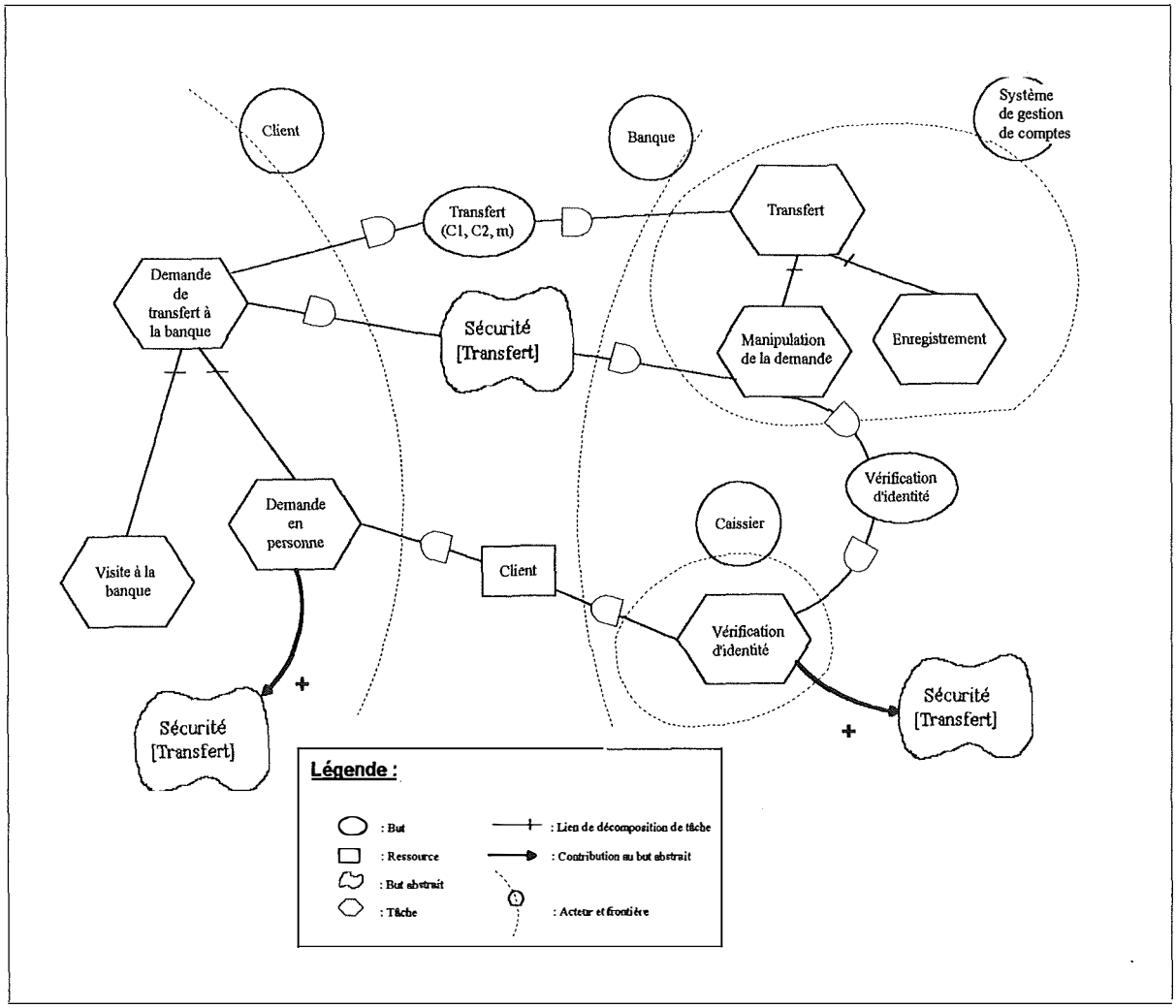


Figure 2.2. Modèle d'analyse stratégique raisonnée de l'exemple.

2.2. Présentation de KAOS

La méthode que l'on va développer est présentée par A. van Lamsweerde dans [9]. Une brève présentation suivie d'une étude de cas portant sur un gestionnaire de réunion peut être trouvé dans [25].

L'approche générale de KAOS recouvre trois composants. On va tout d'abord présenter cette approche pour ensuite s'intéresser plus particulièrement à l'un de ces composants : *le modèle conceptuel* (conceptual model). On montre en effet à travers ce modèle comment KAOS peut prendre en charge les exigences de type non-fonctionnel.

Tout comme i*, KAOS (Knowledge Acquisition in autOmated Specification) s'intéresse principalement à la phase d'acquisition des exigences. Cette méthodologie est structurée en trois composants :

1. Un *modèle conceptuel* qui est à la base de KAOS et permet l'acquisition et la structuration des exigences. Ce modèle est en fait un méta-modèle. Son but consiste à établir, en toute abstraction de cas particuliers à développer, une présentation de tous les concepts que l'on utilise dans la méthodologie, ainsi que les liens qui lient ces concepts. Ce méta-modèle est suffisamment riche pour permettre une représentation des exigences non-fonctionnelles et fonctionnelles d'une façon naturelle (sans biais de représentation) et précise. Une partie ce ses composants sont présentés à la figure 2.3.

2. Un *ensemble de stratégies* qui permettent d'organiser l'acquisition des exigences. Il s'agit en fait de définir les différentes étapes que l'on doit suivre pour pouvoir saisir un modèle des

exigences que l'on est amené à traiter dans un projet. Ce modèle des exigences est perçu comme une instance du méta-modèle. Par exemple, si un des éléments de notre système de gestion d'ascenseur à modéliser est la cage d'ascenseur, on définit "cage d'ascenseur" comme instance d' ENTITE qui est un concept du méta-modèle. Ainsi, les différentes stratégies correspondent aux différentes façons d'aborder le méta-modèle afin de mettre à jour les instances du modèle correspondant à notre cas à modéliser.

3. Une *stratégie de raffinement des buts* destinée à pouvoir rendre ceux-ci opérationnels à travers le concept de contrainte (il s'agit de l'opérationalisation des buts) [8].

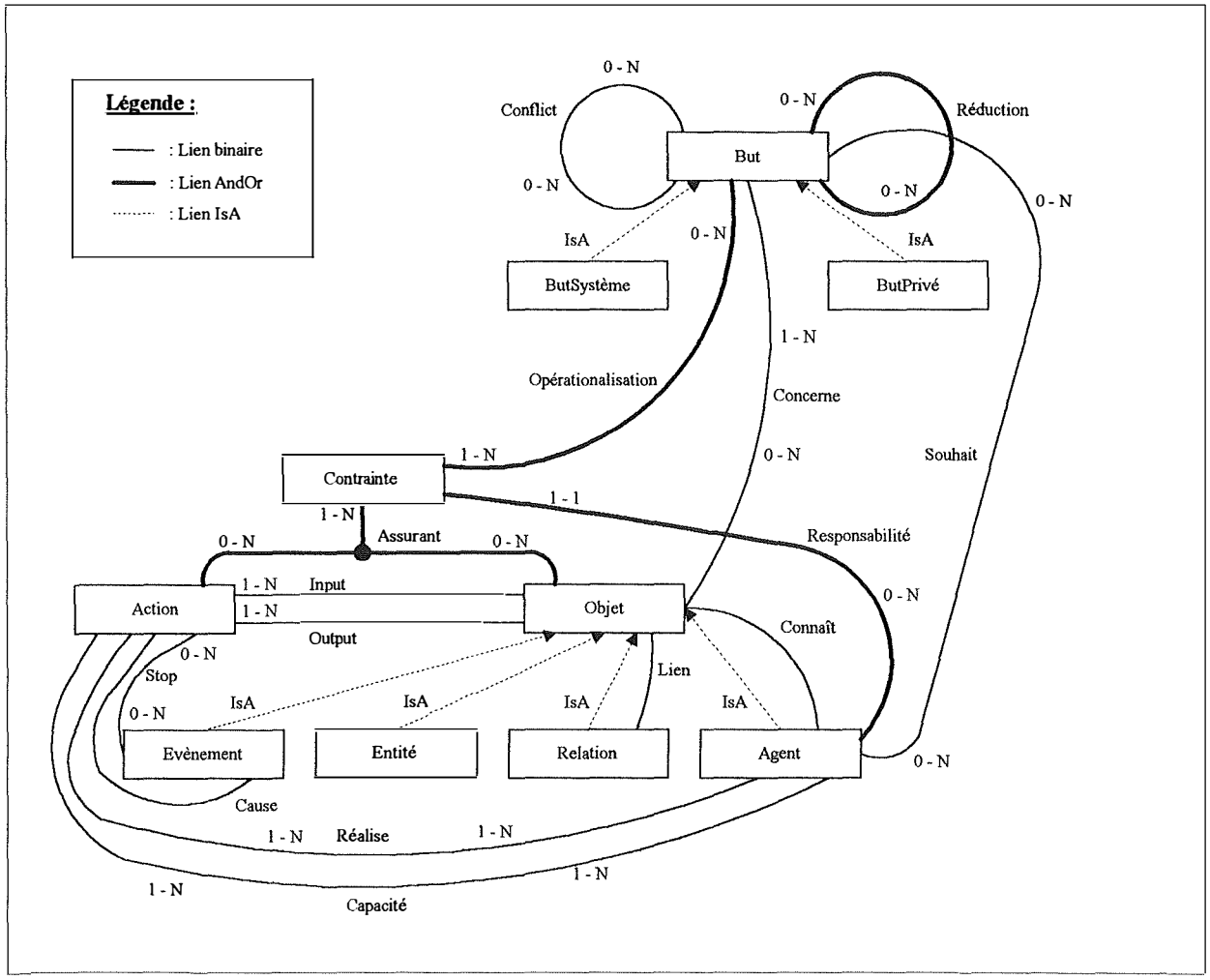


Figure 2.3. Partie du méta-modèle conceptuel de KAOS.

a. Les composants du méta-modèle :

Dans le méta-modèle, on identifie des concepts tels que Objet, But, Action, Contrainte. Certains de ces concepts font l'objet d'une spécialisation. Ainsi Objet est spécialisé en Evénement, Entité, Relation et Agent. Le concept But est quant à lui spécialisé en ButSystème et ButPrivé. Ces concepts vont être définis, mais avant tout une explication est apportée sur la manière dont les liens liant ces concepts doivent être interprétés.

a.1. Les liens :

Trois types de liens sont définis dans le méta-modèle : les *liens binaires*, les *liens AndOr* et les *liens IsA*. Lorsqu'un lien est établi entre concepts, il reçoit un *nom* et une *cardinalité*. La cardinalité est définie par "min - max", où min et max déterminent respectivement le minimum et le maximum du nombre d'instances de la relation dans laquelle chaque instance du concept concerné par la cardinalité est impliquée. Ainsi, chaque instance de but est liée à au moins une instance de objet et au maximum N instances de objet. Inversement, une instance d'objet peut n'être reliée à aucun but et peut être au maximum reliée à N instances de but.

Le lien de type binaire est le lien tel qu'il est défini dans le modèle Entité-Association [3]. Le lien AndOr permet d'exprimer les possibilités d'alternatives, par exemple un but peut être rendu opérationnel par les combinaisons de plusieurs alternatives de contraintes. Le lien IsA exprime pour sa part une spécialisation d'un concept en sous-concept. Ainsi, un agent est un sous-ensemble du concept générique objet. La cardinalité n'est pas renseignée dans le graphe de la figure 2.3, mais elle est toujours 1-1 du coté du concept spécialisé et 0-1 du coté du concept générique.

a.2. Le concept d'objet :

Un objet est un concept dont les instances peuvent évoluer d'état en état. Un objet est spécialisé à travers les concepts d' Evénement, d' Entité, de Relation et d' Agent.

Une *entité* est un objet autonome (objet passif) dont les instances peuvent exister indépendamment de l'existence d'autres instances d'objet. Par exemple, un dossier, une armoire à dossier, etc ...

Une *relation* est un objet subordonné. L'existence d'instances de relation dépend de l'existence d'instances d'objet qui sont reliées par cette relation. Par exemple, la relation S'OccupeDe qui relie l'entité Secrétaire à l'entité Dossier.

Un *événement* est un objet instantané qui est également relié à l'existence d'autres instances d'objet. Ainsi, DossierTraité est un événement qui implique l'instance Secrétaire et une instance Dossier d'entité.

Un *agent* est défini comme le concept (objet actif) qui est à même de traiter les actions. Une instance d'action va définir par son application les transitions d'état(s) d'instance(s) d'objet. L'agent a donc le contrôle de ses transitions d'état(s) et est caractérisé par sa liberté de comportement (au contraire des autres types d'objet).

a.3. Le concept d'action :

Comme on l'a dit, l'action définit les changements d'état(s) des instances d'objet. Ce concept inclut la notion de pré- et de post-condition déterminant les conditions nécessaires à l'application d'une action et déterminant les effets d'une action sur un objet. Ainsi, l'action TransmettreDossier qui a comme pré-condition la nécessité que le dossier soit dans l'état Disponible et qui a comme post-condition entre autre le fait que l'état du dossier devient Indisponible.

a.4. Le concept de but :

C'est ce concept-ci associé à celui de contrainte qui permet de prendre en compte les exigences non-fonctionnelles. Le but est un objectif **non-opérationnel** qui doit être réalisé dans le système modélisé. Non-opérationnel signifie que l'objectif ne peut pas être formulé en terme d'actions disponibles et d'objets. C'est typiquement la situation qui se présente lorsque l'on met à jour une contrainte non-fonctionnelle puisque bien souvent la contrainte non-fonctionnelle est abstraite et ne repose pas sur une description des actions qu'il faut fournir pour assurer une couverture du besoin..

Un exemple de prise en compte d'exigence de sécurité par KAOS est établi dans [9]. On suppose un système d'ascenseur. Si l'on veut modéliser le fait que l'utilisation de l'ascenseur doit se faire en toute sécurité, il s'agit d'un objectif non-opérationnel qui sera formulé par le concept de but. En effet, ceci ne peut être établi à travers les actions disponibles à l'utilisateur telles que : Faire Demande, MonterDedans, Descendre. La formulation ne peut se faire par les actions du ControleurAscenseur qui sont OuvrirPorte, FermerPorte, Arrêt et AllerEtage.

Dans un premier temps, il s'agit de décrire de manière formelle ou informelle ce but. Comme ce type de but est fortement abstrait, il faut ensuite passer par une étape de raffinement destinée à préciser plus finement ce qu'il y a lieu de faire plus concrètement. Ceci se fait par établissement de sous-buts plus précis reliés au but non-fonctionnel par le lien Réduction du méta-modèle.

a.5. Le concept de contrainte :

Une contrainte est un objectif **opérationnel** qui doit être réalisé dans le système modélisé. C'est grâce à ce concept de contrainte que les exigences non-fonctionnelles une fois raffinées sous forme de buts précis vont pouvoir être formulées sous forme d'objets et d'actions disponibles pour certains agents du système.

La correspondance entre un but et une ou plusieurs contrainte(s) se fait par un lien que l'on appelle opérationalisation. Cette formulation appliquée au cas des exigences non-fonctionnelles exprime bien le fait que le but du traitement de ces exigences est d'obtenir une solution qui puisse être opérationnelle, s'exprimer de manière fonctionnelle.

2.3. Présentation de F^3

La méthode développée dans F^3 est présentée dans le manuel de l'ingénierie des exigences [12] édité par F^3 Consortium. Ce qui retient plus particulièrement notre attention dans cette méthode concerne le module développé par l'équipe de P. Loucopoulos et traitant des exigences non-fonctionnelles.

On va d'abord s'attarder à montrer l'environnement global dans lequel se positionne le traitement des exigences non-fonctionnelles. Ensuite on s'attaque à montrer concrètement comment l'on assure la prise en compte des exigences non-fonctionnelles dans F^3.

a. Vue générale sur la structure de F^3 :

F^3 est un ensemble de techniques qui aident à améliorer l'acquisition et la validation des exigences, et un ensemble d'outils pour supporter ces techniques afin de diriger le processus de méthodologie des exigences et pour permettre la réutilisation du travail exécuté en amont. De cet ensemble de techniques, on se focalise seulement sur trois éléments qui sont intéressants à ce niveau. Il s'agit de *la modélisation de l'entreprise, la capture, l'analyse et la validation des exigences* et *la modélisation des contraintes non-fonctionnelles et des contraintes fonctionnelles*.

1. La modélisation de l'entreprise a pour but d'aider le développement, l'acquisition et la communication des premières exigences de l'entreprise et de l'utilisateur. C'est une approche de modélisation, un travail itératif et structuré basé sur le raffinement d'un nombre de sous-

modèles inter-reliés et ce depuis un haut niveau d'abstraction jusqu'au plus bas niveau. Les sous-modèles étudient chacun un aspect de l'entreprise :

- le *modèle des objectifs* étudie les buts, les problèmes et les opportunités de l'entreprise. Il demande aux participants de ce modèle (le client) d'exprimer les intentions à court et long terme de l'entreprise.
- le *modèle des concepts* est utilisé pour définir les éléments dont on parle dans les autres modèles afin de bien comprendre autant du coté du client que de l'analyse quel est la signification des termes utilisés.
- le *modèle des usages et des activités* est établi afin de discuter de la manière dont les processus de l'activité interagissent et manipulent l'information.
- le *modèle des acteurs* est destiné à forcer les analystes à considérer le rôle des acteurs aussi bien dans le développement du système que dans l'utilisation de celui-ci dans le futur.
- le *modèle des exigences du système d'information* est destiné à s'intéresser au système d'information à développer afin de supporter les buts et les activités de l'entreprise.

2. La capture, l'analyse et la validation des exigences est une activité dans laquelle on reconnaît l'importance du domaine de connaissance dans l'acquisition des exigences. L'outil permet une assistance automatisée du raffinement des exigences qui sont bien souvent inexactes, incomplètes, contradictoires, ambigües. Les exigences peuvent avoir comme origine un rapport de spécification des utilisateurs finaux ou avoir été identifiées durant la modélisation de l'entreprise. Ce processus produit des exigences qui sont plus précises, non-ambigües et complètes.

3. La modélisation des contraintes non-fonctionnelles et des contraintes fonctionnelles vient ensuite. L'objectif principal de l'approche de F^3 concernant les contraintes non-fonctionnelles est de permettre l'utilisation de l'information extraite durant la modélisation de l'entreprise et la capture, l'analyse et la validation des exigences afin de réaliser cette modélisation. Pour s'assurer qu'aucune exigence non-fonctionnelle ne soit négligée, le processus de modélisation fournit à l'utilisateur final et aux analystes une plus grande compréhension de ce que ces exigences devraient impliquer sur le système d'information et comment elles sont en relation avec les différents composants du domaine de l'application.

b. La prise en compte des exigences non-fonctionnelles :

La manière de procéder de F^3 consiste à récupérer l'ensemble des exigences qui ont pu être mises à jour grâce à l'étape de modélisation de l'entreprise ainsi qu'à l'activité de capture d'analyse et de validation des exigences pour ensuite commencer le traitement.

La première étape consiste à effectuer une complète **décomposition et classification des exigences**. C'est au cours de cette phase que les exigences non-fonctionnelles vont être raffinées afin d'exprimer plus concrètement le but qui est poursuivi. La classification consiste à diviser les exigences en trois ensembles. Le premier regroupant toutes les exigences architecturales, le deuxième toutes les exigences fonctionnelles et le dernier toutes les exigences non-fonctionnelles.

La deuxième étape va consister à proprement parler en une phase de **modélisation des contraintes non-fonctionnelles**. Les exigences non-fonctionnelles sont pensées dans le modèle comme étant des contraintes qui viennent se greffer sur les autres composants du système. Le but recherché par la modélisation est donc dans F^3 de pouvoir rattacher les contraintes non-fonctionnelles aux composants impliqués par elles.

Il va donc s'agir de pouvoir exprimer chacune de ses contraintes par rapport aux éléments du futur système qui sont visés. Ceci présente une situation difficile car les composants n'ont pas encore été définis et ne le seront pas jusqu'à ce que les exigences non-fonctionnelles soient prises en compte dans le modèle. Une façon de régler le problème consiste à effectuer une analyse des exigences fonctionnelles et architecturales afin d'établir les composants qui devront exister dans le futur système. C'est la manière qui a été choisie dans F^3.

La première phase de la modélisation va consister en l'identification à partir des exigences architecturales et fonctionnelles à retrouver tous les éléments du futur système. Ces

éléments sont appelés ici "cibles" puisqu'ils constituent effectivement les cibles des contraintes non-fonctionnelles.

Ceci fait, on attache les exigences non-fonctionnelles aux cibles. S'il y a lieu, les cibles feront l'objet d'une décomposition si elles ne sont pas assez précises par rapport au exigences non-fonctionnelles. Par exemple, on a établi qu'un des éléments du système serait une base de données des employés. Si la contrainte porte sur un attribut de cette base de données, on décomposera cet éléments en ses différents composants.

Cette méthodologie permet d'identifier quels sont les éléments du système qui sont impliqués par les exigences non-fonctionnelles, mais il faut encore par après définir les alternatives que l'on a à notre disposition afin de les intégrer fonctionnellement dans le système.

2.4. Conclusion

On vient de présenter trois modèles qui permettent de tenir compte des exigences non-fonctionnelles. Ces trois modèles sont focalisés sur l'étape d'acquisition des exigences. Sauf pour F^3 qui se veut plus étendu et qui apporte notamment des outils de simulation pour appuyer la validation dont nous n'avons pas parlé ici.

Ces modèles ont leurs spécificités propres (figure 2.4), ainsi F^3 s'attache surtout à la manière d'aborder le projet pour saisir les exigences tandis que i^* relie les exigences fonctionnelles et non-fonctionnelles à l'organisation sous forme de dépendances et KAOS présente un méta-modèle qui permet d'assurer une guidance lors de l'acquisition des exigences.

Mais la principale critique que l'on peut adresser à ces différents modèles découle du fait que ceux-ci permettent d'exprimer les exigences non-fonctionnelles, voir (pour i^* et KAOS) de tenir compte des éléments fonctionnels qu'elles aspirent à mettre en oeuvre, mais ne permettent pas de gérer très facilement (voir du tout pour i^*) le processus de traitement des contraintes non-fonctionnelles afin de générer une solution fonctionnelle. C'est principalement sur l'établissement d'une méthode permettant d'appuyer le raffinement des exigences non-fonctionnelles que l'on va s'attacher à travailler dans les chapitres qui vont suivre.

	F^3	i^*	KAOS
Niveau du développement principalement visé	**Acquisition des exigences**	**Acquisition des exigences**	**Acquisition des exigences**
Style de spécification	**Déclaratif (présente une déclaration de ce que l'on souhaite voir développé)**	**Déclaratif**	**Déclaratif**
Spécificité	**Recherche une amélioration de la communication entre clients et développeurs par une structuration complète de l'étape d'acquisition**	**Compréhension du pourquoi par des relations organisationnelles**	**Un méta-modèle utilisé avec un "haut" niveau d'abstraction destiné à capturer les objectifs**
Eléments manipulés	**- Modélisation de l'entreprise** **- Capture, Analyse et Validation des exigences** **- Modélisation des contraintes fonctionnelles et non-fonctionnelles**	**- Modèle de dépendance stratégique** **- Modèle d'analyse stratégique raisonnée** **- Acteur, but, ressource, but abstrait, tâche, liens, ...**	**- Modèle conceptuel** **- Ensemble de stratégies** **- Assistant automatique** **- Liens, objet, entité, relation, événement, agent, but, action, contrainte, ...**
Degré de formalisme	**Conventions graphiques**	**Conventions graphiques**	**- Conventions graphiques** **- Logique temporelle** **- Formalisme mathématique**
Critiques	**- Est fortement orientée sur la gestion organisationnelle de l'acquisition des exigences.** **- Permet de lier les exigences non-fonctionnelles aux futurs éléments du système mais ne montre pas comment les répercuter fonctionnellement**	**Est plus ouvert que KAOS, notamment parce que les concepts sont moins cloisonnés dans un modèle pré-défini**	**Offre plus de guidance que i*. Ceci est dû au fait que les exigences acquises viennent se placer dans une structure pré-définie.**

Figure 2.4. Récapitulatif des spécificités.

Chapitre 3

Présentation de la Méthodologie

La méthodologie que l'on propose d'utiliser ici est avant tout appuyée sur le modèle de Chung qui présente les spécificités qui permettent de formaliser et de traiter les contraintes non-fonctionnelles de sécurité et de performance. Dès lors, dans un premier temps on va s'attacher à montrer dans quel environnement s'intégre ce modèle afin de constituer la méthodologie. Ensuite, le modèle est détaillé avant de considérer dans une troisième et dans une dernière partie l'évaluation des solutions potentielles apportées par le modèle et la validation du choix de la solution retenue.

3.1. Introduction

Un outil de traitement de contraintes non-fonctionnelles n'est pas un élément autonome, il s'intègre dans un environnement de développement général de projets informatiques. Ainsi, on peut présenter la méthodologie sur laquelle s'appuiera le modèle de traitement des contraintes non-fonctionnelles comme exprimée à la figure 3.1.

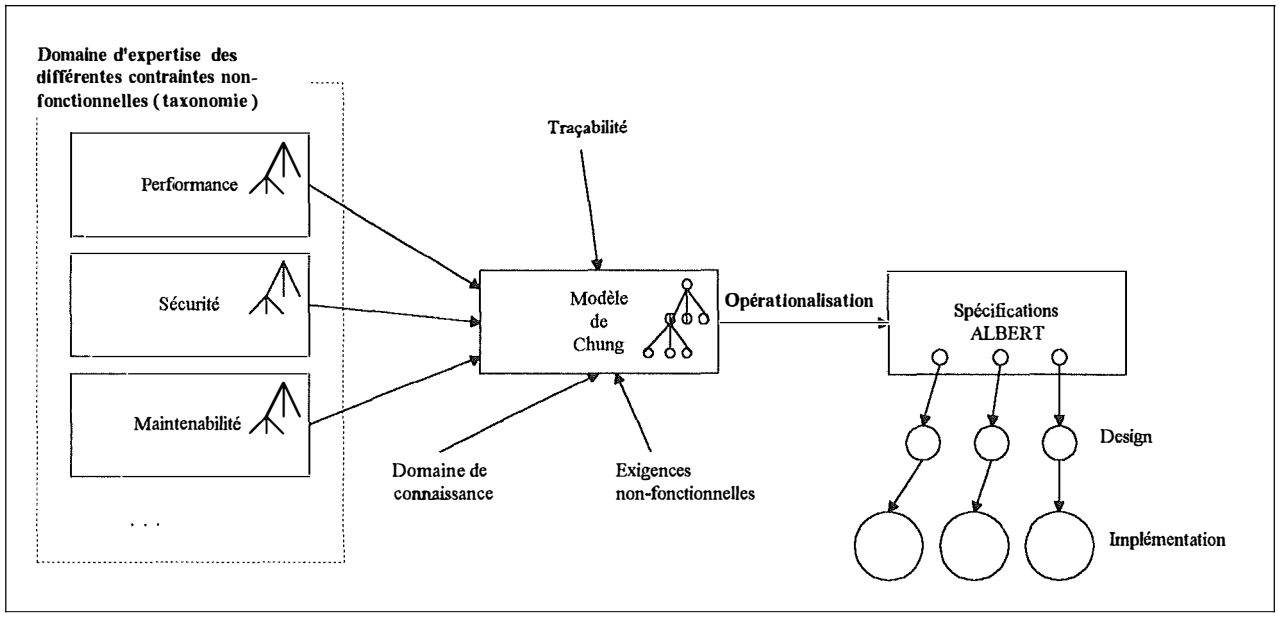


Figure 3.1. Présentation de la méthodologie.

Cette méthode est basée sur un ensemble de techniques que l'on a à notre disposition :

1. *Le langage ALBERT* qui permet de spécifier les fonctions attendues d'un système. On peut trouver la présentation complète du langage dans [10] de Ph. Du Bois. Ce modèle se caractérise par les concepts d'agent, de composant d'état, d'action sur les composants, ainsi que les notions de perception d'action, d'information d'action, de perception d'état et d'information d'état. Une brève introduction est présentée en annexe.

2. *Le modèle de Chung* qui sera à la base du traitement des exigences fonctionnelles. Ce traitement consiste à raffiner une exigences afin de l'enrichir et de la simplifier pour aboutir à une solution répondant au besoin et pouvant être exprimée dans les spécifications d'ALBERT.

3. *Une étape d'opérationalisation* qui a pour but de négocier le passage des exigences non-fonctionnelles rendues opérationnelles vers le modèle de spécification. Une fois après avoir effectuer le raffinement, il s'agit dans notre cas de déterminer quel sera la responsabilité des agents du modèle de spécification qui devront assumer des responsabilités vis à vis des solutions à intégrer. Une introduction à ce type de problème peut être trouvée dans [11].

4. *Une taxonomie* qui correspond à un méta-modèle du modèle de Chung présenté pour chaque type de contraintes non-fonctionnelles. Il s'agit d'un domaine d'expertise propre à la contrainte qui permet d'apporter une assistance dans sa manipulation.

5. *Une procédure d'instanciation* qui partant du modèle de Chung, du domaine de connaissance du système à développer et de la taxonomie permet de construire le raffinement associé au cas particulier de l'analyste.

6. *Un mécanisme de traçabilité* qui est lié à la caractéristique d'interdépendance des exigences non-fonctionnelles et qui nécessite un réexamen des solutions apportées aux contraintes non-fonctionnelles dans le passé et touchant aux éléments du système développé. Cela est important puisque le caractère parfois non conciliable des exigences pousse à prendre des décisions non-optimales mais satisfaisantes pour chacune. Par conséquent, il est tout à fait possible que le projet implique une modification de situation rendant possible un choix plus optimal. Une illustration de ceci peut être trouvée dans [6].

Nous allons maintenant nous préoccuper plus particulièrement du modèle de Chung, mais garder en tête ce modèle peut permettre de mieux comprendre la signification de tous les éléments qui seront abordés dans ces pages.

3.2. La représentation des contraintes non-fonctionnelles

Le modèle que l'on développe ici est issu des recherches effectuées par l'équipe de L. Chung sur les contraintes non-fonctionnelles. Une présentation du modèle peut être trouvé dans [5] ainsi que des illustrations de l'utilisation du modèle au cas particuliers de la sécurité dans [4], de l'exactitude dans [5] et [17] et de la performance dans [17] et [19]. Le modèle est aussi utilisé dans [7] afin d'appuyer la sélection dans les alternatives de conception architecturale et dans [6] pour soutenir le processus de maintenance adaptative d'un système informatique.

Ce modèle s'articule autour de cinq concepts :

1. Une classification de *buts*. Chacun de ces buts exprime soit un besoin non-fonctionnel, soit une décision de conception qui permet de mettre en oeuvre une solution fonctionnelle afin de répondre à un besoin non-fonctionnel ou finalement exprime un argument allant en faveur ou en défaveur de la prise en compte dans la solution d'une instance bien particulière d'un but de décision de conception.

2. Un ensemble de *méthodes*. Le modèle procède par décomposition de buts en sous-buts plus précis et plus simples. Ce processus se répète plusieurs fois jusqu'à l'obtention de solutions fonctionnelles pour le problème des exigences non-fonctionnelles posées. Les méthodes sont donc des procédures génériques qui permettent de décomposer (raffiner) les buts en sous-buts.

3. Une classification de *types de liens*. Puisque le modèle procède par raffinements successifs des buts, il existe un lien entre un but père et son ou ses fils raffinés. Les liens permettent d'identifier les buts fils à leur but père mais aussi, à travers les

différents types de liens, à exprimer le type de relation qu'entretient chaque but avec son but père.

4. Un ensemble de *règles de corrélation*. Une exigence non-fonctionnelle est raffinée par un ensemble de buts reliés par des liens. Mais outre les liens exprimant une relation entre but fils et but père, les liens peuvent exprimer un lien d'interdépendance positive ou négative entre deux buts ayant ou n'ayant pas les mêmes ancêtres. Les règles de corrélation permettent de générer ces liens spéciaux.

5. Une *procédure d'étiquetage*. Il s'agit de la dernière étape du modèle. Cette procédure a pour but d'aider à l'élaboration des différentes alternatives de solutions en déterminant le statut de chaque but par assignation d'une étiquette. En tenant compte des interactions positives et négatives entre buts ainsi qu'entre des liens plus ou moins forts liant but fils et père, il s'agit de voir dans la globalité quel est le niveau de réponse à chaque but.

Une présentation plus complète, enrichie d'exemples et d'une syntaxe formalisant les concepts peut être trouvée en annexe3.

a. Les buts

L'ensemble des buts de la méthodologie est composé de trois sous-ensembles qui sont mutuellement exclusifs. Les trois sous-ensembles sont appelés l'ensemble des ***buts des exigences non-fonctionnelles***, l'ensemble des ***buts de satisfaisabilité*** (que l'on appelle également l'ensemble des buts de décision de conception) et l'ensemble des ***buts d'argumentation***.

Un but est caractérisé par un *nom* et par zéro ou plusieurs *paramètres*. Une des limites du modèle tel que présenté par Chung [6] est liée à ce concept de paramètre qui renseigne mal sur la nature de l'objet manipulé par la contrainte. Afin de résoudre ce problème, on introduit à ce niveau une modification consistant à indiquer le type de chaque paramètre représentant un élément du système à développer. Ainsi, si on considère ce qui peut constituer la cible d'une contrainte non-fonctionnelle dans le système ou dans l'organisation, on identifie les données,

les processus, les éléments architecturaux et la structure organisationnelle. A travers ces quatre cibles potentielles, on définit par conséquent les différents types de paramètres que l'on nomme respectivement DATA, PROCESS, DEVICE et ORGANIZATION.

L'identification d'un paramètre à un de ces types renvoie soit au *schéma conceptuel de la base de données* s'il s'agit du type DATA; aux *fonctionnalités attendues du système* développées durant la phase d'élaboration des exigences informelles s'il s'agit du type PROCESS; au *plan de l'architecture informatique envisagée* s'il s'agit du type DEVICE et finalement à *l'organigramme de l'organisation* s'il s'agit du type ORGANIZATION. Ainsi, le paramètre peut être identifié avec certitude à travers l'ensemble de ces documents.

a.1. Les buts d'exigences non-fonctionnelles :

Ce type de but constitue la partie haut niveau de la méthode. En effet, pour rappel on est, du point de vue des contraintes non-fonctionnelles, bien souvent amené à débuter le traitement à partir d'exigences abstraites exprimées par le client. Ces contraintes ainsi exprimées doivent être, grâce à la méthodologie, raffinées pour aboutir à une solution pouvant s'intégrer "fonctionnellement" dans le système à développer. De sorte que l'on définit les buts d'exigences non-fonctionnelles comme se trouvant au sommet de la méthodologie puisque ce sont ces buts ci qui exprimeront les exigences abstraites des clients.

a.2. Les buts de satisfaisabilité :

Les buts de satisfaisabilité sont ceux qui s'établissent au niveau le plus bas dans le raffinement en présentant les alternatives fonctionnelles dont on dispose pour répondre aux besoins non-fonctionnels posés par le client.

a.3. Les buts d'argumentation :

Ce dernier type de but est destiné à apporter un argument en faveur ou en défaveur d'un autre but ou d'un lien entre buts. Les buts d'argumentation ont toujours comme nom : *ArgFormel* ou *ArgInformel* dans le sens où l'argument présenté est toujours formel ou informel.

b. Les types de liens

Les méthodes que l'on abordera après ce point consistent à raffiner les buts non-fonctionnels en d'autres buts. Ce processus se répète plusieurs fois jusqu'à l'obtention de solutions fonctionnelles pour les contraintes non-fonctionnelles. Dès lors, puisqu'on raffine les buts, il y a de toute évidence un lien qui existe entre un but père et son ou ses buts fils raffiné(s).

On retrouve bien sur les liens classiques que sont OR et AND tel que défini par N. Nilsson dans [18], mais ceci n'est pas suffisant pour couvrir toutes les caractéristiques héritées du processus de raffinement et dont on doit tenir compte. Il faut effectivement des liens qui permettent d'exprimer un renforcement négatif ou positif entre deux buts. A cette fin, nous considérerons les liens OR, AND, sup, sub, und, eql, -sub, -sup, -und, +und. Mais avant d'en dire plus sur leur signification, il faut introduire les concepts suivants : Satisfait, Satisfaisable, Rejeté et Rejetable. Ceux-ci vont permettrent d'exprimer la sémantique de chacun des liens.

Les liens relient un but père à un ou plusieurs buts fils, mais peuvent aussi relier un but d'argumentation à un ou plusieurs liens entre un but d'exigence non-fonctionnelle et un but de satisfaisabilité pour exprimer un support négatif ou positif dans le raffinement du premier vers le deuxième. Durant la procédure d'étiquetage on est amené à vérifier si le choix d'un but de décision de conception rend le but non-fonctionnel satisfaisable (on y répond bien). Pour le savoir il faut propager la notion de satisfaisabilité depuis le bas niveau (décision de conception) vers le haut niveau (niveau abstrait, non-fonctionnel) ce qui implique la connaissance du

caractère satisfaisable ou non du lien entre les buts.

Considérons par conséquent l'ensemble des liens et *Satisfait*, un prédicat qui est vrai pour les buts ET LES LIENS qui ont été montrés satisfaisants. Et considérons également *Rejeté*, un prédicat qui est vrai pour les buts ET LES LIENS qui ont été montrés insatisfaisants.

Une proposition peut parfois être *Satisfaisable* ou *Rejetable* au regard des buts qui lui sont rattachés. En fait si un but fils est satisfait et que le lien le reliant à son père l'est aussi, le but père est potentiellement satisfait, c'est-à-dire qu'il est satisfaisable cela dépend d'autres liens éventuels qu'à le but père.

Au regard de ceci, il reste à présenter la signification de chaque type de lien. Ce à quoi l'on va s'attacher maintenant en exprimant la sémantique de chaque lien en s'appuyant sur les prédicats.

b.1. Le lien AND :

AND : Proposition x $2^{\text{Proposition}}$

Satisfait(G1) $\wedge$ Satisfait (G2) $\wedge$ ... $\wedge$ Satisfait (Gn) $\wedge$
Satisfait (AND (G0, { G1, ..., Gn })) $\rightarrow$ Satisfaisable (G0)

(Rejeté (G1) $\vee$ Rejeté (G2) $\vee$ ... $\vee$ Rejeté (Gn)) $\wedge$
Satisfait (AND (G0, { G1, ..., Gn })) $\rightarrow$ Rejetable (G0)

b.2. Le lien OR :

OR : Proposition x $2^{\text{Proposition}}$

Rejeté (G1) $\wedge$ Rejeté (G2) $\wedge$ ... $\wedge$ Rejeté (Gn) $\wedge$
Satisfait (OR (G0, { G1, ..., Gn })) $\rightarrow$ Rejetable (G0)

(Satisfait (G1) $\vee$ Satisfait (G2) $\vee$ ... $\vee$ Satisfait (Gn)) $\wedge$
Satisfait (OR (G0, { G1, ..., Gn })) $\rightarrow$ Satisfaisable (G0)

b.3. Le lien sup : [condition suffisante]

sup : Proposition x Proposition

Satisfait (G1) ∧ Satisfait (sup (G0, G1)) → Satisfaisable (G0)

Le lien sup est une contribution positive qui est telle que ce lien indique que la satisfaction du but fils est une preuve suffisante pour dire que le but père serait satisfaisable.

b.4. Le lien sub : [condition nécessaire]

sub : Proposition x Proposition

Rejeté (G1) ∧ Satisfait (sub (G0, G1)) → Rejetable (G0)

En somme, le lien sub indique que la satisfaction du but fils est nécessaire pour dire que le but parent serait satisfaisable.

¬ (Satisfait (G2) ∧ ... ∧ Satisfait (Gn) → Satisfaisable (G0))

mais,

Satisfait (G1) ∧ Satisfait (G2) ∧ ... ∧ Satisfait (Gn) ∧

Satisfait (sub (G0, G1)) → Satisfaisable (G0)

b.5. Le lien -sup :

-sup : Proposition x Proposition

Satisfait (G1) ∧ Satisfait (-sup (G0, G1)) → Rejetable (G0)

b.6. Le lien -sub :

-sub : Proposition x Proposition

Rejeté (G1) ∧ Satisfait (-sub (G0, G1)) → Satisfaisable (G0)

¬ (Satisfait (G2) ∧ ... ∧ Satisfait (Gn) ∧ Satisfait (-sub (G0, G1))

→ Rejetable (G0))

mais,

Satisfait (G1) ∧ Satisfait (G2) ∧ ... ∧ Satisfait (Gn) ∧
Satisfait (-sub (G0, G1)) → Rejetable (G0)

b.7. Le lien eql :

eql : Proposition x Proposition

eql (G0, G1) ≡ sup (G0, G1) ∧ sup (G1, G0) ∧
sub (G0, G1) ∧ sub (G1, G0)

b.8. Le lien und :

Ce lien intervient parce qu'il se peut que le concepteur ne sache pas bien pour un but quel sera son impact sur un autre but particulier.

und : Proposition x Proposition

und (G0, G1) montre la présence d'une influence possible entre G0 et G1. Cette influence pourrait être positive ou négative. A ceci, on ajoute +und et -und qui exprime le fait que l'on est en position de préciser que l'influence serait respectivement positive ou négative.

c. Les méthodes

On a vu que le modèle procédait par raffinements successifs de buts. Ces raffinements sot laissés à la responsabilité du concepteur mais l'utilisation répétée du modèle peut permettre de mémoriser et de réutiliser certaines techniques mises à jour. De plus, le modèle propose certaines méthodes qui représentent des procédures génériques de raffinement de buts en un ou plusieurs but(s) fils.

Chaque raffinement est représenté par un lien liant le but père à son ou ses nouveaux buts fils et dont le lien est considéré comme Satisfait. Il y a trois types de méthodes correspondant aux trois types de buts présentés précédemment et que l'on va détailler

maintenant.

c.1. Les méthodes de décomposition :

Une méthode de décomposition peut être basée sur une décomposition du paramètre du but ou sur une décomposition du but lui-même. Généralement, cette création passe par la création d'un lien AND entre le but père et le ou les but(s) fils.

La décomposition basée sur le paramètre reste de la responsabilité intégrale du concepteur car cette décomposition dépend de la structure propre du paramètre.

La bonne question à se poser consiste à se demander quelle peut bien être l'utilité d'une décomposition de paramètre. Si on retranspose la question dans un exemple où l'on explique l'utilité de cette décomposition par le fait qu'Employé est composée de sous-classes Chercheur et Secrétaire qui ne présentent pas la même importance. Par là même, il se peut que endéans un certain niveau de réponse à la couverture des besoins d'exactitude apporté par le concepteur, le client soit prêt à accepter une moins grande couverture des données secrétaire parce que ces données sont moins sensibles.

La décomposition basée sur le but quant à elle peut faire l'objet d'une réutilisation et par conséquent peut être de façon systématique présentée au concepteur lors de la phase de raffinement. Cette décomposition s'appuie sur une typologie qui commence à se construire pour chaque type de contrainte non-fonctionnelle. On peut illustrer cette décomposition à travers le problème de la sécurité qui sera abordé plus en profondeur dans le chapitre suivant. Supposons que l'on veuille organiser la sécurité autour des données de comptes bancaires. Ceci passe par le besoin d'assurer l'intégrité, la confidentialité et la disponibilité de ces données.

c.2. Les méthodes de satisfaction :

De telles méthodes ont pour objectif de raffiner un but en un ensemble de buts de

satisfaisabilité qui impliquent, si la solution est retenue, un engagement du concepteur à mettre en oeuvre une décision de conception afin de répondre à la contrainte non-fonctionnelle.

A ce niveau, il est possible d'obtenir une assistance afin de générer des solutions à apporter aux contraintes non-fonctionnelles soit par une réutilisation des techniques de conception utilisées pour répondre au même type de contraintes dans le passé, soit en observant les solutions proposées dans les autres organisations ou bien également dans la littérature.

c.3. Les méthodes d'argumentation :

Les méthodes d'argumentation raffinent un but ou un lien en un but d'argumentation qui renseigne sur une preuve ou une contre-évidence concernant le caractère satisfaisant ou pas de ce but ou lien. Il faut surtout connaître particulièrement bien l'environnement dans lequel viendra s'intégrer le système en développement afin de pouvoir identifier ce qui constituera un élément positif ou négatif dans le choix d'un but précis.

d. Les règles de corrélation

Comme nous l'avons déjà souligné, les contraintes non-fonctionnelles s'influencent entre elles de manière positive ou négative. Il y aura lieu de reprendre les différents liens qui ont été présentés ci-dessus pour exprimer l'influence existant entre des buts raffinés de contraintes non-fonctionnelles différentes.

On a besoin de guide méthodologique pour mettre à jour de telle relation afin de pouvoir sélectionner les buts de décision de conception en connaissance de cause. Cette guidance se fait à travers la notion de règles de corrélation. Ces règles de corrélation sont tirées de la littérature et de l'expérience acquise (on retrouve l'omniprésence du caractère de réutilisabilité) et peuvent en temps utile être confirmée par le concepteur.

Une illustration du principe se retrouve à travers le besoin de minimiser le coût opérationnel en personne humaine et celui de demande d'une interface utilisateur pouvant être utilisée facilement. Le fait d'utiliser une interface facile à maîtriser ne nécessite pas d'utilisation de personne hautement qualifiée pour exécuter l'application et donc limite le coût opérationnel.

e. La procédure d'étiquetage

La procédure d'étiquetage est la dernière étape du modèle avant d'entamer l'évaluation des solutions potentielles. Cette procédure a pour but d'aider l'élaboration des différentes alternatives en déterminant le statut de chaque but par assignation d'étiquettes.

Un but est étiqueté *Satisfait* s'il est Satisfaisable et pas Rejetable; *Rejeté* s'il est Rejetable et pas Satisfaisable; *Conflictuel* s'il est à la fois Satisfaisable et Rejetable au regard de ses buts fils. Finalement dans un dernier cas, il sera étiqueté *Indéterminé* s'il n'est pas possible de déterminer son statut. Correspondant aux liens +und et -und, on identifie également *Indéterminé plus* et *Indéterminé moins.*

Ce sont les situations conflictuelles qui impliquent la présence d'alternatives puisque l'on posera des choix de décision de conception qui sont tels qu'un but non retenu ayant une influence négative sur un but étiquetté conflictuel peut le rendre satisfait.

f. Le graphe des buts

Il ne s'agit pas d'un nouveau concept de la méthode mais d'un moyen de représenter de manière graphique tout ce qui a été développé dans ce chapitre. La portée du graphe est de permettre une vision étendue du problème afin d'aider le concepteur à se faire une représentation mentale complète de la situation.

Mais l'importance du graphe réside également dans l'aide qu'il pourra fournir lors de l'étape de validation puisqu'il permettra au client de se faire une idée de la solution apportée sans pour autant devoir faire l'effort de compréhension du modèle formel.

Chaque but développé est identifié dans le graphe par un cercle marqué de son libellé formel. Le reste des principes de la représentation est relaté dans la figure 3.2.

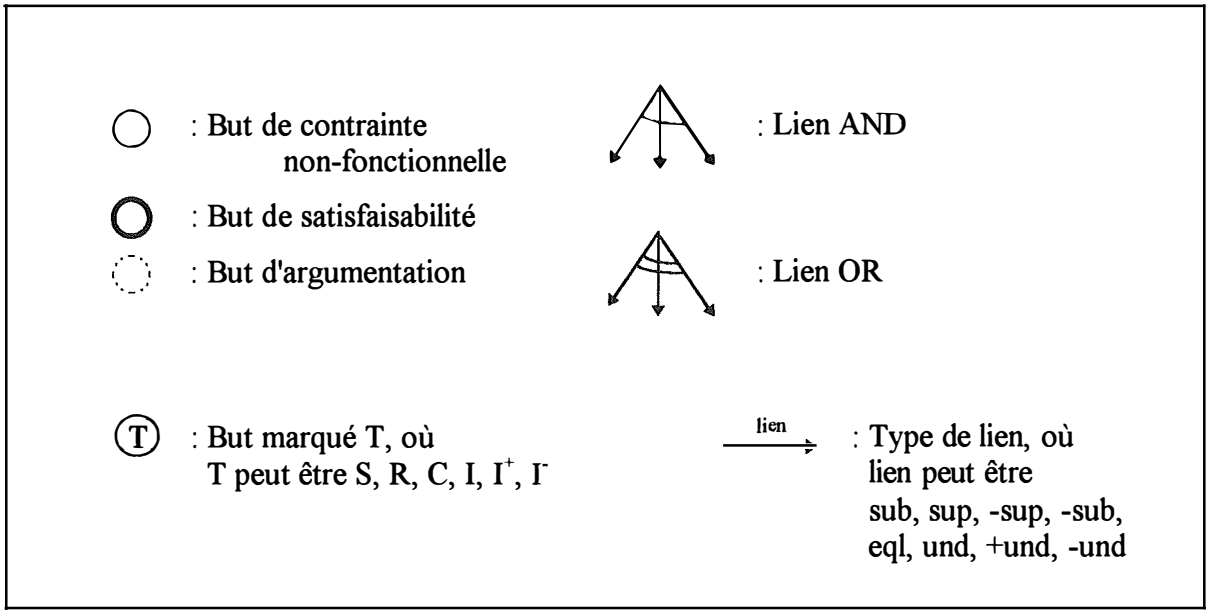


Figure 3.2. Légende de la représentation graphique.

g. Avantages de la méthodologie

En toute généralité, l'apport de cette méthode consiste à faciliter la disponibilité de grandes variétés de techniques connues et moins connues afin de répondre aux exigences non-fonctionnelles (pour autant que l'on dispose de méthodes et de liens de corrélation pré-définis) .

Un des avantages importants de la méthode de Chung réside dans la possibilité de définir des méthodes et des règles de corrélation assurant par là même une réutilisation de l'acquis des traitements d'exigences non-fonctionnelles dans le passé. Ceci est d'autant plus

apprécié que la lourdeur des moyens utilisés globalement lors du développement d'un projet informatique rend ce type d'opportunité de plus en plus indispensable.

On remarque également que le modèle de Chung tel que présenté dans [6] ne fournit pas de syntaxe. Mais tel que reformulé ici, avec une définition de syntaxe pour chacun des concepts et l'aide à la formalisation des buts d'argumentation apporté par la logique des prédicats ainsi que d'opérateurs pré-définis sur les types de données permettent de fournir un formalisme mathématique.

L'assistance graphique est un élément utile dans cette méthode car elle apporte une vision plus globale du problème. Ceci est fort utile pour pouvoir identifier les alternatives de choix de décision de conception ainsi que pour pouvoir juger des interdépendances entre buts ne découlant pas d'un même raffinement d'exigences non-fonctionnelles.

h. Inconvénients de la méthodologie

Le formalisme apporté est assez lourd à lire. Ainsi, sans appuie de la représentation graphique, l'ensemble de la formulation mathématique rend la compréhension du problème particulièrement compliquée. Ceci est imputable à la structure globale du modèle qui implique la création d'un ensemble d'arbres difficilement lisibles lorsqu'ils sont exprimés sous forme mathématique.

De plus, la représentation graphique est, elle aussi, lourde à manipuler car elle implique au fur et à mesure du raffinement un espace nécessaire imposant. Ceci est particulièrement frappant lorsque l'on effectue une décomposition de paramètre, puisque tout le raffinement aval pour chacun de ces buts-fils sera souvent identique (ce qui ne veut pas dire que la solution pour chacun d'eux sera la même).

3.3. L' Evaluation des Solutions

Lorsque le graphe des buts est construit dans son entièreté, intervient la procédure d'étiquetage destinée comme on l'a vu à renseigner sur le degré de satisfaisabilité des buts par rapports aux contraintes non-fonctionnelles posées.

Si la procédure d'étiquetage permet de renseigner sur l'impact d'un choix par rapport aux autres contraintes et moyens disponibles pour les prendre en considération, elle ne permet pas à elle seule de fournir les moyens d'opérer un choix parmi les alternatives que l'on est susceptible de rencontrer. Il est donc de la première importance de relever les éléments de la méthode qui impliquent la survenance d'alternatives et de voir la manière dont la situation peut être abordée afin de poser un choix le plus optimal possible.

a. L'identification des alternatives

Si l'on regarde les différents éléments constituant de la méthodologie, on remarque que les situations de choix découlent de la possibilité, par l'application d'une méthode, de raffiner un but en plusieurs buts liés par une relation OR (figure 3.3). On nomme ce type de situation l'*alternative OR*.

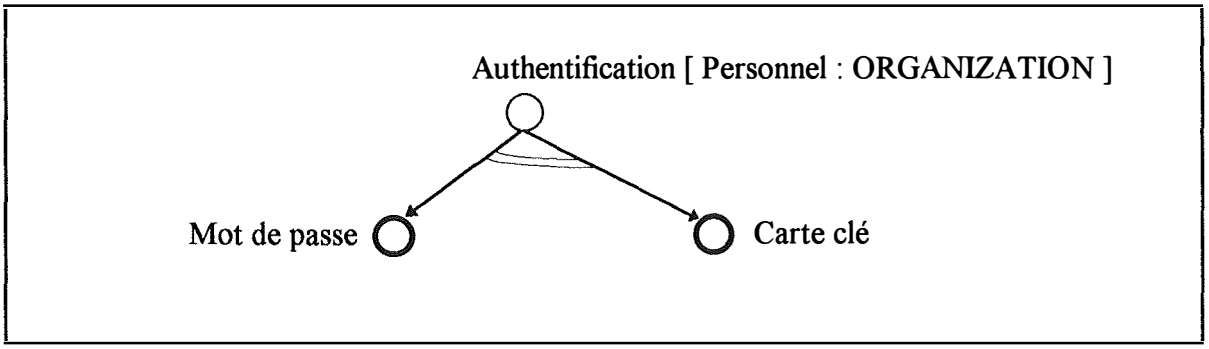


Figure 3.3. Exemple d'alternative OR.

Une deuxième situation de ce genre intervient à travers la notion de règle de corrélation. Cette notion intervient parce que les contraintes non-fonctionnelles sont pour certaines non-conciliables, c'est-à-dire que dans certaine situation il ne sera pas toujours possible de choisir, pour deux (ou plus) contraintes non-fonctionnelles, des buts de décision de conception optimaux. L'existence de liens négatifs entre ces contraintes poussera bien souvent à choisir une situation intermédiaire découlant sur une satisfaisabilité des contraintes plutôt que sur une optimalité impossible à atteindre conjointement. On nomme ce type de situation l'*alternative de non-conciliabilité*. C'est le cas pour ce qui concerne la sécurité et la facilité d'utilisation du produit développé. Car la facilité d'utilisation revendique un système qui soit le moins contraignant possible aux yeux des utilisateurs alors que généralement la sécurité à justement comme conséquence d'en contraindre l'utilisation.

b. Le choix d'une alternative

Généralement lorsque l'on entame une telle analyse, on a à faire à un haut niveau de subjectivité dans ce sens où les contraintes non-fonctionnelles émises par le client se situent toujours autour de concepts qualitatifs et en général pas fonctionnels. Une illustration peut être apportée par l'exemple suivant : on aura à développer la méthodologie autour de la contrainte "Je veux que mes données de client soient bien protégées." plutôt que "Mettez mes données client sur un ordinateur dont l'accès au local se fait par lecteur de mot de passe." Ce que l'on veut exprimer par ce petit exemple, c'est qu'il est extrêmement difficile de juger à partir de quel

moment ce que l'on met en oeuvre est suffisamment puissant pour couvrir réellement (sans aller au-delà) le besoin du client en matière de contraintes non-fonctionnelles.

Au regard de cette considération, on estime donc que connaître la position du client au moment de l'évaluation des alternatives est d'une importance capitale. A cette fin, deux informations sont nécessaires pour permettre un choix optimal parmi les alternatives du graphe des buts. Premièrement, avant de s'attacher à développer les graphes des buts propres à chaque contrainte non-fonctionnelle, on demandera au client de donner un *degré d'importance* (coefficient allant de un à cent) pour chaque contrainte. Finalement, avant de lancer une évaluation des alternatives on demandera au client de formuler un *pourcentage de couverture* de sa contrainte pour chacun des buts de décision de conception.

Partant de cela, il s'agit de trouver dans l'ensemble des graphes de buts, les buts de décision de conception qui permettent de maximiser le pourcentage de couverture tout en respectant le degré d'importance entre contraintes non-fonctionnelles (ceci est représenté graphiquement pour validation au point 3.4).

Du point de vue de la représentation graphique il s'agit simplement de compléter la représentation des buts de la manière exprimée par la figure 3.4.

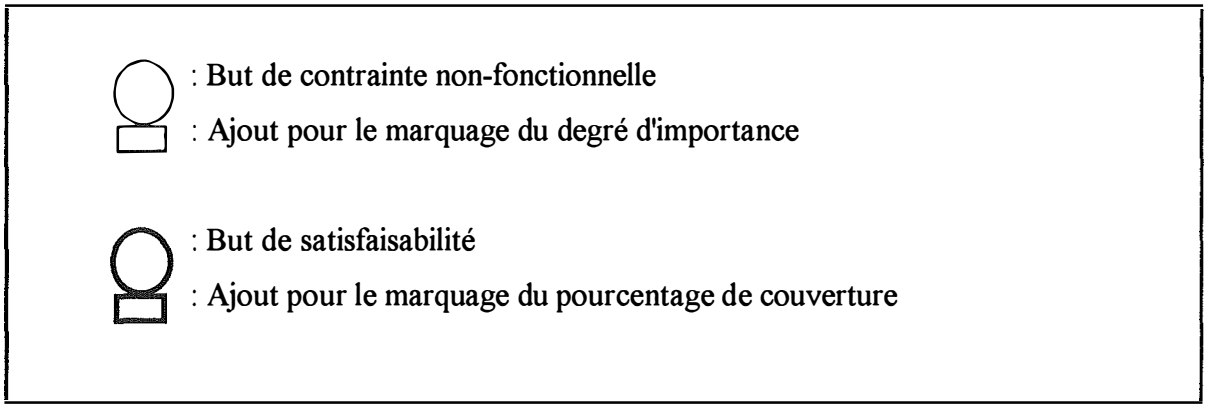


Figure 3.4. Complément de légende de la représentation graphique.

c. L'analyse de coût-qualité

c.1. Le but :

Une analyse de coût-qualité joue un rôle important dans l'évaluation des solutions, car si jusqu'ici on a considéré les alternatives sous l'angle de la réponse aux exigences, il est certain que le coût économique d'une solution est également à considérer. Ceci permet d'affiner le coût estimé de la mise en oeuvre du projet, mais dans le cadre de l'exposé l'intérêt est tout autre.

Premièrement une analyse de coût-qualité exprime une rationalité entre une solution potentielle et le réel besoin à considérer. A cette fin, on a besoin d'effectuer une analyse de risque et une analyse de coût de la solution potentielle. Le but étant normalement d'obtenir une solution ne coûtant pas plus cher que le coût de la perte globale que l'on risquerait de subir sans implémentation d'une solution. Le graphe de coût-qualité permet de vérifier cet état de fait de manière graphique. Il est à remarquer que l'analyse de risque dont on parle ici n'est pas l'analyse du risque encouru en choisissant de développer le projet, mais l'analyse du risque encouru si l'on ne développe pas de solution en réponse à un besoin non-fonctionnel dans le système qui est développé.

Deuxièmement, le coût économique est aussi une contrainte non-fonctionnelle qui doit faire l'objet d'une conciliation avec les autres contraintes non-fonctionnelles. Cette conciliation ne se fait pas par le biais de liens de corrélation puisque le coût ne s'inscrit pas dans la même méthode que celle vue ici, mais se fait par l'envoie des caractéristiques fonctionnelles d'une solution raffinée depuis la contraintes non-fonctionnelle "abstraite" vers l'endroit de la conception où ces nouvelles fonctionnalités auraient une répercussion. Après évaluation économique, il y a lieu au niveau du traitement des contraintes non-fonctionnelles d'accepter la solution comme satisfaisante et de passer à l'étape de validation ou bien de relancer le processus de sélection des buts de décision de conception.

Prenons un exemple plus parlant, si l'on hérite de la contrainte non-fonctionnelle *performance (système : DEVICE)*, on va générer des paramètres de performance que le

système architectural devra respecter. On envoie le ou les paramètre(s) vers l'Ingénierie architecturale qui fait l'évaluation du système à utiliser pour répondre au besoin et expédiera en conséquence le coût d'un tel système.

Un autre exemple pourrait être la décision de conception impliquant la création d'une fonction d'identification des utilisateurs par un mot de passe à l'entrée du système en développement. On fera par conséquent une répercussion en coût (heures de développement) de la mise en oeuvre d'une telle solution pour répondre à un besoin non-fonctionnel.

c.2. Le graphe de coût-qualité :

De la même façon que développé par K.P. Badenhorst dans [1] et [2], on peut représenter l'analyse coût-qualité des différentes alternatives sous forme d'un graphique afin de faciliter la compréhension de l'analyse. Cette vision graphique porte le nom de ***graphe de coût-qualité*** et exprime en abscisse le *degré de qualité* de solutions potentielles et en ordonnée le coût soit résultant de l'analyse de risque pour les solutions, soit résultant de la mise en application des solutions.

En toute généralité, le graphe présente deux courbes; ***la courbe de perte globale*** qui montre que plus le degré de couverture est élevé pour la solution, moins le montant global d'un problème est élevé; et ***la courbe de prise de décision*** qui montre que plus le degré de couverture est élevé pour la solution, plus le coût de la solution est élevée (figure 3.5).

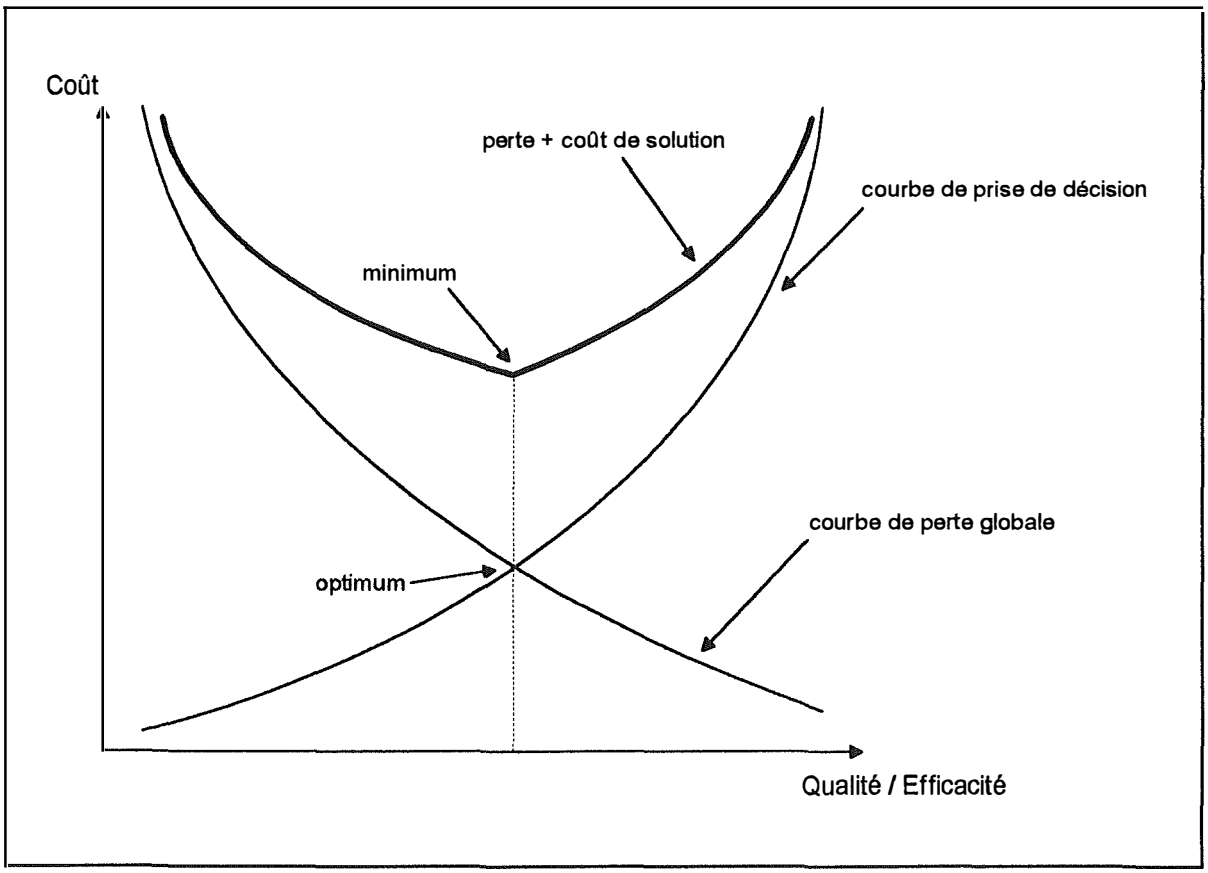


Figure 3.5. Le graphe de coût-qualité.

Le niveau de réponse qu'attend le client par rapport à sa contrainte non-fonctionnelle se trouvera quelque part sur l'abscisse et selon le pourcentage de couverture que la solution apportera, le point de réponse effectif se déplacera vers la gauche ou vers la droite.

c.3. Inconvénients de l'analyse :

Le principal inconvénient vient du fait de la difficulté à chiffrer le coût des moyens que l'on propose d'implémenter et du coût que l'on pourrait globalement supporter si les moyens n'étaient pas mis en oeuvre. De ce fait, toute l'analyse ne peut reposer que sur des tendances et non sur des éléments absolus, intangibles.

Deuxièmement, le graphe de coût-qualité représente avant tout un modèle théorique qui affiche deux courbes linéaires qui ne reposent pas sur quelque chose de concret. En effet, toutes les solutions potentielles que l'on peut apporter au problème des contraintes non-fonctionnelles ne présentent pas autant de points qu'il n'y en a sur les courbes, en fait il s'agit de solutions marquant quelques points ponctuels sur l'espace de ces deux courbes. Par conséquent, le point minimal vers lequel on doit tendre peut ne pas exister en tant que tel et de plus il se peut qu'il ne soit pas souhaitable de s'y ramener. Ce sera par exemple le cas pour le critère de performance où l'on désirera opter pour une solution plus performante que nécessaire afin de garder de la marge dans l'infrastructure pour les développements futurs.

3.4. La validation de la Solution

Lors de l'évaluation des solutions, on avait en main les informations nécessaires afin d'opérer le meilleur choix parmi des alternatives. Pour rappel, on disposait du degré d'importance de chaque contrainte non-fonctionnelle posée par le client afin de connaître les priorités posées par celui-ci; du raffinement formel développé pour chaque contrainte non-fonctionnelle permettant de disposer d'une vue sur les différentes possibilités de solutions globales qui étaient offertes; d'un pourcentage de couverture pour chaque but de décision de conception développé à travers la méthodologie et destiné à mieux cibler encore le niveau de réponse par rapport aux besoins du client; et on disposait finalement d'une analyse de risque et de coût qui permettait de juger de l'opportunité coût-qualité des alternatives mises à jour.

A ce stade, on va devoir exprimer le travail qui a été fait sur les contraintes non-fonctionnelles sans pour autant en venir aux concepts formels propres aux analystes dont le client n'a pas à faire l'effort d'apprentissage. On propose également le *graphe multi-métrique de validation de solution* qui n'est pas en soi un nouvel outil mais qui permet de synthétiser d'une manière graphique le niveau de réponse apporté aux contraintes non-fonctionnelles posées.

a. Les éléments à considérer

Pour présenter un traitement à la validation, il y a deux moyens, le premier consiste à montrer tout le cheminement du traitement. Le deuxième consiste à mettre en avant le résultat.

Pour les contraintes non-fonctionnelles, on peut présenter les deux styles de technique de validation. La remarque qu'il faut néanmoins faire est que la technique consistant à mettre en avant le résultat obtenu par le traitement des contraintes non-fonctionnelles ne se fait pas par une technique particulière puisque toutes les solutions apportées se concrétisent par une prise en compte aux différents niveaux du développement du système. La présentation se fait dans ce cas grâce aux moyens propres développés aux différents niveaux impliqués par les contraintes non-fonctionnelles.

Ainsi si une contrainte non-fonctionnelle a des répercutions sur le niveau fonctionnel, la validation pourra se faire par la technique de prototypage. S'il s'agit d'une contrainte de performance qui se répercute sur le niveau architectural, on pourra faire la validation grâce aux éléments de simulation que l'on retrouve à ce stade.

Le type de validation que l'on considère à proprement parler à cet endroit est donc uniquement une validation du cheminement suivi dans le processus de traitement. Il s'agit en quelque sorte d'une validation par traçabilité qui vérifie que tout les choix étaient conformes à ce que l'on pouvait attendre. A cette fin, on présente les graphes des buts qui permettent de justifier le choix des buts de décision de conception qui auront été choisis afin de répondre aux besoins. Cette présentation peut également être appuyée par les graphes de coût-qualité marquant la viabilité de la décision de conception.

b. Le graphe multi-métrique de validation de la solution

Le graphe multi-métrique de validation de la solution est simplement destiné à restructurer de l'information dont on dispose déjà afin de rendre mieux compte de l'efficacité de la solution proposée. Ce type de graphe est proposé dans l'étude des composants d'exigences de sécurité élaborée par L. Pfleeger dans [22]. Le graphe reprend le degré d'importance des contraintes non-fonctionnelles et le pourcentage de couverture des décisions de conception reprises dans la solution (figure 3.6). On part d'un cercle où chaque contrainte non-fonctionnelle est représentée par un quartier. La grandeur de chaque quartier dépend du degré

d'importance de la contrainte. *La droite de couverture* (la droite pointiée) qui tend à couper chaque quartier en deux exprime le pourcentage de couverture de la décision de conception prise pour couvrir la contrainte non-fonctionnelle impliquée par le quartier. Cette droite est d'autant plus petite que la couverture est importante. Ce que l'on cherche à atteindre à travers la solution retenue est donc de minimiser la forme géométrique inscrite dans le cercle et construite en joignant toutes les extrêmités des droites de couverture.

Le but d'un tel graphe est d'exprimer clairement le caractère de satisfaisabilité des décisions prises. Or, le principe de satisfaisabilité découle du fait que les contraintes non-fonctionnelles sont en général non-conciliables entre elles. Par conséquent on développera dans un graphe toutes les contraintes non-fonctionnelles qui ont été liées dans le graphe des buts par un lien de corrélation négatif. Autrement dis, on élaborera un graphe par ensemble de contraintes non-fonctionnelles liées par un lien de corrélation.

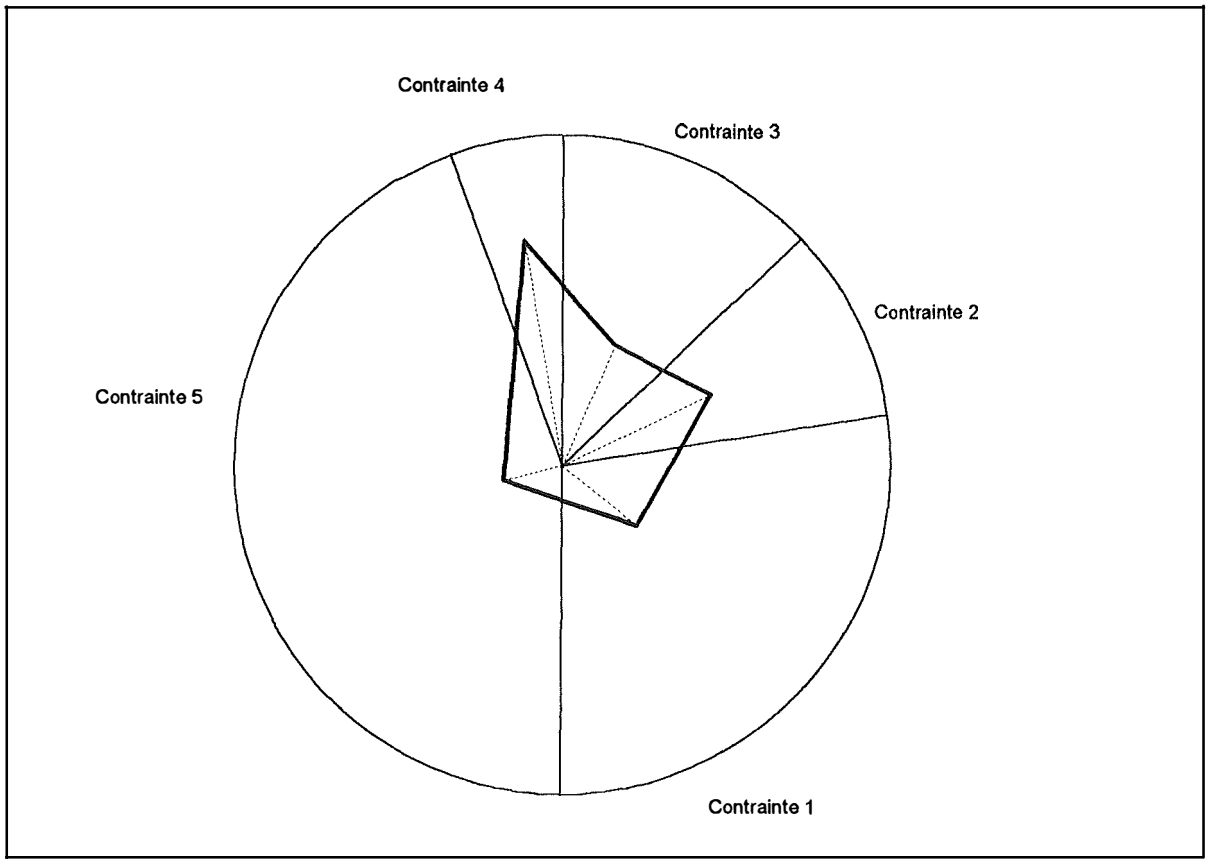


Figure 3.6. Le graphe multi-métrique de validation de la solution

Chapitre 4

Etude des Exigences Non-fonctionnelles de Sécurité et de Performance

Nous avons dans le chapitre 3 jeté les bases méthodologiques du modèle que l'on propose d'utiliser dans ces pages afin d'assurer le traitement de contraintes non-fonctionnelles qui font appel aux techniques de raffinement. Il faut maintenant utiliser tous ces concepts afin identifier les éléments propres à fournir une assistance au raffinement d'exigences en matière de sécurité et de performance.

Nous allons tenter de présenter ce chapitre en toute généralité, dans le but de ne générer que des éléments réutilisables. Ceci est assuré notamment par la présentation de décompositions de buts exprimées indépendamment des paramètres de ces buts et constituant une *taxonomie* pour la contrainte non-fonctionnelle traitée.

4.1. La modélisation des contraintes non-fonctionnelles de sécurité

Parmi les exigences non-fonctionnelles considérées par les recherches actuellement, la sécurité est probablement celle qui a reçu le plus d'attention. Rassemblant l'ensemble des originalités des différentes publications à ce sujet, on s'attache dans ce premier point à présenter une modélisation la plus complète possible.

A travers la publication de E. Dubois et de S. Wu [11], on identifie certains buts que l'on peut mémoriser pour en autoriser la réutilisation. De plus, cet article illustre l'implication des décisions de conception sur le niveau fonctionnel des spécifications logiciels. Dans [4] , outre la présentation du modèle décrit dans ces pages, L. Chung identifie un raffinement de buts pour la sécurité mais il établit surtout une taxonomie des buts de satisfaction. S.L. Pfleeger dans [22] montre qu'en matière de sécurité, il ne suffit pas de considérer les seules exigences non-fonctionnelles mais qu'il faut également tenir compte de la politique en matière de sécurité de l'organisation et des solutions techniques existantes (ce qui est compatible avec le modèle de Chung puisque les décisions de conception ne sont rien d'autre que des solutions techniques). Finalement, dans [14] de H.F. Hofmann et R. Holbein on trouve une présentation des différents composants de la sécurité.

Comme il a été précisé au chapitre 1, la prise en considération des exigences non-fonctionnelles implique une répercution sur les autres niveaux du cycle de vie du système développé. C'est exact, mais l'on ne décrit le résultat que par la seule considération du processus de traitement de l'exigence non-fonctionnelle. Il faut se rendre compte que dans la

mise en oeuvre de la sécurité l'on est tributaire de bien plus de facteurs que les seules exigences de sécurité émisent par le client.

Il est donc important de présenter un modèle renseignant sur les éléments à prendre en considération pour pouvoir aborder le traitement des exigences non-fonctionnelles de sécurité en tout état de cause. Ces différents éléments constituent un ensemble d'entrées et de sortie au modèle de Chung que l'on peut énumérer sous le nom de contraintes, exigences et techniques pour les entrées ainsi que solution pour la sortie.

a. Notion de contrainte, d'exigence et de technique

Nous allons laisser de côté le modèle de Chung et son raffinement des exigences non-fonctionnelles pour se préoccuper dans un premier temps des entrées et sortie que l'on sera amené à manipuler dans la méthodologie.

Si on adapte la vision du problème de S.L. Pfleeger dans [22] à la manière dont on présente les choses dans ces pages, nous pouvons représenter la méthode de raffinement de Chung comme une boîte noire à laquelle se rattachent toute une série d'entrées et de sortie (figure 4.1). Les entrées au modèle sont les ***exigences***, la ***politique de sécurité*** (contrainte) et les ***techniques de sécurité***. Les exigences ne sont rien d'autre que les exigences du client en matière de sécurité que l'on définit la plupart du temps comme étant établies à un haut niveau d'abstraction. Elles constituent la base du processus de raffinement et se retrouvent en conséquence au sommet des arbres du raffinement. Les techniques de sécurité sont constituées de tous les moyens dont l'analyste dispose afin de mettre en pratique des besoins en matière de sécurité. Il s'agit en fait des extêmités terminales de l'arbre de raffinement qui constituent les décisions potentielles de conception (carte d'accès, mot de passe, vigile, ...).

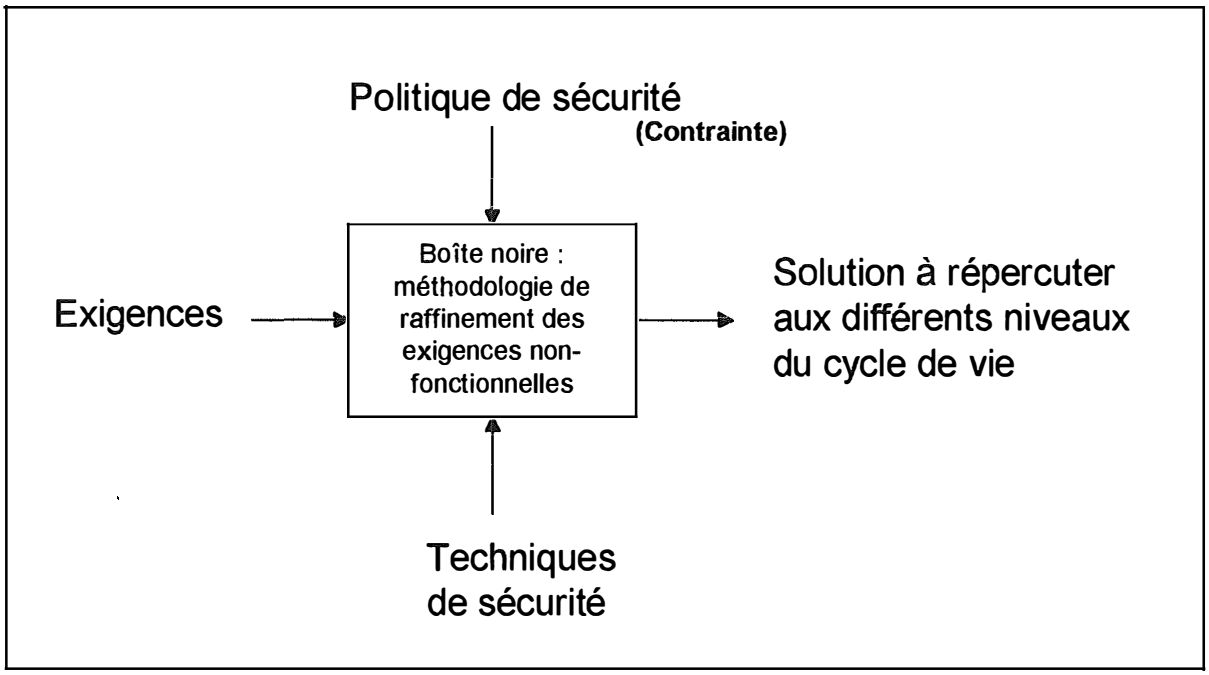


Figure 4.1. Les entrées-sortie du raffinement de sécurité.

Une taxonomie des exigences et techniques de sécurité a été établie et est présentée aux points a.2. et a.3., mais pour l'instant il va s'agir de présenter ce qui se cache derrière la politique de sécurité.

La politique de sécurité se distingue bien de l'ensemble des exigences de sécurité. Pour exprimer la différence entre ce qui constitue les exigences de sécurité et les contraintes, on peut simplement définir la manière dont il faut percevoir les deux termes. On peut dire qu'une exigence constitue un besoin dans un contexte bien précis alors que la contrainte serait un élément à respecter qui est inhérent à l'organisation dans laquelle vient se poser cette exigence. Ainsi, dans les exigences on retrouve les besoins de sécurité qui sont propres au projet lui-même et dans les contraintes on retrouve les exigences qui étaient posées plus globalement au niveau de l'organisation et qui doivent être respectée par le projet.

C'est cette distinction qui rend la prise en compte de la politique de sécurité (si elle existe) très importante car dans la définition du projet il faut tenir compte des besoins spécifiques au projet mais aussi de l'environnement dans lequel la réalisation vient se placer. Il se peut en effet que la sécurité ne soit pas seulement liée au projet et que le développement

adéquat de la sécurité soit fortement lié à la situation générale de l'organisation plutôt qu'au seul niveau de sécurité recquis par le projet et défini indépendamment de toute autre considération.

Par exemple, supposons qu'un département informatique afin de filtrer le personnel pouvant accéder à la salle des ordinateurs ait installé un lecteur de cartes d'accès. A l'époque, les concepteurs du projet ont jugés que la seule identification par carte suffisait puisque l'entrée au batiment utilisant le même moyen nécessitait, quant à lui, une authentification par mot de passe. Si maintenant un projet voit le jour pour modifier la technique de sécurité pour l'accès au batiment, il faudra tenir compte de la sécurité d'ensemble car si l'on décide de s'en tenir à une identification pour l'accès au batiment, alors le simple fait de disposer de la carte d'un informaticien donne maintenant un accès aisé à la salle des ordinateurs.

Plus concrètement, le terme de "politique de sécurité" désigne un engagement en matière de sécurité posé par l'organisation impliquée par le projet informatique et exprimé sous forme de COMPTE RENDU qui doit comporter trois éléments :

1. *une définition claire et non ambigüe des buts poursuivis en matière de sécurité pour l'entreprise*. Comme on vient de le souligner, la sécurité n'est pas une exigence que l'on peut diviser en autant d'éléments à traiter indépendamment les uns des autres. Sous peine d'aboutir à une incohérence ou à une incompatibilité entre les éléments de sécurité synonyme de perte significative pour la sécurité d'ensemble, la sécurité doit rester sous le jour d'une vision en permanence globale de la situation. A cette fin, un compte rendu des buts poursuivis en matière de sécurité par l'entreprise assure la vision globale du problème;

2. *une description de la responsabilité en matière de sécurité dans l'entreprise*. Il devient en effet indispensable de connaître les acteurs ayant la responsabilité en matière de sécurité car il est nécessaire de les impliquer dans les aspects de sécurité du projet. Ils prendront part en temps que tel, et à même titre que les clients du projet, à la phase de validation des décisions prisent en matière de sécurité;

3. *une confirmation de l'implication de l'organisation en matière de sécurité*. Ce point est très important car la mise en oeuvre de techniques de sécurité implique souvent une grande mobilisation de la part de l'organisation d'un point de vue structurel et/ou du personnel et/ou financier. Il faut donc savoir jusqu'où l'organisation est prête à s'investir pour assurer la couverture du besoin de sécurité. Par exemple, connaître la motivation du personnel pour l'instauration de techniques de sécurité joue un rôle dans la décision de conception choisie puisque cette motivation agit comme facteur de risque (de réussite ou non) du projet. Un personnel impliqué est apte à accepter des choix plus contraignants allant dans le sens d'une plus grande sécurité.

La boîte noire constituant le modèle de raffinement des exigences (figure 4.1), on se pose donc la question de savoir à quel endroit du modèle, la politique de sécurité va bien pouvoir jouer un rôle. Les contraintes exprimées à travers la politique de sécurité peuvent en cas de nécessité être formalisées dans le modèle sous forme de buts d'argumentation afin de jouer en faveur ou en défaveur d'une décision de conception potentielle.

b. Une taxonomie pour les exigences

Lorsque l'on parle d'exigences de sécurité dans un projet informatique, cette sécurité peut se décliner sous trois aspects. Soit on recherche *la sécurité d'un élément de la partie logiciel* du système développé (une donnée, un processus), soit on recherche *la sécurité d'un élément ne faisant pas partie de la conception logiciel* (un élément architectural, ...) ou bien finalement on désire utiliser les moyens informatiques pour *réaliser un projet de sécurité*. Pour ce troisième type de sécurité, il ne s'agit pas de contraintes non-fonctionnelles, cette sécurité constitue en elle-même le projet. Il ne s'agit pas d'un élément qui va venir s'appliquer sur d'autres éléments du projet, cet élément est autonome. Ce type de vision de la sécurité est perçu comme un ensemble de contraintes fonctionnelles exprimées par un client connaissant toutes les fonctionnalités que doit contenir son système informatique de sécurité.

Ce qui nous intéresse à notre niveau ne concerne que les deux premiers types de sécurité se rattachant au problème des contraintes non-fonctionnelles. De plus, si l'on regarde de plus près les éléments pouvant faire l'objet d'un besoin de sécurité, on identifie les *données*, les *éléments architecturaux* et les *processus* (dans le sens partie de programme). La donnée est l'élément primordial à protéger dans des systèmes tels que les systèmes d'informations; un élément architectural pourra faire l'objet d'une protection par exemple dans un système de gestion d'ascenseur ou l'on requerra des mesures de sécurité pour la cage d'ascenseur; finalement la sécurité appliquée aux processus pourra intervenir dans des systèmes automatisés de production industrielle pour raison de secret de fabrication.

Partant d'une exigence de sécurité pouvant impliquer un des trois éléments définis au paragraphe précédent, on va voir comment il est possible de décomposer cela pour mettre à jour les critères plus précis qui la composent. Ceci permet de disposer de raffinement de la sécurité autorisant la création de méthodes dans le modèle de Chung.

Si l'on s'attarde sur les travaux de Chung [4], de E. Dubois et de S. Wu [11], de H.F. Hofmann et R. Holbein [14] et encore de Sh. L. Pfleeger [22], on observe qu'il y a unanimité dans l'identification de certains critères de décomposition de la sécurité. Ces critères sont la *confidentialité*, l'*intégrité* et la *disponibilité*. Ces trois critères découlent de la manière dont on entend définir la sécurité. Une exigence de sécurité devra être vue comme appartenant à l'un de ces trois critères.

La confidentialité est caractérisée par un élément (donnée, appareil, ...) qui est reconnu comme secret ou privé. On doit par conséquent vérifier l'accès à ce type d'élément. La confidentialité retenue pour les seules données nous est définie par Sh. L. Pfleeger [22] qui explique qu'elle "... *est l'état qui existe lorsqu'une donnée est tenue confidentielle et est protégée de toute divulgation non-autorisée*."

L'intégrité cherche à garder le caractère authentique de l'élément concerné. Concernant l'exemple du système d'ascenseur, on décompose la sécurité de la cage en intégrité de la cage (indiquant que l'on doit assurer le maintient intègre de la cage en cas de rupture du cable. Par exemple, par un système de frein pour éviter la destruction de la cage). Toujours défini pour

les données, SH. L. Pfleeger [22] nous dit que "*l'intégrité est l'état qui existe lorsqu'une donnée est la même que celle du document source ou qu'elle a été correctement calculée depuis des données sources.* "

La disponibilité est un concept fort proche de la confidentialité et qui exprime que lorsque l'on est autorisé à utiliser un élément, cet élément doit pouvoir être disponible. Sh. L. Pfleeger [22] définit la disponibilité comme "... *l'état qui existe lorsqu'une donnée peut être obtenue dans une période de temps acceptable.*"

Le critère d'intégrité que l'on vient de définir peut à son tour faire l'objet d'une décomposition. Les deux sous-critères qui définissent la notion d'intégrité sont la *complétude* et l' *exactitude.* Il s'agit de deux critères qui sont complémentaires pour garantir l'intégrité dans le cas où le paramètre concerne une donnée (DATA). Cette décomposition n'est donc pas générale à tous les types de paramètres. Puisque ces sous-critères sont complémentaires, une recherche d'intégrité devra passer par la prise en considération des deux sous-critères. La complétude, selon Chung [5], dans la manipulation d'une base de données consiste à y maintenir tous les faits pertinents. L'exactitude, quant à elle, implique que les données concernées représentent effectivement la réalité qu'elles sont censées incarner.

Au regard de toutes ces considérations, on peut définir la taxonomie pour l'exigence non-fonctionnelle de sécurité à travers la figure 4.2. De nouveau, il faut rappeler que l'on exprime ici uniquement ce que l'on peut généraliser par rapport à toutes les situations qui peuvent se présenter à l'utilisation du modèle. Il faut donc bien percevoir que chaque projet particulier est susceptible de présenter des raffinements originaux et propres à lui-même. Ces raffinements originaux pourront être mémorisés grâce aux méthodes du modèle mais ne seront peut-être jamais réutilisés puisque correspondant à une situation bien particulière.

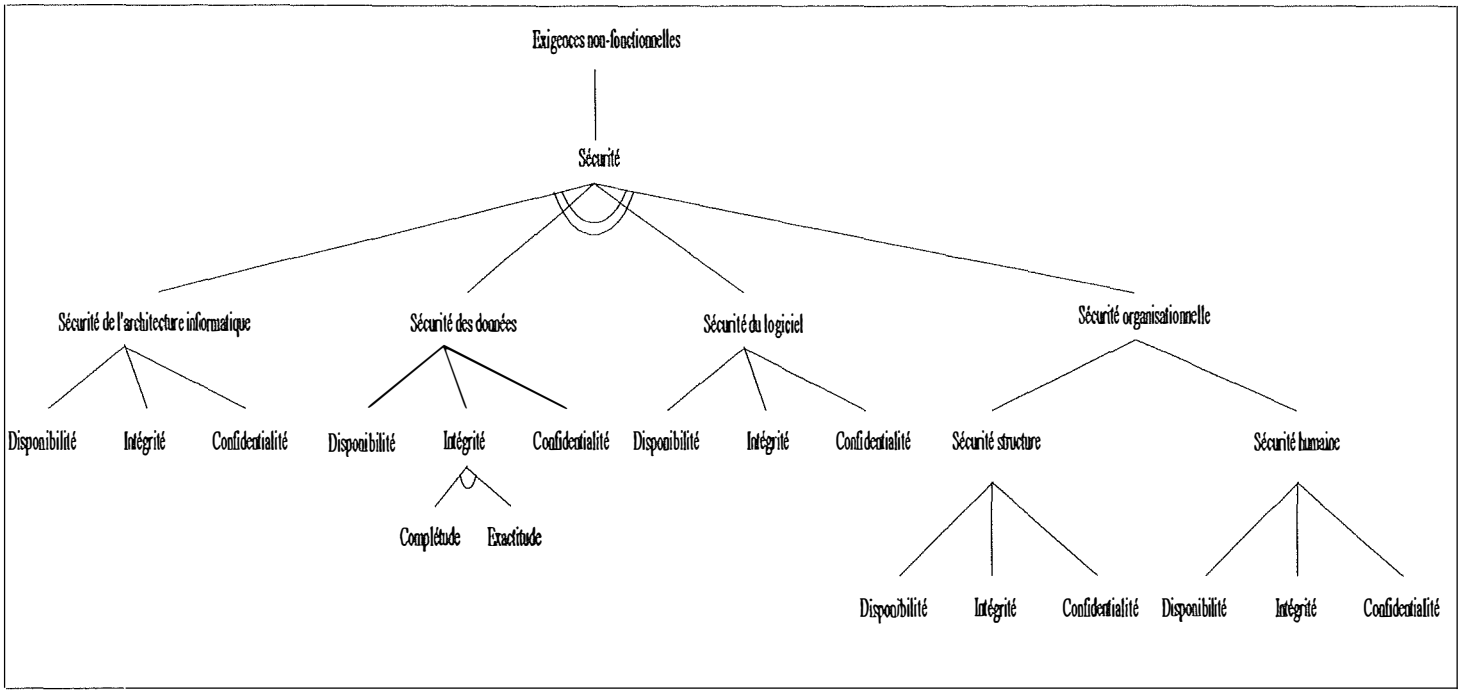


Figure 4.2. La taxonomie de la sécurité.

c. Une taxonomie pour les techniques

Puisque l'on a décomposé la sécurité en exactitude, complétude, confidentialité et disponibilité, on va discuter de manière non-exhaustive des techniques disponible afin de répondre à de tels besoins.

c.1. La confidentialité :

On présente à la figure 4.3, la taxonomie des décisions de conception telle qu'elle nous est donnée par L.Chung dans [4].

1. L'*identification* consiste à déterminer l'identité d'une personne. Ceci peut être assuré par *reconnaissance physiologique*, *mot de passe* et *badge*. Il s'agit en fait de moyens de protection assez faibles puisqu'un badge ou un mot de passe ne garantissent pas que le

détenteur de ces moyens en est bien le propriètaire. Une reconnaissance physiologique n'est pas non plus une certitude quant à l'identité de la personne (jumeaux, sosie, etc...).

2. La procédure d'*authentification* est plus sécurisante que l'identification car il s'agit ici de s'ASSURER de l'identité d'une personne afin de confirmer que cette personne est bien celle qu'elle prétend être. Ceci peut être mis en oeuvre à l'extrême par la prise de *mesures biologiques* (prise d'empruntes, ...) ou bien à moindre niveau par l'utilisation d'une *carte clé* ou d'un mot de passe tel que défini par la *procédure de Earnest*.

   Ce type de mot de passe est en fait fonction d'une donnée aléatoire et d'un contexte qui font que le mot de passe n'est jamais le même empêchant par là même une tierce personne de s'approprier le code. Par exemple pour une personne, le contexte est (3 * X) + 2 et la donnée aléatoire est un nombre généré par le système avant que l'on entre le mot de passe au clavier. Donc si le système vous présente 13, le mot de passe sera 3 * 13 + 2 = **41**. Nous pouvons encore sécuriser l'accès par mot de passe en présentant la procédure en *une passe* (une seule fois utilisée à l'entrée du système) ou bien en *multi-passe* (plusieurs fois empêchant ainsi l'accès par chance).

   Nous remarquons que la carte clé est bien un moyen d'authentification car le seul moyen existant pour tromper le système d'authentification consiste à dérober la carte clé ce qui n'est pas transparent vis-à-vis de l'utilisateur propriétaire de la carte qui peut donner l'alarme.

3. La technique de l'*alarme* est en fait un moyen de confidentialité complémentaire aux autres moyens. Il permet de prévenir toute tentative consistant à contourner les techniques d'authentification, d'identification ou d'autorisation d'accès. Il peut s'agir d'*alarme physique* protégeant les accès physiques (bâtiment, local, ...) mais également d'une *alarme software* protégeant l'accès à un système qui peut se faire, par exemple, par essais multiples d'introduction de mots de passe.

4. L'*autorisation d'accès* est une technique qui peut se faire par identification à l'*accueil*. Cette dernière technique est principalement utilisée pour filtrer les entrées occasionnelles.

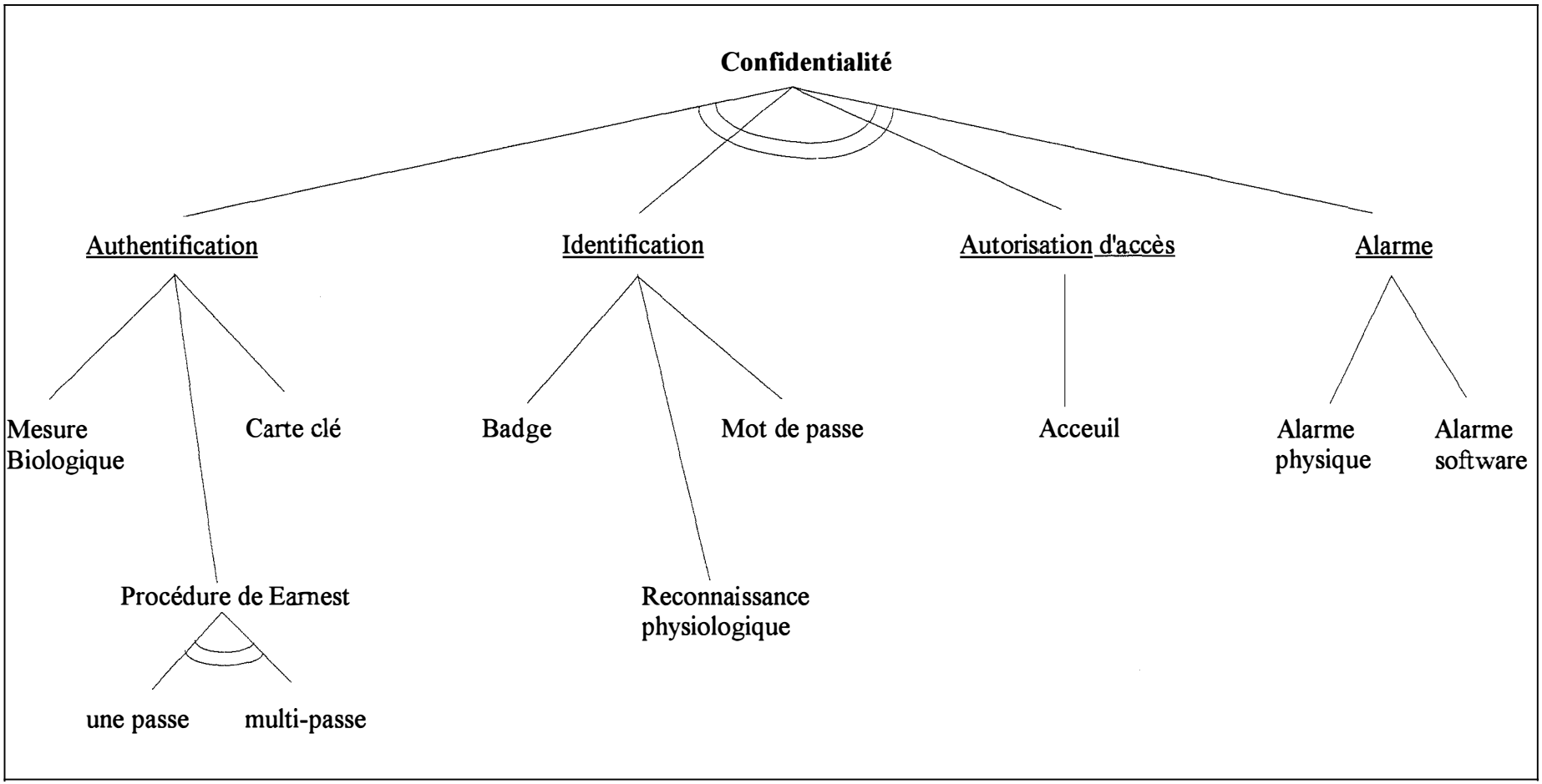


Figure 4.3. Taxonomie des techniques de confidentialité.

c.2. La disponibilité :

La disponibilité des données est fortement dépendante des moyens architecturaux ainsi que des techniques d'accès aux données. Pour les autres types de paramètres cela dépend du type de système que l'on développe. Il y a par conséquent une multitude de moyens assurant la disponibilité. L'étude de la disponibilité est ici limitée au seule donnée et l'on identifie uniquement la prise de back-up et la technologie RAID (Redundant Array of Inexpensive Disk) comme moyen de couverture du besoin de disponibilité (figure 4.4).

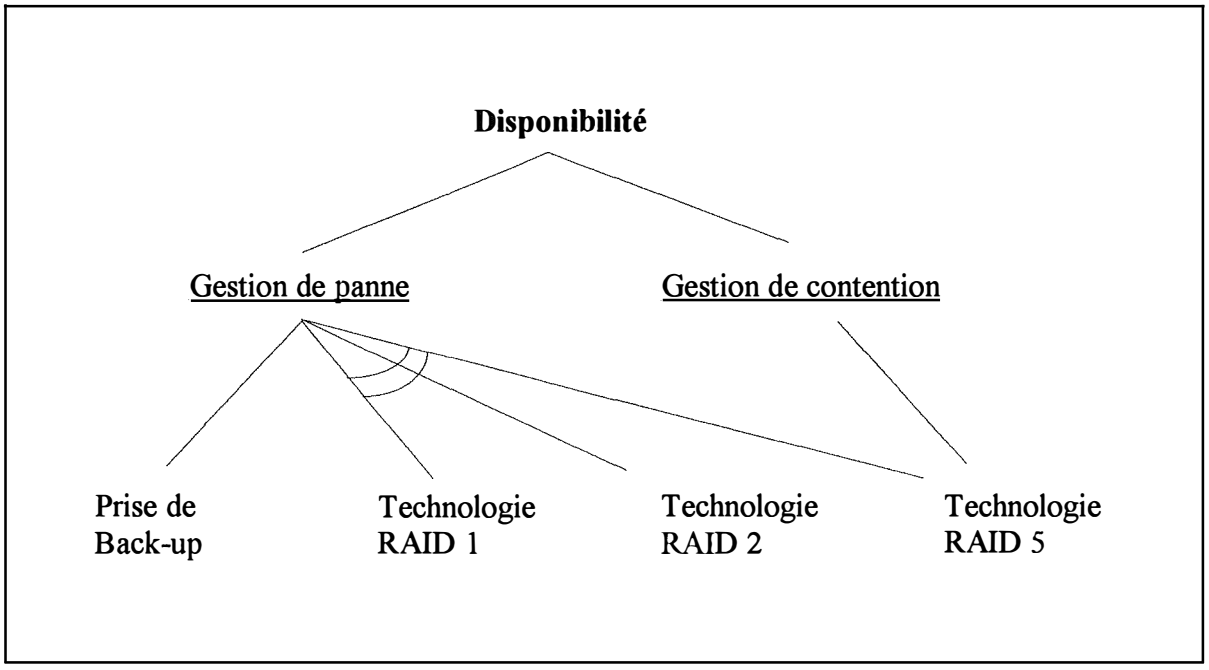


Figure 4.4. Taxonomie des techniques de disponibilité.

La disponibilité peut tout d'abord se décomposer en *gestion de panne* ou en *gestion de contention*. Par la gestion de panne on essaye de trouver un moyen qui permet d'assurer que le service peut toujours être rendu par le système système malgré une panne. La gestion de contention cherche à éviter quant à elle les délais qui découlent de demandes effectuées en même temps et visant un même support.

La technologie RAID apporte une solution du point de vue des exigences en matière de panne et de contention liées aux supports. Le principe repose sur une technique de dédoublement du support des données.

RAID 1 nous permet de gérer les pannes de support. Il s'agit en fait d'*une technique de mirroring*. C'est-à-dire que l'on utilise deux disques en parallèle sur lesquels on trouve deux copies identiques des données. Par conséquent, si l'un des deux supports tombe en panne, le service peut toujours être assuré. Les avantages sont principalement la haute disponibilité des données et le contrôle de disques assez simple à gérer. Le principal problème découle du coût de redondance puisque RAID 1 implique deux copies de chaque données. RAID 2 diffère de RAID 1 par le fait qu'il ne reprend pas la technique de mirroring et évite par conséquent la redondance. Une donnée est répartie sur les disques et un disque (voir plusieurs) constitue une parité pour les données. Ainsi si un disque tombe en panne, on peut retrouver la partie manquante de la donnée à extraire et garantir la disponibilité des données. Cette technique garantit un haut degré de disponibilité des données et également un coût peut élevé de redondance. De plus, le temps de réparation d'un disque est peu élevé. On change le disque endommagé et on le complète chaque fois qu'il y a écriture ou lecture grâce à la redondance du disque de parité qui nous permet de retrouver les informations manquantes. RAID 5 par le même principe que RAID 2 garantit une gestion de panne, mais il diffère de RAID 2 par le fait qu'une donnée est mise entièrement sur un disque. Ceci offre la possibilité de faire des accès en parallèles si on recherche des données qui ne sont pas sur le même disque. Ainsi cela garantit une meilleure disponibilité des données par réduction du temps d'attente à leur fourniture.

Commentaire :

Il est à noter que cette technique se retrouvera également comme but de satisfaisabilité dans l'arbre de disponibilité d'un processus.

c.3. L'intégrité :

Comme il fut expliqué ci-dessus, l'intégrité peut dans le cas de traitement impliquant un paramètre de type DATA se décomposer en problème de complétude et d'exactitude. Voyons maintenant les solutions qui peuvent être apportées à ces différents problèmes.

• La complétude :

Une solution peut consister à imposer que les champs composant l'enregistrement du paramètre mentionné ici soient définis non nuls dans la base de données (figure 4.5).

• L'exactitude :

L'exactitude des données constitue un grand problème car nombreuses sont les causes pouvant mener à une corruption des données. Néanmoins, il existe des moyens d'assurer un suivi des données dont voici les plus communes (figure 4.5) :

1. *La vérification de consistance* est une méthode qui permet de vérifier que l'information manipulée correspond bien à un standard de représentation. On peut prendre comme exemple la manipulation des numéros de compte bancaire si le moyen mis en oeuvre pour assurer l'exactitude consistait à vérifier le nombre de chiffre composant le numéro.

2. *La certification* : il s'agira d'une information additionnelle couplée à une donnée qui garantit que cette donnée est bien exacte. Dans ce cas, il s'agit plus d'une procédure visant à certifier une information à encoder dans le système informatique. C'est la pratique qui est utilisée pour garantir qu'un numéro de compte bancaire manipulé est bien valide. Ceci est certifié par l'utilisation de deux chiffres supplémentaires (check-digit).

3. *L'autorisation* est également une procédure en amont du système informatique. Il s'agit d'accorder ou non à une personne l'encodage des informations dans le système. C'est un moyen de s'assurer que l'on ne met pas n'importe quoi dans la base de données.

4. *Les moyens de manipulation d'exception* visent à supprimer les erreurs d'encodage d'informations dans le système qui surviendraient de par la nature exceptionnelles des

informations elles-mêmes. On a tous connu un centre administratif ne sachant pas comment traiter notre cas particulier aux yeux du système informatique. Il faut donc prévoir des procédures indiquant comment gérer les situations exceptionnelles lors de l'encodage pour assurer une bonne interprétation a posteriori.

5. *L'examen parallèle* n'est pas vraiment un moyen d'assurer l'exactitude mais plutôt un moyen de repérer l'inexactitude. Il s'agit en fait d'une procédure d'analyse de comportement des données. On observe le comportement d'une donnée en se référant à l'*historique* de la donnée ou à un modèle *statistique* de son évolution pour identifier les situations d'inexactitude potentielle. Un système bancaire a par exemple renvoyé un message "N'avez-vous pas commis une erreur?" à un employé ayant introduit dans le système un taux d'intérêt de 80 pourcent au lieu de 8,0 pourcent.

6. *La confirmation* est une procédure impliquant un double contrôle de l'information. Ce contrôle peut se faire soit directement par la même personne (*canaux directs*) soit par une autre personne (*canaux distincts*). On retrouve ce moyen de sécurité lorsque le système demande si l'on est vraiment certain de vouloir remplacer les anciennes données par les nouvelles.

7. *La vérification* est un mécanisme qui nécessite une double entrée de l'information impliquant l'identification d'une erreur si les deux entrées ne sont pas identiques. De nouveau cette procédure peut se faire par canaux identiques ou distincts.

8. *La validation* : le dictionnaire définit la validation comme étant l'action de vérifier la condition de satisfaisabilité aux conditions légales pour produire ses effets. On a donc à faire ici à une procédure de vérification d'informations utilisant certaines directives pour assurer que ces informations rencontrent bien des standards prédéterminés. Cela ne garantit pas que les données seront l'image de la réalité à modéliser, mais garantit que la base de données ne sera pas mise dans un état incohérent. Cela peut par exemple être mis en oeuvre par un outil d'assistance d'encodage d'informations dans la base de données.

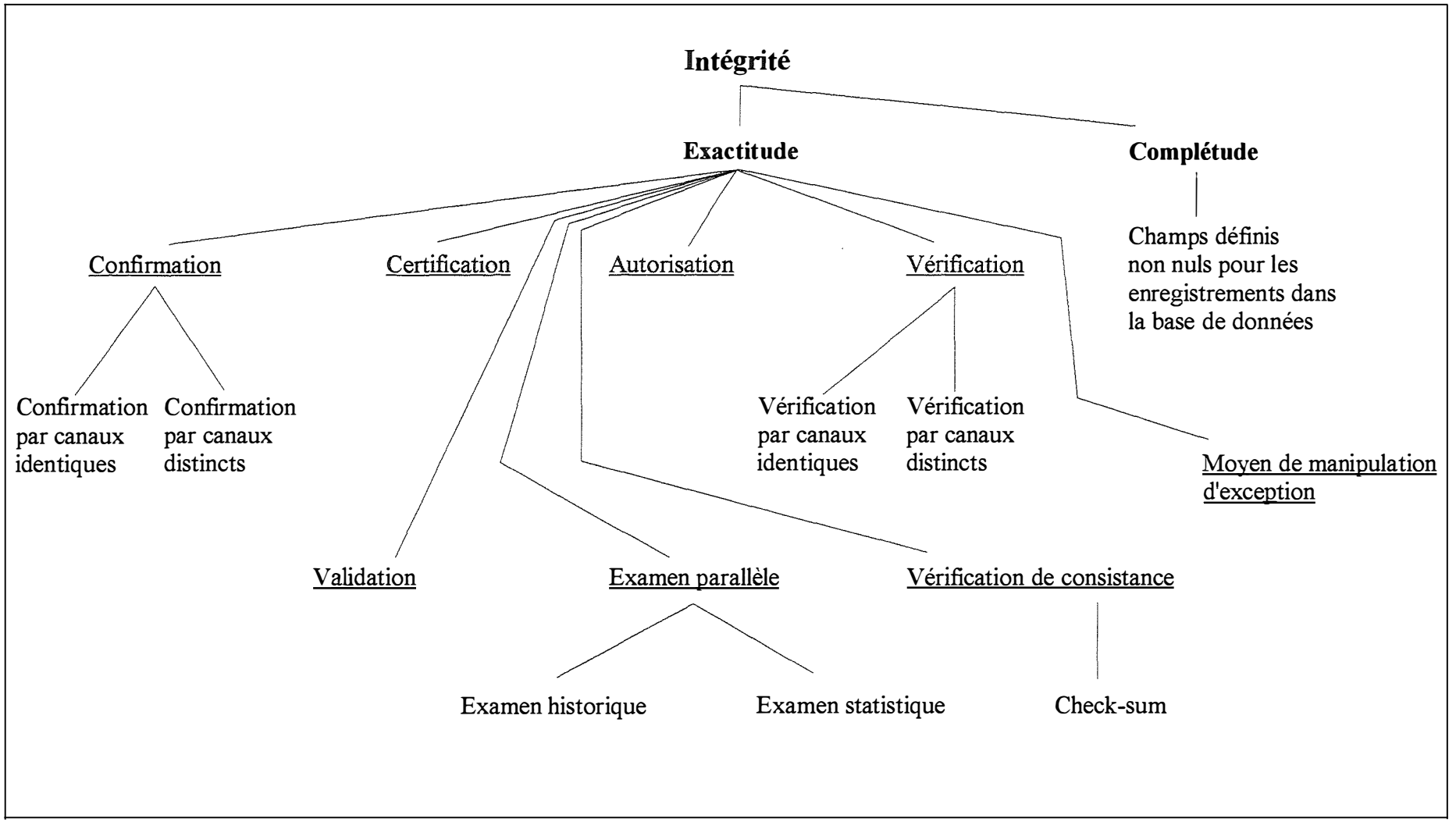


Figure 4.5. Taxonomie des techniques d'intégrité.

d. Répercussion des choix sur les autres niveaux du développement

Nous avons longuement insisté dans le premier chapitre sur le fait que les contraintes non-fonctionnelles se répercutent sur les autres niveaux du développement du système informatique ainsi que sur la structure organisationnelle et sur l'architecture de l'organisation (découpe en départements, infrastructure matériel, ...). Savoir quel niveau du développement du projet et/ou quelle partie de l'environnement de l'organisation doit tenir compte d'une contrainte non-fonctionnelle dépend en fait de la décision de conception qui est retenue dans le raffinement de la contrainte. effectuer cette étude au regard des techniques de prise en compte de la sécurité que l'on vient de développer.

d.1. La confidentialité : (figure 4.6)

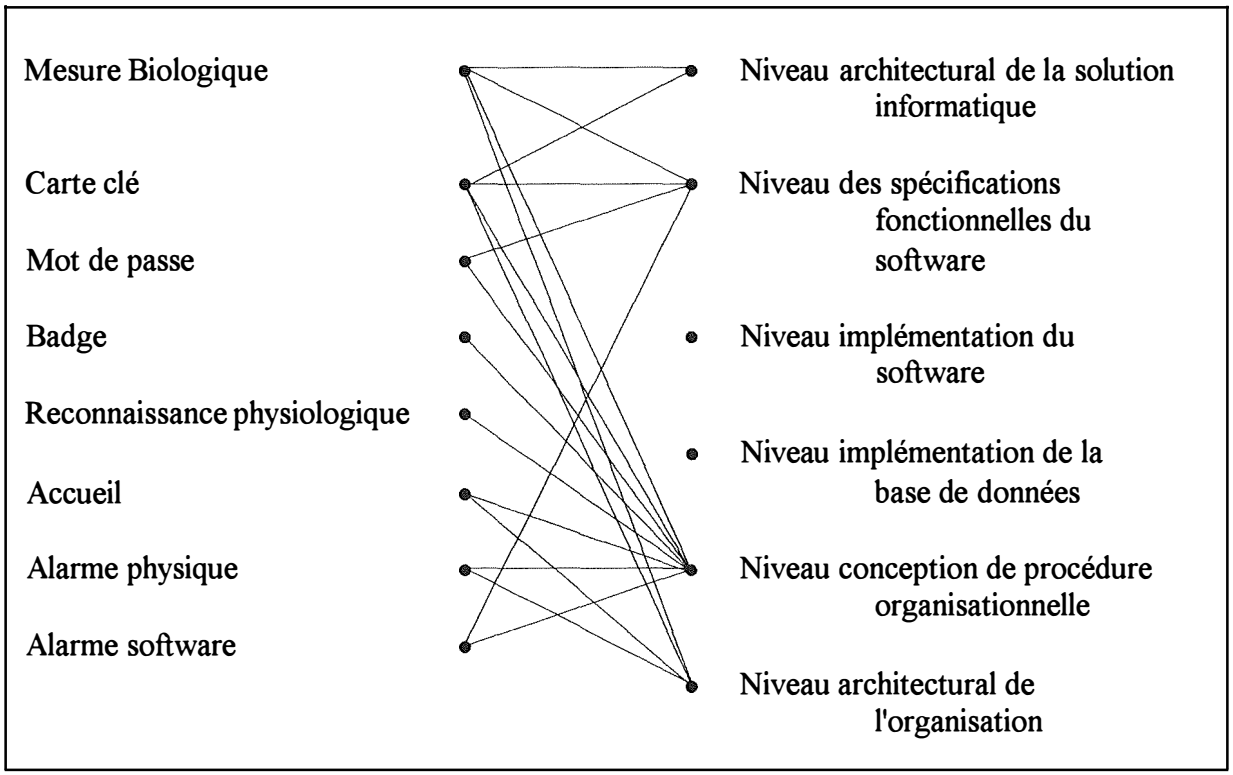


Figure 4.6. La répercussion des techniques de confidentialité.

On ne va pas s'attarder sur cette répercussion car elle est assez parlante en soi, si ce n'est pour souligner que toutes ces techniques peuvent avoir un impact sur le niveau de la conception de procédures organisationnelles puisqu'il se peut qu'aucune structure n'existe avant le projet pour prendre en compte une volonté de confidentialité. De plus, même si cette structure existe déjà il faut prendre en considération les nouvelles procédures à cet égard. Ainsi si l'on retient le port d'un badge, il faud définir la procédure de gestion du système de badge : qui autorise l'émission du badge, qui s'occupe de l'émission, de la restitution, de la gestion de perte, de la gestion de vol, etc ...

d.2. La disponibilité : (figure 4.7)

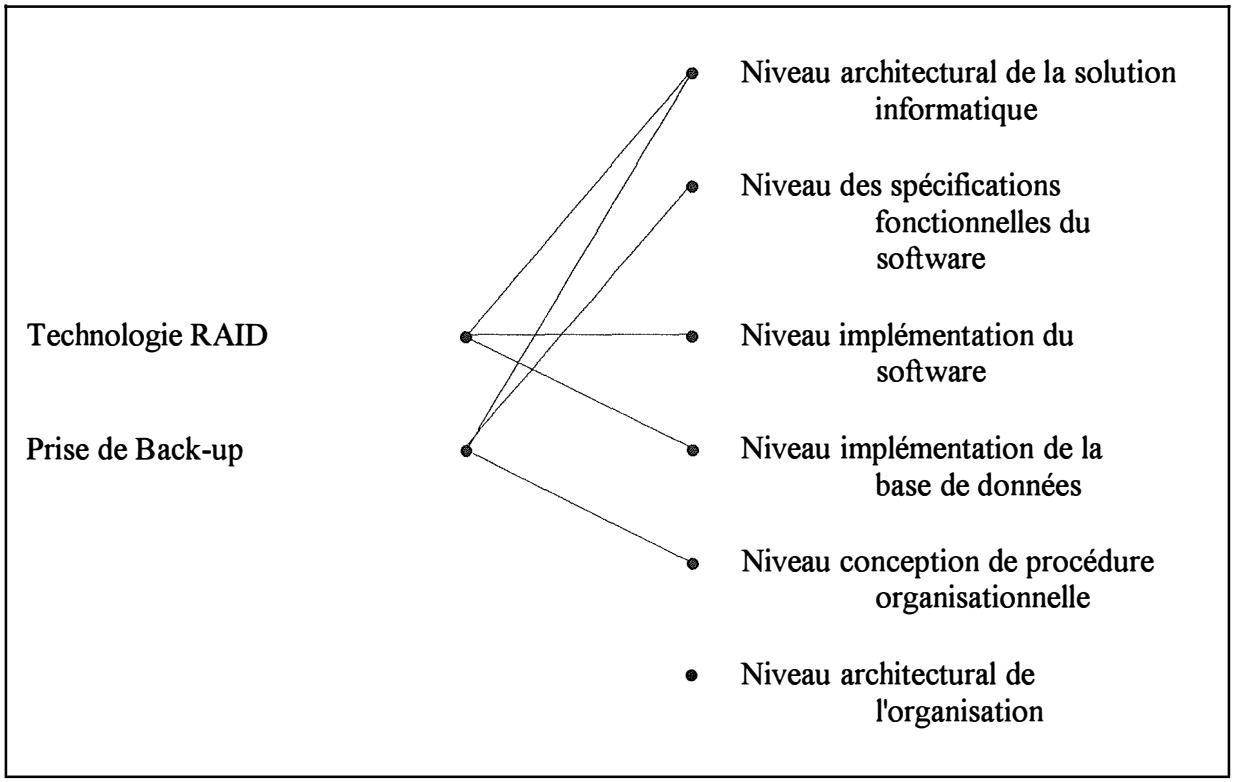


Figure 4.7. La répercussion des techniques de disponibilité.

La technologie RAID ne joue que pour le niveau implémentation puisque précisément il s'agit bien d'un choix de technique d'implémentation des données. Il faut remarquer que les trois niveaux impliqués doivent se concerter pour savoir quel est le nombre de supports à utiliser afin d'atteindre une optimalité dans l'utilisation. Le nombre de support est jugé d'après

les données utilisées par la base de données et le choix du nombre de supports aura une répercussion sur la manière de programmer le fonctionnement des enregistrements et des restitutions des données de la base au niveau implémentation du software (si l'on doit prévoir les gestionnaires).

Quant à la prise de Back-up, elle peut impliquer une nouvelle procédure organisationnelle. Par exemple, une personne ayant la responsabilité de choisir dans les bandes attribuées pour la sauvegarde celles qui selon une procédure déterminée doivent être utilisée pour cette journée. Cela peut également avoir des conséquence sur les fonctionnalités du système à développer : il se peut que l'on demande le développement d'une application batch qui déclenchera la sauvegarde automatiquement chaque soirée.

d.3. La complétude : (figure 4.8)

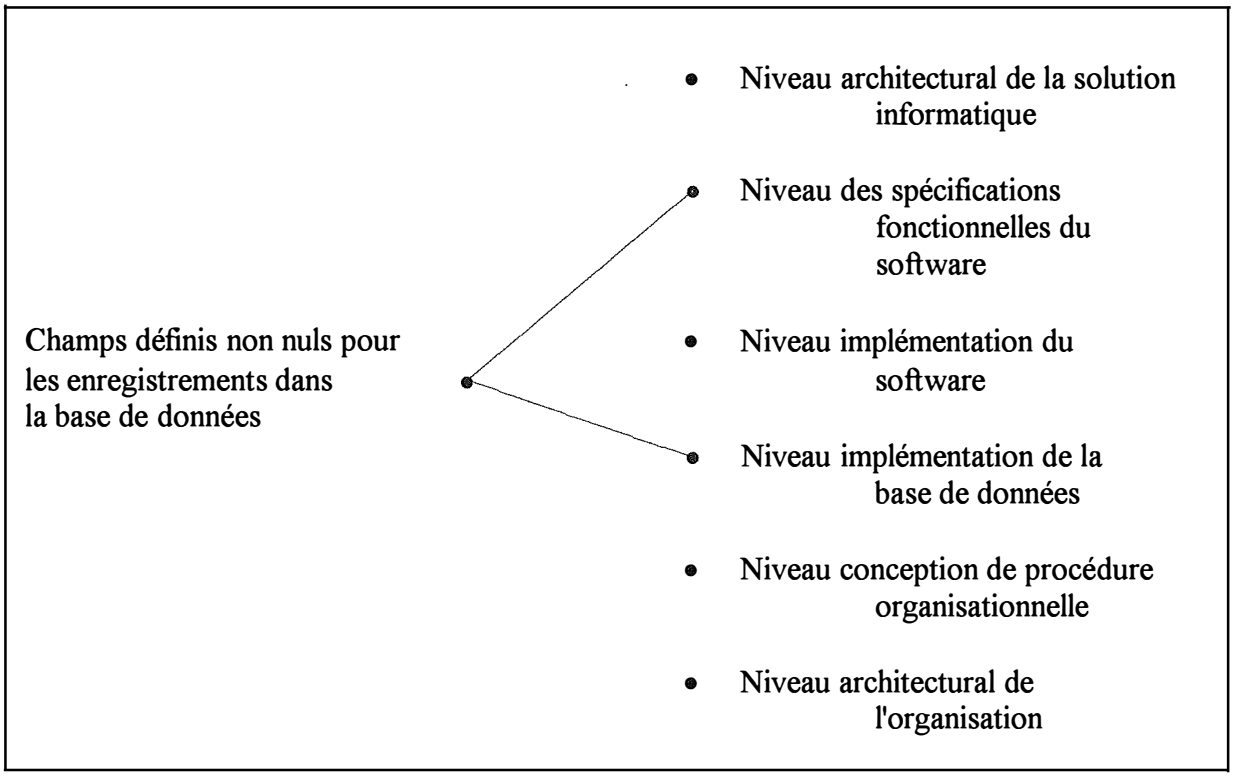


Figure 4.8. La répercussion des techniques de complétude.

Définir des champs de données comme ne pouvant pas prendre de valeur nulle peut être mis en pratique par définition d'une fonctionnalité dans la partie applicative mais aussi faire l'objet d'une spécification lors de la définition de la structure de la base de données.

d.4. L'exactitude : (figure 4.9)

Il n'y a pas grand chose à signaler si ce n'est que l'exactitude met grandement à contribution le niveau conception de procédures organisationnelles. Par conséquent, on remarque qu'au même titre que la confidentialité, l'exactitude peut être coûteuse en moyen humain à déployer (nouveau poste à pourvoir, charge de travail supplémentaire, etc ...). C'est pour cette raison que l'on a insisté auparavant sur l'importance de savoir jusqu'à quel point l'organisation est prête à investir pour assurer la sécurité.

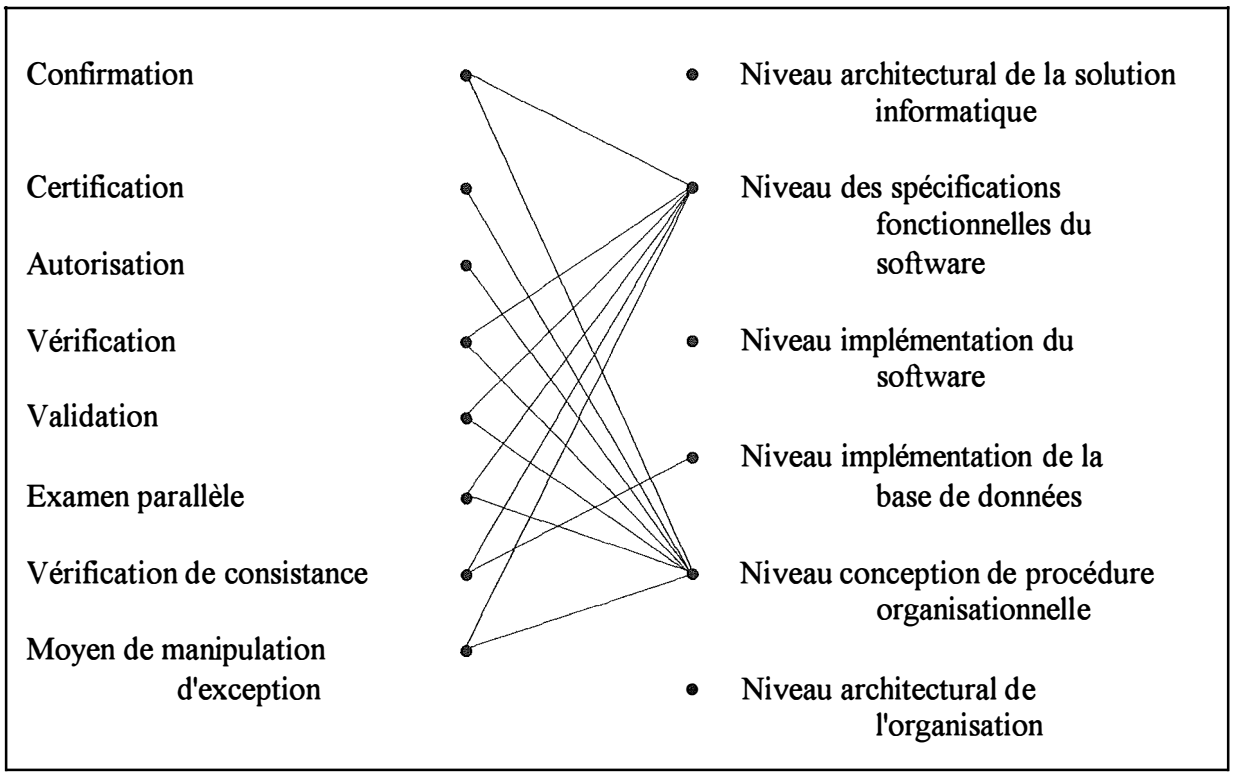


Figure 4.9. La répercussion des techniques d'exactitude.

4.2. La modélisation des contraintes non-fonctionnelles de performance

Ce que l'on va établir dans ces lignes concerne l'identification des différents composants de la performance. Avant de débuter cette analyse il faut néanmoins émettre un certain nombre de remarques.

1.Les exigences en matière de performance sont souvent établies de manière brèves parce que le client s'attend à ce que le système rencontre ses préoccupations IMPLICITES de performance. En conséquence, bien souvent lorsque des besoins de performance sont exprimés, cela implique que le caractère performant est primordial et doit être maximisé dans les limites du possible.

2.La méthodologie va différer par rapport à ce que l'on a vu jusqu'ici dans le sens où le client est peu impliqué dans le raffinement des exigences de performance puisque le résultat exprime des orientations en matière d'implémentation où de mesures de performance qui sont censées être transparentes aux yeux du client. Par conséquent, la validation du résultat se fait pour le client au niveau des résultats de l'implémentation et la validation des décision de conception au niveau du traitement de la performance se fait par les informaticiens.

La taxonomie présentée découle de la publication de J. Mylopoulos, L. Chung et B. Nixon [17] qui présente un traitement de performance associé à une base de données. Les recherches effectuées par A.L. Opdahl [21] aident aussi à l'établissement d'une taxonomie par une présentation plus théorique du problème mais également plus complète.

a. La performance dans le projet informatique

Lorsque l'on veut traiter un problème de performance lié à un projet, on peut considérer ce problème sous l'*angle de l'architecture informatique* qui sera utilisée ou bien sous l'*angle des processus* qui vont sous-tendre les applications ou sous l'*angle des données* (et plus particulièrement leur modèle de placement et de structuration sur le support) et finalement sous l'*angle organisationnel*. La première action du raffinement consiste donc à préciser par ces quatre principes le(s) composant(s) du système qui est (sont) visé(s). On remarque que cette décomposition n'est en aucun cas mutuellement exclusive. Au contraire, il est parfaitement concevable que le problème de performance soit répercuté à travers les quatre critères.

Ces quatre angles de vue du problème sont donc à la base de la taxonomie de support du raffinement que l'on présente à la figure 4.10 et qui découle des recherches effectuées par A.L. Opdahl dans [21]. Il va s'agir maintenant de se pencher sur la signification de toute cette taxonomie exprimant la décomposition de la performance en sous-critères. Il est toutefois très important de noter que dans cette décomposition nous ne sommes pas allé jusqu'à l'identification de solutions opérationnelles. On reste donc au niveau de la découpe incarnant les buts non-fonctionnelles car cette décomposition serait assez lourde à effectuer ici. Dès lors, **on se contente d'amorcer le raffinement jusqu'à l'établissement d'un but représentant le *paramètre* qui sera l'élément du système devant être optimisé en terme de performance**. Au delà de ce point, il s'agit de reformuler les techniques d'optimisation de base de données, d'architecture informatique et logiciel afin de pouvoir les intégrer dans la taxonomie.

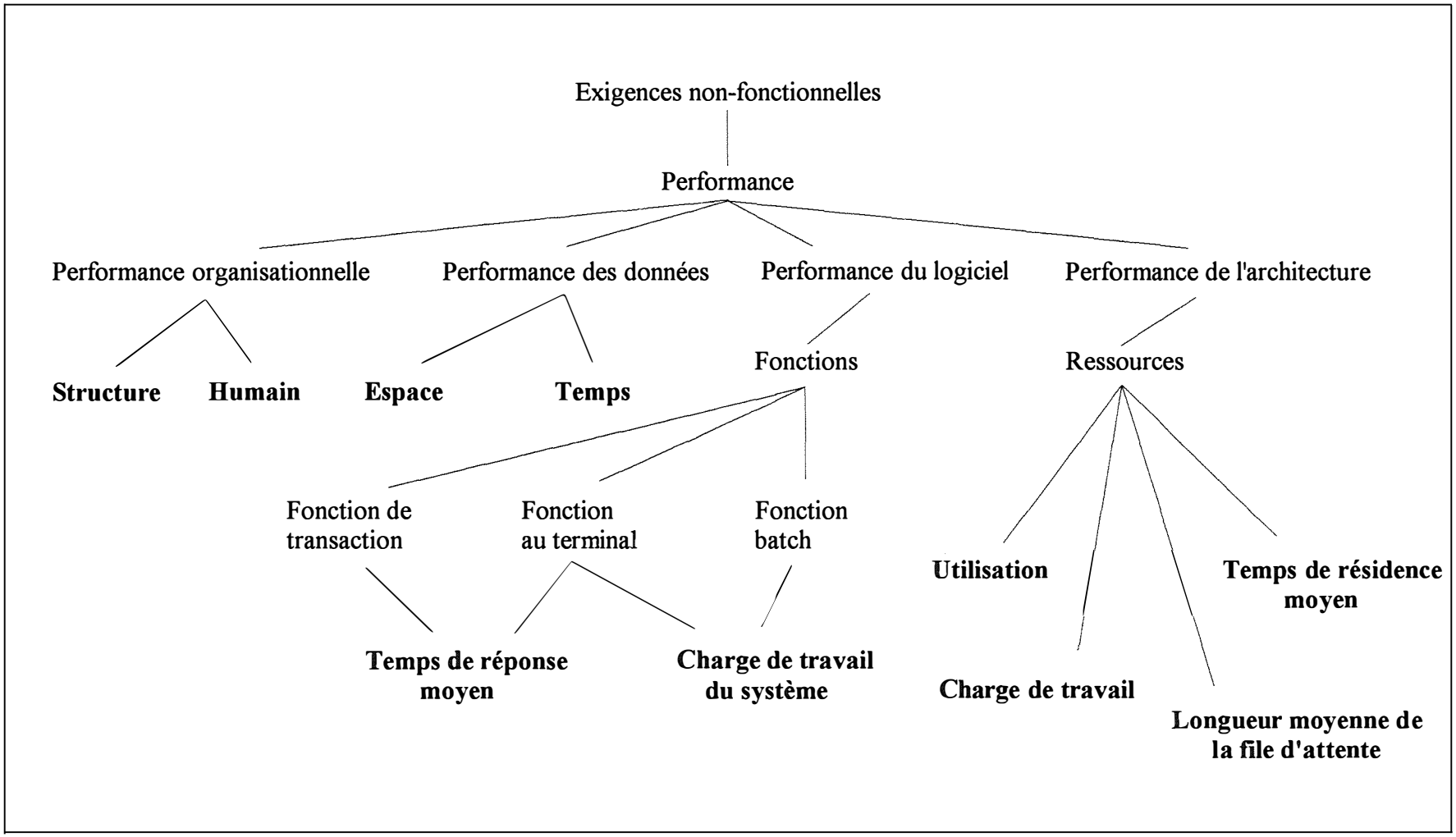


Figure 4.10. La taxonomie de la performance.

a.1. La performance du logiciel :

L'idée sous-jacente ici est que l'on perçoit le concept de performance en terme de ce qui sera observable par les futurs utilisateurs du système en développement. Par extension, comme l'utilisateur interagit avec le système, ce qu'il observe du système c'est le comportement de réponse de celui-ci pour les fonctionnalités qu'il utilise. Par conséquent, on parle de l'établissement de performance au regard des différents types de fonctions que peut contenir un système informatique. On a besoin d'identifier les différents types de fonctions car certains paramètres de mesure de performance sont plus pertinents que d'autres pour un type particulier de fonction. On identifie ainsi les *fonctions de transaction*, les *fonctions batch* et finalement les *fonctions au terminal*.

L'ensemble des fonctions de transaction regroupe toutes les fonctions qui sont utilisées par un grand nombre d'utilisateurs mais non-fréquemment dans le système. Nous pouvons par exemple considérer la fonction de connexion à un réseau comme fonction de transaction. Un grand nombre de personnes utilisent le réseau, elles ont donc besoin de se connecter mais ne passent pas leur temps à répéter les actions de connection-déconnection.

Dans l'ensemble des fonctions batch, on intégre toutes les fonctions qui sont exécutées par un nombre fixe de jobs dans le système. Par exemple, la fonction de mise à jour de données concernant les achats des clients dans une entreprise de vente par correspondance.

Finalement, dans l'ensemble des fonctions au terminal, on retrouve toutes les fonctions qui sont utilisées par un nombre fixe d'utilisateurs et de manière fréquente dans le système. Ce sera le cas pour une fonction qui vérifie les informations concernant les produits commandés dans notre entreprise de vente par correspondance. Le nombre d'utilisateurs est fixe dans notre cas puisque les seules personnes accréditées pour réaliser cette tâche seront du département de codage des commandes afin de vérifier l'existence du produit et du département des fournitures pour vérifier l'état du stock.

Si un grand nombre d'utilisateurs utilisent une fonction spécifique et qu'entre deux utilisations, le *temps de réflexion* des utilisateurs est très important, alors cette fonction est

classée dans l'ensemble des fonctions de transaction parce que l'on considère que les utilisateurs utilisent cette fonction non-fréquemment. Mais si le temps de réflexion est très court, alors on peut peut-être classer cette fonction comme faisant partie de l'ensemble des fonctions au terminal.

Dès lors que l'on a défini à quel type de fonction est rattachée celle dont l'analyste doit assurer la performance, il y a deux types de paramètres de performance qu'il peut considérer comme pertinents. Ces deux paramètres sont d'une part *le temps de réponse moyen* (T.R.M.) et d'autre part *la charge de travail du système* (C.T.S.).

Le temps de réponse moyen signifie que l'utilisateur doit attendre en moyenne x unités de temps pour obtenir une réponse du système pour la fonction spécifiée. Par exemple, le temps de réponse moyen pour vérifier une information au sujet d'un produit dans la base de données est de deux secondes. Donc, c'est le temps écoulé entre la demande concernant le produit et la réponse à la demande.

La charge de travail du système considère le nombre de fois que le système traite une fonction spécifique par unité de temps. Il y a un lien entre le temps de réponse moyen et la charge de travail, mais les deux paramètres ne sont pas deux manières de représenter la même information car la charge de travail prend en compte le temps de réflexion de l'utilisateur ce qui n'est pas le cas pour le temps de réponse moyen.

Comme représenté à la figure 4.10, l'utilisation de tous ces paramètres n'est pas nécessairement pertinente pour l'établissement de performance car il y a un lien entre certains paramètres et les différents types de fonction. Pour une fonction de transaction, le seul paramètre pertinent est le temps de réponse moyen. Ici, si la fonction est exécutée non-fréquemment alors il n'est pas pertinent de se concentrer sur la charge de travail du système parce que ce paramètre considère le temps de réflexion des utilisateurs entre les différentes exécutions de la fonction, mais il considère aussi le temps durant lequel les utilisateurs ne veulent pas utiliser la fonction. Par conséquent, un lien existe seulement entre ce type de fonction et le temps de réponse moyen.

Pour une fonction batch, on peut seulement considérer avec intérêt le paramètre de charge de travail du système. L'argument vient ici du fait que lorsque le système finit son travail sur une fonction batch, il la remplace immédiatement par une autre fonction batch, par conséquent il est plus intéressant d'utiliser le paramètre de charge de travail car on est amené à considérer le nombre de fonctions de ce type qui sont exécutées par unité de temps.

Pour une fonction au terminal, les deux paramètres sont pertinents. Mais comme il a été précisé avant, si la charge de travail considère le temps de réflexion de l'utilisateur entre deux exécutions, alors en accord avec A.L. Opdahl [21], le temps de réponse moyen représente un critère d'optimisation de la performance plus pointu. Néanmoins, se soucier des deux paramètres peut être intéressant car si le temps de réflexion semble être considéré comme un terme parasite, Sa prise en considération implique de se préoccuper de la performance du paramètre humain qui découle sur une prise en charge de l'efficacité de l'interface (plus l'interface est adaptée à l'utilisateur, plus sa performance est optimale).

a.2. La performance de l'architecture :

Dans cette vision particulière de la performance, on va considérer celle-ci en terme des ressources du système dont l'établissement de la performance se base sur *l'utilisation* (Utilisation), *le temps de résidence moyen* (T.R.M.), *la charge de travail* (C.T.) et *la longueur moyenne de la file d'attente* (L.M.F.A.) d'une ressource.

L'utilisation des ressources mesure le pourcentage des ressources utilisées par le système. Supposons que l'on considère un réseau dont la capacité de la ligne constitue la ressource et dont le réseau est composé de deux ordinateurs joints par une ligne point à point. On peut dès lors rechercher la performance en terme d'utilisation de cette ressource correspondant à la charge de travail de la ligne. En effet, on doit assurer que la charge moyenne de cette ligne sera telle qu'il n'y aura pas de surcharge impliquant une file d'attente synonyme de perte de performance du système. Généralement, on affirme que la charge de travail moyenne d'une ligne de communication ne doit pas dépasser les soixante pourcent en moyenne afin d'éviter la saturation de la capacité durant certains pics d'utilisation.

Le temps de résidence moyen pour sa part est un paramètre qui mesure si ce qui utilise une ressource la conserve longtemps ou est susceptible de la restituer rapidement. Si l'on prend l'exemple du buffer d'un contrôleur de disque dur comme ressource, un temps de résidence moyen qui est établi très bas peut témoigner d'une bonne performance dans l'échange d'informations entre le disque dur et la RAM.

Commentaire :

L'exemple présenté traitant du cas d'un buffer est représentatif du fait que l'on peut rechercher la performance simultanément au travers des quatre angles de vue définis précédemment. Si l'on désire assurer la performance pour l'accès à une base de données, ce critère de performance ne sera pas seulement tributaire de la constitution d'un index particulièrement bien choisi pour les données. Il est dès lors intéressant de remarquer, en relation avec la figure 4.10, que l'on peut rechercher la performance de l'accès à la base de données en décomposant le critère de performance à la fois en performance des données mais aussi en performance de l'architecture informantique prenant pour paramètre le temps de résidence moyen d'une donnée dans le buffer qui est utilisé pour l'échange d'information concernant la base de données.

La charge de travail est un paramètre qui mesure le nombre d'éléments qui arrivent en traitement et qui utilisent les ressources par unité de temps. La différence entre la charge de travail et l'utilisation des ressources n'est pas facile à identifier au premier coup d'oeil, mais si le paramètre d'utilisation d'une ressource est une mesure de la capacité utilisée, le paramètre de charge de travail donne une idée sur le caractère dynamique de cette ressource. L'occupation de la ressource est de soixante pourcent, mais est-ce que le roulement des données qui utilisent cette ressource est régulier, rapide ou lent ?

Finalement, la longueur moyenne de la file d'attente concerne le nombre d'éléments qui attendent en moyenne de pouvoir disposer d'une ressource. Pour l'exemple impliquant un réseau, cette recherche de performance sera intéressante si l'utilisation de la ressource est telle qu'il est impossible d'éviter à certains moments un manque de capacité impliquant la formation d'une file d'attente.

Commentaire :

Ce dernier exemple montre que les paramètres sont étroitement liés. Nous pouvons donner quelques exemples typiques qui peuvent aider à comprendre facilement la différence entre tous ces paramètres :

1. Si "Temps de résidence moyen " est élevé
   et "Charge de travail" est élevée
   Alors vous avez de grande chance d'obtenir
   "Utilisation de la ressource" est élevée
   et "longueur moyenne de la file d'attente" existe et est élevée

2. Si "Longueur moyenne de la file d'attente" existe
   Alors "Utilisation de la ressource" est élevée

3. Si "Utilisation de la ressource" est élevée
   et "Temps de résidence moyen" est élevé
   et "Longueur moyenne de la file d'attente" est élevée
   Alors vous avez de grande chance d'obtenir
   "Charge de travail" est bas

a.3. La performance des données :

Lorsque l'on parle de performance de données, la problématique se ramène toujours à savoir si l'on veut obtenir une bonne efficacité en terme d'*espace* ou de *temps*. L'espace fait référence à la quantité d'espace disque nécessaire au stockage des données et le temps fait référence au délai nécessaire afin d'effectuer un accès écriture ou lecture à la donnée.

Dans notre modélisation, il n'y aura lieu qu'à déterminer quel est le type de performance qui est désiré pour la donnée. Les étapes suivantes de la prise en charge de l'efficacité se faisant à travers le processus d'optimisation de la base de données qui doit faire l'objet dans des développements futurs d'une intégration dans le modèle.

a.4. La performance organisationnelle :

On se permet de rappeler ce qui a été dit dans l'introduction de cette modélisation des performances. A savoir que le fait de ne pas spécifier explicitement que l'on veut assurer une bonne performance organisationnelle ne veut pas dire que les modifications apportées ne seront pas faites avec rigueur. Mais préciser dans les exigences non-fonctionnelles qu'il y a lieu de garantir ce type de performance met le doigt sur le caractère ESSENTIEL que recèle ce besoin de performance afin d'assurer le bon aboutissement du projet. Les performances mentionnées ici en terme de besoin organisationnel implique donc au niveau de la gestion des procédures organisationnelles une attention toute particulière.

Cette performance se décompose en deux paramètres. D'une part la performance de la ***structure*** et d'autre part, la performance du paramètre ***humain***.

a.5. Répercussion des choix sur les autres niveaux du développement :

On a précédemment exprimé le fait que la rapidité d'exécution des applications devait être recherchée qu'elle fasse l'objet d'une demande explicite ou non. Mais l'expression d'une demande de performance va permettre de rechercher l'efficacité basée sur le comportement opérationnel qui devra caractériser le système informatique. Par conséquent la recherche de performance impliquera particulièrement le niveau de l'architecture informatique qui constitue le pilier de la performance du système.

Lorsque l'on regarde la répercussion sur les autres couches (figure 4.11), on observe que seul le temps de réponse moyen est un paramètre qui se répercute dans le niveau software. Ce paramètre est importante car il est inconcevable d'imposer des contraintes de performance au niveau de l'architecture informatique sans se soucier d'éventuelles contre-performances d'efficacité dans la couche logiciel.

Comme on l'a dit, la plupart des paramètres mesurent la performance de l'architecture informatique. Ceci montre bien toute la délicatesse du traitement de performance car ces paramètres sont ciblés sur le comportement en temps d'exécution des nouveaux processus ou sur la fréquence d'utilisation de ces derniers ce qui est une question d'estimation plus que de certitude.

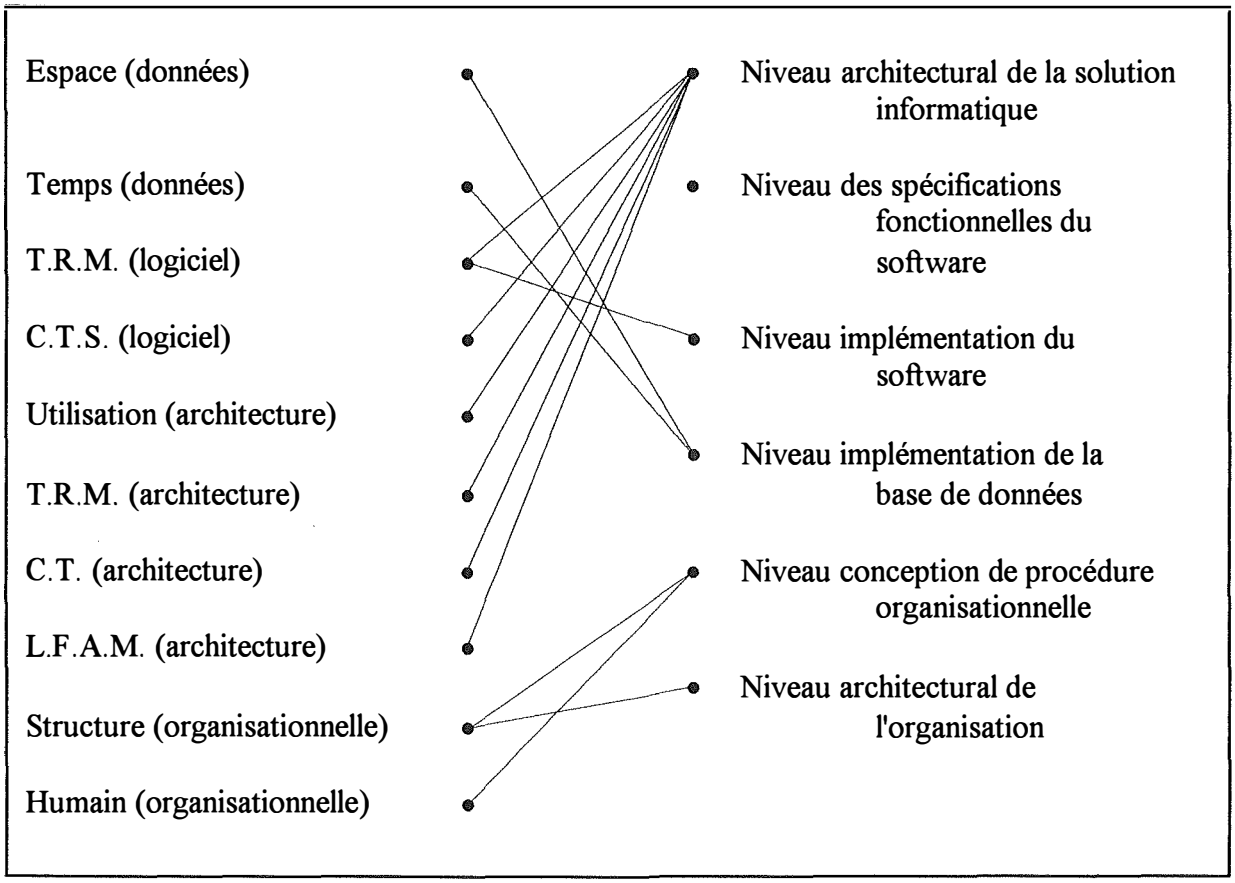


Figure 4.11. La répercussion des paramètres de performance.

b. La quantification de la performance

Maintenant que l'on a fait le choix parmi les paramètres de performance que l'on va considérer pour effectuer l'optimisation, il faudra encore la quantifier. Cette mesure est particulièrement importante et on a besoin de l'établir avec le maximum de discernement, de

sérieux et de méthode car la comparaison entre cette mesure et la performance réelle du système pourrait mener à la validation ou non du produit final.

b.1. La mesure de la performance du logiciel :

Cette étude de la mesure de la performance concernant des éléments logiciels est basée sur les travaux réalisés par A.L. Opdahl dans [21]. Il précise que lorsque l'on essaye de fixer une performance attendue concernant certains éléments du futur système informatique, on doit garder à l'esprit que la performance a un impact sur deux facteurs dont il est indispensable de se préoccuper. Ces deux impacts sont d'ordre *organisationnel* et *psychologique*.

• L'impact organisationnel de la performance : la performance du système a un impact sur la performance de l'organisation. Par conséquent, avoir un système performant est un des paramètres qui mènent à une organisation performante. Mais le problème reste le même on ne crée pas un système performant juste pour le plaisir, il faut assurer que les investissements consentis pour la performance du système informatique aient des effets suffisamment positifs sur l'organisation par rapport à ses coûts.

• L'impact psychologique de la performance : si les exigences de performance considérées n'ont pas d'impact direct sur la performance de l'organisation, alors nécessairement l'impact doit jouer sur la psychologie de l'utilisateur (et a par conséquent un impact indirect sur l'organisation). D'après A.L. Opdahl [21], la performance des utilisateurs ne s'accroît pas linéairement avec la performance du système informatique. Teal et Rudnicky [24], identifie même trois périodes de temps impliquant différents états pour les utilisateurs (figure 4.12).

Durant la période A, le temps de réponse du système est rapide et l'utilisateur n'a pas le temps de préparer toutes les informations qu'il va devoir utiliser par après. Par conséquent le système l'attend après avoir fini son traitement.

Quand le temps de réponse du système passe de la période A à la période B, l'utilisateur a maintenant le temps de préparer ses informations durant le temps de travail du système. Le

fait que la ligne ne rejoint pas la valeur 0 pour le temps d'attente de la part de l'utilisateur est simplement le fait que l'utilisateur utilise du temps pour encoder ses informations dans le système. Dans la période B, lorsque le temps de réponse du système s'allonge, le temps d'attente du système après traitement s'accroît mais plus que proportionnellement. La raison est imputable à l'utilisateur qui doit attendre le système et ne reste pas toujours attentif à cette tâche.

Finalement, dans la période C, le temps de réponse du système est tellement long que le système donne un signal lorsque le traitement suivant peut être introduit. Par conséquent, la période d'attente de l'utilisateur reste constante.

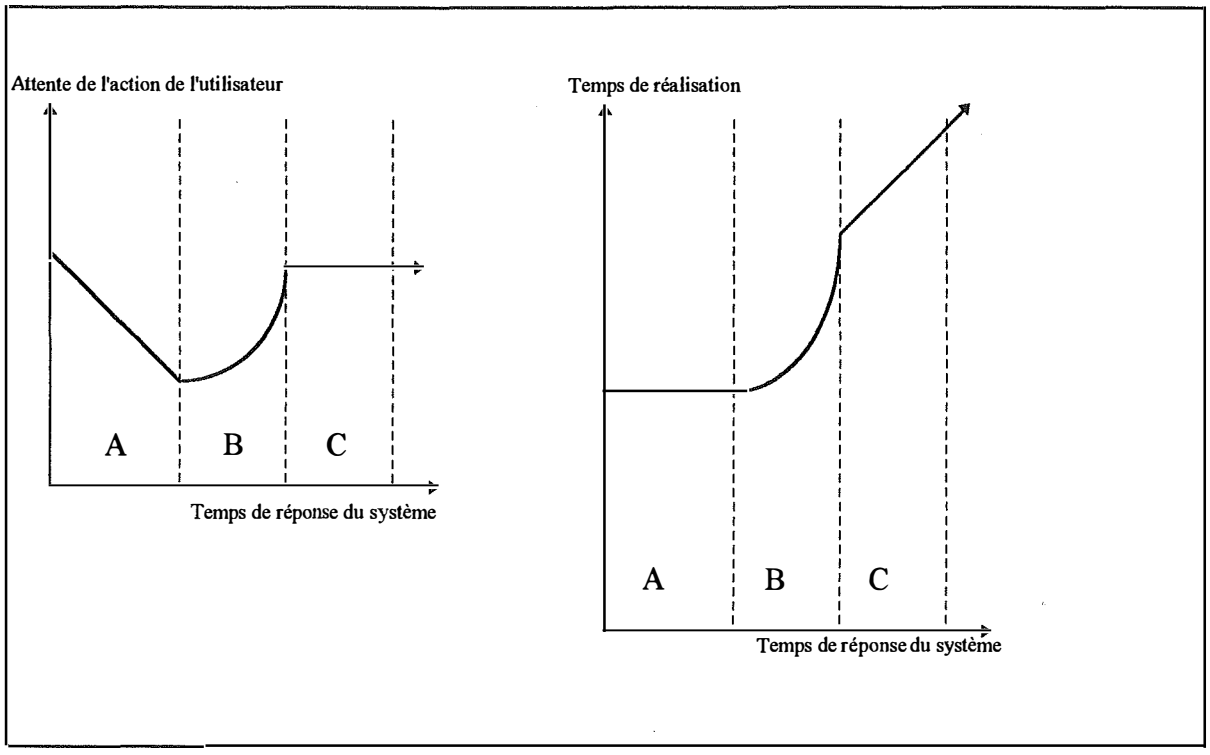


Figure 4.12. Les trois périodes de temps.

Si l'on dessine le second graphe dans lequel le temps de réalisation représente le temps total nécessaire pour l'exécution d'une action (attente d'action de la part de l'utilisateur + temps de réponse du système), on voit que dans la période A, le bon temps de réponse du système n'a pas d'impact sur la rapidité de la réalisation de l'action. Donc, si le processeur n'a pas la possibilité de réaliser une autre tâche durant le temps d'attente, il n'est pas nécessaire de réaliser

un tel effort d'investissement pour obtenir une si grande performance (en dehors de toutes autres considérations tel que la marge de performance pour les développements futurs).

C'est de ce type de graphe que l'on fixe une valeur pour la mesure de performance. Mais notons que si on considère la frontière entre la période B et la période C comme étant fixe, la frontière entre la période A et la période B dépend de l'action à réaliser. Il y a donc une réelle difficulté à produire un tel graphe parce que le temps nécessaire à l'utilisateur pour préparer ses informations et les encoder dépend du niveau intellectuel de l'action à réaliser, du nombre d'informations à manipuler pour cette action, de la clarté de la future interface disponible pour réaliser l'action, ...

Dès lors, tenant compte de ces deux impacts, comment s'y prendre pour fixer une valeur au paramètre sélectionné par la considération d'un type précis de fonction?

• Pour une fonction au terminal : on a retenu le temps de réponse moyen et la charge de travail du système pour cette fonction précise.

1. En ce qui concerne le paramètre de charge de travail du système, il s'agit d'établir de manière statistique, en supposant que l'exécution de la fonction est infinitésimale, le nombre de fois que cette fonction devrait être utilisée par unité de temps. Il faut pour cela analyser le comportement de l'élément de l'organisation qui est impliqué par cette fonction. Ainsi, si la fonction a pour but d'enregistrer une commande passée à une société de livraison à domicile, il s'agit de déterminer le nombre de commande que celle-ci traite en moyenne par unité de temps.

2. Pour le paramètre de temps de réponse moyen, étant donné que ce genre de fonction est utilisée régulièrement, il faut considérer l'impact de l'utilisation de cette fonction sur la psychologie de l'utilisateur. Ainsi, la mesure du paramètre sera fixée par la méthode des trois périodes expliquées ci-dessus.

• Pour une fonction batch : le paramètre retenu est la charge de travail du système. Il s'agit donc de nouveau de déterminer de manière statistique le nombre de fois que cette fonction est amenée à être utilisée par unité de temps.

• Pour une fonction de transition : c' est le temps de réponse moyen qui nous intéresse. Mais ce type de fonctions moins utilisées est susceptible d'avoir le moins d'impact sur la psychologie des utilisateurs. Si l'augmentation de l'exigence de performance accroît le coût de développement, cette augmentation pourra uniquement se justifier par un plus grand retour de bénéfice au niveau de l'organisation. Il s'agit de développer une analyse de coût-efficacité visant à déterminer quel est le temps de réponse moyen qui maximise ce coût-efficacité.

b.2. La mesure de performance de l'architecture :

Etablir une mesure de performance des ressources de l'architecture est extrêmement difficile car il s'agit de regarder ce qui est destiné à utiliser l'architecture (processus et données) afin de déterminer la performance que devront assumer certaines ressources pour ne pas constituer un goulot d'étranglement.

La mesure doit être effectuée tôt dans le développement du système, car si le développement logiciel implique des performances à assurer du point de vue architectural, le niveau architectural peut également impliquer des restrictions au niveau logiciel. Et il n'est pas souhaitable que les modifications interviennent trop tard dan le développement (pour cause de délai et de difficulté). Ceci sera le cas si la mesure de cette performance souhaitée se présente irréalisable lors du choix d'une architecture ou indésirable lors de la validation du paramètre de performance par le graphe coût-efficacité. Cela peut entre autre aboutir à une redéfinition des interfaces graphiques utilisateurs jugées trop coûteuses en ressources.

Dès lors, puisque la mesure des paramètres se base sur la taille des processus, leur fréquence d'utilisation, leur performance en temps d'exécution, sur la taille des informations à manipuler, leur fréquence d'utilisation, cette mesure ne peut être qu'une évaluation approximative qui doit faire l'objet d'un suivi tout au long du développement.

Chapitre 5

Etude de Cas

Nous allons maintenant appliquer la technique de raffinement des contraintes non-fonctionnelles de sécurité et de performance à travers un cas de développement d'un projet informatique. Celui-ci a spécialement été choisi non pour son intérêt particulier mais pour sa capacité à mettre en lumière les principes qui ont été évoqués jusqu'ici.

Dans ce chapitre, nous ne développerons que l'énoncé du cas ainsi que l'étape de raffinement des exigences non-fonctionnelles. L'étude complète se trouve en annexe. Elle comporte une spécification du système exprimée dans le langage ALBERT, suivie du raffinement et finalement de la modification des spécifications afin de prendre en compte les solutions opérationnelles retenues pour les exigences. **On recommande vivement de lire l'étude complète de l'annexe car les buts de raffinement de la méthodologie sont définis sur des paramètres qui sont développés dans la spécification du système (en accord avec la méthodologie).**

Le projet concerne le développement d'un système de pointeuse informatisée impliquant un besoin de contrôle d'accès. Nous allons tout d'abord présenter le contexte dans lequel le projet prend naissance. Nous présenterons dans un deuxième point les exigences liées au projet telles qu'elles pourraient être émises par les futurs utilisateurs (expression informelle). Nous détaillerons par après la spécification formelle du niveau software. Un quatrième point verra le raffinement des contraintes non-fonctionnelles de sécurité et de performance. Finalement nous détaillerons les répercussions de ses contraintes non-fonctionnelles sur les autres niveaux du développement.

Cette étude de cas est basée sur le système d'horaire flottant tel qu'il a été défini au sein de la COBAC S.A., société d'assurance crédit de Belgique. Toutefois, ce cas s'en inspire mais les décisions de conception détaillées dans ce chapitre ne reflètent en rien les dispositions prisent au sein de cette entreprise. De même, le contexte est ici purement fictif.

5.1. Présentation de l'étude de cas

Une organisation fonctionnait jusqu'il y a peu avec un système de pointeuse mécanique assisté par un préposé du service du personnel vérifiant l'utilisation de la pointeuse ainsi qu'effectuant les fonctions de contrôle des heures prestées.

Suite au départ en pension du préposé et suite à la recherche croissante de diminution du coût de structure organisationnelle, l'entreprise acheta un système de pointeuse informatisée qui se révéla fort peu efficace dans son adaptation à notre situation. Notamment par son manque de clarté à l'utilisation. Un exemple, la principale différence entre notre entreprise et celle nous ayant vendu le système venait du fait de la présence dans notre société de personnes pouvant à tout moment partir à l'extérieur afin d'effectuer une mission. Ceci n'étant pas prévu à l'origine, il fut décidé de rajouter des fonctionnalités au système. Ces fonctionnalités furent implémentées de manière tout à fait maladroite empêchant toute utilisation du système de manière intuitive.

La situation actuelle est donc caractérisée par un rejet unanime de la part du personnel. Conscient des nombreux manquements du système, il a donc été décidé de reprendre les choses depuis le début afin d'implémenter un nouveau système satisfaisant pour tous.

Nous ne nous intéresserons pas d'avantage à la solution choisie en premier lieu. Du moins, il est primordial de savoir que :

- le système privilégiait un accès par carte clé;
- l'entrée au bâtiment nécessitait l'introduction d'un code secret en plus de l'utilisation de la carte clé;
- certaines zones du bâtiment étaient d'accès protégé. Cette protection était assurée par un lecteur identique à ceux des entrées du bâtiment.

Toutefois, comme une authentification était assurée à l'entrée du bâtiment, il suffisait vis-à-vis de ces zones protégées de présenter une carte de personne ayant accès pour être libre du passage (simple identification).

Finalement, afin d'avoir une idée précise de la situation, un plan du rez-de-chaussée et du premier étage sont présentés à la figure 5.1 et 5.2.

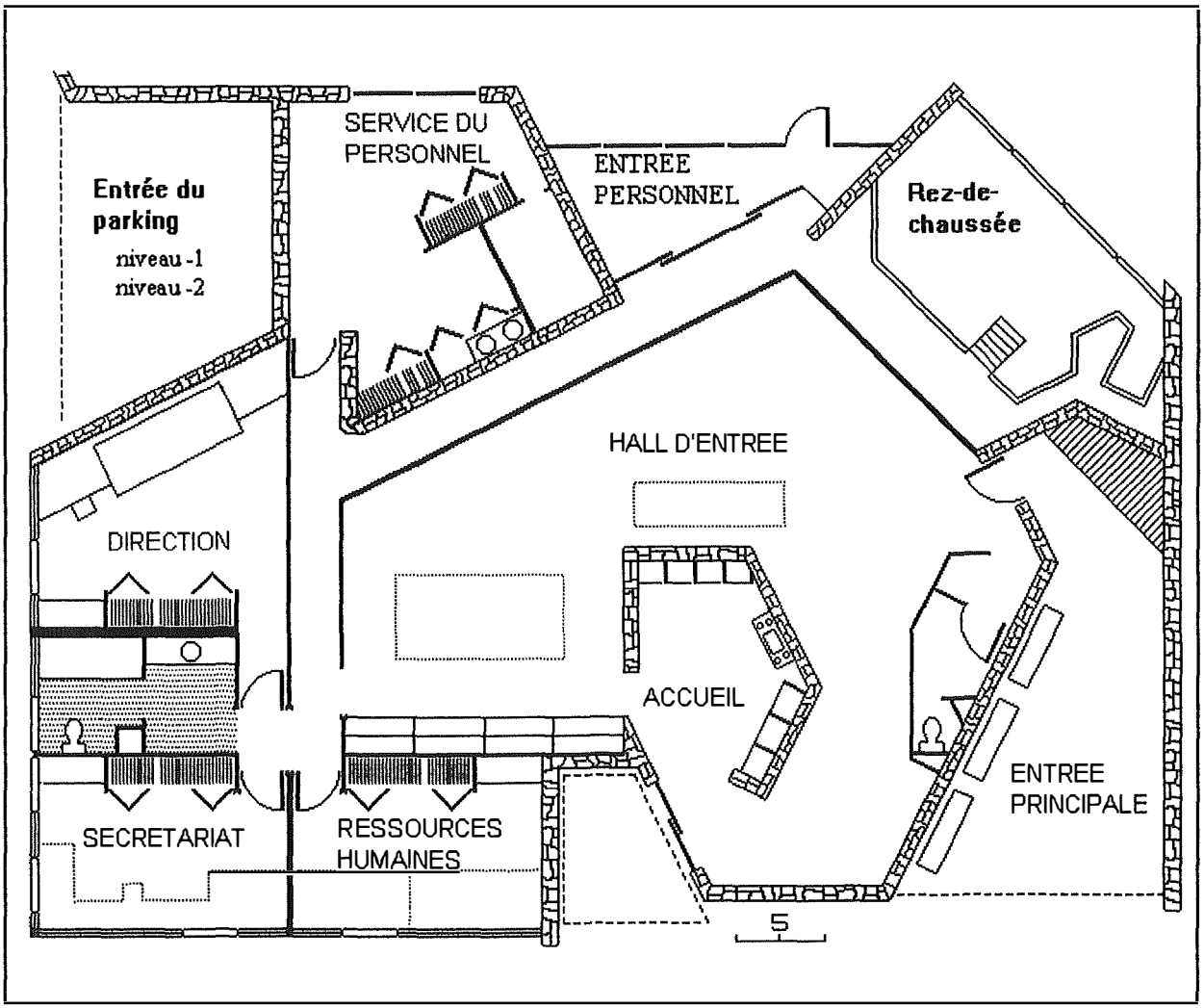


Figure 5.1. Le plan du rez-de-chaussée.

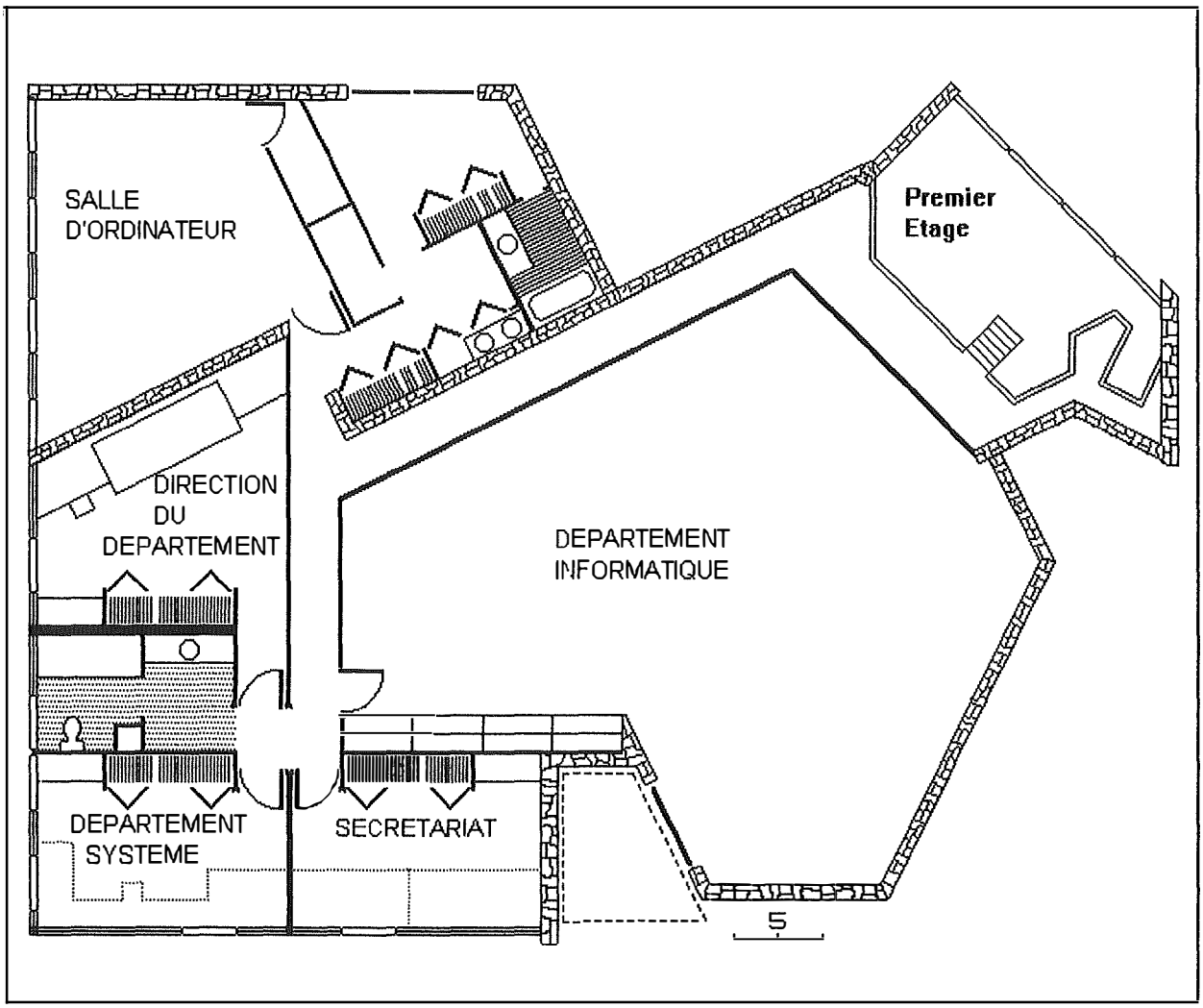


Figure 5.2. Le plan du premier étage.

5.2. Les exigences informelles

Parmi les exigences fonctionnelles visant à établir un système de poiteuse à horaire variable, il faudra prendre en considération le cadre légale définissant le temps de travail ainsi que tous les engagements qui ont été définies dans la convention collective de travail entre les employeurs et les représentants des travailleurs. Une énumération de tous les éléments à respecter constitue une bonne base avant de voir plus précisément les exigences que devront repecter les fonctionnalités du système. Les fonctionnalités émises par le client seront dès lors détaillées en intégrant les contraintes d'ordre légales. Les contraintes non-fonctionnelles que le client désire voir réalisées dans le projet seront ensuite détaillées. Finalement, toutes les exigences en matière de restructuration organisationnelle seront exprimées.

a. Le fonctionnement de l'horaire variable

Dans la définition des dispositions concernant le fonctionnement de l'horaire variable devant faire l'objet d'une informatisation, la présentation est dans ce chapitre découpée en identification du *cadre légal* à respecter, suivi des *règles de fonctionnement* concernant les plages horaires de travail, la manière de *comptabiliser les heures prestées* et finalement des dispositions concernant les *absences, congés et récupérations*.

a.1. Le cadre légal :

De manière générale, la loi prévoit que :

- la journée de travail ne dépasse pas 9 heures,
- la semaine de travail ne dépasse pas 40 heures,
- on ne travaille pas le dimanche,
- on ne travail pas la nuit.

Des dérogations sont possibles dans certains cas :

- sans formalités spéciales (travaux de clôture financière, ...)
- moyennant l'accord de la Délégation Syndicale (heures supplémentaires, ...)
- moyennant une convention de secteur ou d'entreprise (flexibilité du temps de travail, ...)
- moyennant l'accord du ministère du travail (travail de nuit, ...)

a.2. Les règles de fonctionnement :

Heures d'arrivée : Les membres du personnel arrivent entre 8h00' et 9h00' à l'heure de leur choix.

Heures de départ : Du lundi au jeudi, les membres du personnel partent entre 16h00' et 18h00' à l'heure de leur choix. Concernant le vendredi, les membres du personnel partent entre 15h00' et 17h00' à l'heure de leur choix.

Plages fixes : Tout le monde doit être présent :

- entre 9h00' et 12h00'
- entre 14h00' et 16h00' du lundi au jeudi
- entre 14h00' et 15h00' le vendredi

Pause de midi : Entre 12h00' et 14h00', le personnel peut disposer de ce temps pour son déjeuner et son délassement. La pause ne peut pas être inférieure à 45 minutes.

Un résumé de toutes ces dispositions est présenté à la figure 5.3.

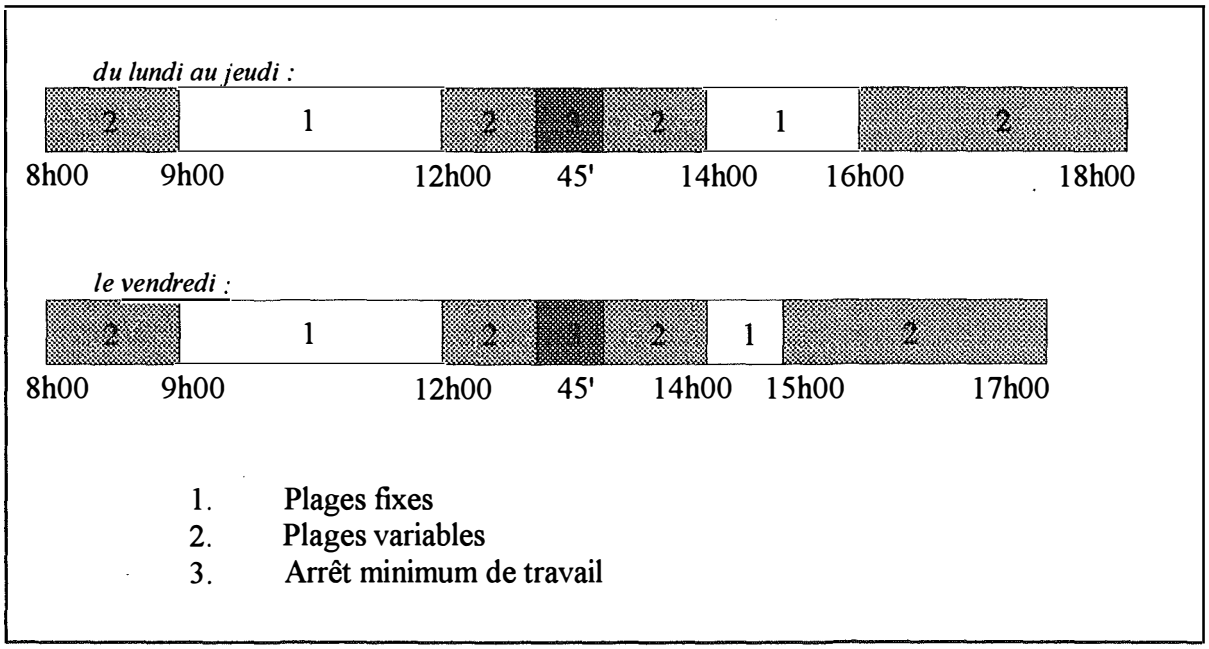


Figure 5.3. Réglementation de l'horaire variable.

Régime spécial : Certaines personnes font l'objet d'un régime spécial du fait de la nécessité d'effectuer des missions à l'extérieur pour le compte de la société. Ainsi, si un employé doit sortir régulièrement de la Société pendant les heures de travail, mais pour le compte de celle-ci, le Service du Personnel donnera un accès mission à la demande du responsable hiérarchique. Différentes situations peuvent se présenter :

- mission en cours de journée mais avec présence pour la plage de repos à midi,
- la journée commence par une mission et celle-ci s'étend sur toute ou plusieurs journées et le retour se fait en début d'une journée.

Quand un membre du personnel est en mission, le temps de travail est calculé dans la limite de l'horaire, c'est-à-dire de 8h00' à 18h00' sauf le vendredi de 8h00' à 17h00'.

a.3. La comptabilisation des heures prestées :

A la fin de chaque mois, plusieurs situations peuvent se présenter :

- le nombre d'heures prestées coïncide avec le nombre d'heures comprises dans la période de référence,
- le nombre d'heures prestées est supérieur au nombre d'heures comprises dans la période de référence. Dans ce cas vous disposez d'un crédit d'heures, que vous pouvez récupérer,
- le nombre d'heures prestées est inférieur au nombre d'heures comprises dans la période de référence, dans ce cas vous disposez d'un débit d'heures, que vous devez prester.

Des limites doivent toutefois être mentionnées :

- le crédit d'heures ne peut dépasser 5 heures en fin de mois. Sauf accord spécial de la hiérarchie, le supplément de crédit sera perdu. Les employé(e)s pourront récupérer leur crédit, une demi-journée par mois. La date est à fixer en accord avec le responsable hiérarchique.
- le débit d'heures ne peut être supérieur à 5 heures en fin de mois. Le cas échéant, il devra être résorbé le mois suivant. Un débit d'heures de moins de 5 heures, ne peut être admis plus de deux mois consécutifs. Pour résorber un débit, il faut accroître les prestations journalières dans les plages variables. Tout manquement constitue une faute.

a.4. Absences - Congés - Récupérations :

Toutes absences doivent être signalées immédiatement au Service du Personnel, qui en avertira le responsable. Celui-ci fera parvenir dans les plus brefs délais une fiche d'absence, reprenant le motif et la durée probable de l'absence. Pendant l'absence, le compte est alimenté manuellement par le Service du Personnel à raison d'une durée journalière normale (7h38' du lundi au jeudi et 5h58' le vendredi).

Toutes demandes de congé doivent parvenir au Service du Personnel trois jours avant le début du congé. La procédure est la même que pour les absences.

Toutes récupérations doivent être signalées au Service du Personnel par le responsable hiérarchique au moyen du formulaire adéquat et dans les plus brefs délais.

b. Les fonctionnalités attendues du système

b.1. Enregistrement arrivée normale

Objectif : L'objectif de cette fonctionnalité consiste à recevoir l'information d'arrivée d'un membre du personnel ne revenant pas de mission.

Message d'entrée : { matricule, 'IN' }

Message de sortie : { temps cumulé global }

b.2. Enregistrement arrivée mission

Objectif : L'objectif de cette fonctionnalité consiste à recevoir l'information d'arrivée d'un membre du personnel revenant de mission et de mettre à jour le champ "est en mission" de la base de donnée.

Message d'entrée : { matricule, 'IN-miss' }

Message de sortie : { temps cumulé global }

b.3. Enregistrement sortie normale

Objectif : L'objectif de cette fonctionnalité consiste à recevoir l'information de sortie d'un membre du personnel ne partant pas en mission.

Message d'entrée : { matricule, 'OUT' }

Message de sortie : { temps cumulé global }

b.4. Enregistrement sortie mission

Objectif : L'objectif de cette fonctionnalité consiste à recevoir l'information d'arrivée d'un membre du personnel revenant de mission et de mettre à jour le champ "est en mission" de la base de donnée.

Message d'entrée : { matricule, 'OUT-miss' }

Message de sortie : { temps cumulé global }

b.5. Enregistrement manuel de prestation

Objectif : Etablir une fonctionnalité dont dispose le Service du Personnel afin de mettre manuellement à jour le temps cumulé.

Message d'entrée : { matricule, décompte temps }

b.6. Cumul des heures prestées

Objectif : Etablir une fonctionnalité qui pour les matricules n'ayant pas été inscrit dans la base de données comme absent ou en congé décompte le nombre de temps presté durant la journée, la soustrait au temps normal à prester en ce jour (temps qui indique la parfaite répartition du temps à prester pour les jours d'un mois afin d'arriver au décompte d'heures devant être effectuées) et en crédite ou débite, cela dépend du solde, le temps cumulé global. Il s'agit également de mettre à jour, pour ces membres du personnel, les champs "Temps cumulé mois antérieur" et "Temps total hebdomadaire" de la base de données. Ayant calculé le total presté par chaque membre, il faudra introduire le matricule des membres du personnel n'ayant pas respecté la durée de pause de midi ou la durée maximale de travail d'une journée respectivement dans "Cal-pause" et "Cal-cumul".

b.7. Réduction cumul mensuel

Objectif : Cette fonctionnalité a pour but de ramener le temps cumulé à 5h00' pour tous les membres du personnel ayant un solde établit supérieurement à ces 5h00'.

b.8. Contrôle de légalité journée

Objectif : Le but est d'analyser pour les membres du personnel ceux qui n'ont pas respecté les réglementations en matière d'horaire journalier :

- entrée / sortie effectuées dans les plages fixes,
- entrée avant 8h00'
- sortie après 18h00' du lundi au jeudi et après 17h00' le vendredi,
- cumul de plus de 9h00',
- plage de midi plus grande que 45',
- absence non justifiée.

afin d'en établir la liste accompagniée de la justification.

Message de sortie : { matricule, respect plage fixe, entrée trop tôt, entrée trop tard, respect midi, cumul trop grand }

b.9. Contrôle de légalité semaine

Objectif : Le but est d'analyser pour les membres du personnel ceux qui n'ont pas respecté les réglementations en matière d'horaire hebdomadaire :

- cumul hebdomadaire de plus de 40 heures.

Message de sortie : { matricule, temps cumulé hebdomadaire }

b.10. Gestion déficience mensuelle

Objectif : Le but est d'analyser pour les membres du personnel ceux qui n'ont pas respecté les réglementations en matière de cumul de prestation mensuelle :

- cumul négatif global de plus de 5 heures,
- cumul négatif global consécutif de plus de 5 heures.

Message de sortie : { matricule, temps cumulé global, consécutif }

b.11. Gestion des absences - Congés - Récupération

Objectif : Le but consiste à introduire dans la base de données les demandes de congés, de récupérations ainsi que les justificatifs d'absences. Ceci sera utile notamment pour le calcul du cumul horaire afin de ne pas débiter le temps aux personnes dont l'absence est justifiée.

Message d'entrée : { matricule, motif, dates }

b.12. Traitement des absences non justifiées

Objectif : Le but est de retirer le temps normal à prester en ce jour du temps cumulé global pour l'ensemble des membres du personnel faisant l'objet d'une absence non justifiée. (Toute justification postérieure à la date d'absence injustifiée fera l'objet d'une rectification de temps cumulé global manuel grâce à la fonctionnalité Enregistrement manuel de prestation). Il s'agit également de mettre à jour, pour ces membres du personnel, les champs "Temps cumulé mois antérieur" et "Temps total hebdomadaire" de la base de données.

b.13. Traitement des congés et absences justifiées

Objectif : Etablir une fonctionnalité qui pour les matricules ayant été inscrit dans la base de données comme absent justifié ou en congé met à jour les champs "Temps cumulé mois antérieur" et "Temps total hebdomadaire" de la base de données en considérant le temps normal à prester en ce jour comme référence pour le crédit d'heure.

c. Les contraintes non-fonctionnelles liées au système

c.1. Le contrôle d'accès - sécurité :

Un système d'accès centralisé :

Les zones qui doivent faire l'objet d'un filtrage d'accès sont :

- les entrées du garage (niveau -1 et niveau -2),
- la sortie par le hall.
- l'entrée du personnel

Il faut en outre que les enregistrements d'horaire fassent l'objet d'une protection afin d'assurer le bon fonctionnement de la fonction de contrôle de l'horaire. De plus, le système pour lequel on optera ne devra pas, pour des raisons de sécurité physique (et légale), être une entrave à l'évacuation des locaux en cas de problèmes.

c.2. La performance requise :

Il s'agira d'assurer que le système soit suffisamment efficace que pour éviter toute formation de file d'attente au dispositif de pointage. Le système devra également permettre une bonne performance d'évacuation du batîment.

d. Les exigences organisationnelles

d.1. Accès exceptionnel :

Pour obtenir un accès en dehors des plages normales, la demande signée par le responsable de département devra parvenir, 48 heures à l'avance, au responsable de la sécurité. Celui-ci comme au paravant assurera la sécurité de nuit et verrouillera toutes les portes manuellement 30 minutes après la fin des heures de bureau. Toute personne bloqué après cette heure devra se justifier auprès du responsable de la sécurité qui la raccompagnera à la sortie. Le membre du personnel ne devra toutefois pas omettre de pointer sa sortie.

5.3. Le raffinement des contraintes non-fonctionnelles de sécurité et de performance.

Lors de l'acquisition des exigences, on a pu mettre à jour cinq besoins non-fonctionnels à prendre en compte. Nous allons maintenant raffiner ces besoins afin d'aboutir à un ensemble de solutions à intégrer dans les spécifications du système.

Les besoins non-fonctionnels établis étaient les suivants :

1. Assurer la sécurité du personnel. Il s'agit en fait d'assurer pour des raisons de sécurité physique et légale que le système ne sera pas une entrave à l'évacuation des locaux.
2. Assurer la performance d'évacuation. Le système doit permettre une bonne performance de l'évacuation des locaux.
3. Assurer la performance d'accès. Il faut que l'accès soit rapide pour éviter la formation de files d'attente devant les pointeuses.
4. Assurer la protection des pointages. Les enregistrements des horaires doivent être protégés afin d'assurer le bon fonctionnement de la fonction de contrôle de l'horaire .
5. Assurer le filtrage d'accès. Comme il avait été précisé dans les exigences informelles, il faut assurer une fonction de filtrage à l'entrée du bâtiment.

Par nécessité de simplicité, le problème de sélection d'alternative et de validation de solution ne sera pas développé ici.

a. Sécurité du personnel :

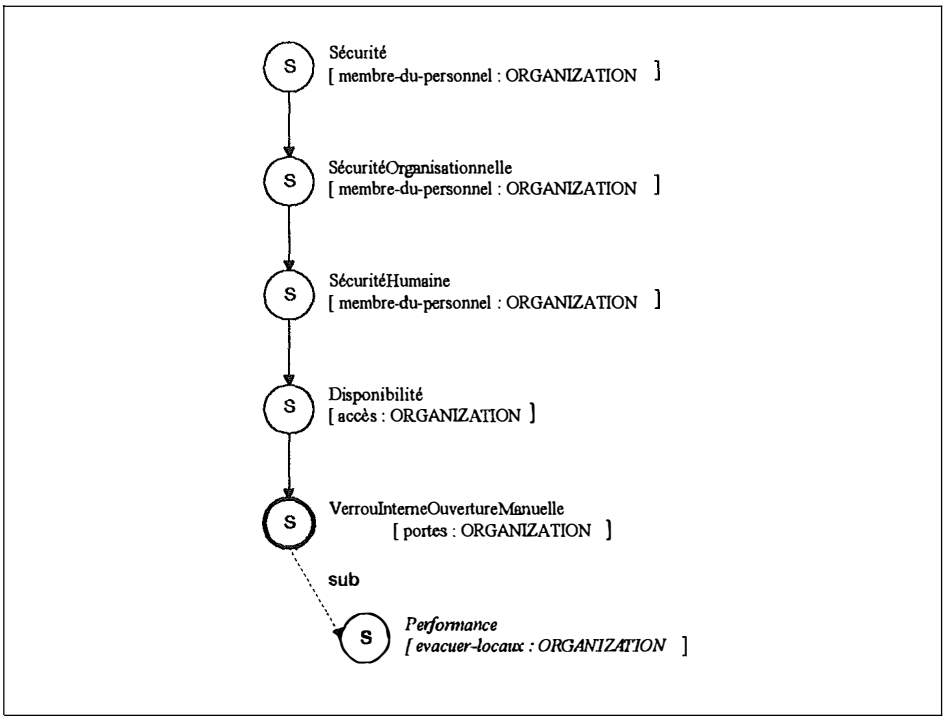


Figure 5.4. La sécurité du personnel.

Partant du besoin de fournir une solution au besoin de sécurité pour les membres du personnel modélisé par "Sécurité du personnel [membre du personnel : ORGANIZATION]" dans la figure 5.4, on a appliqué la taxonomie définie dans le chapitre 4. On a pu dès lors sélectionner la bonne décomposition jusqu'au but non-fonctionnel de SécuritéHumaine. Comme il s'agit d'assurer que le personnel puisse évacuer le bâtiment à tout moment, cela se concrétise par le raffinement de SécuritéHumaine en Disponibilité du moyen d'accès. Cela implique que l'accès ne doit pas être une entrave pour l'évacuation du personnel.

Afin de simplifier le cas, nous n'avons retenu qu'un but de satisfaisabilité. Il s'agit de poser un verrou manuel pour néanmoins autoriser la sortie si le système est en panne. On remarque également que le fait de choisir cette solution apporte une contribution nécessaire à la satisfaisabilité du besoin de performance dans l'évacuation des locaux.

b. Performance de l'évacuation :

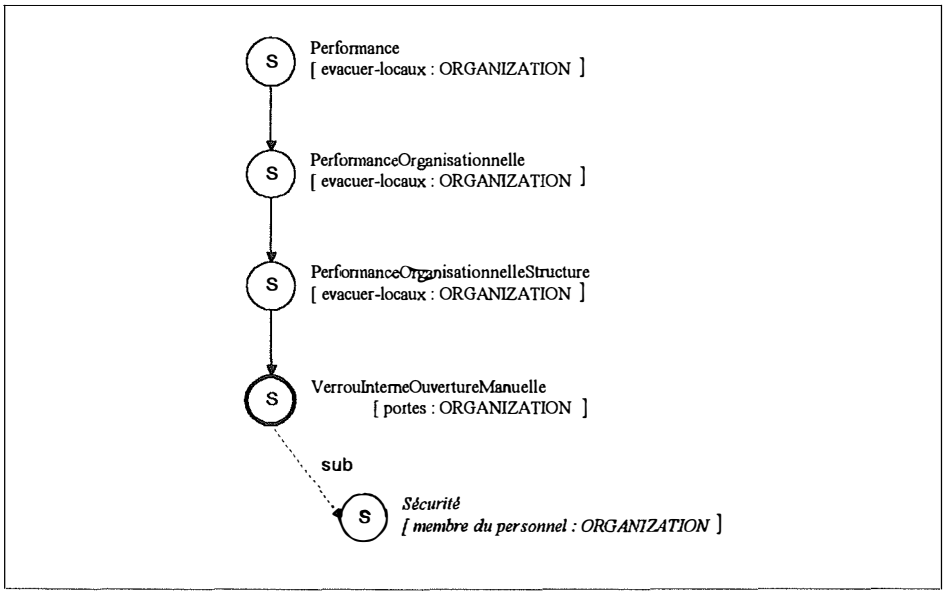


Figure 5.5. Performance de l'évacuation.

Le système doit permettre une bonne performance de l'évacuation des locaux. On remarque que cette performance touche à la structure organisationnelle. La solution identifiée est la même que pour le point précédant. On identifie aussi un lien de "contribution nécessaire" qui lie la solution de performance à l'exigence de sécurité des membres du personnel.

Ceci nous amène à faire deux remarques. Deux contraintes non-fonctionnelles différentes peuvent amenées à une même solution potentielle à adapter sur le système. Mais encore, on voit que performance et sécurité peuvent signifier la même chose, ou du moins poursuivre un même objectif dans certains cas. Ici, la sécurité du personnel quant aux possibilités d'évacuation passe par la performance dans le délai d'évacuation et la performance d'évacuation implique dans cette solution ci une sécurité pour le personnel.

c. Performance d'accès :

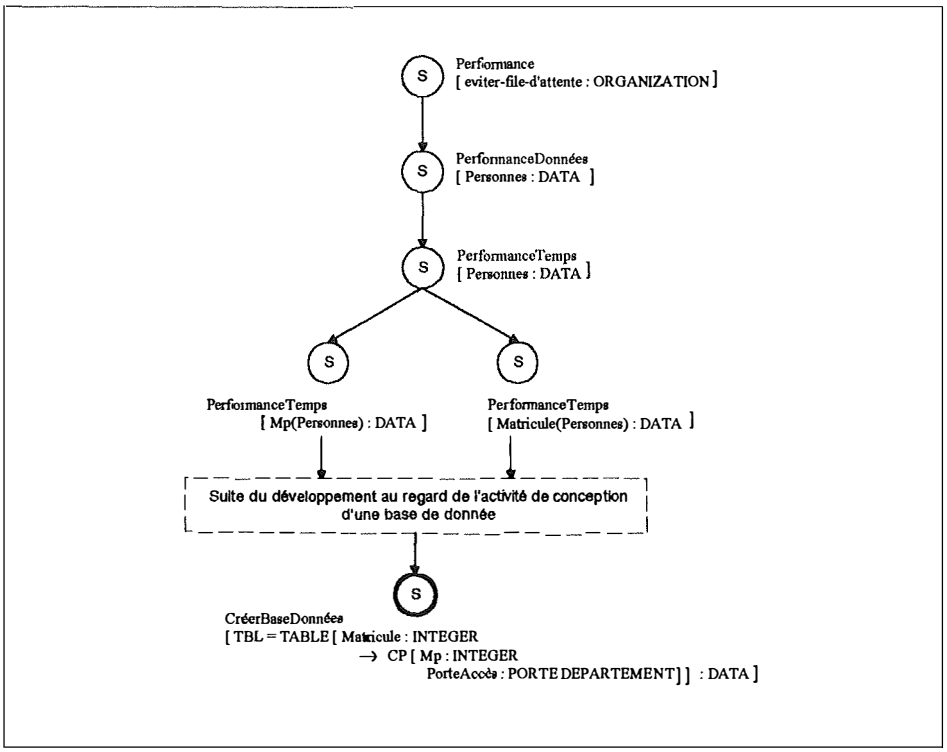


Figure 5.6. La performance d'accès.

Il faut assurer qu'il n'y aura pas de formation de files d'attente devant les pointeuses. On a orienté le raffinement vers le développement d'une solution liée à l'efficacité en temps pour les pointeuses de départements car elles doivent recevoir un mot de passe ce qui allonge le traitement. Une solution serait de créer une base de données à part de tous les enregistrements de l'horaire flottant pour en optimiser l'accès puisqu'il n'y a pas de mouvement à effectuer. (De nouveau cette solution ne serait pas suffisante dans un cas réel)

Remarquons que l'identification du paramètre Mp de Personnes vient du raffinement du besoin de filtrage d'accès (point e) dans lequel il a été décidé de recourir à une technique de mot de passe. Ceci montre bien que toute cette activité ne peut pas être perçue comme séquentielle D'autant plus que le raffinement de la performance d'accès influencera par lien de corrélation une décision dans le raffinement du filtrage.

d. Protection des pointages :

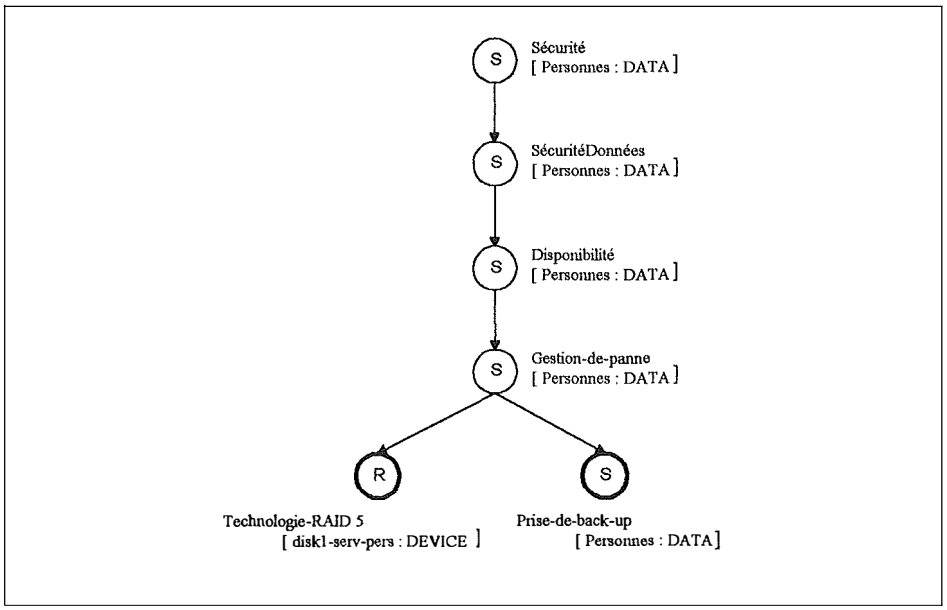


Figure 5.7. La protection des pointages.

On a identifié la sécurité de la base de données comme besoin pour assurance le bon fonctionnement de la fonction de contrôle de l'horaire. Cette exigence peut donc se décomposer en besoin de disponibilité de la base de données. Et principalement, il faut couvrir celle-ci par un mécanisme de gestion de pannes, ce qui peut être garanti par la prise de back-up ou par le recours à l'emploi de la technologie RAID. Dans ce cas précis, on se trouve devant deux alternatives. L'analyse de coût-efficacité devient nécessaire pour mettre à jour la solution la mieux adaptée à la situation.

Si l'on regarde l'allure générale du graphe de manière intuitive (figure 5.8), on observe que la perte en cas de panne ne devrait pas être trop élevée alors que les solutions apportées peuvent coûter très cher. En ce sens, si l'on prend la technologie RAID comme moyen d'assurer la disponibilité de la base de donnée, le coût sera (toujours de manière intuitive) trop important par rapport au gain fourni (la disponibilité de l'horaire flottant n'est pas quelque chose de vital pour une entreprise). Par conséquent, si l'analyse était faite en toute rigueur, la solution adoptée pencherait vers le choix du back-up. Solution que nous allons retenir.

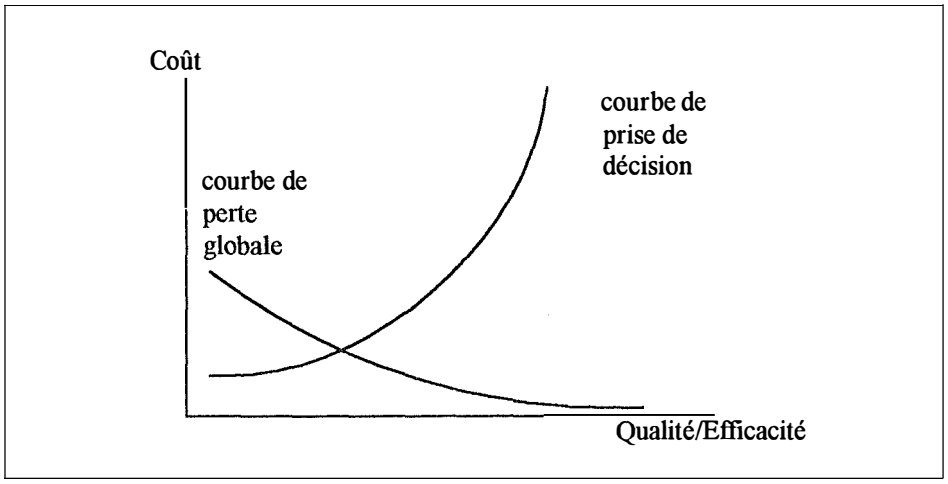


Figure 5.8. Graphe de coût/efficacité.

On remarquera que la sécurité demandée pour un élément de donnée peut être assuré par une solution apportée au niveau de l'architecture informatique. Ainsi, ce n'est pas parce que l'on recherche une couverture pour un besoin non-fonctionnel de type bien particulier que cette couverture impliquera seulement les éléments de ce type là.

e. Filtrage d'accès :

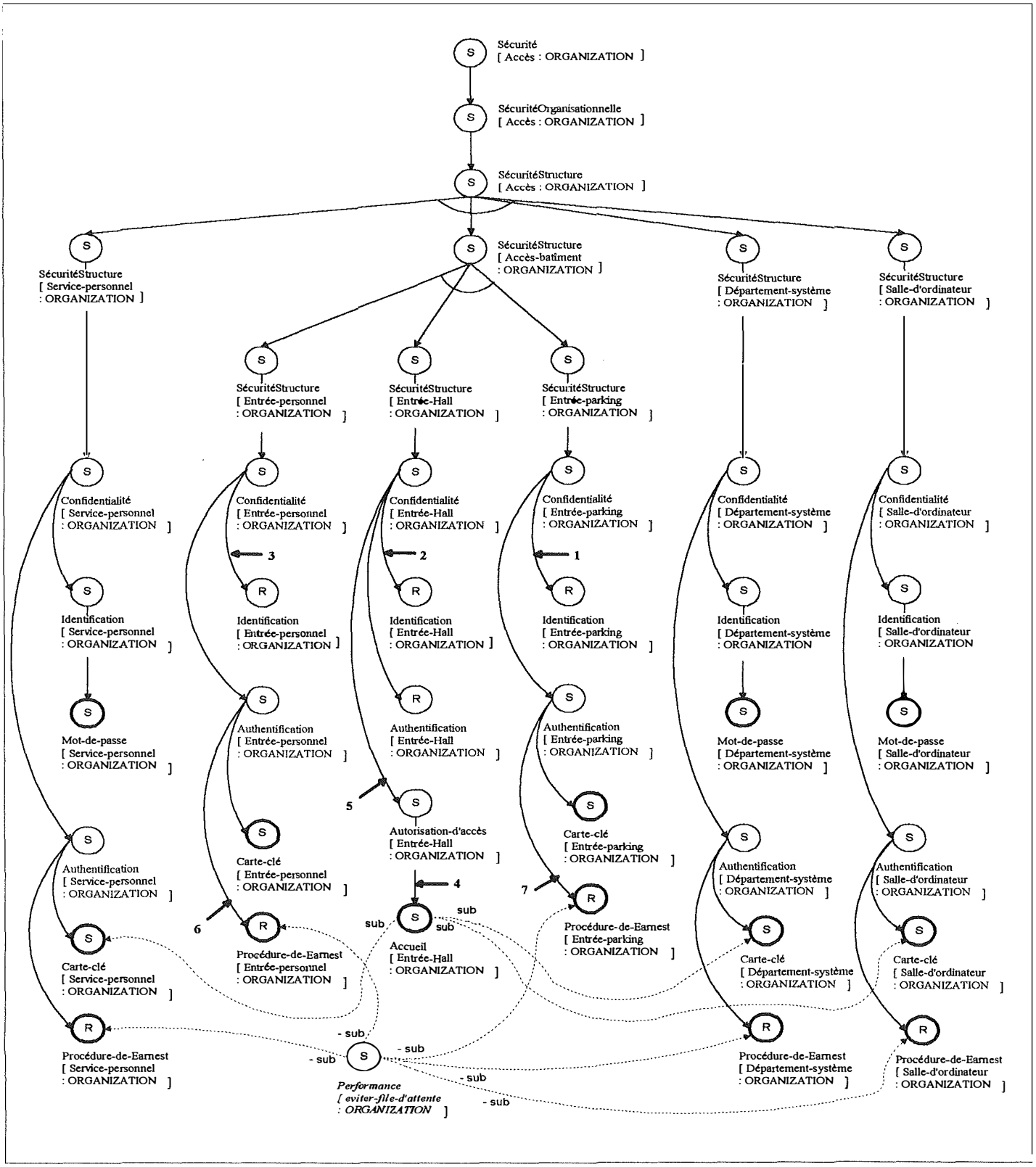


Figure 5.9. Le filtrage d'accès.

La maintenance évolutive ou adaptative ainsi que la définition d'un nouveau projet peut avoir pour conséquence de modifier les circonstances qui ont été à la base de prise de décisions de conception lors du développement d'autres projets. Ainsi, dans notre cas, la protection de départements avait été établie dans le précédant projet de gestion de l'horaire flottant. Puisque nous modifions les éléments qui avaient été à la base de la décision de l'époque, il nous faut reconsidérer la protection de ses départements dans le projet actuel. En effet, toute la protection de ses départements reposait sur le fait qu'un contrôle d'authentification par mot de passe était effectué à l'entrée du bâtiment. Si nous en arrivons à modifier cette état des faits, la protection perdra toute sa consistance.

Un autre élément important est fourni par les contraintes (voir chapitre 4, page 4.6) qui influencent le traitement des exigences de sécurité. Il s'agit de la motivation du personnel, qui ayant subi un système inadapté n'est plus apte à fournir des efforts dont il ne cible pas bien l'intérêt. Seule les membres du personnel des départements à protéger sont déclarées prêts à accepter une solution plus lourde vu qu'ils sont impliqués par le besoin de sécurité de leur département. Cette constatation permet de définir les buts d'argumentation 6, 7 déclarant que le personnel ne veut pas assumer une solution lourde. Rejetant par-là même la proposition d'établir un système de mot de passe selon la technique de Earnest.

Nous pouvons considérer également que les techniques de sécurité basées sur la simple identification sont jugées insuffisantes. Ce qui est exprimé par les buts d'argumentation 1, 2, 3. Néanmoins, pour ce qui concerne la sécurité du hall d'entrée, on estime que les personnes qui utilisent cette entrée sont externes et ne font pas l'objet d'un accès courant. Par conséquent, l'accès par une méthode d'autorisation d'accès serait plus adaptée. Cette méthode passe par une utilisation d'un service d'accueil. Ce raffinement est de plus renforcé par le fait qu'un accueil existe déjà dans la situation actuelle de la société. Cela est exprimé par le but d'argumentation 4.

On trouve également un lien de corrélation -sub liant la recherche d'une performance visant à éviter la constitution d'une file d'attente à l'utilisation de la procédure d'Earnest jugée

trop coûteuse. Et finalement le choix d'un accueil pour le hall d'entrée impliquerait la nécessité d'utiliser une carte clé afin de garantir la sécurité des départements.

La procédure d'étiquetage retient de tout cela une solution possible impliquant :

- un accès à toutes les entrées (bâtiment, départements) par carte clé
- un mot de passe pour les départements
- l'utilisation d'un service d'accueil dans le hall.

Chapitre 6

Conclusion

6.1. Résumé et contribution

L'objectif de cette thèse consistait à présenter une méthodologie permettant de prendre en considération les contraintes non-fonctionnelles et de pouvoir mettre en pratique ceci par le développement d'une illustration mettant en jeu tous les principes établis.

a. La méthodologie développée

Partant d'une exigence abstraite, mal définie et donc finalement non-opérationnelle (au sens défini par KAOS), il s'agissait de mette au point une méthode permettant de traiter les exigences en les décomposant et en les enrichissant afin d'aboutir à une solution (décision de conception) qui permet leur prise en compte dans une spécification. Ce processus les rend en effet exprimables en terme d'actions à exécuter et d'objets impliqués (au sens de KAOS) du système spécifié.

Cet exposé avait pour but de montrer la manière dont pouvait être abordé le problème des exigences non-fonctionnelles de sécurité et de performance. Ceci se fait tout d'abord par l'utilisation d'un modèle de raffinement des exigences et ensuite par l'élaboration d'une taxonomie des types d'exigences. Le modèle de raffinement constituant la base de la méthode en fournissant toute la syntaxe nécessaire à la formalisation du traitement. La taxonomie permet quant à elle de présenter le domaine d'expertise propre au type d'exigence manipulé. Le but de cette taxonomie est de pouvoir fournir une assistance à l'analyste dans son travail de raffinement. Cette assistance est absolument nécessaire au niveau des exigences non-fonctionnelles si l'on veut prendre une décision d'opérationalisation les concernant en toute connaissance de cause. On explique cela par le fait que le champ des techniques disponibles

s'accroît avec l'évolution des technologies et la taxonomie permet d'avoir une vision globale du domaine et des façons de les mettre en oeuvre de manière opérationnelle. En d'autres termes, la taxonomie reprend les solutions techniques et doit par conséquent évoluer avec elles.

La combinaison des deux éléments, *modèle de raffinement* et *taxonomie*, permet ainsi d'obtenir un raffinement type, ou autrement dit, un méta-modèle des possibilité globales de raffinement et de solutions potentielles pour les exigences. Il s'agit dès lors de pouvoir instancier ce méta-modèle au problème posé et à son domaine de connaissance propre afin de retenir les décompositions et enrichissements du méta-modèle qui sont pertinents pour l'analyste.

La méthode se caractérise finalement par l'aboutissement potentiel du raffinement en plusieurs alternatives de conception. Ceci implique, avant de passer à une mise en oeuvre opérationnelle, la nécessité d'effectuer un choix parmi les alternatives. Ce choix a été abordé comme étant avant tout une question de mesure de coût/efficacité des différentes alternatives.

b. L'illustration développée

Le but poursuivi consistait à présenter tous les concepts vus lors du développement de la méthodologie. L'illustration mettait en oeuvre la décision de développer un système de pointeuse à horaire flottant dans une société. Les points intéressants comprenaient le fait que :

1. Ce projet s'accompagnait du besoin de garantir la sécurité de l'accès au bâtiment et de certaines zones protégées. Ainsi, on a pu identifier que pour un même besoin de sécurité (sécurité d'accès), les décisions prisent en matière de sécurité n'étaient pas les mêmes pour tous les accès, mettant en évidence le problème de l'évaluation du niveau de couverture souhaité par un client à travers l'établissement d'une contrainte non-fonctionnelle .

2. La performance demandée en matière de temps nécessaire à un accès à permis de mettre à jour le fait qu'une solution potentielle d'une contrainte non-fonctionnelle peut être

abandonnée pour raison de forte incompatibilité. Ceci mit en valeur la possibilité de lien d'interdépendance entre les contraintes non-fonctionnelles.

3. La considération du personnel à permis de ne pas retenir l'accès par mot de passe aux entrées du bâtiment. Ceci montre qu'il faut considérer les éléments externes (ou qui paraissent externes) au projet comme pouvant jouer un rôle dans le choix d'une décision de conception.

6.2. Futurs développements

Un travail consistant reste à effectuer dans l'espoir de rendre cette méthodologie utilisable dans un besoin de traitement complet des différents types d'exigences non-fonctionnelles. Il s'agit avant tout de réaliser et de faire évoluer une taxonomie complète des exigences non-fonctionnelles. Mais il s'agit de pouvoir également enrichir le modèle pour pouvoir prendre en compte l'ensemble des exigences non-fonctionnelles.

On a présenté dans ces pages un moyen adapté au traitement des contraintes de sécurité et de performance. Mais le champ des exigences non-fonctionnelles ne se limite pas à ces seules types de problèmes. Il s'agit de pouvoir assurer la modélisation des autres types de contraintes non-fonctionnelles. Un premier temps dans la poursuite de l'élaboration de ce modèle pourrait passer par l'*établissement d'une taxonomie* complète pour chacun des types de ces contraintes. Il faut ajouter à cela que l'on a présenté l'amorçage de la taxonomie pour la performance, mais qu'il faut encore la compléter en considérant les différentes spécialités qui s'impliquent dans le raffinement tel que nous l'avons laissé.

De plus, la méthode s'étend dans l'élaboration d'une taxonomie jusqu'à présenter les moyens dont dispose l'analyste pour rendre le problème opérationnel. Un exemple est constitué par la possibilité de recourir à la technologie RAID (voir chapitre 4) afin de résoudre le problème de disponibilité décomposé de la sécurité des données. Cette caractéristique implique que le modèle n'est pas statique, c'est-à-dire que l'évolution de la technologie est susceptible de rendre inadapté ou de renforcer mais également de créer des moyens nouveaux dont nous pouvons disposer à un moment donné pour résoudre de manière opérationnelle un problème d'exigence non-fonctionnelle. Il s'agit par conséquent de veiller à l'évolution et à la mise à jour de la taxonomie.

Un autre problème consistera à établir la manière d'*enrichir le modèle du raffinement* présenté afin de le rendre applicable pour les autres exigences non-fonctionnelles. Ceci risque d'être assez important si l'on observe par exemple le cas particulier de la gestion de groupe. Il s'agit principalement de pouvoir manipuler des contraintes de formation d'équipes, de responsabilités à partager, et de délais à respecter. Dans ce cas de figure, des spécificités pourraient être apportées afin de modéliser les contraintes temporelles de délais, ainsi que les liens modélisant l'attribution de responsabilités à une équipe et également une manière plus opportune de présenter la réponse au problème lors de la validation (par l'équipe travaillant sur le projet).

Bibliographie

[1] : K. P. BADENHORST, J.H.P. ELOFF, *Framework of a Methodology for the Life Cycle of Computer Security in a Organisation*, Computers & Security, Vol. 8, n°5, August 1989, p.433-442

[2] : K. P. BADENHORST, J.H.P. ELOFF, *Computer Security Methodology : Risk Analysis and Project Definition*, Computers & Security, Vol. 9, n°4, June 1990, p.339-346

[3] : F. BODART, Y. PIGNEUR. *Conception Assistée des Systèmes d'Information - Méthode, Modèles, Outils*. 2ème édition, Méthodes Informatiques et Pratiques des Systèmes, MASSON, 1993.

[4] : L. CHUNG. *Dealing with Security Requirements During the Development of Information Systems*. In Advanced Information Systems Eng., Proc., 5th Int. Conf. CAISE'93, Paris, France. Berlin: Springer-Verlag, 1993, pp. 234-251.

[5] : L. CHUNG. *Representing and Using Non-functional Requirements : A Process-Oriented Approach*. PhD Thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, June 1993.

[6] : L. CHUNG. *Using Non-functional Requirements to Systematically Support Change*. Proceedings of the Second IEEE International Symposium on Requirements Engineering (RE'95), York (UK), March 27-29, 1995.

[7] : L. CHUNG, B. NIXON, E. YU. *Using Non-functional Requirements to Systematically Select Among Alternatives in Architectural Design*. ICSE-17 Workshop on Architectures for Software Systems, 24-25 April 1995, Seattle, Washington, U.S.A.

[8] : R. DARIMONT. *Process Support for Requirements Elaboration*. PhD Thesis, Département d'Ingénierie Informatique, Faculté des Sciences Appliquées, Juin 1995.

[9] : A. DARDENNE, A. vAN LAMSWEERDE, S. FICKAS. *Goal-directed Requirements Acquisition*. Science of Computer Programming, 20:3-50, 1993.

[10] : Ph. DU BOIS. *The Albert II Language : On the Design and the Use of a Formal Specification Language for Requirements Analysis*. PhD Thesis, Dept. of Computer Science, University of Namur, 1995.

[11] : E. DUBOIS, S. WU. *A Framework for Dealing with and Specifying Security Requirements in Information Systems*. Technical Report, Dept. of Computer Science, University of Namur, 1995.

[12] : F^3 Consortium. *From Fuzzy to Formal :Requirements Engineering Handbook*. Esprit III Project 6612, F^3 partners, 1994.

[13] : C. GHEZZI, M. JAZAYERI and D. MANDRIOLI. *Fundamentals of Software Engineering*. Prentice Hall International Editions, Edition 1991.

[14] : H. F. HOFMANN, R. HOLBEIN. *Seven Ways to Quality : A Framework for Specifying Security Requirements*. Proceedings of the First International Workshop on Requirements Engineering : Foundation of Software Quality, REFSQ'94, Utrecht, Netherland, 6-7 June, 1994.

[15] : B. JUNGEN. *The Albert II Refernce Manual*. Version 1.1 CAT Project, 1996.

[16] : N. G. LEVESON, *Software Safety : Why, What, and How*, ACM Computing Surveys, Vol. 18, n°2, June 1986, p.125-163

[17] : J. MYLOPOULOS, L. CHUNG, B. NIXON. *Representing and Using Non-functional Requirements : A Process-Oriented Approach*. IEEE Transactions on Software Engineering, Vol. 18 n°6, 1992

[18] : N. NILSSON. *Problem-Solving Methods in Artificial Intelligence*. New York, MCGraw-Hill, 1971.

[19] : B. NIXON. *Dealing with Performance Requirements During the Development of Information Systems*. Proc. IEEE International Symposium on Requirements Engineering, San Diego, CA, January 1993.

[20] : P. O'NIEL. *Database - Principles, Programming, Performance*. Morgan Kaufmann, 1994.

[21] : A. L. OPDAHL. *Requirements Engineering for Software Performance*. Proceedings of the First International Workshop on Requirements Engineering : Foundation of Software Quality, REFSQ'94, Utrecht, Netherland, 6-7 June, 1994.

[22] : Sh. L. PFLEEGER, *A Framework for Security Requirements*, Computers & Security, Vol. 10, 1991, p. 515-523

[23] : E. RICH, K. KNIGHT. *Artificial Intelligence*. 2nd Edition, McGraw-Hill, 1991.

[24] : S. L. TEAL, A. I. RUDNICKY. *A Performance Model of System Delay and User Strategy Selection*. CHI'92, pages 295-305. ACM, 3-7 May 1992. 0-89791-513-5/92/0005-0295.

[25] : A. vAN LAMSWEERDE, R. DARIMONT, Ph. MASSONET. *Goal-directed Elaboration of Requirements for a Meeting Scheduler: Problems and Lessons Learnt*. Proceedings of the Second IEEE International Symposium on Requirements Engineering (RE'95), York (UK), March 27-29, 1995.

[26] : P.H. WINSTON. *Artificial Intelligence*. 2nd Edition, Addison-Wesley, 1984.

[27] : Ch. WOOD. *Principles of Secure Information Systems Design*. Computers & Security, 9 (1990) 13-24.

[28] : S. WU, E. DUBOIS, J. RAMAEKERS. *Using a Formal Agent-Oriented Requirements Language for Message Handling System Security*. Dept. of Computer Science, University of Namur, 1996.

[29] : E. YU. *Modelling Strategic Relationship for Process Reengineering. PhD thesis, Computer Science Department, University of Toronto, Toronto (Canada), 1995.*

[30] : E. YU, Ph. DU BOIS, E. DUBOIS, J. Mylopoulos. *From Organization Models to System Requirements : A "Cooperating Agents" Approach*. Proc. of the Third International Conference on Cooperative Information System - COOPIS'95, Vienne (Autriche), Mai 9-12, 1995.

Facultés Universitaires
Notre-Dame de la Paix
Institut d' Informatique
Namur

Analyse et Modélisation des Exigences Non-fonctionnelles de Sécurité et de Performance

(TOME II)

Eric THIRY

Promoteur : M. Eric DUBOIS

Mémoire présenté en vue de l'obtention du diplôme de Maître en Informatique
Année Académique 1995 - 1996

TOME II : Annexes.

Annexe 1

Présentation du langage de spécification ALBERT

b. Les contraintes sur un agent

Il existe trois types de contraintes : - *les contraintes de base*
- *les contraintes locales*
- *les contraintes de coopération*

b.1. Les contraintes de base :

On définit : - *les composants dérivés*. Ce sont des composants dont la valeur est liée de manière statique à un ou des autres composant(s). Cela peut être défini uniquement entre composants internes d'un agent. Le montant exprimé ci-dessous verra sa valeur statiquement définie par l'évolution de la valeur des composants prix et quantité.
(Montant $\hat{=}$ Quantité * Prix)

- *les valeurs initiales*. Il s'agit pour un composant de déterminer quel est sa valeur lors de la naissance de l'agent. (exemple, Prix = 200)

b.2. Les contraintes locales :

On définit : - *le comportement d'état*. Il s'agit ici de définir un invariant sur un composant d'état. (exemple, Prix $\leq$ 200)

On peut aussi recourir aux connectivités temporelles pour exprimer le comportement d'état. Ainsi, nous définissons :

- $\Diamond$ x (x est vrai un moment dans le futur)
- $\blacklozenge$ x (x est vrai un moment dans le passé)

- □ x (x est toujours vrai dans le futur)
- ■ x (x est toujours vrai dans le passé)
- x U y (x est vrai jusqu'à ce que y devienne vrai)
- x S y (x est vrai depuis que y était vrai)

Ainsi, nous pouvons définir qu'une machine ne peut tomber en panne qu'une fois dans les comportements d'état (◆ Statut = Panne ⇒ □ Statut ≠ Panne)

On peut également définir des connectivités temps-réel :

- ✧ <10' x (x est vrai dans les 10')
- □ <10' x (x est vrai pendant au moins 10')
- etc...

- *les effets des actions*. C'est ici que l'on décrit l'effet que produit la survenance d'une action sur l'état de l'agent. Si une action externe a des effets sur un composant de notre agent, cet effet est repris ici. (on retrouve notre personne qui pousse la porte par : Personne.PousserPorte : EtatPorte : = Ouvert)

- *les "capabilities"*. Permet de renseigner sur les condition de survenance pour les actions internes. Il existe trois possibilités,
  - soit on oblige à réaliser tout le temps une action lorsqu'un état particulier est atteint.
  - soit on défend de réaliser une action lorsqu'un état particulier est atteint.
  - soit on a une combinaison des deux autres.

  Les différentes possibilités sont reprisent dans la figure 1.

	Défaut	F (1 / oui)	O (1 / oui)	XO (1 / oui)
oui	1 ou pas 1	pas 1	1	1
non	1 ou pas 1	1 ou pas 1	1 ou pas 1	pas 1

Figure 1. Capability.

Ainsi, si nous considérons le XO (1 / oui), dans le cas où oui survient, on exécute l'action 1. Pour F (1 / oui), dans le cas où oui survient, le choix est laissé pour l'exécution de 1.

- *la composition d'actions*. Elle permet de décomposer les actions en actions plus simples. Cette composition peut se réaliser de différentes manières :
  - composition séquentielle (a1 ; a2)
  - composition simultanée (a1 $\otimes$ a2)
  - composition parallèle (a1 $\|$ a2)
  - l'alternative (a1 $\oplus$ a2)
  - la répétition ($\{a1\}^n$)

Ainsi la décomposition de l'action Entrer en (PousserPorte ; Avancer) implique que l'action PousserPorte sera réalisée avant l'action Avancer.

- *l'action duration*. Cela permet d'exprimer une contrainte de longueur d'action. (| Produire | = 1' : implique que l'action de produire dure une minute).

b.3. Les contraintes de coopération :

On définit :

- *la perception d'action.* Ce type de contrainte restreint la perception des agents sur les actions qui ont lieu à l'extérieur de ces agents. Il existe trois possibilités,
  - soit on oblige à percevoir tout le temps une action lorsqu'un état particulier est atteint.
  - soit on défend de percevoir une action lorsqu'un état

particulier est atteint.

- soit on a une combinaison des deux autres.

Les différentes possibilités sont reprisent dans la figure 2

	Défaut	I (1 / oui)	K (1 / oui)	XK (1 / oui)
oui	1 ou pas 1	pas 1	1	1
non	1 ou pas 1	1 ou pas 1	1 ou pas 1	pas 1

Figure 2. Contraintes de coopération.

Ainsi, si nous considérons le XK (1 / oui), dans le cas où oui survient, on perçoit l'action 1. Pour I (1 / oui), dans le cas où oui survient, la perception de 1 est laissé au choix.

- *la perception d'état*. Ce type de contrainte restreint la perception des agents sur les états extérieurs de ces agents. Cette contrainte s'exprime également avec les concepts de la figure 2.

- *l'information d'action*. C'est une contrainte qui indique ce que l'on montre à l'extérieur.

- *L'information d'état*. Cela détermine quand un composant d'état de l'agent est visible par les autres agents.

Annexe 2

Présentation des opérateurs de types pré-définis

2. Présentation des opérateurs de types pré-définis

a. Basic types

BOOLEAN

* true constant *
True: → BOOLEAN

* false constant *
False: → BOOLEAN

* negation *
¬ #: BOOLEAN → BOOLEAN

* and *
# ∧ # : BOOLEAN x BOOLEAN → BOOLEAN

* or *
# ∨ # : BOOLEAN x BOOLEAN → BOOLEAN

* implies *
# ⇒ # : BOOLEAN x BOOLEAN → BOOLEAN

* equivails *
# ⇔ # : BOOLEAN x BOOLEAN → BOOLEAN

* alternatives *
If # then # else # : BOOLEAN x BOOLEAN xBOOLEAN → BOOLEAN

INTEGER

* addition *

+ # : INTEGER x INTEGER → INTEGER

* substraction *

− # : INTEGER x INTEGER → INTEGER

* multiplication *

* # : INTEGER x INTEGER → INTEGER

* division *

÷ # : INTEGER x INTEGER → INTEGER

* modulo *

mod # : INTEGER x INTEGER → INTEGER

* is a odd number ? *

Odd? : INTEGER → BOOLEAN

* is an even number ? *

Even? : INTEGER → BOOLEAN

* is divisible by ? *

Is-divisible-by # : INTEGER x INTEGER → BOOLEAN

* is equal to ? *

= # : INTEGER x INTEGER → BOOLEAN

* is not equal to ? *

≠ # : INTEGER x INTEGER → BOOLEAN

* is lower ? *

< # : INTEGER x INTEGER → BOOLEAN

* is greater ? *

> # : INTEGER x INTEGER → BOOLEAN

* is lower or equal ? *

≤ # : INTEGER x INTEGER → BOOLEAN

* is greater or equal ? *

≥ # : INTEGER x INTEGER → BOOLEAN

CHAR

* is equal to ? *
    # = # : CHAR x CHAR → BOOLEAN

* is not equal to ? *
    # ≠ # : CHAR x CHAR → BOOLEAN

* is lexically before ? *
    # < # : CHAR x CHAR → BOOLEAN

* is lexically after ? *
    # > # : CHAR x CHAR → BOOLEAN

* is lexically before or equal ? *
    # ≤ # : CHAR x CHAR → BOOLEAN

* is lexically after or equal ? *
    # ≥ # : CHAR x CHAR → BOOLEAN

STRING

* is equal to ? *
    # + # : STRING x STRING → STRING

* length *
    Length : STRING → INTEGER

* extraction of a sub-string ? *
    Sub-string : STRING x INTEGER x INTEGER → STRING

* is equal to ? *
    # = # : STRING x STRING → BOOLEAN

* is not equal to ? *
    # ≠ # : STRING x STRING → BOOLEAN

* is lexically before ? *
    # < # : STRING x STRING → BOOLEAN

* is lexically after ? *
    # > # : STRING x STRING → BOOLEAN

* is lexically before or equal ? *
    # ≤ # : STRING x STRING → BOOLEAN

* is lexically after or equal ? *
    # ≥ # : STRING x STRING → BOOLEAN

REAL

. . .

DATE
à définir

Enumerated Type
Exemple : COLOR = { Red, Blue, Green }

b. Types Constructors

For a matter of understandability, for each Type Constructor, we mention the name and the arity of formal operation parameters used.

The Set Type Constructor

The formal operation parameters used are:

    Eqt : T x T → BOOLEAN (equality on T)
    Prop? : T → BOOLEAN
    Op1 : T → T
    Op2 : T → T

* empty set *
    {}: → SET[T]

* set constructor *
    { # , # , # }: T x T x ... T → SET[T]

* is member of ? (use EqT) *
    # ∈ # : T x SET[T] → BOOLEAN

* is empty *
    Empty?: SET[T] → BOOLEAN

* cardinality *
Card: SET[T] → INTEGER

* intersection (use EqT) *
# ∩ # : SET[T] x SET[T] → SET[T]

* union *
# ∪ # : SET[T] x SET[T] → SET[T]

* substraction *
# \ # : SET[T] x SET[T] → SET[T]

* adding an element *
Add : SET[T] x T → SET[T]

* removing an element (use EqT) *
Remove: SET[T] x T → SET[T]

* is equal to ? (use EqT) *
# = # : SET[T] x SET[T] → BOOLEAN

* is not equal to ? (use EqT) *
# ≠ # : SET[T] x SET[T] → BOOLEAN

* is included ? (use EqT) *
# ⊂ # : SET[T] x SET[T] → BOOLEAN

* contains ? (use EqT) *
# ⊃ # : SET[T] x SET[T] → BOOLEAN

* is included or equal ? (use EqT) *
# ⊆ # : SET[T] x SET[T] → BOOLEAN

* contains or is equal ? (use EqT) *
# ⊇ # : SET[T] x SET[T] → BOOLEAN

* filtering (use Prop?) *
Filter : SET[T] → SET[T]

* mapping (use Op1) *
Map : SET[T] → SET[T]

* associative iteration (use Op2) *
Iter-assoc : SET[T] → SET[T]
(Exemples: +/, */, and/, or/, min/, max/)

The Cartesian Product Type Constructor

The formal operation parameters used are:

EqT1: T1 x T1 → BOOLEAN (Equality on T1)
EqT2: T2 x T2 → BOOLEAN (Equality on T2)
...
EqTn: Tn x Tn → BOOLEAN (Equality on Tn)

* cartesian pruduct constructor *
< # , #, ... # >: T1 x T2 x ... Tn → CP[T1, T2, ... Tn]

* is equal to ? (use EqTi) *
# = # : CP[T1, T2, ... TN] x CP[T1, T2, ... TN] → BOOLEAN

* is not equal to ? (use EqTi) *
# ≠ # : CP[T1, T2, ... TN] x CP[T1, T2, ... TN] → BOOLEAN

* access to the first component *
Sel1 : CP[T1, T2, ... TN] → T1

...

The Discriminant Union Type Constructor

The formal operation parameters used are:

EqT1: T1 x T1 → BOOLEAN (Equality on T1)
EqT2: T2 x T2 → BOOLEAN (Equality on T2)
...
EqTn: Tn x Tn → BOOLEAN (Equality on Tn)

* is equal to ? (use EqTi on Ti) *
# = # : UNION[T1, T2, ... TN] x UNION[T1, T2, ... TN] → BOOLEAN

* is not equal to ? (use EqTi on Ti) *
# ≠ # : UNION[T1, T2, ... TN] x UNION[T1, T2, ... TN] → BOOLEAN

* test of the type *
IsofT1 : UNION[T1, T2, ... TN] → BOOLEAN

...

The Bag Type Constructor

The formal operation parameters used are:

EqT: T x T → BOOLEAN (Equality on T)
Prop?: T → BOOLEAN
Op1: T → T
Op2: T → T

* empty bag *
$\{\}^+$: → BAG[T]

* bag constructor *
$\{ \# , \# , \# \}^+$: T x T x ... T → BAG[T]

* is member of ? (use EqT) *
$\# \in \#$: T x BAG[T] → BOOLEAN

* number of occurences ? (use EqT) *
Count: T x BAG[T] → BOOLEAN

* is empty *
Empty?: BAG[T] → BOOLEAN

* intersection (use EqT) *
$\# \cap \#$: BAG[T] x BAG[T] → BAG[T]

* union *
$\# \cup \#$: BAG[T] x BAG[T] → BAG[T]

* substraction *
$\# \setminus \#$: BAG[T] x BAG[T] → BAG[T]

* adding an element *
Add : BAG[T] x T → BAG[T]

* removing an occurence of element (use EqT) *
Remove: BAG[T] x T → BAG[T]

* removing all occurences of element (use EqT) *
Remove-all: BAG[T] x T → BAG[T]

* is equal to ? (use EqT) *
$\# = \#$: BAG[T] x BAG[T] → BOOLEAN

* is not equal to ? (use EqT) *
$\# \neq \#$: BAG[T] x BAG[T] → BOOLEAN

* is included in? (use EqT) *
$\# \subset \#$: BAG[T] x BAG[T] → BOOLEAN

* contains ? (use EqT) *
$\# \supset \#$: BAG[T] x BAG[T] → BOOLEAN

* is included in or equal ? (use EqT) *
$\# \subseteq \#$: BAG[T] x BAG[T] → BOOLEAN

* contains or is equal ? (use EqT) *
$\# \supseteq \#$: BAG[T] x BAG[T] → BOOLEAN

* filtering (use Prop?) *
Filter : BAG[T] → BAG[T]

* mapping (use Op1) *
Map : BAG[T] → BAG[T]

* associative iteration (use Op2) *
Iter-assoc : BAG[T] → BAG[T]
(Exemples: +/, */, and/, or/, min/, max/)

The Sequence Type Constructor

The formal operation parameters used are:

EqT: T x T → BOOLEAN (Equality on T)
Prop?: T → BOOLEAN
Op1: T → T
Op2: T → T

* empty seq *
[]: → SEQ[T]

* seq constructor *
[# , # , #]: T x T x ... T → SEQ[T]

* is member of ? (use EqT) *
$\# \in \#$: T x SEQ[T] → BOOLEAN

* is empty *

Empty?: SEQ[T] → BOOLEAN

* length *

Length: SEQ[T] → BOOLEAN

* first element *

First : SEQ[T] → T

* last element *

Last : SEQ[T] → T

* returns the head of the sequence *

Head : SEQ[T] → SEQ[T]

* returns the tail of the sequence *

Tail: SEQ[T] → SEQ[T]

* i^{th} element *

S_i : SEQ[T] x INTEGER → T

* part of a sequence *

Sub-seq : SEQ[T] x INTEGER x INTEGER → SEQ[T]

* concatenation *

+ # : SEQ[T] x SEQ[T] → SEQ[T]

* adding an element at end *

Append: SEQ[T] x T → SEQ[T]

* inserting an element at the i^{th} position *

Add-ith: SEQ[T] x T x INTEGER → SEQ[T]

* removing the i^{th} element *

Remove-ith: SEQ[T] x INTEGER → SEQ[T]

* is equal to ? (use EqT) *

= # : SEQ[T] x SEQ[T] → BOOLEAN

* is not equal to ? (use EqT) *

≠ # : SEQ[T] x SEQ[T] → BOOLEAN

* filtering (use Prop?) *

Filter : SEQ[T] → SEQ[T]

* mapping (use Op1) *
Map : SEQ[T] → SEQ[T]

* associative iteration (use Op2) *
Iter-assoc : SEQ[T] → T
(Exemples: +/, */, and/, or/, min/, max/)

The Table Type Constructor

The formal operation parameters used are:

EqT: T x T → BOOLEAN (Equality on T)
EqID: ID x ID → BOOLEAN (Equality on ID)
Prop?: T → BOOLEAN
PropID?: ID → BOOLEAN
Op: T → T

* empty table *
[]: → TBL(ID,T)

* is in range ? (use EqT) *
# ∈ # : T x TBL(ID,T) → BOOLEAN

* is in domain ? (use EqID) *
In: ID x TBL(ID,T) → BOOLEAN

* the domain *
Dom: TBL(ID,T) → SEQ(ID)

* the range *
Codom: TBL(ID,T) → SEQ(T)

* the length *
Length: TBL(ID,T) → INT

* is empty ?*
Empty? : TBL(ID,T) → BOOLEAN

* access to an element (use EqID)*
# [#] [EqID]: TBL(ID,T) x ID → T

* insersion (use EqID) *

Insert: TBL(ID,T) x ID x T → TBL(ID,T)

* suppression (use EqID) *

Delete: TBL(ID,T) x ID → TBL(ID,T)

* modification (use EqID) *

Modify: TBL(ID,T) x ID x T → TBL(ID,T)

* is equal to ? (use EqT and EqID) *

= # : TBL(ID,T) x TBL(ID,T) → BOOLEAN

* is not equal to ? (use EqT and EqID) *

≠ # : TBL(ID,T) x TBL(ID,T) → BOOLEAN

* filtering (use Prop?) *

Filter : TBL(ID,T) → TBL(ID,T)

* filtering (use PropID?) *

FilterID : TBL(ID,T) → TBL(ID,T)

* mapping (use Op) *

Map : TBL(ID,T) → TBL(ID,T)

Annexe 3

Présentation de la Méthode de Chung

Dans cette annexe, une présentation est faite du modèle qui constitue le pilier central de la méthodologie et dont les spécificités permettent de formaliser et de traiter les contraintes non-fonctionnelles de sécurité et de performance.

3. La représentation des contraintes non-fonctionnelles

Le modèle que l'on développe ici est issu des recherches effectuées par l'équipe de L. Chung sur les contraintes non-fonctionnelles. Une présentation du modèle peut être trouvé dans [5] ainsi que des illustrations de l'utilisation du modèle au cas particuliers de la sécurité dans [4], de l'exactitude dans [5] et [17] et de la performance dans [17] et [19]. Le modèle est aussi utilisé dans [7] afin d'appuyer la sélection dans les alternatives de conception architecturale et dans [6] pour soutenir le processus de maintenance adaptative d'un système informatique.

Ce modèle s'articule autour de cinq concepts :

1. Une classification de *buts*. Chacun de ces buts exprime soit un besoin non-fonctionnel, soit une décision de conception qui permet de mettre en oeuvre une solution fonctionnelle afin de répondre à un besoin non-fonctionnel ou finalement exprime un argument allant en faveur ou en défaveur de la prise en compte dans la solution d'une instance bien particulière d'un but de décision de conception.

2. Un ensemble de *méthodes*. Le modèle procède par décomposition de buts en sous-buts plus précis et plus simples. Ce processus se répète plusieurs fois jusqu'à l'obtention de solutions fonctionnelles pour le problème des exigences non-fonctionnelles posées. Les méthodes sont donc des procédures génériques qui permettent de décomposer (raffiner) les buts en sous-buts.

3. Une classification de *types de liens*. Puisque le modèle procède par raffinements successifs des buts, il existe un lien entre un but père et son ou ses fils raffinés. Les liens permettent d'identifier les buts fils à leur but père mais aussi, à travers les

différents types de liens, à exprimer le type de relation qu'entretient chaque but avec son but père.

4. Un ensemble de *règles de corrélation*. Une exigence non-fonctionnelle est raffinée par un ensemble de buts reliés par des liens. Mais outre les liens exprimant une relation entre but fils et but père, les liens peuvent exprimer un lien d'interdépendance positive ou négative entre deux buts ayant ou n'ayant pas les mêmes ancêtres. Les règles de corrélation permettent de générer ces liens spéciaux.

5. Une *procédure d'étiquetage*. Il s'agit de la dernière étape du modèle. Cette procédure a pour but d'aider à l'élaboration des différentes alternatives de solutions en déterminant le statut de chaque but par assignation d'une étiquette. En tenant compte des interactions positives et négatives entre buts ainsi qu'entre des liens plus ou moins forts liant but fils et père, il s'agit de voir dans la globalité quel est le niveau de réponse à chaque but.

Dans le but d'apporter un maximum d'éclaircissement concernant les concepts du modèle, une illustration appuie chacun d'entre eux. En voici un énoncé informel : "*Disposant d'une base de données regroupant des informations concernant des employés, on désire assurer que les données soient caractérisées par leur exactitude*". De sorte que dans ce qui suit, on retrouve une présentation fournie des concepts du modèle supportés par l'illustration, suivie de la description de la représentation graphique supportant la méthodologie et finalement de la critique du modèle.

a. Les buts

L'ensemble des buts de la méthodologie est composé de trois sous-ensembles qui sont mutuellement exclusifs. Les trois sous-ensembles sont appelés l'ensemble des ***buts des exigences non-fonctionnelles***, l'ensemble des ***buts de satisfaisabilité*** (que l'on appelle également l'ensemble des buts de décision de conception) et l'ensemble des ***buts d'argumentation***.

Un but est caractérisé par un *nom* et par zéro ou plusieurs *paramètres*. Une des limites du modèle tel que présenté par Chung [5] est liée à ce concept de paramètre qui renseigne mal sur la nature de l'objet manipulé par la contrainte. Afin de résoudre ce problème, on introduit à ce niveau une modification consistant à indiquer le type de chaque paramètre représentant un élément du système à développer. Ainsi, si on considère ce qui peut constituer la cible d'une contrainte non-fonctionnelle dans le système ou dans l'organisation, on identifie les données, les processus, les éléments architecturaux et la structure organisationnelle. A travers ces quatre cibles potentielles, on définit par conséquent les différents types de paramètres que l'on nomme respectivement DATA, PROCESS, DEVICE et ORGANIZATION.

L'identification d'un paramètre à un de ces types renvoie soit au *schéma conceptuel de la base de données* s'il s'agit du type DATA; aux *fonctionnalités attendues du système* développées durant la phase d'élaboration des exigences informelles s'il s'agit du type PROCESS; au *plan de l'architecture informatique envisagée* s'il s'agit du type DEVICE et finalement à *l'organigramme de l'organisation* s'il s'agit du type ORGANIZATION. Ainsi, le paramètre peut être identifié avec certitude à travers l'ensemble de ces documents.

Par exemple, nous pouvons définir Sécurité (employé : DATA) où "Sécurité" est le nom attribué au but d'exigence non-fonctionnelle; où "employé" est le nom de l'attribut de la base de données devant faire l'objet de mesure de sécurité; et où finalement "DATA" représente le type du paramètre manipulé, renvoyant ainsi l'identification du paramètre à la lecture du schéma conceptuel de la base de donnée.

a.1. Les buts d'exigences non-fonctionnelles :

Ce type de but constitue la partie haut niveau de la méthode. En effet, pour rappel on est, du point de vue des contraintes non-fonctionnelles, bien souvent amené à débuter le traitement à partir d'exigences abstraites exprimées par le client. Ces contraintes ainsi exprimées doivent être, grâce à la méthodologie, raffinées pour aboutir à une solution pouvant s'intégrer "fonctionnellement" dans le système à développer. De sorte que l'on définit les buts d'exigences non-fonctionnelles comme se trouvant au sommet de la méthodologie puisque ce

sont ces buts ci qui exprimeront les exigences abstraites des clients.

Reprenant l'exemple d'une contrainte non-fonctionnelle d'exactitude tel que développée dans [4], un tel type de but peut être représenté par :

Exactitude [Employé : DATA]

Selon la norme BNF, on établit la représentation syntaxique des buts d'exigences non-fonctionnelles de la manière suivante :

```
<but> ::=   <but exigences non-fonctionnelles> |
            <but satisfaisabilité> |
            <but argumentation>

<but exigences non-fonctionnelles> :: =   <nom but> [ <parametre> ] |
                                          <nom but>

<parametre> ::=  <symbole de variable> : <type> |
                 { <symbole de variable> : <type> , <parametre> }

<type> ::= DATA | PROCESS | DEVICE | ORGANIZATION
```

a.2. Les buts de satisfaisabilité :

Les buts de satisfaisabilité sont ceux qui s'établissent au niveau le plus bas dans le raffinement en présentant les alternatives fonctionnelles dont on dispose pour répondre aux besoins non-fonctionnels posés par le client.

Reprenant l'exemple cité ci-dessus, un moyen fonctionnel de satisfaire l'exactitude peut passer par : Validation [Employé : DATA] ,
qui peut encore être raffiné par :

ValidéPar [Sec1 : ORGANIZATION, Employé : DATA] ,

ce qui signifie que ce sont les secrétaires de classe 1 qui sont tenues d'effectuer une validation des données Employé.

La syntaxe de formaliation des buts de satisfaisabilié s'établit comme suit :

<but satisfaisabilité> ::= <nom but> [<parametre>] |
<nom but>

a.3. Les buts d'argumentation :

Ce dernier type de but est destiné à apporter un argument en faveur ou en défaveur d'un autre but ou d'un lien entre buts. Les buts d'argumentation ont toujours comme nom : *ArgFormel* ou *ArgInformel* dans le sens où l'argument présenté est toujours formel ou informel.

Reprenant l'exemple, on définit :

ArgFormel [∃e ∃s (ValidéPar [e : ORGANIZATION, Employé : DATA] ∧
EmpStatus (e : ORGANIZATION, s : ORGANIZATION) ∧
Classification_La_Plus_Haute (s : ORGANIZATION,
Sec2 : ORGANIZATION))]

ce qui signifie que le travail de validation doit être assuré par une personne dont le travail est évalué à un niveau supérieur d'une secrétaire de Classe 2.

D'une manière informelle, on pourrait également dire :

ArgInformel ["Une examination rigoureuse est recommandée lors de la modification des données Employé."]

Voici la syntaxe que l'on propose d'utiliser, et telle que définie par la logique des prédicats [23, 26] :

<but argumentation> ::= <Informel> | <Formel>

<Informel> ::= ArgInformel [" <Texte> " | <lien visé>]

<Formel> ::= ArgFormel [<corps> | <lien visé>]

```
<corps> ::= <formule atomique> |
            ( ¬ <corps> ) |
            ( <corps> <connecteur binaire> <corps> ) |
            ∃ <symbole de variable> <corps> |
            ∀ <symbole de variable> <corps>

<connecteur binaire> ::= ∧ | ∨ | ⊃ | ≡

<formule atomique> ::= <symbole de prédicat> ( <terme> ) |
                       <but>

<lien visé> ::= <lien>

<terme> ::= <symbole constante> |
            <symbole constante> , <terme> |
            <symbole de variable> : <type> |
            <symbole de variable> : <type> , <terme> |
            <symbole fonction> ( <terme> ) |
            <symbole fonction> ( <terme> ) , <terme>
```

Une base de types prédéfinis et d'opérateurs sur ces types est présentée en annexe 2 et peut être utilisée comme support des types ainsi que des prédicats et fonctions que nous sommes amenés à manipuler ici.

b. Les types de liens

Les méthodes que l'on abordera après ce point consistent à raffiner les buts non-fonctionnels en d'autres buts. Ce processus se répète plusieurs fois jusqu'à l'obtention de solutions fonctionnelles pour les contraintes non-fonctionnelles. Dès lors, puisqu'on raffine les buts, il y a de toute évidence un lien qui existe entre un but père et son ou ses buts fils raffiné(s).

On retrouve bien sur les liens classiques que sont OR et AND tel que défini par N. Nilsson dans [18], mais ceci n'est pas suffisant pour couvrir toutes les caractéristiques héritées du processus de raffinement et dont on doit tenir compte. Il faut effectivement des liens qui permettent d'exprimer un renforcement négatif ou positif entre deux buts. A cette fin, nous

considérerons les liens OR, AND, sup, sub, und, eql, -sub, -sup, -und, +und. Mais avant d'en dire plus sur leur signification, il faut introduire les concepts suivants : Satisfait, Satisfaisable, Rejeté et Rejetable. Ceux-ci vont permettrent d'exprimer la sémantique de chacun des liens.

Les liens relient un but père à un ou plusieurs buts fils, mais peuvent aussi relier un but d'argumentation à un ou plusieurs liens entre un but d'exigence non-fonctionnelle et un but de satisfaisabilité pour exprimer un support négatif ou positif dans le raffinement du premier vers le deuxième. Durant la procédure d'étiquetage on est amené à vérifier si le choix d'un but de décision de conception rend le but non-fonctionnel satisfaisable (on y répond bien). Pour le savoir il faut propager la notion de satisfaisabilité depuis le bas niveau (décision de conception) vers le haut niveau (niveau abstrait, non-fonctionnel) ce qui implique la connaissance du caractère satisfaisable ou non du lien entre les buts.

Considérons par conséquent l'ensemble des liens et *Satisfait*, un prédicat qui est vrai pour les buts ET LES LIENS qui ont été montrés satisfaisants. Et considérons également *Rejeté*, un prédicat qui est vrai pour les buts ET LES LIENS qui ont été montrés insatisfaisants.

La syntaxe de tels prédicats est définie par :

<prédicat> ::= Rejeté (<proposition>) |
Satisfait (<proposition>)

<proposition> ::= <but> | <lien>

Ou un lien sera défini par :

<lien> ::= <lien normal> | <lien argumentation>

<lien argumentation> ::=
<type du lien> (<but argumentation> , <lien>)

<lien normal> ::= <type du lien> (<but père> , <but fils>)

<but père> ::= <but>

<but fils> ::= <but> | { <but> , <liste> }

<type du lien> ::= AND | OR | sup | sub | -sup | -sub | eql | und | -und | +und

Figure 1 : Exactitude [Employé : DATA] $\xrightarrow{AND}$
{Exactitude [Chercheur : DATA] ,
Exactitude [Secrétaire : DATA] }

de sorte que nous pouvons mémoriser la méthode suivante :

x : DATA, x_i : DATA : Exactitude [x : DATA] $\xrightarrow{AND}$
{ Exactitude [x_i : DATA] | $\forall$ x_i : IsA(x_i, x) }

où IsA (,) est un prédicat vérifiant que le deuxième paramètre est générique du premier.

Figure 2 : Sécurité [CptBanc : DATA] $\xrightarrow{AND}$
{ Intégrité [CptBanc : DATA] ,
Confidentialité [CptBanc : DATA] ,
Disponibilité [CptBanc : DATA] }

de sorte que nous pouvons mémoriser la méthode suivante :

x : DATA : Sécurité [x : DATA] $\xrightarrow{AND}$
{ Intégrité [x : DATA] ,
Confidentialité [x : DATA] ,
Disponibilité [x : DATA] }

c.2. Les méthodes de satisfaction :

De telles méthodes ont pour objectif de raffiner un but en un ensemble de buts de satisfaisabilité qui impliquent, si la solution est retenue, un engagement du concepteur à mettre en oeuvre une décision de conception afin de répondre à la contrainte non-fonctionnelle.

A ce niveau, il est possible d'obtenir une assistance afin de générer des solutions à apporter aux contraintes non-fonctionnelles soit par une réutilisation des techniques de conception utilisées pour répondre au même type de contraintes dans le passé, soit en observant les solutions proposées dans les autres organisations ou bien également dans la littérature.

conséquent peut être de façon systématique présentée au concepteur lors de la phase de raffinement. Cette décomposition s'appuie sur une typologie qui commence à se construire pour chaque type de contrainte non-fonctionnelle. On peut illustrer cette décomposition à travers le problème de la sécurité qui sera abordé plus en profondeur dans le chapitre suivant. Supposons que l'on veuille organiser la sécurité autour des données de comptes bancaires. Ceci passe par le besoin d'assurer l'intégrité, la confidentialité et la disponibilité de ces données. (figure 2)

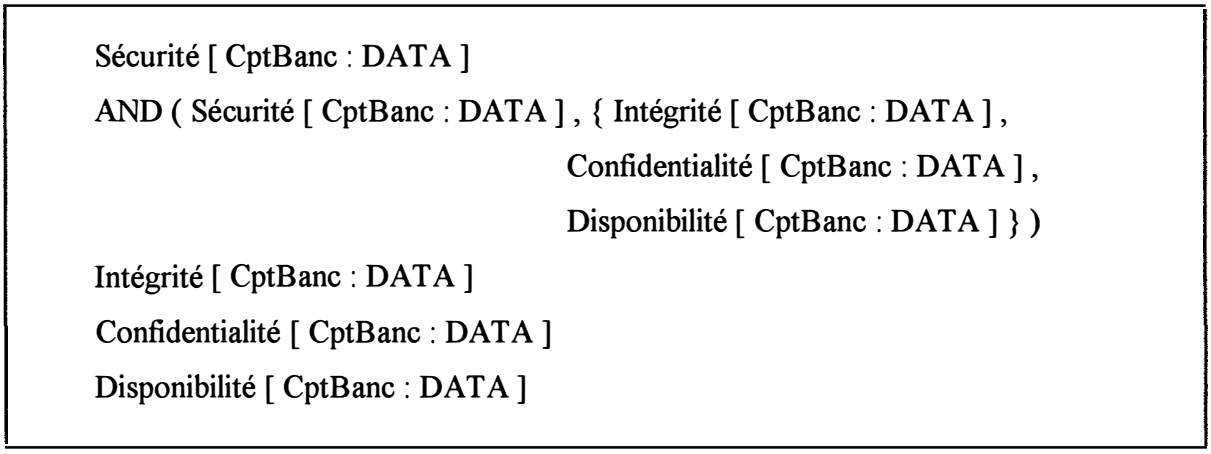


Figure 2. Décomposition de but.

Le formalisme que l'on propose afin de modéliser les méthodes est le suivant :

```
<méthode> ::= <nom but> [ <parametre> ]
                  --<type de lien>--> { <liste attributs fils> } |
              <parametre> : <nom but> [ <parametre> ]
                  --<type de lien>--> { <liste attributs fils> | <condition> }

<liste attributs fils> ::= <nom but> [ <parametre> ] |
                           <nom but> [ <parametre> ] ,
                                <liste attributs fils>

<condition> ::= <corps>
```

A l'aide de ce formalime, les méthodes qui ont permis de traiter nos exemples de la figure 1 et 2 peuvent être éditées.

maintenant.

c.1. Les méthodes de décomposition :

Une méthode de décomposition peut être basée sur une décomposition du paramètre du but ou sur une décomposition du but lui-même. Généralement, cette création passe par la création d'un lien AND entre le but père et le ou les but(s) fils.

La décomposition basée sur le paramètre reste de la responsabilité intégrale du concepteur car cette décomposition dépend de la structure propre du paramètre.

Prenons l'exemple de notre donnée Employé : Exactitude [Employé : DATA]. Il se peut que l'on décide de prendre la décision de décomposer "Employé" en ses sous-classes "Chercheur" et "Secrétaire". (figure 1)

```
Exactitude [ Employé : DATA ]
AND ( Exactitude [ Employé : DATA ] ,
        { Exactitude [ Chercheur : DATA ] , Exactitude [ Secrétaire : DATA ] } )
Exactitude [ Chercheur : DATA ]
Exactitude [ Secrétaire : DATA ]
```

Figure 1. Décomposition du paramètre.

La bonne question à se poser consiste à se demander quelle peut bien être l'utilité d'une décomposition de paramètre. Si on retranspose la question dans l'exemple, on explique l'utilité de cette décomposition par le fait qu'Employé est composée de sous-classes Chercheur et Secrétaire qui ne présentent pas la même importance. Par là même, il se peut que endéans un niveau de réponse à l'exactitude apporté par le concepteur, le client soit prêt à accepter une moins grande couverture des données secrétaire parce que ces données sont moins sensibles.

La décomposition basée sur le but quant à elle peut faire l'objet d'une réutilisation et par

Satisfait (G1) ∧ Satisfait (G2) ∧ ... ∧ Satisfait (Gn) ∧
Satisfait (-sub (G0, G1)) → Rejetable (G0)

b.7. Le lien eql :

eql : Proposition x Proposition

eql (G0, G1) ≡ sup (G0, G1) ∧ sup (G1, G0) ∧
sub (G0, G1) ∧ sub (G1, G0)

b.8. Le lien und :

Ce lien intervient parce qu'il se peut que le concepteur ne sache pas bien pour un but quel sera son impact sur un autre but particulier.

und : Proposition x Proposition

und (G0, G1) montre la présence d'une influence possible entre G0 et G1. Cette influence pourrait être positive ou négative. A ceci, on ajoute +und et -und qui exprime le fait que l'on est en position de préciser que l'influence serait respectivement positive ou négative.

c. Les méthodes

On a vu que le modèle procédait par raffinements successifs de buts. Ces raffinements sot laissés à la responsabilité du concepteur mais l'utilisation répétée du modèle peut permettre de mémoriser et de réutiliser certaines techniques mises à jour. De plus, le modèle propose certaines méthodes qui représentent des procédures génériques de raffinement de buts en un ou plusieurs but(s) fils.

Chaque raffinement est représenté par un lien liant le but père à son ou ses nouveaux buts fils et dont le lien est considéré comme Satisfait. Il y a trois types de méthodes correspondant aux trois types de buts présentés précédemment et que l'on va détailler

b.3. Le lien sup : [condition suffisante]

sup : Proposition x Proposition

Satisfait (G1) $\wedge$ Satisfait (sup (G0, G1)) $\rightarrow$ Satisfaisable (G0)

Le lien sup est une contribution positive qui est telle que ce lien indique que la satisfaction du but fils est une preuve suffisante pour dire que le but père serait satisfaisable.

b.4. Le lien sub : [condition nécessaire]

sub : Proposition x Proposition

Rejeté (G1) $\wedge$ Satisfait (sub (G0, G1)) $\rightarrow$ Rejetable (G0)

En somme, le lien sub indique que la satisfaction du but fils est nécessaire pour dire que le but parent serait satisfaisable.

$\neg$ (Satisfait (G2) $\wedge$ ... $\wedge$ Satisfait (Gn) $\rightarrow$ Satisfaisable (G0))

mais,

Satisfait (G1) $\wedge$ Satisfait (G2) $\wedge$ ... $\wedge$ Satisfait (Gn) $\wedge$

Satisfait (sub (G0, G1)) $\rightarrow$ Satisfaisable (G0)

b.5. Le lien -sup :

-sup : Proposition x Proposition

Satisfait (G1) $\wedge$ Satisfait (-sup (G0, G1)) $\rightarrow$ Rejetable (G0)

b.6. Le lien -sub :

-sub : Proposition x Proposition

Rejeté (G1) $\wedge$ Satisfait (-sub (G0, G1)) $\rightarrow$ Satisfaisable (G0)

$\neg$ (Satisfait (G2) $\wedge$ ... $\wedge$ Satisfait (Gn) $\wedge$ Satisfait (-sub (G0, G1))

$\rightarrow$ Rejetable (G0))

mais,

<liste> ::= <but> | <but> , <liste>

Une proposition peut parfois être *Satisfaisable* ou *Rejetable* au regard des buts qui lui sont rattachés. En fait si un but fils est satisfait et que le lien le reliant à son père l'est aussi, le but père est potentiellement satisfait, c'est-à-dire qu'il est satisfaisable cela dépend d'autres liens éventuels qu'à le but père.

Dès lors, on complète la syntaxe par :

<prédicat> ::= **Rejeté (** <proposition> **) |**
Satisfait (<proposition> **) |**
Satisfaisable (<proposition> **) |**
Rejetable (<proposition> **)**

Au regard de ceci, il reste à présenter la signification de chaque type de lien. Ce à quoi l'on va s'attacher maintenant en exprimant la sémantique de chaque lien en s'appuyant sur les prédicats.

b.1. Le lien AND :

AND : Proposition x $2^{\text{Proposition}}$

Satisfait(G1) $\wedge$ Satisfait (G2) $\wedge$ ... $\wedge$ Satisfait (Gn) $\wedge$
Satisfait (AND (G0, { G1, ..., Gn })) $\rightarrow$ Satisfaisable (G0)

(Rejeté (G1) $\vee$ Rejeté (G2) $\vee$ ... $\vee$ Rejeté (Gn)) $\wedge$
Satisfait (AND (G0, { G1, ..., Gn })) $\rightarrow$ Rejetable (G0)

b.2. Le lien OR :

OR : Proposition x $2^{\text{Proposition}}$

Rejeté (G1) $\wedge$ Rejeté (G2) $\wedge$ ... $\wedge$ Rejeté (Gn) $\wedge$
Satisfait (OR (G0, { G1, ..., Gn })) $\rightarrow$ Rejetable (G0)

(Satisfait (G1) $\vee$ Satisfait (G2) $\vee$ ... $\vee$ Satisfait (Gn)) $\wedge$
Satisfait (OR (G0, { G1, ..., Gn })) $\rightarrow$ Satisfaisable (G0)

Reprenant l'exemple, il se pourrait que l'organisation d'une vérification des données apporte une influence positive sur l'exactitude des données. De même, la validation des données serait un critère suffisant pour garantir l'exactitude. (figure 3)

Exactitude [Chercheur : DATA]
+und (Exactitude [Chercheur : DATA] , Vérification [Chercheur : DATA])
sup (Exactitude [Chercheur : DATA] , Validation [Chercheur : DATA])
Vérification [Chercheur : DATA]
Validation [Chercheur : DATA]

Figure 3. Méthode de satisfaction

c.3. Les méthodes d'argumentation :

Les méthodes d'argumentation raffinent un but ou un lien en un but d'argumentation qui renseigne sur une preuve ou une contre-évidence concernant le caractère satisfaisant ou pas de ce but ou lien. Il faut surtout connaître particulièrement bien l'environnement dans lequel viendra s'intégrer le système en développement afin de pouvoir identifier ce qui constituera un élément positif ou négatif dans le choix d'un but précis.

d. Les règles de corrélation

Comme nous l'avons déjà souligné, les contraintes non-fonctionnelles s'influencent entre elles de manière positive ou négative. Il y aura lieu de reprendre les différents liens qui ont été présentés ci-dessus pour exprimer l'influence existant entre des buts raffinés de contraintes non-fonctionnelles différentes.

On a besoin de guide méthodologique pour mettre à jour de telle relation afin de

pouvoir sélectionner les buts de décision de conception en connaissance de cause. Cette guidance se fait à travers la notion de règles de corrélation. Ces règles de corrélation sont tirées de la littérature et de l'expérience acquise (on retrouve l'omniprésence du caractère de réutilisabilité) et peuvent en temps utile être confirmée par le concepteur.

Une illustration du principe se retrouve à travers le besoin de minimiser le coût opérationnel en personne humaine et celui de demande d'une interface utilisateur pouvant être utilisée facilement. Le fait d'utiliser une interface facile à maîtriser ne nécessite pas d'utilisation de personne hautement qualifiée pour exécuter l'application et donc limite le coût opérationnel. Nous pourrions créer un lien sub :

sub (CoûtOpérationnel [Personnel : ORGANIZATION] ,
InterfaceUtilisateurSimple [Personnel : ORGANIZATION, Donnée : DATA]

et établir une règle de corrélation qui sera réutilisable dans le futur :

CoûtOpérationnel [Personnel : ORGANIZATION] ⟹
InterfaceUtilisateurSimple [Personnel : ORGANIZATION, Donnée : DATA]

La syntaxe se présenterait comme suite :

```
<règle de corrélation> ::= ( <expression gauche> ) ⟹
                             ( <expression droite> )

<expression droite> ::=    ( <lien> ∧ <liste lien> ) |
                           <lien>

<liste lien> ::= <lien> |
                 <lien> ∧ <liste lien>

<expression gauche> ::= <corps>
```

e. La procédure d'étiquetage

La procédure d'étiquetage est la dernière étape du modèle avant d'entamer l'évaluation des solutions potentielles. Cette procédure a pour but d'aider l'élaboration des différentes alternatives en déterminant le statut de chaque but par assignation d'étiquettes.

Un but est étiqueté *Satisfait* s'il est Satisfaisable et pas Rejetable; *Rejeté* s'il est Rejetable et pas Satisfaisable; *Conflictuel* s'il est à la fois Satisfaisable et Rejetable au regard de ses buts fils. Finalement dans un dernier cas, il sera étiqueté *Indéterminé* s'il n'est pas possible de déterminer son statut. Correspondant aux liens +und et -und, on identifie également *Indéterminé plus* et *Indéterminé moins.*

On établit dès lors la syntaxe de la manière suivante :

<procédure étiquetage> ::= <but> = <étiquette> |
<lien> = <étiquette>

<étiquette> ::= S | R | C | I | I^+ | I^-

Ce sont les situations conflictuelles qui impliquent la présence d'alternatives puisque l'on posera des choix de décision de conception qui sont tels qu'un but non retenu ayant une influence négative sur un but étiquetté conflictuel peut le rendre satisfait.

f. Le graphe des buts

Il ne s'agit pas d'un nouveau concept de la méthode mais d'un moyen de représenter de manière graphique tout ce qui a été développé dans ce chapitre. La portée du graphe est de permettre une vision étendue du problème afin d'aider le concepteur à se faire une représentation mentale complète de la situation.

Mais l'importance du graphe réside également dans l'aide qu'il pourra fournir lors de l'étape de validation puisqu'il permettra au client de se faire une idée de la solution apportée sans pour autant devoir faire l'effort de compréhension du modèle formel.

Chaque but développé est identifié dans le graphe par un cercle marqué de son libellé formel. Le reste des principes de la représentation est relaté dans la figure 4.

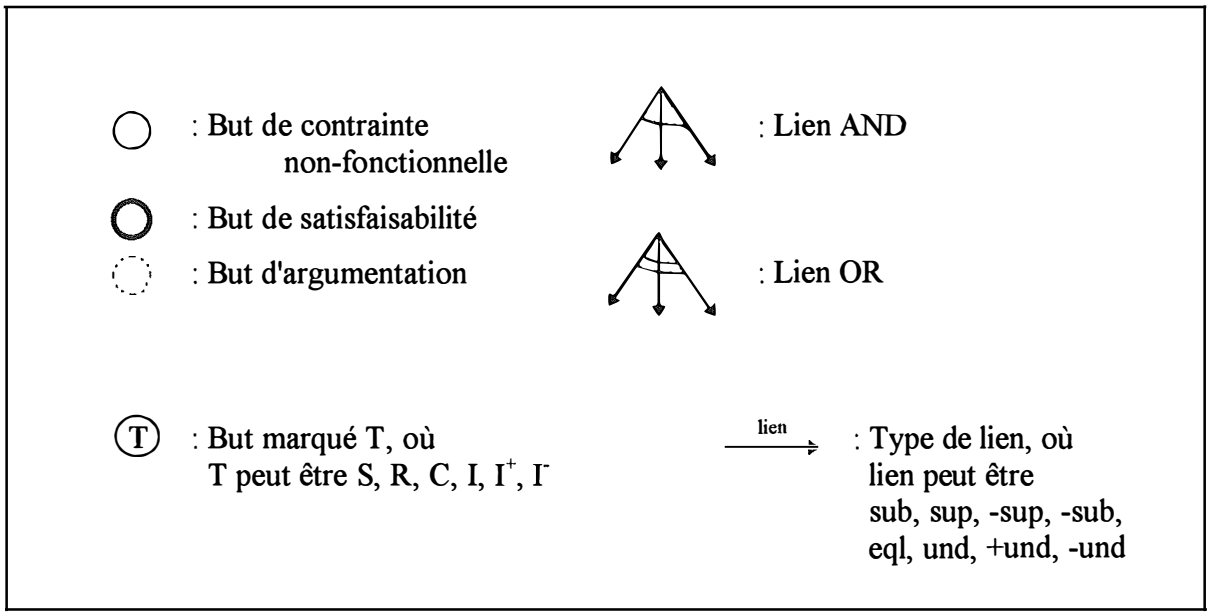


Figure 4. Légende de la représentation graphique.

g. Avantages de la méthodologie

En toute généralité, l'apport de cette méthode consiste à faciliter la disponibilité de grandes variétés de techniques connues et moins connues afin de répondre aux exigences non-fonctionnelles (pour autant que l'on dispose de méthodes et de liens de corrélation pré-définis) .

Un des avantages importants de la méthode de Chung réside dans la possibilité de définir des méthodes et des règles de corrélation assurant par là même une réutilisation de l'acquis des traitements d'exigences non-fonctionnelles dans le passé. Ceci est d'autant plus apprécié que la lourdeur des moyens utilisés globalement lors du développement d'un projet informatique rend ce type d'opportunité de plus en plus indispensable.

On remarque également que le modèle de Chung tel que présenté dans [5] ne fournit pas de syntaxe. Mais tel que reformulé ici, avec une définition de syntaxe pour chacun des concepts et l'aide à la formalisation des buts d'argumentation apporté par la logique des prédicats ainsi que d'opérateurs pré-définis sur les types de données permettent de fournir un

formalisme mathématique.

L'assistance graphique est un élément utile dans cette méthode car elle apporte une vision plus globale du problème. Ceci est fort utile pour pouvoir identifier les alternatives de choix de décision de conception ainsi que pour pouvoir juger des interdépendances entre buts ne découlant pas d'un même raffinement d'exigences non-fonctionnelles.

h. Inconvénients de la méthodologie

Le formalisme apporté est assez lourd à lire. Ainsi, sans appuie de la représentation graphique, l'ensemble de la formulation mathématique rend la compréhension du problème particulièrement compliquée. Ceci est imputable à la structure globale du modèle qui implique la création d'un ensemble d'arbres difficilement lisibles lorsqu'ils sont exprimés sous forme mathématique.

De plus, la représentation graphique est, elle aussi, lourde à manipuler car elle implique au fur et à mesure du raffinement un espace nécessaire imposant. Ceci est particulièrement frappant lorsque l'on effectue une décomposition de paramètre, puisque tout le raffinement aval pour chacun de ces buts-fils sera souvent identique (ce qui ne veut pas dire que la solution pour chacun d'eux sera la même).

Annexe 4

Etude de Cas

Nous allons maintenant appliquer la technique de raffinement des contraintes non-fonctionnelles de sécurité et de performance à travers un cas de développement d'un projet informatique. Celui-ci a spécialement été choisi non pour son intérêt particulier mais pour sa capacité à mettre en lumière les principes qui ont été évoqués jusqu'ici.

Le projet concerne le développement d'un système de pointeuse informatisée impliquant un besoin de contrôle d'accès. Nous allons tout d'abord présenter le contexte dans lequel le projet prend naissance. Nous présenterons dans un deuxième point les exigences liées au projet telles qu'elles pourraient être émises par les futurs utilisateurs (expression informelle). Nous détaillerons par après la spécification formelle du niveau software. Un quatrième point verra le raffinement des contraintes non-fonctionnelles de sécurité et de performance. Finalement nous détaillerons les répercussions de ses contraintes non-fonctionnelles sur les autres niveaux du développement.

Cette étude de cas est basée sur le système d'horaire flottant tel qu'il a été défini au sein de la COBAC S.A., société d'assurance crédit de Belgique. Toutefois, ce cas s'en inspire mais les décisions de conception détaillées dans ce chapitre ne reflètent en rien les dispositions prisent au sein de cette entreprise. De même, le contexte est ici purement fictif.

4.1. Présentation de l'étude de cas

Une organisation fonctionnait jusqu'il y a peu avec un système de pointeuse mécanique assisté par un préposé du service du personnel vérifiant l'utilisation de la pointeuse ainsi qu'effectuant les fonctions de contrôle des heures prestées.

Suite au départ en pension du préposé et suite à la recherche croissante de diminution du coût de structure organisationnelle, l'entreprise acheta un système de pointeuse informatisée qui se révéla fort peu efficace dans son adaptation à notre situation. Notamment par son manque de clarté à l'utilisation. Un exemple, la principale différence entre notre entreprise et celle nous ayant vendu le système venait du fait de la présence dans notre société de personnes pouvant à tout moment partir à l'extérieur afin d'effectuer une mission. Ceci n'étant pas prévu à l'origine, il fut décidé de rajouter des fonctionnalités au système. Ces fonctionnalités furent implémentées de manière tout à fait maladroite empêchant toute utilisation du système de manière intuitive.

La situation actuelle est donc caractérisée par un rejet unanime de la part du personnel. Conscient des nombreux manquements du système, il a donc été décidé de reprendre les choses depuis le début afin d'implémenter un nouveau système satisfaisant pour tous.

Nous ne nous intéresserons pas d'avantage à la solution choisie en premier lieu. Du moins, il est primordial de savoir que :

- le système privilégiait un accès par carte clé;
- l'entrée au bâtiment nécessitait l'introduction d'un code secret en plus de l'utilisation de la carte clé;
- certaines zones du bâtiment étaient d'accès protégé. Cette protection était assurée par un lecteur identique à ceux des entrées du bâtiment.

Toutefois, comme une authentification était assurée à l'entrée du bâtiment, il suffisait vis-à-vis de ces zones protégées de présenter une carte de personne ayant accès pour être libre du passage (simple identification).

Finalement, afin d'avoir une idée précise de la situation, un plan du rez-de-chaussée et du premier étage sont présentés à la figure 1 et 2.

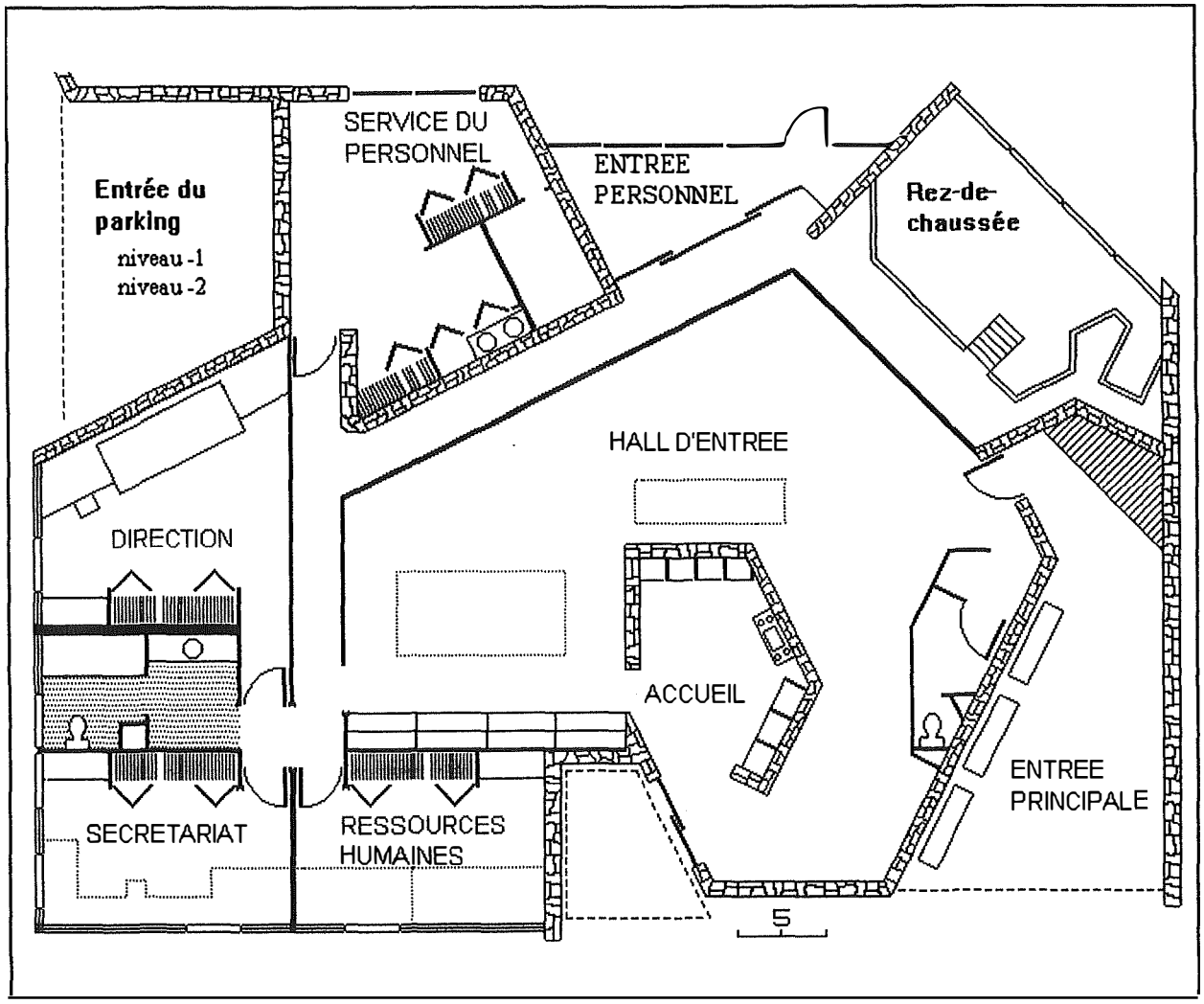


Figure 1. Le plan du rez-de-chaussée.

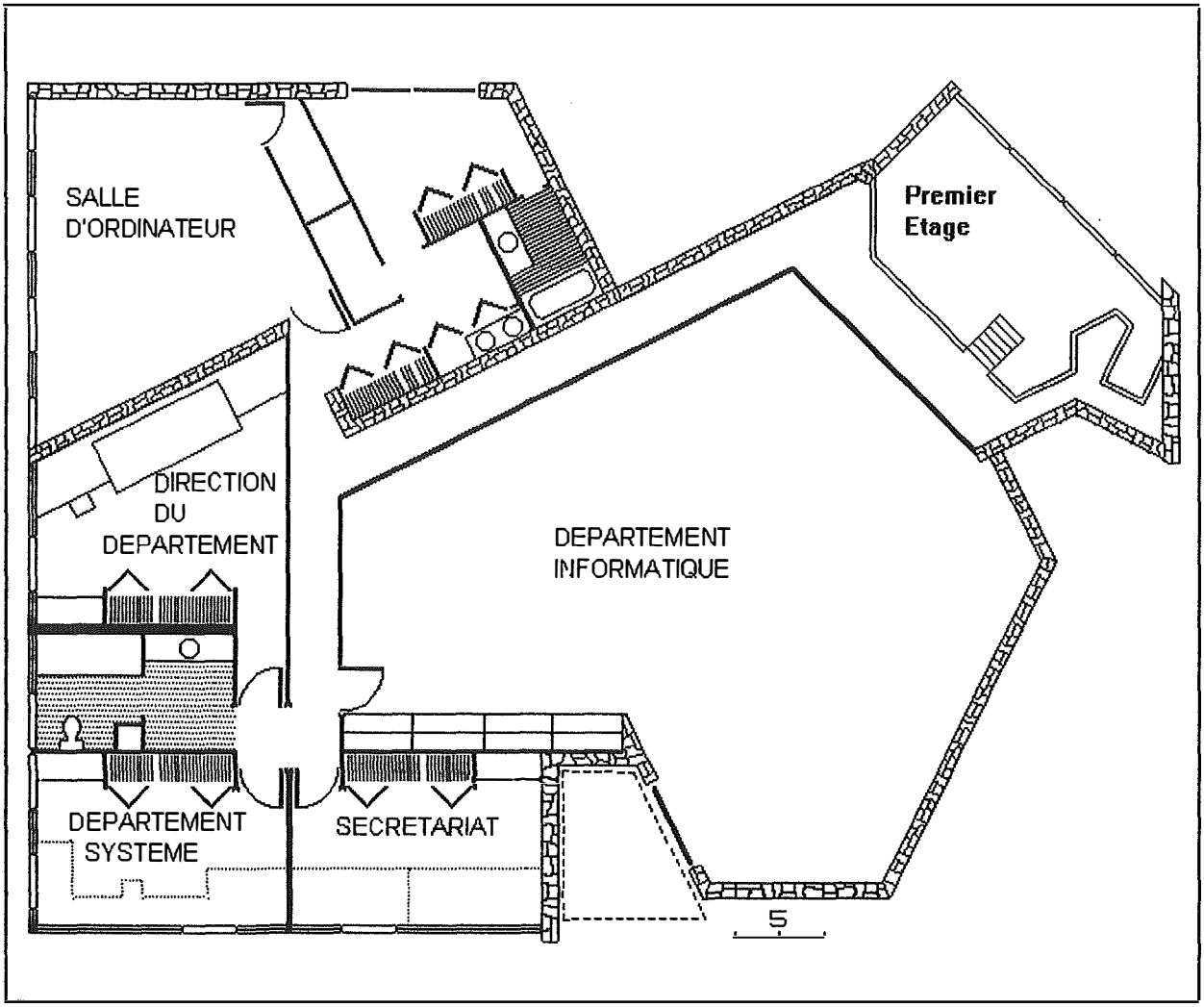


Figure 2. Le plan du premier étage.

4.2. Les exigences informelles

Parmi les exigences fonctionnelles visant à établir un système de poiteuse à horaire variable, il faudra prendre en considération le cadre légale définissant le temps de travail ainsi que tous les engagements qui ont été définies dans la convention collective de travail entre les employeurs et les représentants des travailleurs. Une énumération de tous les éléments à respecter constitue une bonne base avant de voir plus précisément les exigences que devront repecter les fonctionnalités du système. Les fonctionnalités émises par le client seront dès lors détaillées en intégrant les contraintes d'ordre légales. Les contraintes non-fonctionnelles que le client désire voir réalisées dans le projet seront ensuite détaillées. Finalement, toutes les exigences en matière de restructuration organisationnelle seront exprimées.

a. Le fonctionnement de l'horaire variable

Dans la définition des dispositions concernant le fonctionnement de l'horaire variable devant faire l'objet d'une informatisation, la présentation est dans ce chapitre découpée en identification du *cadre légal* à respecter, suivi des *règles de fonctionnement* concernant les plages horaires de travail, la manière de *comptabiliser les heures prestées* et finalement des dispositions concernant les *absences, congés et récupérations*.

a.1. Le cadre légal :

De manière générale, la loi prévoit que :

- la journée de travail ne dépasse pas 9 heures,
- la semaine de travail ne dépasse pas 40 heures,
- on ne travaille pas le dimanche,
- on ne travail pas la nuit.

Des dérogations sont possibles dans certains cas :

- sans formalités spéciales (travaux de clôture financière, ...)
- moyennant l'accord de la Délégation Syndicale (heures supplémentaires, ...)
- moyennant une convention de secteur ou d'entreprise (flexibilité du temps de travail, ...)
- moyennant l'accord du ministère du travail (travail de nuit, ...)

a.2. Les règles de fonctionnement :

Heures d'arrivée : Les membres du personnel arrivent entre 8h00' et 9h00' à l'heure de leur choix.

Heures de départ : Du lundi au jeudi, les membres du personnel partent entre 16h00' et 18h00' à l'heure de leur choix. Concernant le vendredi, les membres du personnel partent entre 15h00' et 17h00' à l'heure de leur choix.

Plages fixes : Tout le monde doit être présent :

- entre 9h00' et 12h00'
- entre 14h00' et 16h00' du lundi au jeudi
- entre 14h00' et 15h00' le vendredi

Pause de midi : Entre 12h00' et 14h00', le personnel peut disposer de ce temps pour son déjeuner et son délassement. La pause ne peut pas être inférieure à 45 minutes.

Un résumé de toutes ces dispositions est présenté à la figure 3.

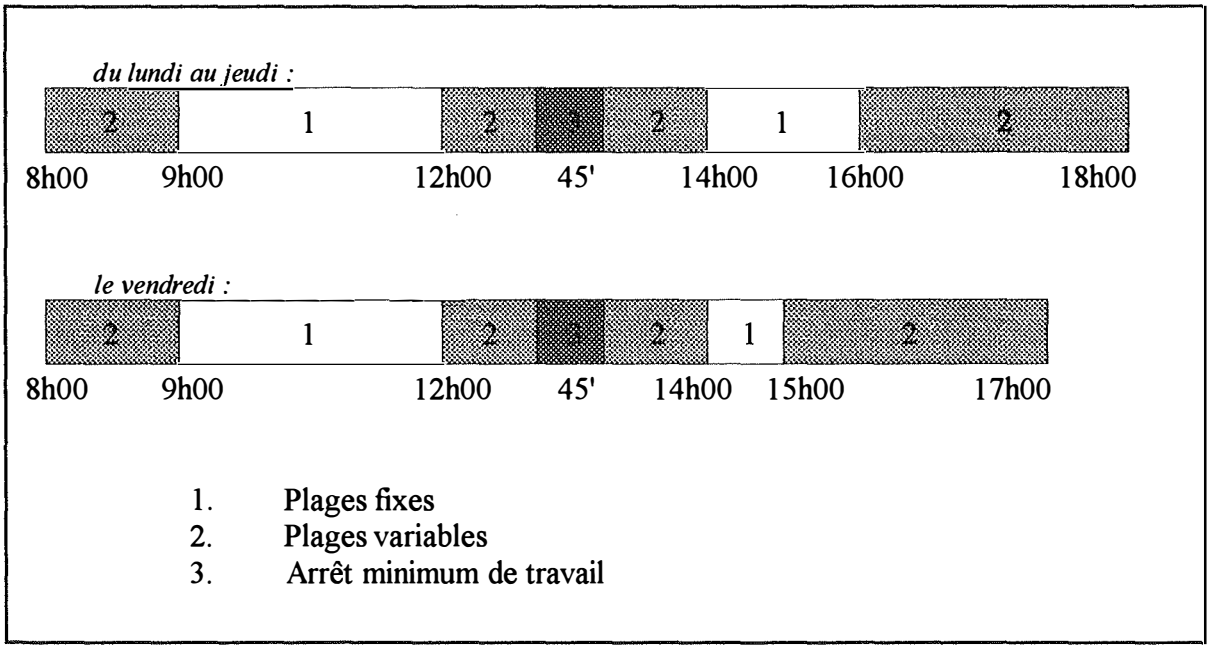


Figure 3. Réglementation de l'horaire variable.

Régime spécial : Certaines personnes font l'objet d'un régime spécial du fait de la nécessité d'effectuer des missions à l'extérieur pour le compte de la société. Ainsi, si un employé doit sortir régulièrement de la Société pendant les heures de travail, mais pour le compte de celle-ci, le Service du Personnel donnera un accès mission à la demande du responsable hiérarchique. Différentes situations peuvent se présenter :

- mission en cours de journée mais avec présence pour la plage de repos à midi,
- la journée commence par une mission et celle-ci s'étend sur toute ou plusieurs journées et le retour se fait en début d'une journée.

Quand un membre du personnel est en mission, le temps de travail est calculé dans la limite de l'horaire, c'est-à-dire de 8h00' à 18h00' sauf le vendredi de 8h00' à 17h00'.

a.3. La comptabilisation des heures prestées :

A la fin de chaque mois, plusieurs situations peuvent se présenter :

- le nombre d'heures prestées coïncide avec le nombre d'heures comprises dans la période de référence,
- le nombre d'heures prestées est supérieur au nombre d'heures comprises dans la période de référence. Dans ce cas vous disposez d'un crédit d'heures, que vous pouvez récupérer,
- le nombre d'heures prestées est inférieur au nombre d'heures comprises dans la période de référence, dans ce cas vous disposez d'un débit d'heures, que vous devez prester.

Des limites doivent toutefois être mentionnées :

- le crédit d'heures ne peut dépasser 5 heures en fin de mois. Sauf accord spécial de la hiérarchie, le supplément de crédit sera perdu. Les employé(e)s pourront récupérer leur crédit, une demi-journée par mois. La date est à fixer en accord avec le responsable hiérarchique.
- le débit d'heures ne peut être supérieur à 5 heures en fin de mois. Le cas échéant, il devra être résorbé le mois suivant. Un débit d'heures de moins de 5 heures, ne peut être admis plus de deux mois consécutifs. Pour résorber un débit, il faut accroître les prestations journalières dans les plages variables. Tout manquement constitue une faute.

a.4. Absences - Congés - Récupérations :

Toutes absences doivent être signalées immédiatement au Service du Personnel, qui en avertira le responsable. Celui-ci fera parvenir dans les plus brefs délais une fiche d'absence, reprenant le motif et la durée probable de l'absence. Pendant l'absence, le compte est alimenté manuellement par le Service du Personnel à raison d'une durée journalière normale (7h38' du lundi au jeudi et 5h58' le vendredi).

Toutes demandes de congé doivent parvenir au Service du Personnel trois jours avant le début du congé. La procédure est la même que pour les absences.

Toutes récupérations doivent être signalées au Service du Personnel par le responsable hiérarchique au moyen du formulaire adéquat et dans les plus brefs délais.

b. Les fonctionnalités attendues du système

b.1. Enregistrement arrivée normale

Objectif : L'objectif de cette fonctionnalité consiste à recevoir l'information d'arrivée d'un membre du personnel ne revenant pas de mission.

Message d'entrée : { matricule, 'IN' }

Message de sortie : { temps cumulé global }

b.2. Enregistrement arrivée mission

Objectif : L'objectif de cette fonctionnalité consiste à recevoir l'information d'arrivée d'un membre du personnel revenant de mission et de mettre à jour le champ "est en mission" de la base de donnée.

Message d'entrée : { matricule, 'IN-miss' }

Message de sortie : { temps cumulé global }

b.3. Enregistrement sortie normale

Objectif : L'objectif de cette fonctionnalité consiste à recevoir l'information de sortie d'un membre du personnel ne partant pas en mission.

Message d'entrée : { matricule, 'OUT' }

Message de sortie : { temps cumulé global }

b.4. Enregistrement sortie mission

Objectif : L'objectif de cette fonctionnalité consiste à recevoir l'information d'arrivée d'un membre du personnel revenant de mission et de mettre à jour le champ "est en mission" de la base de donnée.

Message d'entrée : { matricule, 'OUT-miss' }

Message de sortie : { temps cumulé global }

b.5. Enregistrement manuel de prestation

Objectif : Etablir une fonctionnalité dont dispose le Service du Personnel afin de mettre manuellement à jour le temps cumulé.

Message d'entrée : { matricule, décompte temps }

b.6. Cumul des heures prestées

Objectif : Etablir une fonctionnalité qui pour les matricules n'ayant pas été inscrit dans la base de données comme absent ou en congé décompte le nombre de temps presté durant la journée, la soustrait au temps normal à prester en ce jour (temps qui indique la parfaite répartition du temps à prester pour les jours d'un mois afin d'arriver au décompte d'heures devant être effectuées) et en crédite ou débite, cela dépend du solde, le temps cumulé global. Il s'agit également de mettre à jour, pour ces membres du personnel, les champs "Temps cumulé mois antérieur" et "Temps total hebdomadaire" de la base de données. Ayant calculé le total presté par chaque membre, il faudra introduire le matricule des membres du personnel n'ayant pas respecté la durée de pause de midi ou la durée maximale de travail d'une journée respectivement dans "Cal-pause" et "Cal-cumul".

b.7. Réduction cumul mensuel

Objectif : Cette fonctionnalité a pour but de ramener le temps cumulé à 5h00' pour tous les membres du personnel ayant un solde établit supérieurement à ces 5h00'.

b.8. Contrôle de légalité journée

Objectif : Le but est d'analyser pour les membres du personnel ceux qui n'ont pas respecté les réglementations en matière d'horaire journalier :

- entrée / sortie effectuées dans les plages fixes,
- entrée avant 8h00'
- sortie après 18h00' du lundi au jeudi et après 17h00' le vendredi,
- cumul de plus de 9h00',
- plage de midi plus grande que 45',
- absence non justifiée.

afin d'en établir la liste accompagniée de la justification.

Message de sortie : { matricule, respect plage fixe, entrée trop tôt, entrée trop tard, respect midi, cumul trop grand }

b.9. Contrôle de légalité semaine

Objectif : Le but est d'analyser pour les membres du personnel ceux qui n'ont pas respecté les réglementations en matière d'horaire hebdomadaire :

- cumul hebdomadaire de plus de 40 heures.

Message de sortie : { matricule, temps cumulé hebdomadaire }

b.10. Gestion déficience mensuelle

Objectif : Le but est d'analyser pour les membres du personnel ceux qui n'ont pas respecté les réglementations en matière de cumul de prestation mensuelle :

- cumul négatif global de plus de 5 heures,
- cumul négatif global consécutif de plus de 5 heures.

Message de sortie : { matricule, temps cumulé global, consécutif }

b.11. Gestion des absences - Congés - Récupération

Objectif : Le but consiste à introduire dans la base de données les demandes de congés, de récupérations ainsi que les justificatifs d'absences. Ceci sera utile notamment pour le calcul du cumul horaire afin de ne pas débiter le temps aux personnes dont l'absence est justifiée.

Message d'entrée : { matricule, motif, dates }

b.12. Traitement des absences non justifiées

Objectif : Le but est de retirer le temps normal à prester en ce jour du temps cumulé global pour l'ensemble des membres du personnel faisant l'objet d'une absence non justifiée. (Toute justification postérieure à la date d'absence injustifiée fera l'objet d'une rectification de temps cumulé global manuel grâce à la fonctionnalité Enregistrement manuel de prestation). Il s'agit également de mettre à jour, pour ces membres du personnel, les champs "Temps cumulé mois antérieur" et "Temps total hebdomadaire" de la base de données.

b.13. Traitement des congés et absences justifiées

Objectif : Etablir une fonctionnalité qui pour les matricules ayant été inscrit dans la base de données comme absent justifié ou en congé met à jour les champs "Temps cumulé mois antérieur" et "Temps total hebdomadaire" de la base de données en considérant le temps normal à prester en ce jour comme référence pour le crédit d'heure.

c. Les contraintes non-fonctionnelles liées au système

c.1. Le contrôle d'accès - sécurité :

Un système d'accès centralisé :

Les zones qui doivent faire l'objet d'un filtrage d'accès sont :

- les entrées du garage (niveau -1 et niveau -2),
- la sortie par le hall.
- l'entrée du personnel

Il faut en outre que les enregistrements d'horaire fassent l'objet d'une protection afin d'assurer le bon fonctionnement de la fonction de contrôle de l'horaire. De plus, le système pour lequel on optera ne devra pas, pour des raisons de sécurité physique (et légale), être une entrave à l'évacuation des locaux en cas de problèmes.

c.2. La performance requise :

Il s'agira d'assurer que le système soit suffisamment efficace que pour éviter toute formation de file d'attente au dispositif de pointage. Le système devra également permettre une bonne performance d'évacuation du batîment.

d. Les exigences organisationnelles

d.1. Accès exceptionnel :

Pour obtenir un accès en dehors des plages normales, la demande signée par le responsable de département devra parvenir, 48 heures à l'avance, au responsable de la sécurité. Celui-ci comme au paravant assurera la sécurité de nuit et verrouillera toutes les portes manuellement 30 minutes après la fin des heures de bureau. Toute personne bloqué après cette heure devra se justifier auprès du responsable de la sécurité qui la raccompagnera à la sortie. Le membre du personnel ne devra toutefois pas omettre de pointer sa sortie.

4.3. Formalisation des exigences logicielles

A ce stade, nous allons développer toute la formalisation des exigences fonctionnelles qui se rattachent à la spécification du niveau système. A cette fin nous allons utiliser le language Albert de spécification. Ce language est de type déclaratif, c'est-à-dire qu'il décrit les propriétés attendues de l'application. Une présentation du language est développée dans la partie des annexes.

Normalement, le processus de spécification fonctionnel ne doit pas être perçu comme se déroulant avant le raffinement des exigences non-fonctionnelles. Les deux processus peuvent se développer en parallèle. Mais ici, il est jugé plus intéressant de réintroduire les implications venues du raffinement des exigences non-fonctionnelles par après pour bien réaliser l'influence de ces dernières sur les autres niveaux du développement du projet.

a. Déclaration des types de données :

TYPETAT = ENUM [Libre, Occupé]

BOUTON = ENUM [Poussé , Relaché]

ETATPORTE = ENUM [Ouverte, Fermée]

D-BASE = TABLE [Matricule : INTEGER
→ CP [Nom : STRING ,
Prénom : STRING ,
Temps-cumulé-global : INTEGER ,
Est-en-mission : BOOLEAN ,
Mouvement : MOUV ,
Congé-date : SDATE ,

Absence-date : SABS ,
Récup-date : SREC ,
Temps-total-hebdomadaire : INTEGER ,
Temps-cumulé-mois-antérieur : INTEGER]

TEMPS = CP [Heure : INTEGER, Minute : INTEGER, Seconde : INTEGER]

MOUV = SEQ [Pointage : POINT]

DATE = CP [Jour : INTEGER, Mois : INTEGER, Année : INTEGER]

ABS = CP [Journée : DATE, Justification : BOOLEAN]

REC = CP [Journée : DATE]

POINT = CP [Moment : TEMPS, Action : ACT]

ACT = ENUM [In, In-miss, Out, Out-miss]

SDATE = SET [DATE]

SABS = SET [ABS]

SREC = SET [REC]

MESSOUT1 = SEQ (CP (Matricule : INTEGER,
Temps-cumulé-global : TEMPS,
Consécutif : BOOLEAN))

MESSOUT2 = SEQ (CP (Matricule : INTEGER,
Temps-cumulé-hebdomadaire : TEMPS))

MESSOUT3 = TABLE [Matricule : INTEGER
→ CP [Respect-plage-fixe : BOOLEAN,
Entrée-trop-tôt : BOOLEAN,
Sortie-trop-tard : BOOLEAN,
Respect-plage-midi : BOOLEAN,
Cumul-respecté : BOOLEAN,
Absence-non-justifiée : BOOLEAN]]

CUMUL = SET (Matricule : INTEGER)

TOTPREST = TABLE [Matricule : INTEGER → Presté : INTEGER]

TYPEMOTIF = ENUM [C, R, A]

TYPEDATES = SET [DATE]

DATE = CP [Jour : INTEGER , Mois : INTEGER , Année : INTEGER]

b. Déclaration des opérations :

AssignerPorteParking : POINTEUSE PARKING → PORTE GARAGE ;

| Nous indiquons par le biais d'une opération, le fait que une seule porte est associée à une seule pointeuse. Ceci
| n'est pas très élégeant mais il est difficile de gérer autrement les contraintes d'ordre globales avec ALBERT.

AssignerPorteParking (p) = val
with
¬ ∃ p' : (p ≠ p')
∧ (AssignerPorteParking (p') = val)

CréeInter1 : D-BASE x INTEGER x INTEGER x INTEGER → D-BASE ;

| Le but consiste à générer une base de données *O* reprenant toutes les instances de la base *Int* dont les membre
| du personnel représentés non pas fait de mouvement en ce jour et ne sont ni en mission, ni en absence
| justifiée.

CréeInter1 (Int, J, M, A) = O
with
∀ i : In (i, O)
⇒ ∃ j : In (j, Int)
∧ ¬(< <J, M, A>, TRUE > ∈ Sel7 (Int[j]))
∧ Est-en-mission (Int[j]) = FALSE
∧ ¬(<J, M, A> ∈ Congé-date (Int[j]))
∧ j = i
∧ Empty? (Mouvement (O[j]))
∧ Int[j] = O[i]
∧ ∀ j : In (j, Int)
∧ ¬(< <J, M, A>, TRUE > ∈ Sel7 (Int[j]))
∧ Est-en-mission (Int[j]) = FALSE
∧ ¬(<J, M, A> ∈ Congé-date (Int[j]))
∧ Empty? (Mouvement (Int[j]))
⇒ ∃ i : In (i, O)
∧ Int[j] = O[i]
∧ i = j

CréeInter2 : D-BASE → D-BASE ;

| On met dans *O* tout les enregistrements de *I* caractérisés par des membres du personnel qui ont soit fait un
| mouvement en ce jour, soit qui étaient en mission toute la journée.

CréeInter2 (I) = O
with
∀ i : In (i, I)
∧ (¬Empty? (Mouvement (I[i]))
∨ Est-en-mission (I[i]) = TRUE)
⇒ ∃ j : In (j, O)
∧ i = j
∧ O[j] = I[i]

∧ ∀ j : In (j, O)
⇒ ∃ i : In (i, I)
∧ i = j
∧ (¬Empty? (Mouvement (I[i]))
∨ Est-en-mission (I[i]) = TRUE)
∧ O[j] = I[i]

CréeInter3 : D-BASE x INTEGER x INTEGER x INTEGER → D-BASE ;

| On indique dans *O* les membres du personnel inscris dans I qui sont en congé ou en absence justifiée.

CréeInter3 (I, J, M, A) = O
with
∀ i : In (i, O)
⇒ ∃ j : In (j, I)
∧ (< <J, M, A>, TRUE > ∈ Absence-date (I[j])
∨ <J, M, A> ∈ Congé-date (I[j]))
∧ j = i
∧ I[j] = O[i]

∧ ∀ j : In (j, I)
∧ (< <J, M, A>, TRUE > ∈ Absence-date (I[j])
∨ <J, M, A> ∈ Congé-date (I[j]))
⇒ ∃ i : In (i, O)
∧ I[j] = O[i]
∧ i = j

EnregistreManuel : D-BASE x INTEGER x INTEGER x INTEGER x INTEGER → D-BASE ;

| On enregistre manuellement le Temps-cumulé-global à partir d'une heure *h*, d'une minute *m* et d'une seconde
| *s*. Ceci pour une personne identifiée par son matricule *mat* et enregistrée dans la base de données *I*.

EnregistreManuel (I, mat, h, m, s) = O

with
O = Modify (I, mat, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens') = Temps-total-hendomadaire (rens)
∧ Temps-cumulé-mois-antérieur (rens') = Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens') = (h * 3600) + (m * 60) + s
∧ rens = I[mat]

FaireUnion : MESSOUT3 x MESSOUT3 → MESSOUT3 ;

| Le but consiste à fusionner deux tableaux.

FaireUnion (I1, I2) = O
with
∀ i : In (i, O)
⇒ ∃ j : (O[i] = I1[j] ∧ i = j)
∨ (O[i] = I2[j] ∧ i = j)

∧ ∀ i : In (i, I1)
⇒ ∃ j : (O[j] = I1[i] ∧ i = j)

∧ ∀ i : In (i, I2)
⇒ ∃ j : (O[j] = I1[i] ∧ i = j)

GénèreHeure : D-BASE x INTEGER → INTEGER ;

| Le but consiste à extraire pour un enregistrement, identifié par *mat* dans la base de données *I*, le nombre
| d'heures qui sont contenues dans Temps-cumulé-global.

GénèreHeure (I, mat) = O
with
∃ i : In (i , I)
∧ i = mat
∧ Temps-cumulé-global (I[i]) ≥ 0
⇒ O = [(Temps-cumulé-global (I[i]) - GénèreSeconde (I, mat)) / 60
- GénèreMinute (I, mat)] / 60

∨ ∃ i : In (i , I)
∧ i = mat
∧ Temps-cumulé-global (I[i]) < 0
⇒ O = [(Temps-cumulé-global (I[i]) + GénèreSeconde (I, mat)) / 60
+ GénèreMinute (I, mat)] / 60

GénèreMinute : D-BASE x INTEGER → INTEGER ;

| Le but consiste à extraire pour un enregistrement, identifié par *mat* dans la base de données *I*, le nombre de
| minutes qui sont contenues dans Temps-cumulé-global.

GénèreMinute (I, mat) = O
with
∃ i : In (i , I)
∧ i = mat
∧ Temps-cumulé-global (I[i]) ≥ 0
⇒ O = [(Temps-cumulé-global (I[i])
- GénèreSeconde (I, mat)) / 60] mod 60

∨ ∃ i : In (i , I)
∧ i = mat
∧ Temps-cumulé-global (I[i]) < 0
⇒ O = [(Temps-cumulé-global (I[i])
+ GénèreSeconde (I, mat)) / 60] mod 60

GénèreSeconde : D-BASE x INTEGER → INTEGER ;

| Le but consiste à extraire pour un enregistrement, identifié par *mat* dans la base de données *I*, le nombre de
| secondes qui sont contenues dans Temps-cumulé-global si l'on exprime cette valeur de Temps-cumulé-global
| en heures/minutes/secondes.

GénèreSeconde (I, mat) = O
with
∃ i : In (i , I)
∧ i = mat
⇒ O = Temps-cumulé-global (I[i]) mod 60

GestACR : D-BASE x INTEGER x TYPEMOTIF x TYPEDATES → D-BASE ;

| Pour un membre du personnel identifié par *mat* et enregistré dans la base de données *I*, on enregistre la (les)
| date(s) *d* dans Congé-date, Récup-date ou Absence-date d'après le motif *motif*..

GestACR (I, mat, modif, d) = O

with

$\exists$ i : In (i , I)

$\wedge$ i = mat

$\wedge$ modif = C

$\Rightarrow$ $\forall$ j : (j $\geq$ 1)

(j $\leq$ Length (d))

$\Rightarrow$ O = Modify (I, i, rens')

$\wedge$ Nom (rens') = Nom (rens)

$\wedge$ Prénom (rens') = Prénom (rens)

$\wedge$ Est-en-mission (rens') = Est-en-mission (rens)

$\wedge$ Mouvement (rens') = Mouvement (rens)

$\wedge$ Congé-date (rens') = Congé-date (rens) $\cup$ S_j (d)

$\wedge$ Absence-date (rens') = Absence-date (rens)

$\wedge$ Récup-date (rens') = Récup-date (rens)

$\wedge$ Temps-total-hebdomadaire (rens')

= Temps-total-hendomadaire (rens)

$\wedge$ Temps-cumulé-mois-antérieur (rens')

= Temps-cumulé-mois-antérieur (rens)

$\wedge$ Temps-cumulé-global (rens')

= Temps-cumulé-global (rens)

$\wedge$ rens = I[mat]

$\exists$ i : In (i , I)

$\wedge$ i = mat

$\wedge$ modif = R

$\Rightarrow$ $\forall$ j : (j $\geq$ 1)

(j $\leq$ Length (d))

$\Rightarrow$ O = Modify (I, i, rens')

$\wedge$ Nom (rens') = Nom (rens)

$\wedge$ Prénom (rens') = Prénom (rens)

$\wedge$ Est-en-mission (rens') = Est-en-mission (rens)

$\wedge$ Mouvement (rens') = Mouvement (rens)

$\wedge$ Congé-date (rens') = Congé-date (rens)

$\wedge$ Absence-date (rens') = Absence-date (rens)

$\wedge$ Récup-date (rens') = Récup-date (rens) $\cup$ S_j (d)

$\wedge$ Temps-total-hebdomadaire (rens')

= Temps-total-hendomadaire (rens)

$\wedge$ Temps-cumulé-mois-antérieur (rens')

= Temps-cumulé-mois-antérieur (rens)

$\wedge$ Temps-cumulé-global (rens')

= Temps-cumulé-global (rens)

$\wedge$ rens = I[mat]

$\exists$ i : In (i , I)
$\wedge$ i = mat
$\wedge$ modif = A
$\Rightarrow$ $\forall$ j : (j $\geq$ 1)
(j $\leq$ Length (d))
$\Rightarrow$ O = Modify (I, i, rens')
$\wedge$ Nom (rens') = Nom (rens)
$\wedge$ Prénom (rens') = Prénom (rens)
$\wedge$ Est-en-mission (rens') = Est-en-mission (rens)
$\wedge$ Mouvement (rens') = Mouvement (rens)
$\wedge$ Congé-date (rens') = Congé-date (rens)
$\wedge$ Absence-date (rens')
= Absence-date (rens) $\cup$ < S_j (d), TRUE >
$\wedge$ Récup-date (rens') = Récup-date (rens)
$\wedge$ Temps-total-hebdomadaire (rens')
= Temps-total-hendomadaire (rens)
$\wedge$ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
$\wedge$ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
$\wedge$ rens = I[mat]

GestCLS : D-BASE $\rightarrow$ MESSOUT2 ;

| Le but consiste à répertorier toutes les personnes enregistrées dans *I* et dont Temps-total-hebdomadaire
| dépasse la limite des 40 heures légales.

GestCLS (I) = O
with
$\forall$ i : In (i, I)
$\wedge$ Temps-total-hebdomadaire (I[i]) $\geq$ 144000
$\Rightarrow$ $\exists$ j : (j $\geq$ 0) $\wedge$ (j $\leq$ Length (O))
$\wedge$ Matricule (S_j (O)) = i
$\wedge$ Seconde (Temps-cumulé-hebdomadaire (S_j (O)))
= Temps-total-hebdomadaire (I[i]) mod 60
$\wedge$ Minute (Temps-cumulé-hebdomadaire (S_j (O)))
= **[** [Temps-total-hebdomadaire (I[i])
- Seconde (Temps-cumulé-hebdomadaire (S_j (O)))]
/ 60 **]** mod 60
$\wedge$ Heure (Temps-cumulé-hebdomadaire (S_j (O)))
= **[[** [Temps-total-hebdomadaire (I[i])
- Seconde (Temps-cumulé-hebdomadaire (S_j (O)))]

/ 60] - Minute (Temps-cumulé-hebdomadaire (S_j (O))
)] / 60

∧ ∀ j : (j ≥ 0) ∧ (j ≤ Length (O))
⇒ ∃ i : In (i, I)
∧ Matricule (S_j (O)) = i
∧ Temps-total-hebdomadaire (I[i]) ≥ 144000
∧ Seconde (Temps-cumulé-hebdomadaire (S_j (O)))
= Temps-total-hebdomadaire (I[i]) mod 60
∧ Minute (Temps-cumulé-hebdomadaire (S_j (O)))
= [[Temps-total-hebdomadaire (I[i])
- Seconde (Temps-cumulé-hebdomadaire (S_j (O)))]
/ 60] mod 60
∧ Heure (Temps-cumulé-hebdomadaire (S_j (O)))
= [[[Temps-total-hebdomadaire (I[i])
- Seconde (Temps-cumulé-hebdomadaire (S_j (O)))]
/ 60] - Minute (Temps-cumulé-hebdomadaire (S_j (O))
)] / 60

GestDef : D-BASE → MESSOUT1 ;

| Le but consiste à générer une séquence reprenant le Matricule, le Temps-cumulé-global et le caractère
| consécutif de la présence d'un déficite horaire pour chaque instance de la base de données. Ce déficite est
| caractérisé par une valeur -18000 ou plus grande pour le Temps-cumulé-global.

GestDef (I) = O
with
(∀ i : In (i, I) ∧ (Temps-cumulé-global (I[i])) ≤ -18000
∧ (Temps-cumulé-mois-antérieur (I[i])) ≤ -18000
⇒ ∃ j : Matricule (S_j (O)) = i
∧ Seconde (Temps-cumulé-global (S_j (O))) = GénèreSeconde (I, i)
∧ Minute (Temps-cumulé-global (S_j (O))) = GénèreMinute (I, i)
∧ Heure (Temps-cumulé-global (S_j (O))) = GénèreHeure (I, i)
∧ Consécutif (S_j (O)) = TRUE
∧ ∀ j : (j ≥ 0) ∧ (Length (O) ≥ j)
⇒ ∃ i : In (i, I)
∧ Matricule (S_j (O)) = i
∧ Seconde (Temps-cumulé-global (S_j (O))) = GénèreSeconde (I, i)
∧ Minute (Temps-cumulé-global (S_j (O))) = GénèreMinute (I, i)
∧ Heure (Temps-cumulé-global (S_j (O))) = GénèreHeure (I, i)
∧ Temps-cumulé-global (I [i]) ≤ -18000
)

$\wedge$ ($\forall$ i : In (i, I) $\wedge$ (Temps-cumulé-global (I[i])) $\leq$ -18000
$\wedge$ (Temps-cumulé-mois-antérieur (I[i])) > -18000
$\Rightarrow$ $\exists$ j : Matricule (S_j (O)) = i
$\wedge$ Seconde (Temps-cumulé-global (S_j (O))) = GénèreSeconde (I, i)
$\wedge$ Minute (Temps-cumulé-global (S_j (O))) = GénèreMinute (I, i)
$\wedge$ Heure (Temps-cumulé-global (S_j (O))) = GénèreHeure (I, i)
$\wedge$ Consécutif (S_j (O)) = FALSE
$\wedge \forall$ j : (j $\geq$ 0) $\wedge$ (Length (O) $\geq$ j)
$\Rightarrow$ $\exists$ i : In (i, I)
$\wedge$ Matricule (S_j (O)) = i
$\wedge$ Seconde (Temps-cumulé-global (S_j (O))) = GénèreSeconde (I, i)
$\wedge$ Minute (Temps-cumulé-global (S_j (O))) = GénèreMinute (I, i)
$\wedge$ Heure (Temps-cumulé-global (S_j (O))) = GénèreHeure (I, i)
$\wedge$ Temps-cumulé-global (I [i]) $\leq$ -18000
)

InMiss : D-BASE x INTEGER x INTEGER x INTEGER x INTEGER $\rightarrow$ D-BASE ;

| Pour une instance de la base de données *I* déterminée par le matricule *mat* de la personne concernée et une
| heure *h*, minute *m*, seconde *s*, le but est d'enregistrer ce temps comme moment d'arrivée de mission.

InMiss (I, mat, h, m, s) = O
with
$\exists$ i : In (i, I)
$\wedge$ i = mat
$\Rightarrow$ O = Modify (I, mat, rens')
$\wedge$ Nom (rens') = Nom (rens)
$\wedge$ Prénom (rens') = Prénom (rens)
$\wedge$ Est-en-mission (rens') = FALSE
$\wedge$ Mouvement (rens')
= Append (Mouvement (rens), < <h, m, s>, In-miss >)
$\wedge$ Congé-date (rens') = Congé-date (rens)
$\wedge$ Absence-date (rens') = Absence-date (rens)
$\wedge$ Récup-date (rens') = Récup-date (rens)
$\wedge$ Temps-total-hebdomadaire (rens')
= Temps-total-hendomadaire (rens)
$\wedge$ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
$\wedge$ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
$\wedge$ rens = I[mat]

InNormal : D-BASE x INTEGER x INTEGER x INTEGER x INTEGER → D-BASE ;

| Pour une instance de la base de données *I* déterminée par le matricule *mat* de la personne concernée et une | heure *h*, minute *m*, seconde *s*, le but est d'enregistrer ce temps comme moment d'arrivée normale.

InNormal (I, mat, h, m, s) = O
with
∃ i : In (i, I)
∧ i = mat
⇒ O = Modify (I, mat, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens')
= Append (Mouvement (rens), < <h, m, s>, In >)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hendomadaire (rens)
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
∧ rens = I[mat]

InscrireAInJ : D-BASE x D-BASE x INTEGER x INTEGER x INTEGER → D-BASE ;

| Pour toutes les personnes enregistrées dans *Int*, on indique dans la base de données *I* (pour le jour J, le mois | *M* et l'année *A*) qu'ils ont fait l'objet d'une absence injustifiée.

InscrireAInJ (I, Int, J, M, A) = O
with
∀ i : In (i, Int)
⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens')
= Add (Absence-date (rens), < <J, M, A>, FALSE >)
∧ Récup-date (rens') = Récup-date (rens)

∧ Temps-total-hebdomadaire (rens')
= Temps-total-hendomadaire (rens)
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens') = Temps-cumulé-global (rens)
∧ rens = I[i]

OutMiss : D-BASE x INTEGER x INTEGER x INTEGER x INTEGER → D-BASE ;

| Pour une instance de la base de données *I* déterminée par le matricule *mat* de la personne concernée et une
| heure *h*, minute *m*, seconde *s*, le but est d'enregistrer ce temps comme moment de sortie mission.

OutMiss (I, mat, h, m, s) = O
with
∃ i : In (i, I)
∧ i = mat
⇒ O = Modify (I, mat, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = TRUE
∧ Mouvement (rens')
= Append (Mouvement (rens), < <h, m, s>, Out-miss >)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hendomadaire (rens)
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
∧ rens = I[mat]

OutNormal : D-BASE x INTEGER x INTEGER x INTEGER x INTEGER → D-BASE ;

| Pour une instance de la base de données *I* déterminée par le matricule *mat* de la personne concernée et une
| heure *h*, minute *m*, seconde *s*, le but est d'enregistrer ce temps comme moment de sortie normale.

OutNormal (I, mat, h, m, s) = O
with
∃ i : In (i, I)
∧ i = mat

⇒ O = Modify (I, mat, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens')
= Append (Mouvement (rens), < <h, m, s>, Out >)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hendomadaire (rens)
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
∧ rens = I[mat]

RédCumul : D-BASE → D-BASE ;

| RédCumul ramène Temps-cumulé-global à 18000 (secondes) pour tous les enregistrements dont le Temps-
| cumulé-global dépasse cette valeur.

RédCumul (I) = O
with
∀ i : In (i, I)
∧ (Temps-cumulé-global (I[i]) ≥ 18000)
⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hendomadaire (rens)
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens') = 18000
∧ rens = I[i]

RetraitMiss : D-BASE x D-BASE → D-BASE ;

| Le but consite à retirer pour toutes les personnes enregistrées dans *I*, tous les mouvement de rentrée ou de
| sortie mission.

RetraitMiss (Int, I) = O
with
∀ i, j, k : In (i, I)
∧ ¬Empty? (Mouvement (I[i]))
∧ (k ≥ 1)
∧ (k ≤ Length (Mouvement (I[i]))
∧ (j ≥ 1)
∧ (j ≤ Length (Mouvement (I[i]))
∧ Action (S_k (Mouvement (I[i]))) = In-miss
∧ Action (S_k (Mouvement (I[i]))) = Out-miss
⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens')
= Remove-ith (Remove-ith (Mouvement (rens), k), j)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hendomadaire (rens)
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens') = Temps-cumulé-global (rens)
∧ rens = I[i]

TraiteCC : TOTPREST x INTEGER → CUMUL ;

| On indique dans *O* les I dont le cumul de travail de la journée dépasse le maximum légal.

TraiteCC (P, J) = O
with
(∀ i : In (i, P)
∧ J = 6
∧ P[i] > 21480
⇒ i ⊂ O

∨ ∀ i : In (i, P)
∧ (J ≠ 6

$\vee$ J ≠ 7
$\vee$ J ≠ 1)
$\wedge$ P[i] > 27480
$\Rightarrow$ i $\subset$ O
)

$\wedge$
($\forall$ i : i $\subset$ O
$\Rightarrow$ In (i, P)
$\wedge$ J = 6
$\wedge$ P[i] > 21480

$\vee$ $\forall$ i : i $\subset$ O
$\Rightarrow$ In (i, P)
$\wedge$ (J ≠ 6
$\vee$ J ≠ 7
$\vee$ J ≠ 1)
$\wedge$ P[i] > 21480
)

TraiteCJ : CUMUL $\rightarrow$ MESSOUT3 ;

| Le but but consiste à créer *O* qui est *Int* ajouté de la liste des personnes enregistrées dans *C* comme n'ayant par
| respecté le maximum de du temps de travail imparti pour la journée.

TraiteCJ (C) = O
with
$\forall$ i : In (i, O)
$\wedge$ Cumul-respecté (O[i]) = TRUE
$\Rightarrow$ i $\in$ C

$\wedge$ $\forall$ i : i $\in$ C
$\Rightarrow$ In (i, O)
$\wedge$ Cumul-respecté (O[i]) = TRUE

TraiteCP : D-BASE x D-BASE $\rightarrow$ CUMUL ;

| Le but consiste à mettre dans *O* toutes les I qui n'étaient pas inscrites dans *I* comme étant en mission
| pour toute la journée et dont le temps de pause calculé dans *Int* est trop petit.

TraiteCP (Int, I) = O
with
$\forall$ i : In (i, Int)

$\wedge \neg$Empty? (Mouvement (I[i]))
$\wedge$ Length (Mouvement (Int[i])) = 4
$\wedge$ [(Heure (Moment (S_3 (Mouvement (Int[i])))) * 3600
+ Minute (Moment (S_3 (Mouvement (Int[i])))) * 60
+ Seconde (Moment (S_3 (Mouvement (Int[i])))))
- (Heure (Moment (S_2 (Mouvement (Int[i])))) * 3600
+ Minute (Moment (S_2 (Mouvement (Int[i])))) * 60
+ Seconde (Moment (S_2 (Mouvement (Int[i])))))] < 2700
$\Rightarrow$ i $\subset$ O

$\wedge$ $\forall$ i : i $\subset$ O
$\Rightarrow$ In (i, Int)
$\wedge \neg$Empty? (Mouvement (I[i]))
$\wedge$ Length (Mouvement (Int[i])) = 4
$\wedge$ [(Heure (Moment (S_3 (Mouvement (Int[i])))) * 3600
+ Minute (Moment (S_3 (Mouvement (Int[i])))) * 60
+ Seconde (Moment (S_3 (Mouvement (Int[i])))))
- (Heure (Moment (S_2 (Mouvement (Int[i])))) * 3600
+ Minute (Moment (S_2 (Mouvement (Int[i])))) * 60
+ Seconde (Moment (S_2 (Mouvement (Int[i])))))] < 2700

TraiteETT : D-BASE $\rightarrow$ MESSOUT3 ;

| Le but consiste à créer *O* qui est *Int* ajouté de la liste des personnes enregistrées dans *I* comme étant arrivées
| trop tôt durant la journée (avant 8h00').

TraiteETT (I) = O
with
$\forall$ i : In (i, O)
$\wedge$ Entrée-trop-tôt (O[i]) = TRUE
$\Rightarrow$ $\exists$ j : In (j, I)
$\wedge$ i = j
$\wedge$ $\exists$ k : (k $\geq$ 0) $\wedge$ (k $\leq$ Length (Mouvement (I[j])))
$\Rightarrow$ Heure (Moment (S_k (Mouvement (I[j])))) < 8

$\wedge$ $\forall$ j, k : In (j, I)
$\wedge$ (k $\geq$ 0) $\wedge$ (k $\leq$ Length (Mouvement (I[j])))
$\wedge$ Heure (Moment (S_k (Mouvement (I[j])))) < 8
$\Rightarrow$ $\exists$ i : In (i, O)
$\wedge$ i = j
$\wedge$ Entrée-trop-tôt (O[i]) = TRUE

TraiteLANJ : D-BASE x INTEGER x INTEGER x INTEGER → MESSOUT3 ;

| Le but consiste à reprendre dans *O* la liste des personnes enregistrées dans *I* comme étant en absence
| injustifiée pour ce jour.

TraiteLANJ (I, J, M, A) = O
with
∀ i : In (i, O)
⇒ ∃ j : In (j, I)
∧ i = j
∧ < <J, M, A>, FALSE > ∈ Absence-date (I[j])
∧ Absence-non-justifiée (O[i]) = TRUE

∧ ∀ j : In (j, I)
∧ < <J, M, A>, FALSE > ∈ Absence-date (I[j])
⇒ ∃ i : In (i, O)
∧ i = j
∧ Absence-non-justifiée (O[i]) = TRUE

TraitePFR : D-BASE x INTEGER → MESSOUT3 ;

| Le but consiste à créer *O* qui est *Int* ajouté de la liste des personnes enregistrées dans *I* comme ayant fait un
| mouvement dans les plages fixes.

TraitePFR (I, J) = O
with
∀ i : In (i, O) ∧ Respect-plage-fixe (O[i]) = TRUE
⇒ ∃ j : In (j, I)
∧ i = j
∧ ∃ k : (k ≥ 0) ∧ (k ≤ Length (Mouvement (I[j])))
⇒ (Heure (Moment (S_k (Mouvement (I[j])))) ≥ 8
∧ Heure (Moment (S_k (Mouvement (I[j])))) < 12
∧ ¬(Action (S_k (Mouvement (I[j]))) = IN-miss
∨ Action (S_k (Mouvement (I[j]))) = OUT-miss))
∨ (Heure (Moment (S_k (Mouvement (I[j])))) ≥ 14
∧ Heure (Moment (S_k (Mouvement (I[j])))) < 15
∧ J = 6
∧ ¬(Action (S_k (Mouvement (I[j]))) = IN-miss
∨ Action (S_k (Mouvement (I[j]))) = OUT-miss))
∨ (Heure(Moment (S_k (Mouvement (I[j])))) ≥ 14
∧ Heure (Moment (S_k (Mouvement (I[j])))) < 16
∧ J ≠ 6
∧ J ≠ 7

$\wedge J \neq 1$
$\wedge \neg($ Action (S_k (Mouvement (I[j]))) = IN-miss
$\vee$ Action (S_k (Mouvement (I[j]))) = OUT-miss))

$\wedge$ $\forall$ j, k : In (j, I)
$\wedge$ (k $\geq$ 0) $\wedge$ (k $\leq$ Length (Mouvement (I[j])))
$\wedge$ ((Heure (Moment (S_k (Mouvement (I[j])))) $\geq$ 8
$\wedge$ Heure (Moment (S_k (Mouvement (I[j])))) < 12
$\wedge \neg($ Action (S_k (Mouvement (I[j]))) = IN-miss
$\vee$ Action (S_k (Mouvement (I[j]))) = OUT-miss))
$\vee$ (Heure (Moment (S_k (Mouvement (I[j])))) $\geq$ 14
$\wedge$ Heure (Moment (S_k (Mouvement (I[j])))) < 15
$\wedge J = 6$
$\wedge \neg($ Action (S_k (Mouvement (I[j]))) = IN-miss
$\vee$ Action (S_k (Mouvement (I[j]))) = OUT-miss))
$\vee$ (Heure (Moment (S_k (Mouvement (I[j])))) $\geq$ 14
$\wedge$ Heure (Moment (S_k (Mouvement (I[j])))) < 16
$\wedge J \neq 6$
$\wedge J \neq 7$
$\wedge J \neq 1$
$\wedge \neg($ Action (S_k (Mouvement (I[j]))) = IN-miss
$\vee$ Action (S_k (Mouvement (I[j]))) = OUT-miss)))

$\Rightarrow \exists$ i : In (i, O)
$\wedge$ Respect-plage-fixe (O[i]) = TRUE
$\wedge$ i = j

TraiteRM : CUMUL $\rightarrow$ MESSOUT3 ;

| Le but but consiste à créer *O* qui est *Int* ajouté de la liste des personnes enregistrées dans *C* comme n'ayant par
| respecté le minimum de 45' pour la pause de midi.

TraiteRM (C) = O
with
$\forall$ i : In (i, O)
$\wedge$ Respect-plage-midi (O[i]) = TRUE
$\Rightarrow$ i $\in$ C

$\wedge$ $\forall$ i : i $\in$ C
$\Rightarrow$ In (i, O)
$\wedge$ Respect-plage-midi (O[i]) = TRUE

TraiteSTT : D-BASE x INTEGER → MESSOUT3 ;

| Le but but consiste à créer *O* qui est *Int* ajouté de la liste des personnes enregistrées dans *I* comme étant sorties
| trop tard durant la journée (après 18h00' du lundi au jeudi et après 17h00' le vendredi).

TraiteSTT (I, J) = O
with
∀ i : In (i, O)
∧ Sortie-trop-tard (O[i]) = TRUE
⇒ ∃ j : In (j, I)
∧ i = j
∧ ∃ k : (k ≥ 0) ∧ (k ≤ Length (Sel5 (I[j])))
⇒ (Heure (Moment (S_k (Mouvement (I[j])))) ≥ 18
∧ J ≠ 6
∧ J ≠ 7
∧ J ≠ 1)
∨ (Heure (Moment (S_k (Mouvement (I[j])))) ≥ 17
∧ J = 6)

∧ ∀ j, k : In (j, I)
∧ (k ≥ 0) ∧ (k ≤ Length (Mouvement (I[j])))
∧ (Heure (Moment (S_k (Mouvement (I[j])))) ≥ 18
∧ J ≠ 6
∧ J ≠ 7
∧ J ≠ 1)
∨ (Heure (Moment (S_k (Mouvement (I[j])))) ≥ 17
∧ J = 6)
⇒ ∃ i : In (i, O)
∧ i = j
∧ Sortie-trop-tard (O[i]) = TRUE

TraiteTCG1 : D-BASE x D-BASE x INTEGER → D-BASE ;

| Pour les personnes enregistrées dans *Int*, on retire au Temps-cumulé-global de leur enregistrement dans *I*, le
| total de 5h58' (21480 secondes) si l'on est un vendredi et 7h38' (27480 secondes) si l'on est lundi, mardi,
| mercredi ou jeudi.

TraiteTCG1 (I, Int, J) = O
with
∀ i : In (i, Int)
∧ J = 6
⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)

∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hendomadaire (rens)
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens) - 21480
∧ rens = I[i]

∨ ∀ i : In (i, Int)
∧ J ≠ 6
∧ J ≠ 7
∧ J ≠ 1
⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hendomadaire (rens)
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens) - 27480
∧ rens = I[i]

TraiteTCG2 : D-BASE x TOTPREST x INTEGER → D-BASE ;

| Pour les personnes identifiées dans *P* et qui par conséquent ont nécessairement pointés dans la journée, on met
| le Temps-cumulé-global à jour dans *I* (en fonction du jour *J*).

TraiteTCG2 (I, P, J) = O
with
∀ i : In (i, P)
∧ J = 6

⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hebdomadaire (rens)
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens) - 21480 + P[i]
∧ rens = I[i]

∨ ∀ i : In (i, P)
∧ J ≠ 6
∧ J ≠ 7
∧ J ≠ 1
⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hebdomadaire (rens)
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens) - 27480 + P[i]
∧ rens = I[i]

TraiteTCMA : D-BASE x D-BASE x INTEGER x INTEGER → D-BASE ;

| Consiste pour les personnes enregistrées dans Int à reporter le Temps-cumulé-global dans le Temps-cumulé-
| mois-antérieur de leur enregistrement dans I. Ceci est fait si l'on est le premier jour du mois (j = 1) ou le
| premier jour ouvrable du mois ((J = 2 ∨ J = 3) ∧ J-sem = 2).

TraiteTCMA (I, Int, J, J-sem) = O

with
∀ i : In (i, Int)
∧ J = 1
∨ (In (i, Int)
∧ [J = 2 ∨ J = 3]
∧ J-sem = 2)
⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hebdomadaire (rens)
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-global (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
∧ rens = I[i]

TraiteTempsPresté : D-BASE x D-BASE → TOTPREST ;

| On calcule le temps presté par les I ayant pointés en ce jour

TraiteTempsPresté (I, Int) = O
with
(∀ i : In (i, Int)
∧ ¬Empty? (Mouvement (I[i]))
∧ Length (Mouvement (Int[i])) = 4
⇒ O[i] = (Heure (Moment (S_4 (Mouvement (Int[i])))) * 3600
+ Minute (Moment (S_4 (Mouvement (Int[i])))) * 60
+ Seconde (Moment (S_4 (Mouvement (Int[i])))))
- (Heure (Moment (S_3 (Mouvement (Int[i])))) * 3600
+ Minute (Moment (S_3 (Mouvement (Int[i])))) * 60
+ Seconde (Moment (S_3 (Mouvement (Int[i])))))
+ (Heure (Moment (S_2 (Mouvement (Int[i])))) * 3600
+ Minute (Moment (S_2 (Mouvement (Int[i])))) * 60
+ Seconde (Moment (S_2 (Mouvement (Int[i])))))
- (Heure (Moment (S_1 (Mouvement (Int[i])))) * 3600
+ Minute (Moment (S_1 (Mouvement (Int[i])))) * 60
+ Seconde (Moment (S_1 (Mouvement (Int[i])))))

∨ ∀ i : In (i, Int)
∧ ¬Empty? (Mouvement (I[i]))
∧ Length (Mouvement (Int[i])) = 2
⇒ O[i] = (Heure (Moment (S_2 (Mouvement (Int[i])))) * 3600
+ Minute (Moment (S_2 (Mouvement (Int[i])))) * 60
+ Seconde (Moment (S_2 (Mouvement (Int[i])))))
- (Heure (Moment (S_1 (Mouvement (Int[i])))) * 3600
+ Minute (Moment (S_1 (Mouvement (Int[i])))) * 60
+ Seconde (Moment (S_1 (Mouvement (Int[i])))))

)

∧ ∀ i : In (i, O)
⇒ In (i, Int)
∧ ¬Empty? (Mouvement (Int[i]))

TraiteTTH1 : D-BASE x D-BASE x INTEGER → D-BASE ;

| Pour les personnes enregistrées dans *Int*, on remet dans *I* le champs Temps-total-hebdomadaire à 0 si l'on est
| un lundi.

TraiteTTH1 (I, Int, J) = O
with
∀ i : In (i, Int)
∧ J = 2
⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= 0
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
∧ rens = I[i]

TraiteTTH2 : D-BASE x D-BASE x INTEGER → D-BASE ;

| Pour les personnes identifiées dans *Int* et qui sont en mission pour la journée, on met le Temps-total-
| hebdomadaire de leur enregistrement dans *I* à jour (en fonction du jour *J*).

TraiteTTH2 (I, Int, J) = O
with
∀ i : In (i, Int)
∧ Est-en-mission (Int[i]) = TRUE
∧ J = 2
⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= 27480
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
∧ rens = I[i]

∨ ∀ i : In (i, Int)
∧ Est-en-mission (Int[i]) = TRUE
∧ J = 6
⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hebdomadaire (rens) + 21480
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
∧ rens = I[i]

∨ ∀ i : In (i, Int)
∧ Est-en-mission (Int[i]) = TRUE
∧ J ≠ 6
∧ J ≠ 2
∧ J ≠ 7
∧ J ≠ 1
⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hebdomadaire (rens) + 27480
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
∧ rens = I[i]

TraiteTTH3 : D-BASE x TOTPREST x INTEGER → D-BASE ;

| Pour les personnes identifiées dans *P* et qui par conséquent ont nécessairement pointés dans la journée, on met
| le Temps-total-hebdomadaire à jour dans *I* (en fonction du jour *J*).

TraiteTTH3 (I, P, J) = O
with
∀ i : In (i, P)
∧ J = 2
⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= P[i]
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)

∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
∧ rens = I[i]

∨ ∀ i : In (i, P)
∧ J ≠ 2
∧ J ≠ 7
∧ J ≠ 1
⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hebdomadaire (rens) + P[i]
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
∧ rens = I[i]

TraiteTTH4 : D-BASE x D-BASE x INTEGER → D-BASE ;

| Pour les personnes identifiées dans *Int*, on met le Temps-total-hebdomadaire de leur enregistrement dans *I* à
| jour (en fonction du jour *J*).

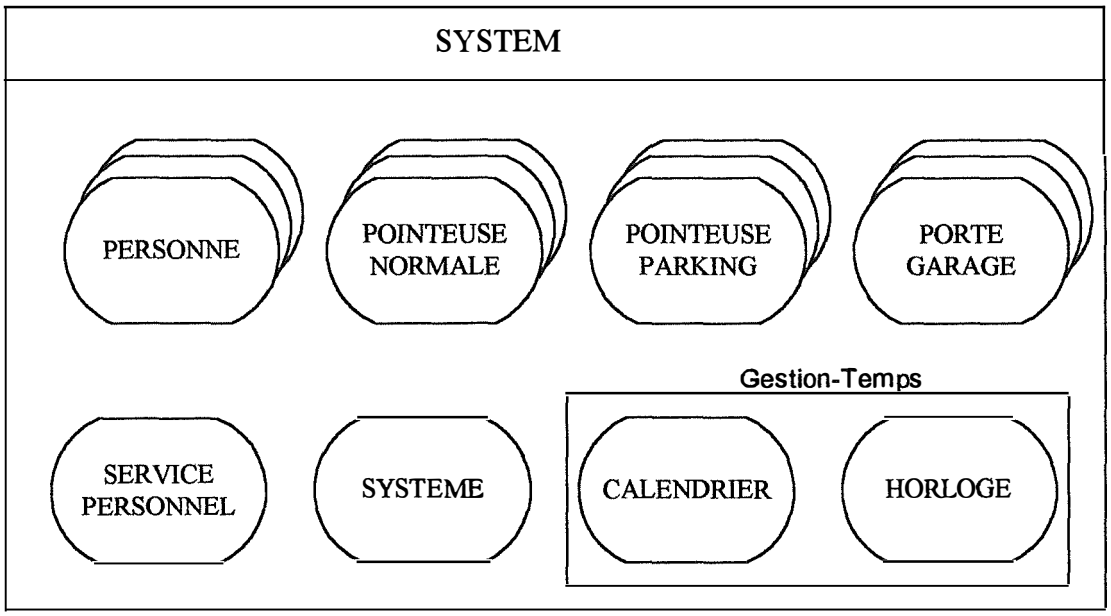
TraiteTTH4 (I, Int, J) = O
with
∀ i : In (i, Int)
∧ J = 2
⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= 27480

∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
∧ rens = I[i]

∨ ∀ i : In (i, Int)
∧ J = 6
⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hebdomadaire (rens) + 21480
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
∧ rens = I[i]

∨ ∀ i : In (i, Int)
∧ J ≠ 6
∧ J ≠ 2
∧ J ≠ 7
∧ J ≠ 1
⇒ O = Modify (I, i, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens') = Mouvement (rens)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hebdomadaire (rens) + 27480
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
∧ rens = I[i]

c. Déclaration de société :



Remarques générales:

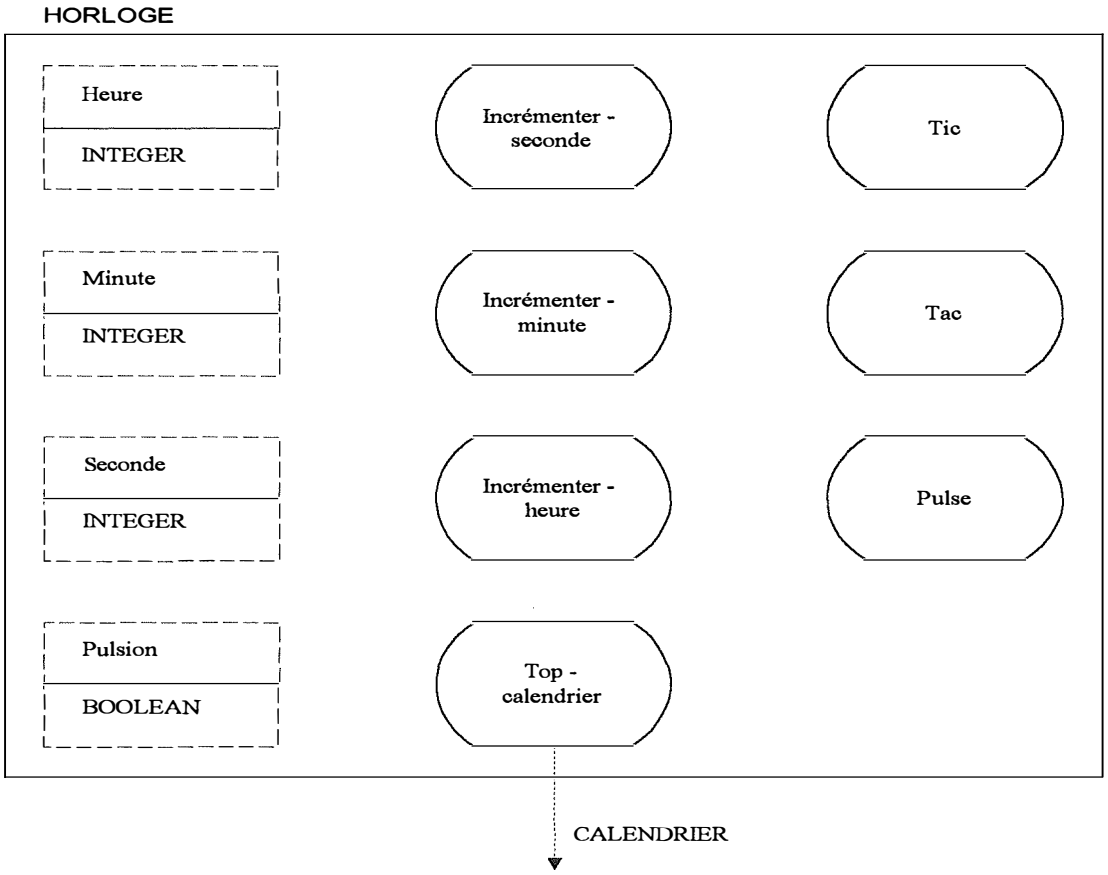
- on n'a pas établi de mécanisme de vérification de validité de matricule entre l'agent personne et pointeuse normale ou parking, car on considère que le problème ne se pose pas encore puisque l'on fait à ce niveau l'abstraction des problèmes de sécurités.

- pour les mêmes raisons que la première remarque, nous ne considérons pas les portes autres que celles du parking voitures.

- on ne sait pas encore comment l'on va rendre opérationnel le passage de l'information matricule entre l'agent personne et l'agent pointeuse normale et parking. Par conséquent, on modélise faussement mais simplement la transmission de cette information par un state information du paramètre défini state component de personne vers l'agent pointeuse.

d. Spécification des agents :

HORLOGE



Déclaration de la structure de l'agent

State Components :

Heure	instance-of	INTEGER
Minute	instance-of	INTEGER
Seconde	instance-of	INTEGER
Pulsion	instance-of	BOOLEAN

Actions :

Tic
Incrémenter-heure
Tac
Incrémenter-minute

Pulse
Incrémenter-seconde
Top-calendrier → CALENDRIER

Contraintes sur l'agent

Initial Valuation :

Heure = 0
Minute = 0
Seconde = 0
Pulsion = FALSE

State Behaviour :

Heure ≥ 0 ∧ Heure ≤ 23
Minute ≥ 0 ∧ Minute ≤ 59
Seconde ≥ 0 ∧ Seconde ≤ 59

Effects of Actions :

Pulse	:	Pulsion : = TRUE
Incrémenter-seconde	:	Pulsion : = FALSE ; Seconde : = Seconde + 1
Incrémenter-minute	:	Pulsion : = FALSE ; Seconde : = 0 ; Minute : = Minute + 1
Incrémenter-heure	:	Pulsion : = FALSE ; Seconde : = 0 ; Minute : = 0 ; Heure : = Heure + 1
Top-calendrier	:	Pulsion : = FALSE ; Seconde : = 0 ; Minute : = 0 ; Heure : = 0

Capability :

XO (Incrémenter-seconde / Pulsion = TRUE ∧
Seconde < 59)

XO (Incrémenter-minute / Pulsion = TRUE ∧
Seconde = 59 ∧
Minute < 59)

XO (Incrémenter-heure / Pulsion = TRUE ∧
Seconde = 59 ∧
Minute = 59 ∧
Heure < 23)

XO (Top-calendrier / Pulsion = TRUE ∧
Seconde = 59 ∧
Minute = 59 ∧
Heure = 23)

Action Composition :

Tic <-> Pulse ; Tac

Tac <-> Pulse ; Tic

Action Duration :

| Pulse | = 1 sec

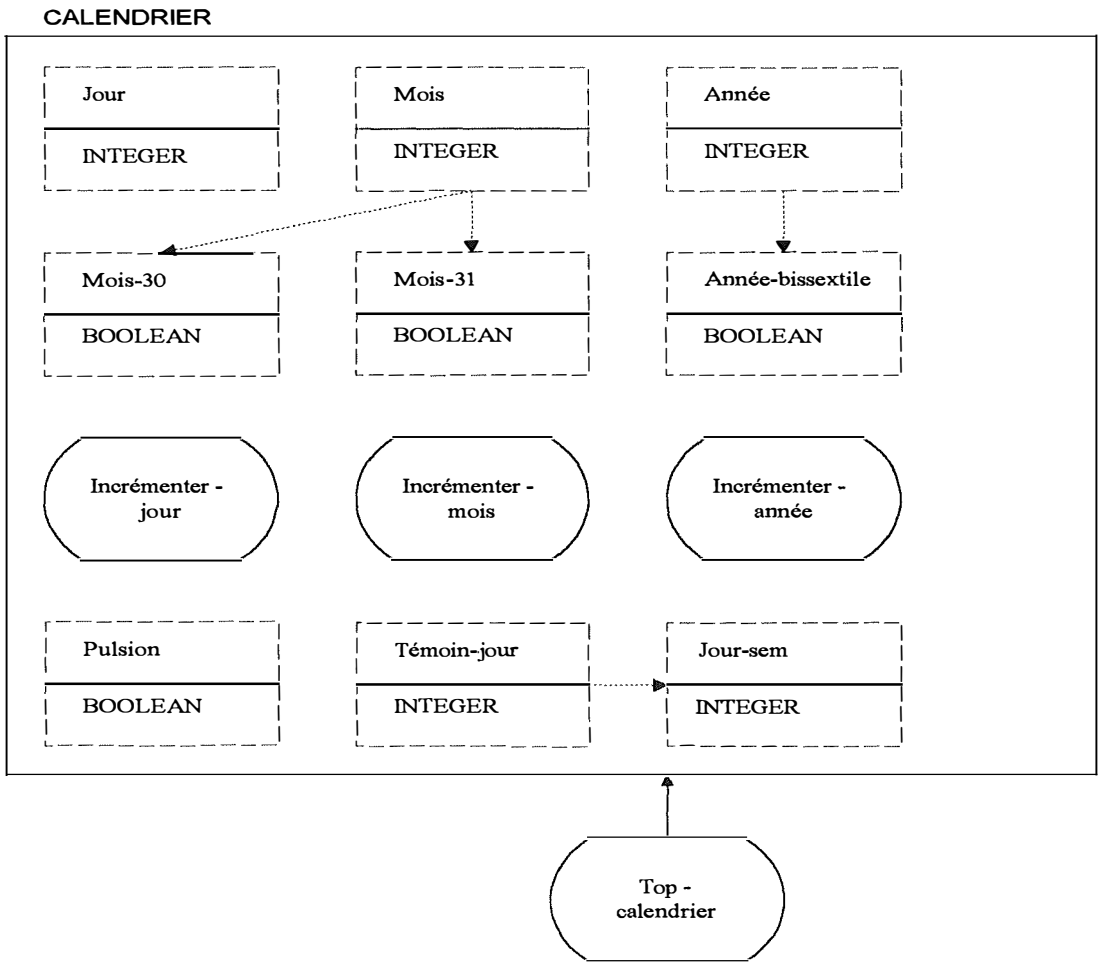
Action Information :

XK (Top-calendrier.Cal / TRUE)

State Information :

XK (Heure.Syst / TRUE)
XK (Minute.Syst / TRUE)
XK (Seconde.Syst / TRUE)

CALENDRIER



Déclaration de la structure de l'agent

State Composents :

Jour	instance-of	INTEGER
Mois	instance-of	INTEGER
Année	instance-of	INTEGER
Mois-30	instance-of	BOOLEAN
Mois-31	instance-of	BOOLEAN
Année-bissextile	instance-of	BOOLEAN
Jour-sem	instance-of	INTEGER
Témoin-jour	instance-of	INTEGER
Pulsion	instance-of	BOOLEAN

Actions :

Incrémenter-jour
Incrémenter-mois
Incrémenter-année

Contraintes sur l'agent

Derived Components :

Mois-31 $\triangleq$ Mois = 1 $\vee$ Mois = 3 $\vee$ Mois = 5 $\vee$ Mois = 7 $\vee$ Mois = 8 $\vee$ Mois = 10 $\vee$ Mois = 12

Mois-30 $\triangleq$ Mois = 4 $\vee$ Mois = 6 $\vee$ Mois = 9 $\vee$ Mois = 11

Année-bissextile $\triangleq$ Année = 1996 + (4 * valeur)

Jour-sem $\triangleq$ (Témoin-jour mod 7) + 1

Initial Valuation :

Jour = 1
Mois = 1
Année = 1996
Jour-sem = Lundi
Témoin-jour = 1
Pulsion = FALSE

State Behaviour :

Jour $\geq$ 1 $\wedge$ Jour $\leq$ 31
Mois $\geq$ 1 $\wedge$ Mois $\leq$ 12
Jour-sem $\geq$ 1 $\wedge$ Jour-sem $\leq$ 7
Année $\geq$ 1996

Effects of Actions :

Hor.Top-calendrier	:	Pulsion : = TRUE
Incrémenter-jour	:	Pulsion : = FALSE ; Jour : = Jour + 1 ; Témoin-jour : = Témoin-jour + 1
Incrémenter-mois	:	Pulsion : = FALSE ; Jour : = 1 ; Mois : = Mois + 1 Témoin-jour : = Témoin-jour + 1

Incrémenter-année : Pulsion : = FALSE ; Jour : = 1 ; Mois : = 1 ;
Année : = Année + 1
Témoin-jour : = Témoin-jour + 1

Capability :

XO (Incrémenter-jour / (Pulsion = TRUE ∧ Mois-31 = TRUE ∧ Jour < 31)
∨ (Pulsion = TRUE ∧ Mois-30 = TRUE ∧ Jour < 30)
∨ (Pulsion = TRUE ∧ Mois-31 = FALSE ∧
Mois-30 = FALSE ∧
Année-bissextile = FALSE ∧ Jour < 28)
∨ (Pulsion = TRUE ∧ Mois-31 = FALSE ∧
Mois-30 = FALSE ∧
Année-bissextile = TRUE ∧ Jour < 29))

XO (Incrémenter-mois / Pulsion = TRUE ∧ Mois ≠ 12 ∧
((Mois-30 = TRUE ∧ Jour = 30)
∨ (Mois-31 = TRUE ∧ Jour = 31)
∨ (Année-bissextile = TRUE ∧ Mois = 2 ∧
Jour = 29)
∨ (Année-bissextile = FALSE ∧ Mois = 2 ∧
Jour = 28))

XO (Incrémenter-année / Pulsion = TRUE ∧ Mois = 12 ∧ Jour = 31)

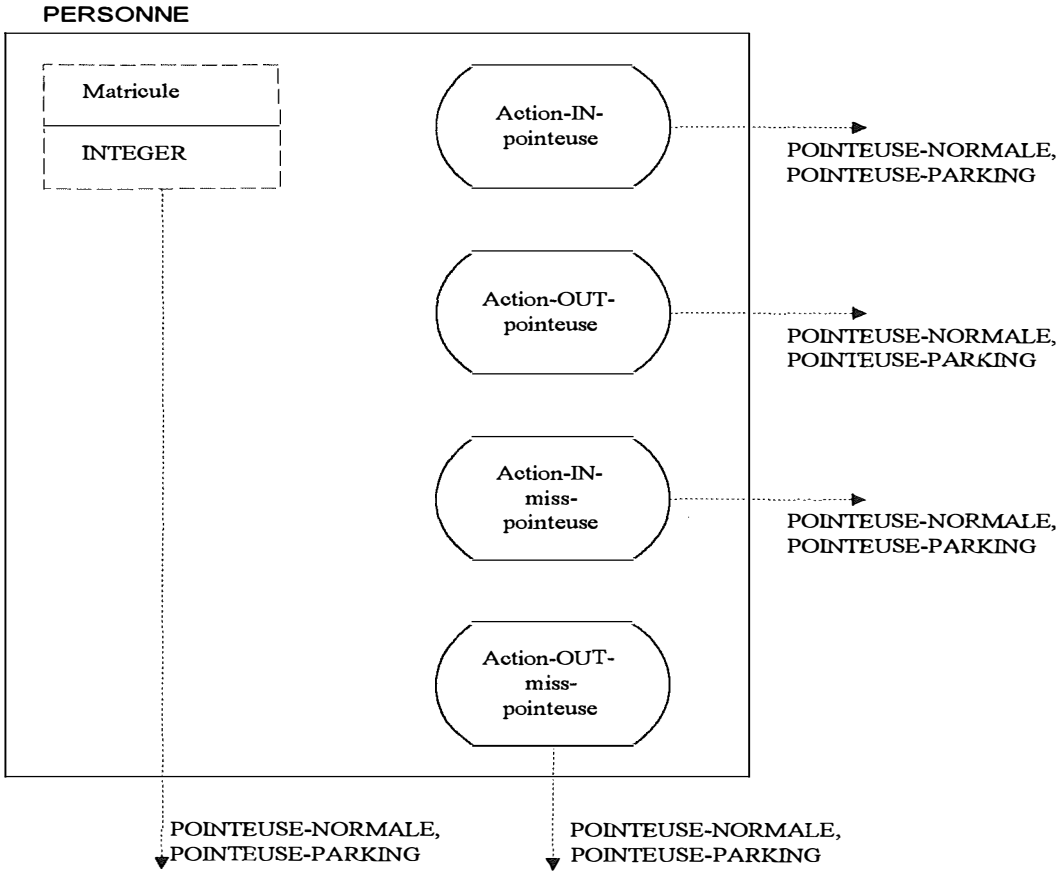
Action Perception :

XK (Hor.Top-calendrier / Pulsion)

State Information :

XK (Jour-sem.Syst / TRUE)
XK (Jour.Syst / TRUE)
XK (Mois.Syst / TRUE)
XK (Année.Syst / T RUE)

PERSONNE



Déclaration de la structure de l'agent

State Components :

Matricule instance-of INTEGER

Actions :

Action-IN-pointeuse → POINTEUSE-NORMALE* , POINTEUSE-PARKING*
Action-IN-miss-pointeuse → POINTEUSE-NORMALE* , POINTEUSE-PARKING*
Action-OUT-miss-pointeuse → POINTEUSE-NORMALE* , POINTEUSE-PARKING*
Action-OUT-pointeuse → POINTEUSE-NORMALE* , POINTEUSE-PARKING*

Contraintes sur l'agent

Action Information :

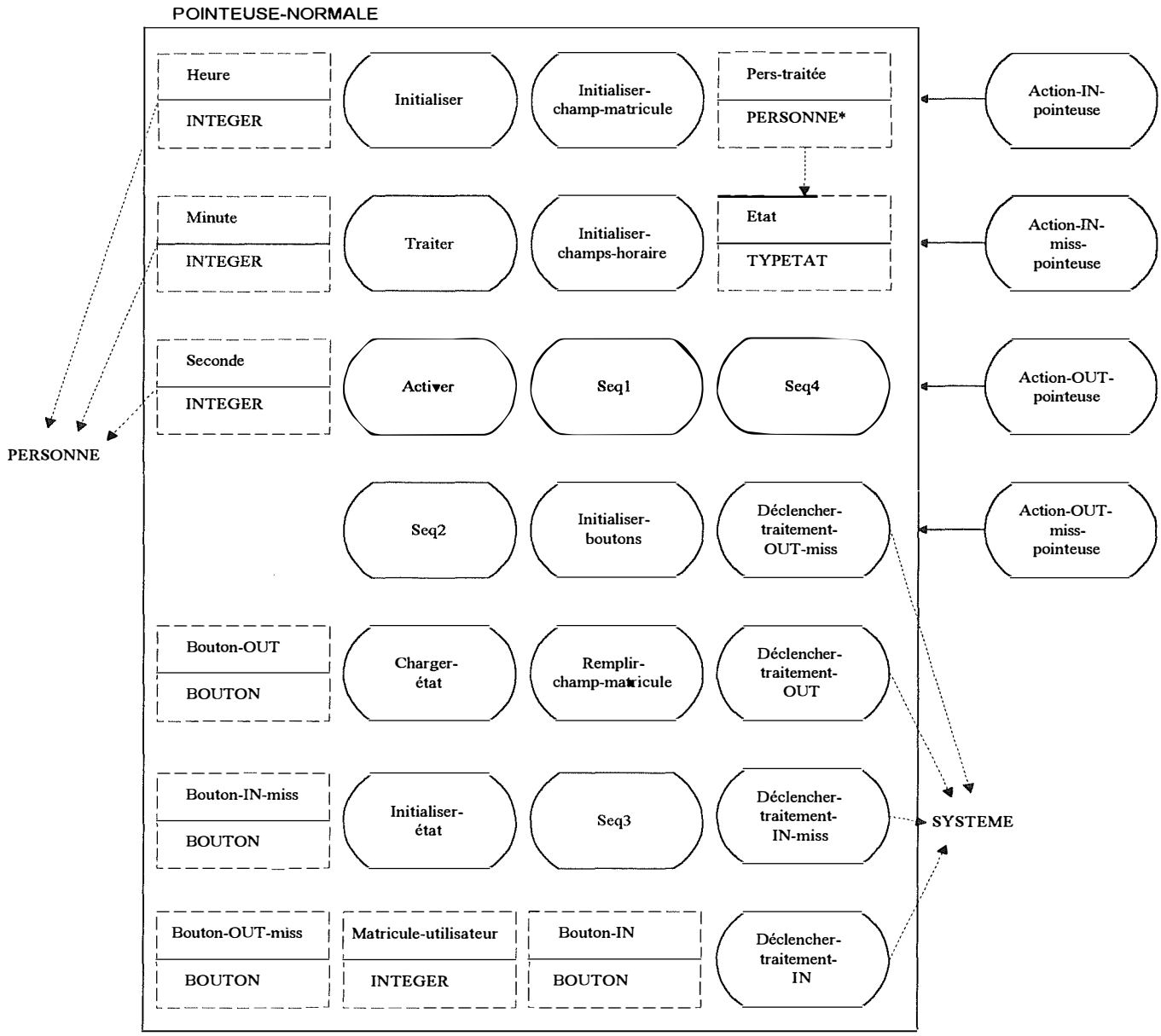
XK (Action-IN-pointeuse.Pn / TRUE) { Pn : POINTEUSE-NORMALE }
XK (Action-OUT-pointeuse.Pn / TRUE) { Pn : POINTEUSE-NORMALE }

XK (Action-IN-miss-pointeuse.Pn / TRUE) { Pn : POINTEUSE-NORMALE }
XK (Action-OUT-miss-pointeuse.Pn / TRUE) { Pn : POINTEUSE-NORMALE }
XK (Action-IN-pointeuse.Pp / TRUE) { Pp : POINTEUSE-PARKING }
XK (Action-OUT-pointeuse.Pp / TRUE) { Pp : POINTEUSE-PARKING }
XK (Action-IN-miss-pointeuse.Pp / TRUE) { Pp : POINTEUSE-PARKING }
XK (Action-OUT-miss-pointeuse.Pp / TRUE) { Pp : POINTEUSE-PARKING }

State Information :

XK (Matricule.Pn / TRUE) { Pn : POINTEUSE-NORMALE }
XK (Matricule.Pp / TRUE) { Pp : POINTEUSE-PARKING }

POINTEUSE-NORMALE



Déclaration de la structure de l'agent

State Components :

Heure	instance-of	INTEGER
Minute	instance-of	INTEGER
Seconde	instance-of	INTEGER
Pers-traitée	instance-of	PERSONNE*

Matricule-utilisateur	instance-of	INTEGER
Etat	instance-of	TYPETAT
Bouton-IN	instance-of	BOUTON
Bouton-IN-miss	instance-of	BOUTON
Bouton-OUT	instance-of	BOUTON
Bouton-OUT-miss	instance-of	BOUTON

Actions :

Traiter
Initialiser-champs-horaire
Activer (PERSONNE)
Initialiser-champ-matricule
Initialiser-état
Changer-état (PERSONNE)
Initialiser-boutons
Initialiser
Remplir-champ-matricule (PERSONNE)
Seq1
Seq2
Seq3
Seq4
Déclencher-traitement-IN (INTEGER) → SYSTEME*
Déclencher-traitement-IN-miss (INTEGER) → SYSTEME*
Déclencher-traitement-OUT (INTEGER) → SYSTEME*
Déclencher-traitement-OUT-miss (INTEGER) → SYSTEME*

Contraintes sur l'agent

Derived components :

Etat $\hat{=}$ Pers-traitée = UNDEF

Initial Valuation :

Heure = 0
Minute = 0
Seconde = 0
Bouton-OUT = Relaché
Pers-traitée = UNDEF
Bouton-OUT-miss = Relaché
Matricule-utilisateur = 0
Bouton-IN = Relaché
Bouton-IN-miss = Relaché

State Behaviour :

Minute $\geq 0 \wedge$ Minute ≤ 59
Seconde $\geq 0 \wedge$ Seconde ≤ 59

Effects of Actions :

Pers.Action-IN-pointeuse : Bouton-IN : = Poussé

Pers.Action-IN-miss-pointeuse : Bouton-IN-miss : = Poussé

Pers.Action-OUT-pointeuse : Bouton-OUT : = Poussé

Pers.Action-OUT-miss-pointeuse : Bouton-OUT-miss : = Poussé

Syst.AfficherTemps (h, m, s, point) : Heure : = h ;
Minute : = m ;
Seconde : = s

Initialiser-champs-horaire : Heure : = 0 ;
Minute : = 0 ;
Seconde : = 0

Initialiser-champ-matricule : Matricule-utilisateur : = 0

Initialiser-état : Pers-traitée : = UNDEF

Remplir-champ-matricule (pers) : Matricule-utilisateur : = pers.Matricule

Initialiser-bouton : Bouton-IN : = Relaché ;
Bouton-IN-miss : = Relaché ;
Bouton-OUT : = Relaché ;
Bouton-OUT-miss : = Relaché

Changer-état (pers) : Pers-traitée : = pers

Action Composition :

Seq1 <-> pers.Action-IN-pointeuse ; Activer (pers) ;
Déclencher-traitement-IN (matricule-utilisateur) ; Initialiser

Seq2 <-> pers.Action-IN-miss-pointeuse ; Activer (pers) ;
Déclencher-traitement-IN-miss (matricule-utilisateur) ; Initialiser

Seq3 <-> pers.Action-OUT-pointeuse ; Activer (pers) ;
Déclencher-traitement-OUT (matricule-utilisateur) ; Initialiser

Seq4 <-> pers.Action-OUT-miss-pointeuse ; Activer (pers) ;
Déclencher-traitement-OUT-miss (matricule-utilisateur) ; Initialiser

Activer (pers) <-> Changer-état (pers) ; Remplir-champ-matricule (pers)

Initialiser <-> Initialiser-champs-horaire ;
Initialiser-champ-matricule ;
Initialiser-boutons ; Initialiser-état

Action Perception :

XK (Pers.Action-IN-pointeuse / Etat = Libre)
XK (Pers.Action-IN-miss-pointeuse / Etat = Libre)
XK (Pers.Action-OUT-pointeuse / Etat = Libre)
XK (Pers.Action-OUT-miss-pointeuse / Etat = Libre)
XK (Syst.AfficherTemps (h, m, s, point) / TRUE)

State Perception :

XK (Pers.Matricule / Pers-traitée = Pers)

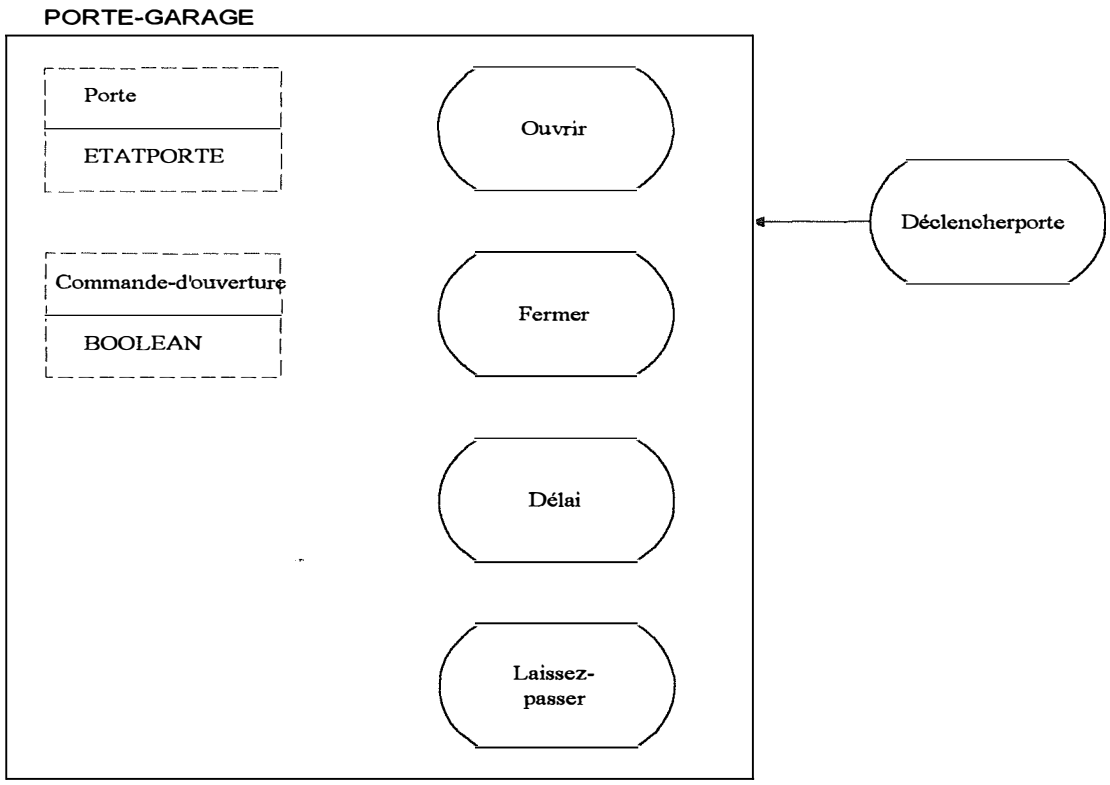
State Information :

XK (Etat.Pers / TRUE)
XK (Seconde.Pers / TRUE)
XK (Minute.Pers / TRUE)
XK (Heure.Pers / TRUE)

Action Information :

XK (Déclencher-traitement-IN (matricule-utilisateur).Syst / TRUE)
XK (Déclencher-traitement-IN-miss (matricule-utilisateur).Syst / TRUE)
XK (Déclencher-traitement-OUT (matricule-utilisateur).Syst / TRUE)
XK (Déclencher-traitement-OUT-miss (matricule-utilisateur).Syst / TRUE)

PORTE-GARAGE



Déclaration de la structure de l'agent

State Components :

Commande-d'ouverture	instance-of	BOOLEAN
Porte	instance-of	ETATPORTE

Actions :

Ouvrir
Fermer
Délai
Laissez-passer

Contraintes sur l'agent

Initial Valuation :

Commande-d'ouverture = FALSE

Porte = Fermée

Effects of Actions :

Syst.Déclencheporte (prt) : Commande-d'ouverture : = TRUE

Ouvrir : Porte : = Ouverte ; Commande-d'ouverture : = FALSE

Fermer : Porte : = Fermée

Capability :

XO (Laissez-passer / Commande-d'ouverture = TRUE)

Action Composition :

Laissez-passer < - > Ouvrir ; Délai ; Fermer

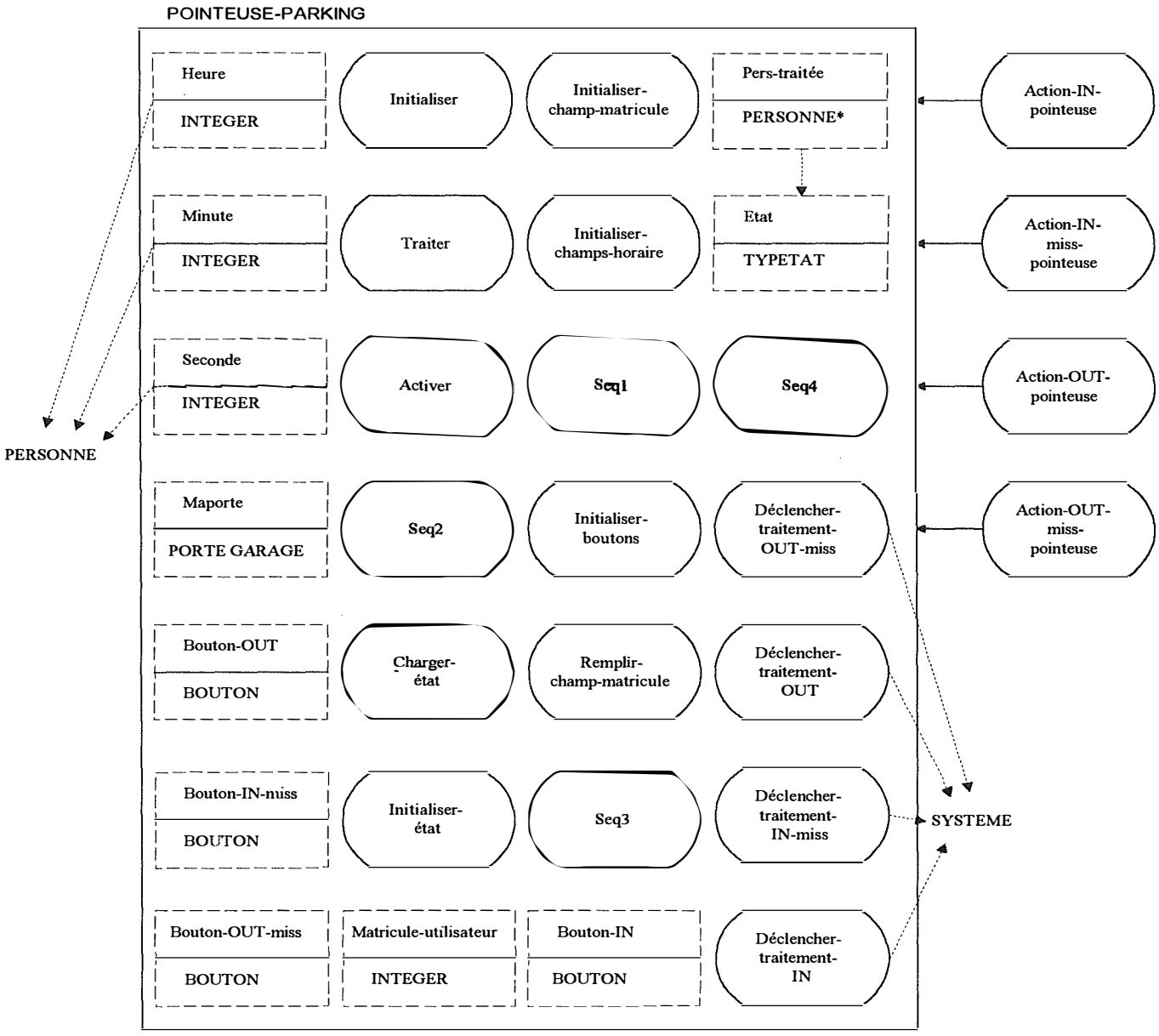
Action Duration :

| Délai | = 30 sec

Action Perception :

XK (Syst.Déclencheporte (prt) / TRUE)

POINTEUSE-PARKING



Déclaration de la structure de l'agent

State Components :

Heure	instance-of	INTEGER
Minute	instance-of	INTEGER
Seconde	instance-of	INTEGER
Matricule-utilisateur	instance-of	INTEGER
Pers-traitée	instance-of	PERSONNE*

Etat	instance-of	TYPETAT
Maporte	instance-of	PORTE GARAGE
Bouton-IN	instance-of	BOUTON
Bouton-IN-miss	instance-of	BOUTON
Bouton-OUT	instance-of	BOUTON
Bouton-OUT-miss	instance-of	BOUTON

Actions :

Traiter
Initialiser-champs-horaire
Activer (PERSONNE)
Initialiser-champ-matricule
Initialiser-état
Changer-état
Initialiser-boutons
Initialiser
Remplir-champ-matricule (PERSONNE)
Seq1
Seq2
Seq3
Seq4
Déclencher-traitement-IN (INTEGER) → SYSTEME*
Déclencher-traitement-IN-miss (INTEGER) → SYSTEME*
Déclencher-traitement-OUT (INTEGER) → SYSTEME*
Déclencher-traitement-OUT-miss (INTEGER) → SYSTEME*

Contraintes sur l'agent

L'agent Pointeuse Parking reprend les mêmes actions et composants d'états que l'agent Pointeuse normale mais intégre par rapport à cela un nouveau composant d'état. De ce fait nous retrouvons la même présentation que pour l'agent Pointeuse avec le complément Maporte qui indique l'instance de l'agent PORTE qui est reliée à une instance particulière de l'agent POINTEUSE PARKING.

Nous allons nous limiter dans la spécification de l'agent Pointeuse Parking à présenter ce qui est à rajouter par rapport aux spécification de la Pointeuse normale.

Action Composition :

Maporte instance-of PORTE-GARAGE

SYSTEME

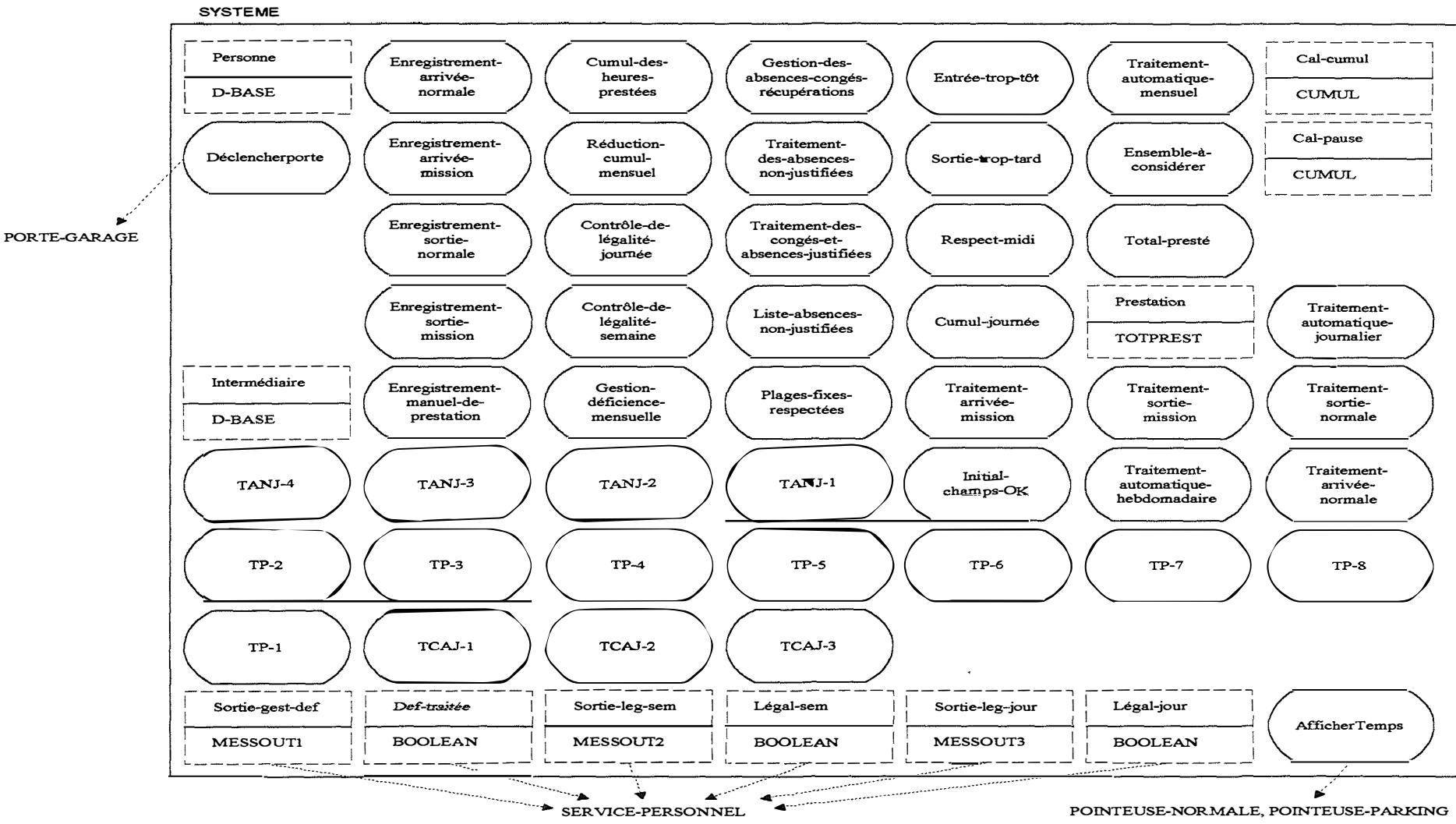
Déclaration de la structure de l'agent

State Components :

Personne	instance-of	D-BASE
Intermédiaire	instance-of	D-BASE
Sortie-gest-def	instance-of	MESSOUT1
Def-traitée	instance-of	BOOLEAN
Sortie-leg-sem	instance-of	MESSOUT2
Légal-sem	instance-of	BOOLEAN
Sortie-leg-jour	instance-of	MESSOUT3
Légal-jour	instance-of	BOOLEAN
Prestation	instance-of	TOTPREST
Cal-cumul	instance-of	CUMUL
Cal-pause	instance-of	CUMUL

Actions :

Enregistrement-arrivée-normale
Enregistrement-arrivée-mission
Enregistrement-sortie-normale
Enregistrement-sortie-mission
*Traitement-arrivée-normale (UNION [POINTEUSE-NORMALE, POINTEUSE-PARKING])
*Traitement-arrivée-mission (UNION [POINTEUSE-NORMALE, POINTEUSE-PARKING])
*Traitement-sortie-normale (UNION [POINTEUSE-NORMALE, POINTEUSE-PARKING])
*Traitement-sortie-mission (UNION [POINTEUSE-NORMALE, POINTEUSE-PARKING])
Enregistrement-manuel-de-prestation → SERVICE-PERSONNEL
Cumul-des-heures-prestées
Réduction-cumul-mensuel
Contrôle-de-légalité-journée → SERVICE-PERSONNEL
Contrôle-de-légalité-semaine → SERVICE-PERSONNEL
Gestion-déficience-mensuelle → SERVICE-PERSONNEL
Gestion-absences-congés-récupérations → SERVICE-PERSONNEL
Traitement-des-absences-non-justifiées
Liste-absences-non-justifiées
Plages-fixes-respectées
Entrée-trop-tôt
Sortie-trop-tard


SYSTEME
Personne
D-BASE
Déclencherporte
PORTE-GARAGE
Enregistrement-arrivée-normale
Enregistrement-arrivée-mission
Enregistrement-sortie-normale
Enregistrement-sortie-mission
Enregistrement-manuel-de-prestation
Cumul-des-heures-prestées
Réduction-cumul-mensuel
Contrôle-de-légalité-journée
Contrôle-de-légalité-semaine
Gestion-déficience-mensuelle
Gestion-des-absences-congés-récupérations
Traitement-des-absences-non-justifiées
Traitement-des-congés-et-absences-justifiées
Liste-absences-non-justifiées
Plages-fixes-respectées
Entrée-trop-tôt
Sortie-trop-tard
Respect-midi
Cumul-journée
Initial-champs-OK
Traitement-automatique-mensuel
Ensemble-à-considérer
Total-presté
Prestation
TOTPREST
Traitement-arrivée-mission
Cal-cumul
CUMUL
Cal-pause
CUMUL
Traitement-automatique-journalier
Traitement-sortie-mission
Traitement-automatique-hebdomadaire
Traitement-sortie-normale
Traitement-arrivée-normale
Intermédiaire
D-BASE
TANJ-4
TANJ-3
TANJ-2
TANJ-1
TP-2
TP-3
TP-4
TP-5
TP-6
TP-7
TP-8
TP-1
TCAJ-1
TCAJ-2
TCAJ-3
Sortie-gest-def
MESSOUT1
Def-traitée
BOOLEAN
Sortie-leg-sem
MESSOUT2
Légal-sem
BOOLEAN
Sortie-leg-jour
MESSOUT3
Légal-jour
BOOLEAN
AfficherTemps
SERVICE-PERSONNEL
POINTEUSE-NORMALE, POINTEUSE-PARKING

*Déclencherporte (PORTE-GARAGE) → PORTE-GARAGE
*AfficherTemps (INTEGER, INTEGER, INTEGER,
UNION [POINTEUSE-NORMALE, POINTEUSE-PARKING])
→ POINTEUSE-NORMALE, POINTEUSE-PARKING
Respect-midi
Cumul-journée
Traitement-automatique-journalier
Traitement-automatique-mensuel
Ensemble-à-considérer
Total-presté
Traitement-des-congés-et-absences-justifiées
TP-1
TP-2
TP-3
TP-4
TP-5
TP-6
TP-7
TP-8
TANJ-1
TANJ-2
TANJ-3
TANJ-4
TCAJ-1
TCAJ-2
TCAJ-3
Initial-champs-OK
Traitement-automatique-hebdomadaire

Contraintes sur l'agent

Initial Valuation :

Def-traitée = FALSE
Légal-sem = FALSE
Légal-jour = FALSE

Effects of Actions :

Traitement-arrivée-normale (mat) :

Personnes : = InNormal (Personnes, mat,
Horloge.Heure, Horloge.Minute,
Horloge.Seconde)

Traitement-arrivée-mission (mat) :

Personnes : = InMiss (Personnes, mat,
Horloge.Heure, Horloge.Minute,
Horloge.Seconde)

Traitement-sortie-normale (mat) :

Personnes : = OutNormal (Personnes, mat,
Horloge.Heure, Horloge.Minute,
Horloge.Seconde)

Traitement-sortie-mission (mat) :

Personnes : = OutMiss (Personnes, mat,
Horloge.Heure, Horloge.Minute,
Horloge.Seconde)

Enregistrement-manuel-de-prestation :

Personnes : = EnregistreManuel(Personne, Service-personnel.Matricule,
Service-personnel.Heure,
Service-personnel.Minute,
Service-personnel.Seconde)

Réduction-cumul-mensuel :

Personnes : = RédCumul (Personnes)

Gestion-déficience-mensuelle :

Sortie-gest-def : = GestDef (Personnes) ;
Def-traitée : = TRUE

TANJ-1 :

| On met dans Intermédiaire tous les membres du personnel qui n'ont pas fait de mouvement en
| ce jour et qui ne sont ni en mission ni en congé ni en absence justifiée.

```
Intermédiaire : = CréeInter1 ( Personnes,
                               Calendrier.Jour,
                               Calendrier.Mois,
                               Calendrier.Année )
```

TANJ-2 :

| On traite le "Temps total hebdomadaire" et le "Temps cumulé mois antérieur" pour les
| membres du personnel qui n'ont pas fait de mouvement en ce jour et qui ne sont ni en mission
| ni en congé ni en absence justifiée. Ces personnes sont enregistrées dans Intermédiaire.

```
Personnes : = TraiteTTH1 ( Personnes,
                           Intermédiaire ,
                           Calendrier.Jour-sem ) ;
Personnes : = TraiteTCMA ( Personnes,
                           Intermédiaire,
                           Calendrier.Jour,
                           Calendrier.Jour-sem )
```

TANJ-3 :

| On met à jour le "Temps cumulé global" pour les membres du personnel qui n'ont pas fait de
| mouvement en ce jour et qui ne sont ni en mission ni en congé ni en absence justifiée. Ces
| personnes sont enregistrées dans Intermédiaire.

```
Personnes : = TraiteTCG1 ( Personnes,
                           Intermédiaire ,
                           Calendrier.Jour-sem )
```

TANJ-4 :

| On indique dans la base de données l'absence injustifiée pour les membres du personnel qui
| n'ont pas fait de mouvement en ce jour et qui ne sont ni en mission ni en congé ni en absence
| justifiée. Ces personnes sont enregistrées dans Intermédiaire.

```
Personnes : = InscrireAInJ ( Personnes,
                             Intermédiaire,
                             Calendrier.Jour,
                             Calendrier.Mois
                             Calendrier.Année )
```

Gestion-des-absences-congés-récupérations :

Personnes : = GestACR (Personnes,
Service-personnel.Matricule,
Service-presonnel.Motif,
Service-personnel.Dates)

Contrôle-de-légalité-semaine :

Sortie-leg-sem : = GestCLS (Personnes) ;
Légal-sem : = TRUE

Liste-absences-non-justifiées :

Sortie-leg-jour : = TraiteLANJ (Personnes,
Calendrier.Jour,
Calendrier.Mois
Calendrier.Année)

Plages-fixes-respectées :

Sortie-leg-jour : = FaireUnion (Sortie-leg-jour, TraitePFR (Personnes,
Calendrier.Jour-sem))

Entrée-trop-tôt :

Sortie-leg-jour : = FaireUnion (Sortie-leg-jour, TraiteETT (Personnes))

Sortie-trop-tard :

Sortie-leg-jour : = FaireUnion (Sortie-leg-jour, TraiteSTT (Personnes,
Calendrier.Jour-sem))

Respect-midi :

Sortie-leg-jour : = FaireUnion(Sortie-leg-jour, TraiteRM (Cal-pause))

Cumul-journée :

Sortie-leg-jour : = FaireUnion (Sortie-leg-jour, TraiteCJ (Cal-cumul) ;
Légal-jour : = TRUE

Ensemble-à-considérer :

| On met dans intermédiaire les membres du personnel qui on soit fait un mouvement, soit sont
| en mission.

Intermédiaire : = CréeInter2 (Personnes)

TP-1 :

| On traite le champ Temps-cumulé-mois-antérieur pour les membres du personnel qui on soit
| fait un mouvement, soit sont en mission.

Personnes : = TraiteTCMA (Personnes,
Intermédiaire,
Calendrier.Jour,
Calendrier.Jour-sem)

TP-2 :

| On met à jour le temps-total-hebdomadaire pour les membres du personnel partis en mission
| toute la journée.

Personnes : = TraiteTTH2 (Personnes,
Intermédiaire,
Calendrier.Jour-sem)

TP-3 :

| Pour les personnes qui ont soit fait un mouvement et sont inscris dans intermédiaire, on retire
| les entrées sorties pointées mission.

Intermédiaire : = RetraitMiss (Intermédiaire, Personnes)

TP-4 :

| On met à jour la variable Cal-pause. C'est-à-dire que l'on enregistre dans Cal-pause toutes les
| personnes qui ont faits un mouvement pendant la journée et dont le temps de pause de midi
| minimum (45 minutes) n'a pas été respecté.

Cal-pause : = TraiteCP (Intermédiaire, Personnes)

TP-5 :

| On calcule le temps presté par les membres du personnel ayant pointés.

Prestation : = TraiteTempsPresté (Personnes, Intermédiaire)

TP-6 :

| On traite le Cal-cumul. C'est-à-dire que l'on enregistre dans Cal-cumul toutes les
| personnes qui ont faits un mouvement pendant la journée et dont le temps de travail
| dépasse le maximum autorisé (9 heures).

Cal-cumul : = TraiteCC (Prestation, Calendrier.Jour-sem)

TP-7 :

| On calcule le Temps-total-hebdomadaire pour les personnes ayant pointés.

Personnes : = TraiteTTH3 (Personnes,
Prestation,
Calendrier.Jour-sem)

TP-8 :

| On met à jour le Temps-cumulé-global pour les personnes qui étaient présentes dans la
| société.

Personnes : = TraiteTCG2 (Personnes,
Prestation,
Calendrier.Jour-sem)

TCAJ-1 :

| On met dans Intermédiaire toutes les personnes en congé ou en absence justifiée.

Intermédiaire : = CréeInter3 (Personnes,
Calendrier.Jour,
Calendrier.Mois,
Calendrier.Année)

TCAJ-2 :

| On met à jour le Temps-cumulé-mois-antérieur pour toutes les personnes en congé ou en
| absence justifiée.

Personnes : = TraiteTCMA (Personnes,
Intermédiaire,
Calendrier.Jour,
Calendrier.Jour-sem)

TCAJ-3 :

| On met à jour le Temps-total-hebdomadaire pour toutes les personnes en congé ou en
| absence justifiée.

Personnes : = TraiteTTH4 (Personnes,
Intermédiaire,
Calendrier.Jour-sem)

SERVICE PERSONNEL.Lecture-de-traitement-journalier :

Légal-jour : = FALSE

SERVICE PERSONNEL.Lecture-de-traitement-hebdomadaire :

Légal-sem : = FALSE

SERVICE PERSONNEL.Lecture-de-traitement-mensuel :

Def-traitée : = FALSE

Capability :

XO (Traitement-automatique-journalier / Service-personnel.OK-jour)
XO (Traitement-automatique-hebdomadaire / Service-personnel.OK-hebd)
XO (Traitement-automatique-mensuel / Service-personnel.OK-mens)

Action Composition :

Cumul-des-heures-prestées < - > Ensemble-à-considérer ;
Total-presté

Contrôle-de-légalité-journée < - > Liste-absences-non-justifiées ;
Plages-fixes-respectées ;
Entrée-trop-tôt ;
Sortie-trop-tard ;
Respect-midi ;
Cumul-journée

Traitement-automatique-journalier
< - > Cumul-des-heures-prestées ;
Traitement-des-congés-et-absences-justifiées ;
Traitement-des-absences-non-justifiées ;
Contrôle-de-légalité-journée

Traitement-automatique-mensuel
< - > Réduction-cumul-mensuel ;
Gestion-déficience-mensuelle

Traitement-automatique-hebdomadaire
< - > Contrôle-de-légalité-semaine

Enregistrement-arrivée-normale < - > point.Déclencher-traitement-IN (mat) ;
Traitement-arrivée-normale (mat) ;
AfficherTemps (h, m, s, point) ;
Déclencherporte (porte)
{ point : POINTEUSE-PARKING }
with porte = AssignerPorteParking (point)
∧ h = GénèreHeure (Personnes, mat)
∧ m = GénèreMinute (Personnes, mat)
∧ s = GénèreSeconde (Personnes, mat)

Enregistrement-arrivée-normale < - > point.Déclencher-traitement-IN (mat) ;
Traitement-arrivée-normale (mat) ;
AfficherTemps (h, m, s, point)
{ point : POINTEUSE-NORMALE }
with h = GénèreHeure (Personnes, mat)
∧ m = GénèreMinute (Personnes, mat)
∧ s = GénèreSeconde (Personnes, mat)

Enregistrement-arrivée-mission < - > point.Déclencher-traitement-IN-miss (mat) ;
Traitement-arrivée-mission (mat) ;
AfficherTemps (h, m, s, point) ;
Déclencherporte (porte)
{ point : POINTEUSE-PARKING }
with porte = AssignerPorteParking (point)
∧ h = GénèreHeure (Personnes, mat)
∧ m = GénèreMinute (Personnes, mat)
∧ s = GénèreSeconde (Personnes, mat)

Enregistrement-arrivée-mission < - > point.Déclencher-traitement-IN-miss (mat) ;
Traitement-arrivée-mission (mat) ;
AfficherTemps (h, m, s, point)
{ point : POINTEUSE-NORMALE }
with h = GénèreHeure (Personnes, mat)
∧ m = GénèreMinute (Personnes, mat)
∧ s = GénèreSeconde (Personnes, mat)

Enregistrement-sortie-normale < - > point.Déclencher-traitement-OUT (mat) ;
Traitement-sortie-normale (mat) ;
AfficherTemps (h, m, s, point) ;
Déclencherporte (porte)
{ point : POINTEUSE-PARKING }
with porte = AssignerPorteParking (point)
∧ h = GénèreHeure (Personnes, mat)
∧ m = GénèreMinute (Personnes, mat)
∧ s = GénèreSeconde (Personnes, mat)

Enregistrement-sortie-normale < - > point.Déclencher-traitement-OUT (mat) ;
Traitement-sortie-normale (mat) ;
AfficherTemps (h, m, s, point)
{ point : POINTEUSE-NORMALE }
with h = GénèreHeure (Personnes, mat)
∧ m = GénèreMinute (Personnes, mat)
∧ s = GénèreSeconde (Personnes, mat)

Enregistrement-sortie-mission < - > point.Déclencher-traitement-OUT-miss (mat) ;
Traitement-sortie-mission (mat) ;
AfficherTemps (h, m, s, point) ;
Déclencherporte (porte)
{ point : POINTEUSE-PARKING }
with porte = AssignerPorteParking (point)
∧ h = GénèreHeure (Personnes, mat)
∧ m = GénèreMinute (Personnes, mat)
∧ s = GénèreSeconde (Personnes, mat)

Enregistrement-sortie-mission < - > point.Déclencher-traitement-OUT-miss (mat) ;
Traitement-sortie-mission (mat) ;
AfficherTemps (h, m, s, point)
{ point : POINTEUSE-NORMALE }
with h = GénèreHeure (Personnes, mat)
∧ m = GénèreMinute (Personnes, mat)
∧ s = GénèreSeconde (Personnes, mat)

Traitement-des-congés-et-absences-justifiées < - > TCAJ-1 ;
TCAJ-2 ;
TCAJ-3

Total-presté < - > TP-1 ; TP-2 ; TP-3 ; TP-4 ;
TP-5 ; TP-6 ; TP-7 ; TP-8

Traitement-des-absences-non-justifiées < - > TANJ-1 ; TANJ-2 ;
TANJ-3 ; TANJ-4

State Perception :

XK (Calendrier.Jour / TRUE)
XK (Calendrier.Mois / TRUE)
XK (Calendrier.Année / TRUE)
XK (Horloge.Heure / TRUE)
XK (Horloge.Minute / TRUE)
XK (Horloge.Seconde / TRUE)
XK (Calendrier.Jour-sem / TRUE)
XK (Service-personnel.OK-mens / TRUE)
XK (Service-personnel.OK-jour / TRUE)
XK (Service-personnel.OK-hebd / TRUE)
XK (Service-personnel.OK-acr / TRUE)
XK (Service-personnel.OK-manuel / TRUE)

State Information :

XK (Sortie-gest-def.Service-personnel / TRUE)
XK (Def-traitée.Service-personnel / TRUE)
XK (Sortie-leg-jour.Service-personnel / TRUE)
XK (Légal-sem.Service-personnel / TRUE)
XK (Sortie-leg-sem.Service-personnel / TRUE)
XK (Légal-jour.Service-personnel / TRUE)

Action perception :

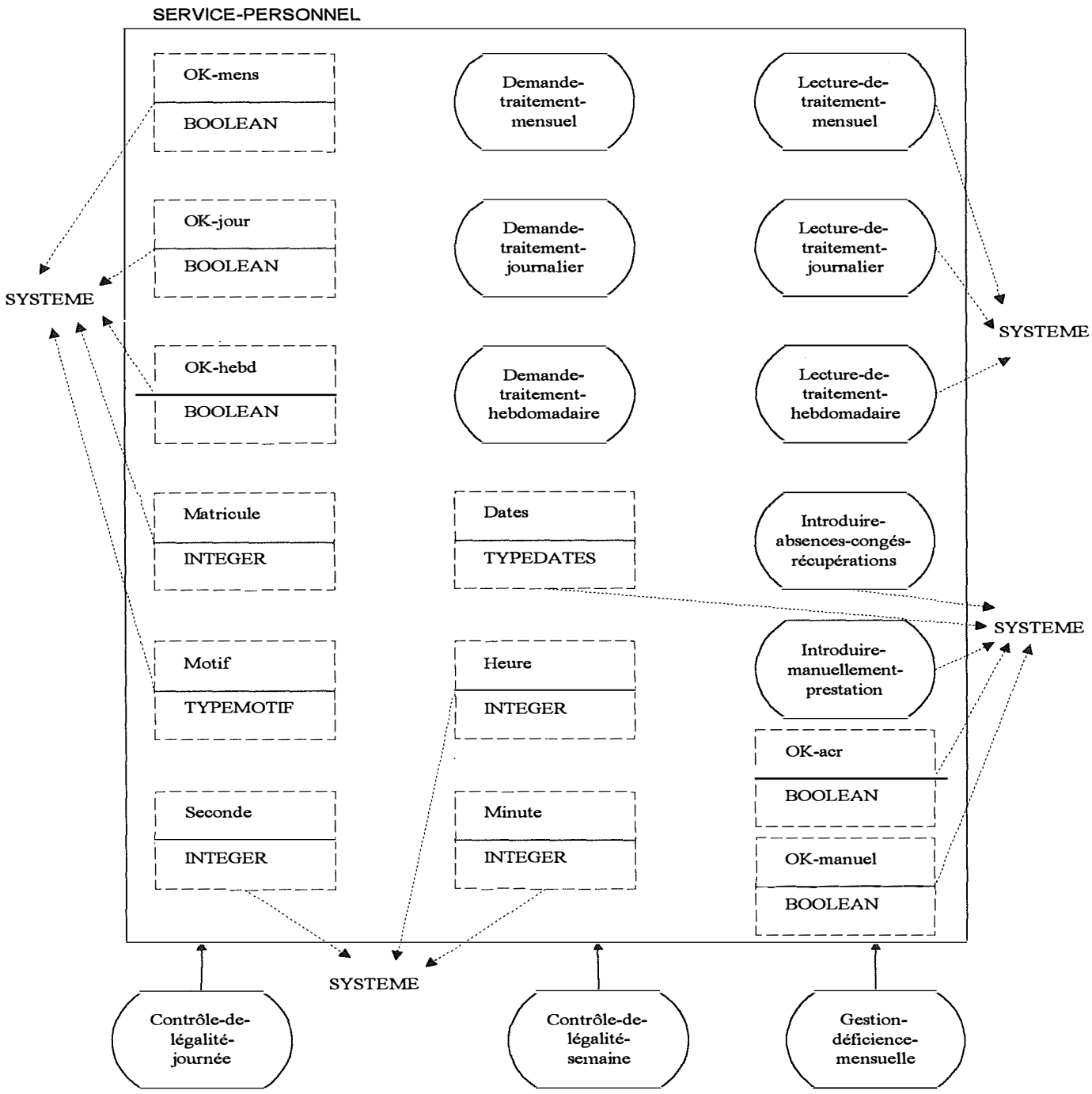
XK (Service-personnel.Lecture-de-traitement-journalier / TRUE)
XK (Service-personnel.Lecture-de-traitement-hebdomadaire / TRUE)
XK (Service-personnel.Lecture-de-traitement-mensuel / TRUE)
XK (Pn.Déclencher-traitement-IN (mat) / TRUE) { Pn : POINTEUSE-NORMALE }
XK (Pn.Déclencher-traitement-IN-miss (mat) / TRUE) { Pn : POINTEUSE-NORMALE }
XK (Pn.Déclencher-traitement-OUT (mat) / TRUE) { Pn : POINTEUSE-NORMALE }
XK (Pn.Déclencher-traitement-OUT-miss (mat) / TRUE) { Pn : POINTEUSE-NORMALE }
XK (Pp.Déclencher-traitement-IN (mat) / TRUE) { Pp : POINTEUSE-PARKING }
XK (Pp.Déclencher-traitement-IN-miss (mat) / TRUE) { Pp : POINTEUSE-PARKING }
XK (Pp.Déclencher-traitement-OUT (mat) / TRUE) { Pp : POINTEUSE-PARKING }
XK (Pp.Déclencher-traitement-OUT-miss (mat) / TRUE) { Pp : POINTEUSE-PARKING }

Action information :

XK (Gestion-déficience-mensuelle.Service-personnel / TRUE)
XK (Contrôle-de-légalité-journée. Service-personnel / TRUE)
XK (Contrôle-de-légalité-semaine.Service-personnel / TRUE)
XK (Gestion-des-absences-congés-récupérations.Service-personnel / TRUE)
XK (AfficherTemps (h, m, s, point).point' / point = point')
{ point : POINTEUSE-NORMALE, point' : POINTEUSE-NORMALE }

XK (AfficherTemps (h, m, s, point).point' / point = point')
{ point : POINTEUSE-PARKING, point' : POINTEUSE-PARKING }
XK (Déclencherporte (porte).prt / porte = prt)
XK (Enregistrement-manuel-de-prestation. Service-personnel / TRUE)

SERVICE-PERSONNEL



Déclaration de la structure de l'agent

State Components :

OK-mens instance-of BOOLEAN

OK-jour	instance-of	BOOLEAN
OK-hebd	instance-of	BOOLEAN
OK-acr	instance-of	BOOLEAN
OK-manuel	instance-of	BOOLEAN
Heure	instance-of	INTEGER
Minute	instance-of	INTEGER
Seconde	instance-of	INTEGER
Dates	instance-of	TYPEDATES
Motif	instance-of	TYPEMOTIF
Matricule	intance-of	INTEGER

Actions :

Demande-traitement-mensuel	
Introduire-manuellement-prestation	→ SYSTEME
Demande-traitement-journalier	
Introduire-absences-congés-récupérations	→ SYSTEME
Demande-traitement-hebdomadaire	
Lecture-de-traitement-journalier	→ SYSTEME
Lecture-de-traitement-mensuel	→ SYSTEME
Lecture-de-traitement-hebdomadaire	→ SYSTEME

Contraintes sur l'agent

Initial Valuation :

OK-mens = FALSE
OK-jour = FALSE
OK-manuel = FALSE
OK-acr = FALSE
OK-hebd = FALSE

Effects of Actions :

Syst.Gestion-déficience-mensuelle	:	OK-mens : = FALSE
Syst.Contrôle-de-légalité-journée	:	OK-jour : = FALSE
Syst.Contrôle-de-légalité-semaine	:	OK-hebd : = FALSE
Demande-traitement-mensuel	:	OK-mens : = TRUE
Demande-traitement-journalier	:	OK-jour : = TRUE
Demande-traitement-hebdomadaire	:	OK-hebd : = TRUE

Introduire-absences-congés-récupérations : Motif : = X ; Matricule : = Y ;
Dates : = Z ; OK-acr : = TRUE

Introduire-manuellement-prestation : Matricule : = Y ; Heure : = X ;
Minute : = Z ; Seconde : = W :
OK-manuel : = TRUE

Syst.Gestion-des-absences-congés-récupérations : OK-acr : = FALSE

Syst.Enregistrement-manuel-de-prestation : OK-manuel : = FALSE

Capability :

XO (Lecture-de-traitement-journalier / Syst.Légal-jour)
XO (Lecture-de-traitement-hebdomadaire / Syst.Légal-sem)
XO (Lecture-de-traitement-mensuel / Syst.Def-traitée)

Action perception :

XK (Syst.Gestion-déficience-mensuelle / TRUE)
XK (Syst.Contrôle-de-légalité-journée / TRUE)
XK (Syst.Contrôle-de-légalité-semaine / TRUE)
XK (Syst.Gestion-des-absences-congés-récupérations / TRUE)
XK (Syst.Enregistrement-manuel-de-prestation / TRUE)

State perception :

XK (Syst.Légal-jour / TRUE)
XK (Syst.Légal-sem / TRUE)
XK (Syst.Def-traitée / TRUE)
XK (Syst.Sortie-gest-def / TRUE)
XK (Syst.Sortie-leg-sem / TRUE)
XK (Syst.Sortie-leg-jour / TRUE)

Action Information :

XK (Lecture-de-traitement-journalier.Syst / TRUE)
XK (Lecture-de-traitement-hebdomadaire.Syst / TRUE)
XK (Lecture-de-traitement-mensuel.Syst / TRUE)

State Information :

XK (OK-mens.Syst / TRUE)
XK (OK-jour.Syst / TRUE)
XK (OK-hebd.Syst / TRUE)
XK (Matricule.Syst / TRUE)
XK (Heure.Syst / TRUE)
XK (Minute.Syst / TRUE)

XK (Seconde.Syst / TRUE)
XK (Motif.Syst / TRUE)
XK (Dates.Syst / TRUE)
XK (OK-acr.Syst / TRUE)
XK (OK-manuel.Syst / TRUE)

4.4. Le raffinement des contraintes non-fonctionnelles de sécurité et de performance.

Lors de l'acquisition des exigences, on a pu mettre à jour cinq besoins non-fonctionnels à prendre en compte. Nous allons maintenant raffiner ces besoins afin d'aboutir à un ensemble de solutions à intégrer dans les spécifications du système.

Les besoins non-fonctionnels établis étaient les suivants :

1. Assurer la sécurité du personnel. Il s'agit en fait d'assurer pour des raisons de sécurité physique et légale que le système ne sera pas une entrave à l'évacuation des locaux.
2. Assurer la performance d'évacuation. Le système doit permettre une bonne performance de l'évacuation des locaux.
3. Assurer la performance d'accès. Il faut que l'accès soit rapide pour éviter la formation de files d'attente devant les pointeuses.
4. Assurer la protection des pointages. Les enregistrements des horaires doivent être protégés afin d'assurer le bon fonctionnement de la fonction de contrôle de l'horaire .
5. Assurer le filtrage d'accès. Comme il avait été précisé dans les exigences informelles, il faut assurer une fonction de filtrage à l'entrée du bâtiment.

Par nécessité de simplicité, le problème de sélection d'alternative et de validation de solution ne sera pas développé ici.

a. Sécurité du personnel :

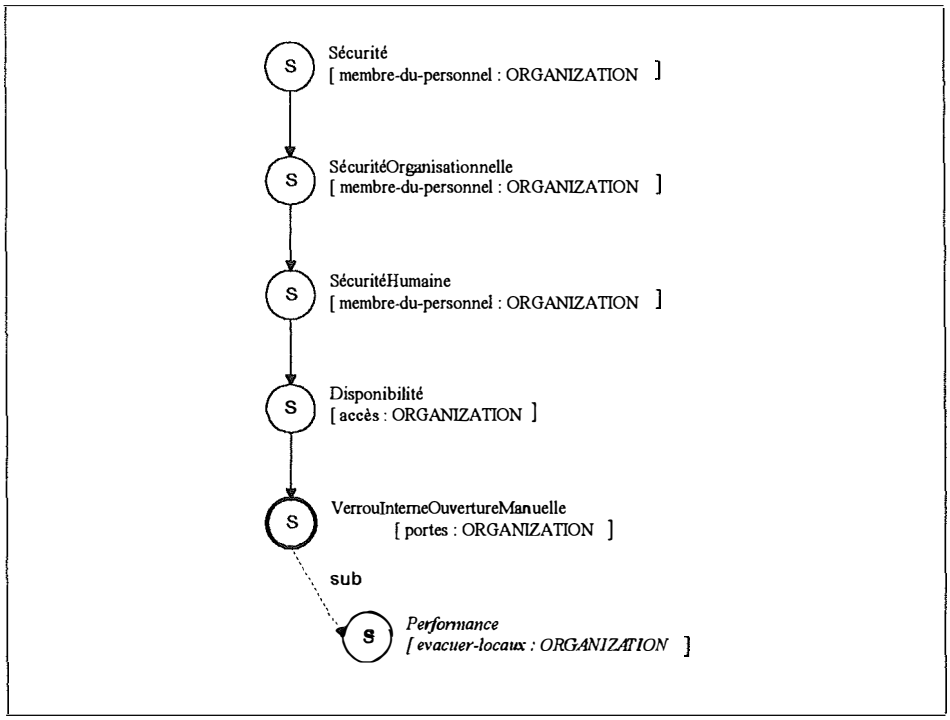


Figure 4. La sécurité du personnel.

Partant du besoin de fournir une solution au besoin de sécurité pour les membres du personnel modélisé par "Sécurité du personnel [membre du personnel : ORGANIZATION]" dans la figure 4, on a appliqué la taxonomie définie dans le chapitre 4. On a pu dès lors sélectionner la bonne décomposition jusqu'au but non-fonctionnel de SécuritéHumaine. Comme il s'agit d'assurer que le personnel puisse évacuer le bâtiment à tout moment, cela se concrétise par le raffinement de SécuritéHumaine en Disponibilité du moyen d'accès. Cela implique que l'accès ne doit pas être une entrave pour l'évacuation du personnel.

Afin de simplifier le cas, nous n'avons retenu qu'un but de satisfaisabilité. Il s'agit de poser un verrou manuel pour néanmoins autoriser la sortie si le système est en panne. On remarque également que le fait de choisir cette solution apporte une contribution nécessaire à la satisfaisabilité du besoin de performance dans l'évacuation des locaux.

b. Performance de l'évacuation :

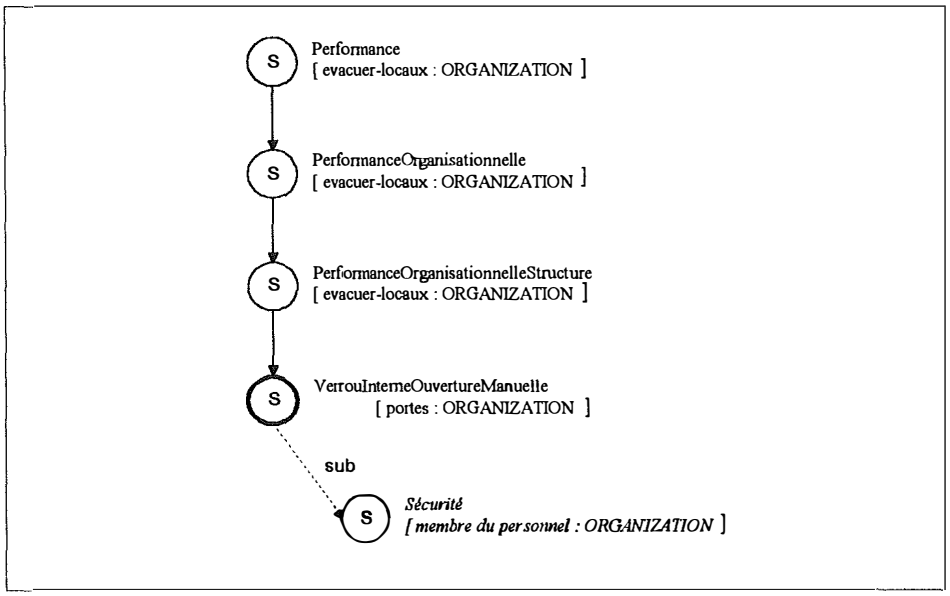


Figure 5. Performance de l'évacuation.

Le système doit permettre une bonne performance de l'évacuation des locaux. On remarque que cette performance touche à la structure organisationnelle. La solution identifiée est la même que pour le point précédant. On identifie aussi un lien de "contribution nécessaire" qui lie la solution de performance à l'exigence de sécurité des membres du personnel.

Ceci nous amène à faire deux remarques. Deux contraintes non-fonctionnelles différentes peuvent amenées à une même solution potentielle à adapter sur le système. Mais encore, on voit que performance et sécurité peuvent signifier la même chose, ou du moins poursuivre un même objectif dans certains cas. Ici, la sécurité du personnel quant aux possibilités d'évacuation passe par la performance dans le délai d'évacuation et la performance d'évacuation implique dans cette solution ci une sécurité pour le personnel.

c. Performance d'accès :

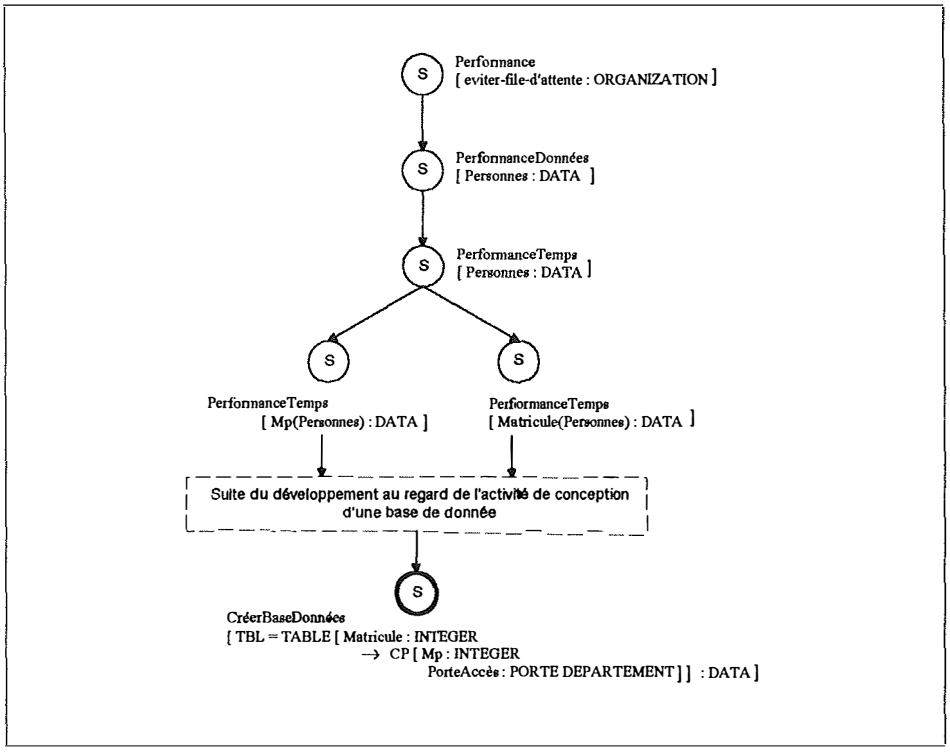


Figure 6. La performance d'accès.

Il faut assurer qu'il n'y aura pas de formation de files d'attente devant les pointeuses. On a orienté le raffinement vers le développement d'une solution liée à l'efficacité en temps pour les pointeuses de départements car elles doivent recevoir un mot de passe ce qui allonge le traitement. Une solution serait de créer une base de données à part de tous les enregistrements de l'horaire flottant pour en optimiser l'accès puisqu'il n'y a pas de mouvement à effectuer. (De nouveau cette solution ne serait pas suffisante dans un cas réel)

Remarquons que l'identification du paramètre Mp de Personnes vient du raffinement du besoin de filtrage d'accès (point e) dans lequel il a été décidé de recourir à une technique de mot de passe. Ceci montre bien que toute cette activité ne peut pas être perçue comme séquentielle D'autant plus que le raffinement de la performance d'accès influencera par lien de corrélation une décision dans le raffinement du filtrage.

d. Protection des pointages :

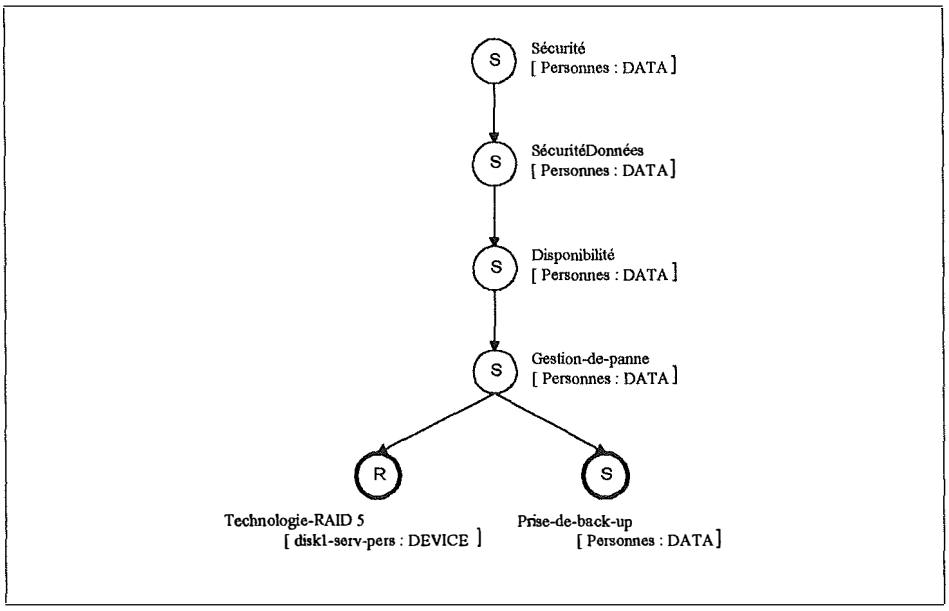


Figure 7. La protection des pointages.

On a identifié la sécurité de la base de données comme besoin pour assurance le bon fonctionnement de la fonction de contrôle de l'horaire. Cette exigence peut donc se décomposer en besoin de disponibilité de la base de données. Et principalement, il faut couvrir celle-ci par un mécanisme de gestion de pannes, ce qui peut être garanti par la prise de back-up ou par le recours à l'emploi de la technologie RAID. Dans ce cas précis, on se trouve devant deux alternatives. L'analyse de coût-efficacité devient nécessaire pour mettre à jour la solution la mieux adaptée à la situation.

Si l'on regarde l'allure générale du graphe de manière intuitive (figure 8), on observe que la perte en cas de panne ne devrait pas être trop élevée alors que les solutions apportées peuvent coûter très cher. En ce sens, si l'on prend la technologie RAID comme moyen d'assurer la disponibilité de la base de donnée, le coût sera (toujours de manière intuitive) trop important par rapport au gain fourni (la disponibilité de l'horaire flottant n'est pas quelque chose de vital pour une entreprise). Par conséquent, si l'analyse était faite en toute rigueur, la solution adoptée pencherait vers le choix du back-up. Solution que nous allons retenir.

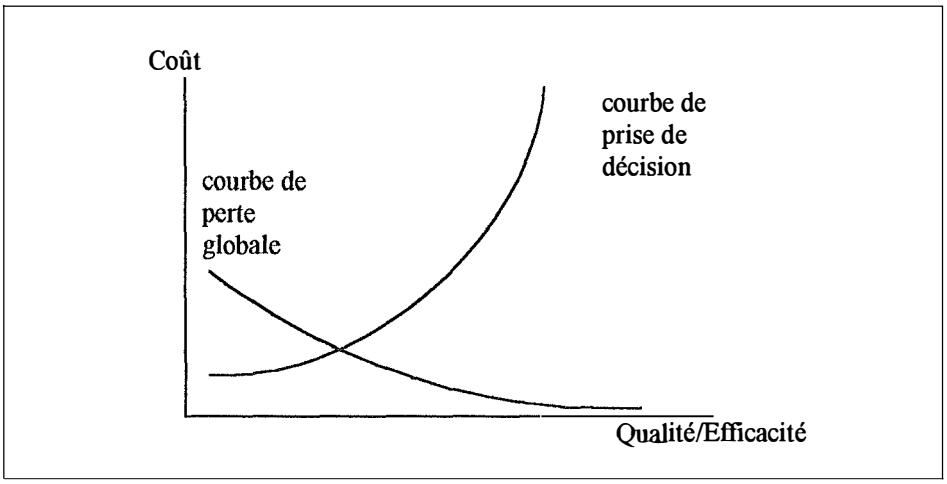


Figure 8. Graphe de coût/efficacité.

On remarquera que la sécurité demandée pour un élément de donnée peut être assuré par une solution apportée au niveau de l'architecture informatique. Ainsi, ce n'est pas parce que l'on recherche une couverture pour un besoin non-fonctionnel de type bien particulier que cette couverture impliquera seulement les éléments de ce type là.

e. Filtrage d'accès :

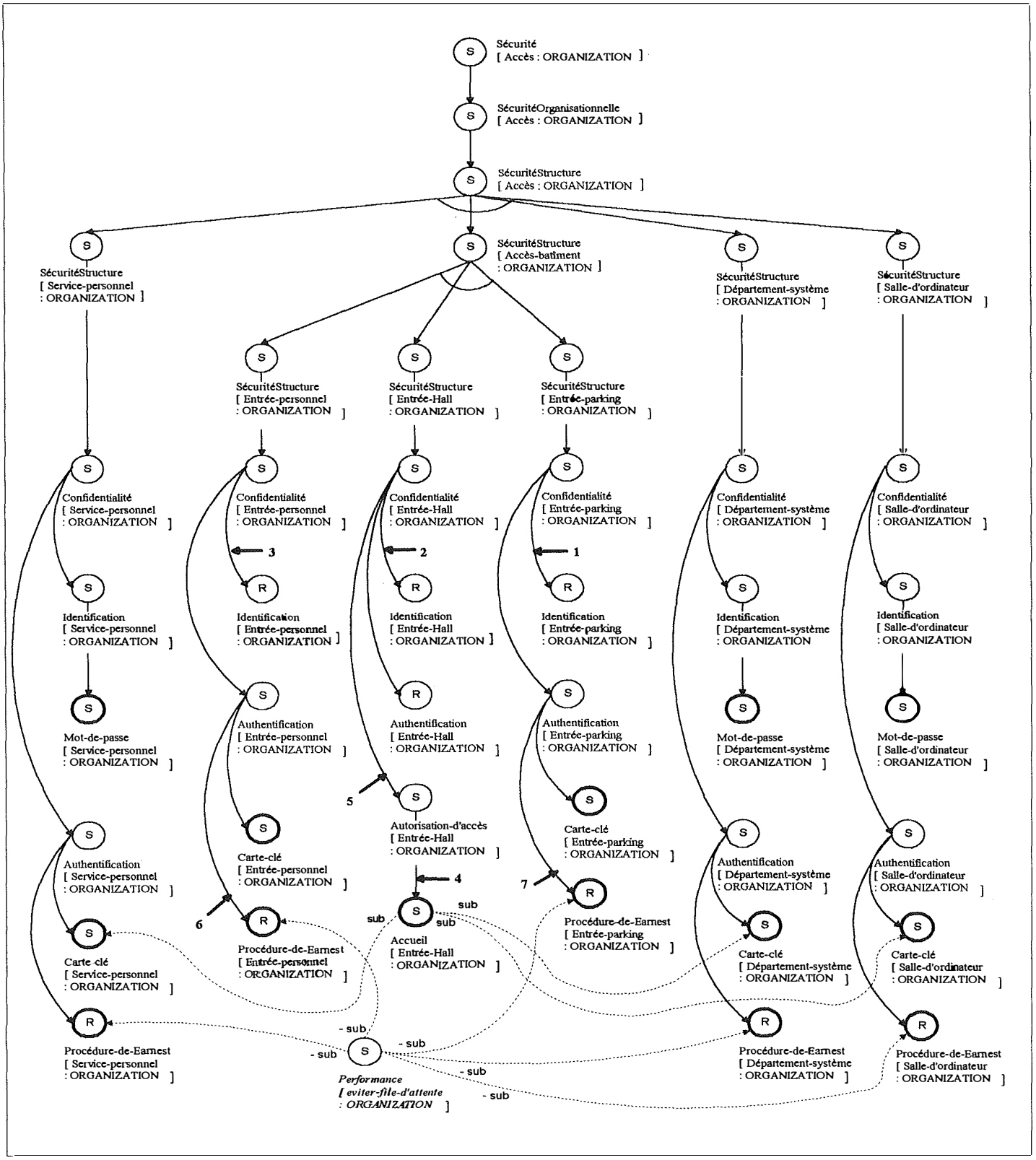


Figure 9. Le filtrage d'accès.

La maintenance évolutive ou adaptative ainsi que la définition d'un nouveau projet peut avoir pour conséquence de modifier les circonstances qui ont été à la base de prise de décisions de conception lors du développement d'autres projets. Ainsi, dans notre cas, la protection de départements avait été établie dans le précédant projet de gestion de l'horaire flottant. Puisque nous modifions les éléments qui avaient été à la base de la décision de l'époque, il nous faut reconsidérer la protection de ses départements dans le projet actuel. En effet, toute la protection de ses départements reposait sur le fait qu'un contrôle d'authentification par mot de passe était effectué à l'entrée du bâtiment. Si nous en arrivons à modifier cette état des faits, la protection perdra toute sa consistance.

Un autre élément important est fourni par les contraintes (voir chapitre 4, page 4.6) qui influencent le traitement des exigences de sécurité. Il s'agit de la motivation du personnel, qui ayant subi un système inadapté n'est plus apte à fournir des efforts dont il ne cible pas bien l'intérêt. Seule les membres du personnel des départements à protéger sont déclarées prêts à accepter une solution plus lourde vu qu'ils sont impliqués par le besoin de sécurité de leur département. Cette constatation permet de définir les buts d'argumentation 6, 7 déclarant que le personnel ne veut pas assumer une solution lourde. Rejetant par-là même la proposition d'établir un système de mot de passe selon la technique de Earnest.

Nous pouvons considérer également que les techniques de sécurité basées sur la simple identification sont jugées insuffisantes. Ce qui est exprimé par les buts d'argumentation 1, 2, 3. Néanmoins, pour ce qui concerne la sécurité du hall d'entrée, on estime que les personnes qui utilisent cette entrée sont externes et ne font pas l'objet d'un accès courant. Par conséquent, l'accès par une méthode d'autorisation d'accès serait plus adaptée. Cette méthode passe par une utilisation d'un service d'accueil. Ce raffinement est de plus renforcé par le fait qu'un accueil existe déjà dans la situation actuelle de la société. Cela est exprimé par le but d'argumentation 4.

On trouve également un lien de corrélation -sub liant la recherche d'une performance visant à éviter la constitution d'une file d'attente à l'utilisation de la procédure d'Earnest jugée

trop coûteuse. Et finalement le choix d'un accueil pour le hall d'entrée impliquerait la nécessité d'utiliser une carte clé afin de garantir la sécurité des départements.

La procédure d'étiquetage retient de tout cela une solution possible impliquant :

- un accès à toutes les entrées (bâtiment, départements) par carte clé
- un mot de passe pour les départements
- l'utilisation d'un service d'accueil dans le hall.

4.5. Répercussion du traitement des exigences non-fonctionnelles

a. Déclaration des types de données :

TBL-MP = TABLE [Matricule : INTEGER
→ CP [Mp : INTEGER,
PorteAccès : PORTE DEPARTEMENT]]

| On part de l'hypothèse qu'une personne authorisée ne peut l'être que pour un seul des
| départements concernés par la sécurité.

b. Déclaration des opérations :

AssignerCarte : PERSONNE → CARTE-CLE ;

| Nous indiquons par le biais d'une opération, le fait que une seule carte clé est associée à une seule personne.
| Ce mécanisme est utilisé afin de gérer les contraintes d'ordre globales avec ALBERT.

AssignerCarte (p) = val
with
$\neg \exists$ p' : (p ≠ p')
$\wedge$ (AssignerCarte (p') = val)

AssignerPorteDep : POINTEUSE-DEPARTEMENT → PORTE-DEPARTEMENT ;

| Nous indiquons par le biais d'une opération, le fait que une seule porte département est associée à une seule
| pointeuse département.
| Ce mécanisme est utilisé afin de gérer les contraintes d'ordre globales avec ALBERT.

AssignerPorteDep (p) = val
with
$\neg \exists$ p' : (p $\neq$ p')
$\wedge$ (AssignerPorteDep (p') = val)

AssignerPorteNormale : POINTEUSE-NORMALE $\rightarrow$ PORTE-NORMALE ;

| Nous indiquons par le biais d'une opération, le fait que une seule porte normale (d'accès au batiment) est
| associée à une seule pointeuse normale.
| Ce mécanisme est utilisé afin de gérer les contraintes d'ordre globales avec ALBERT.

AssignerPorteNormale (p) = val
with
$\neg \exists$ p' : (p $\neq$ p')
$\wedge$ (AssignerPorteNormale (p') = val)

Est-valide : INTEGER x INTEGER x TBL-MP x PORTE-DEPARTEMENT
$\rightarrow$ BOOLEAN ;

| Renvoie TRUE si le mot de passe *mp* est bien celui de la personne ayant le matricule *mat* et ayant
| effectivement accès à cette porte *p*.

Est-valide (mat, mp, table-mp, p) = O
with
$\exists$ i : In (i, table-mp)
$\wedge$ (i = mat)
$\wedge$ (mp = Mp (table-mp[i])
$\wedge$ (p = PorteAccès (table-mp[i])
$\Rightarrow$ O = TRUE

$\vee$ $\neg \exists$ i : In (i, table-mp)
$\wedge$ (i = mat)
$\wedge$ (mp = Mp (table-mp[i])
$\wedge$ (p = PorteAccès (table-mp[i])
$\Rightarrow$ O = FALSE

Est-val : INTEGER x TBL-MP $\rightarrow$ BOOLEAN ;

| Renvoie TRUE si le matricule *mat* est bien un matricule répertorié dans la base des accès *table-mp.*

Est-val (mat, table-mp) = O
with

$\exists$ i : In (i, table-mp)
$\wedge$ (i = mat)
$\Rightarrow$ O = TRUE

$\vee$ $\neg \exists$ i : In (i, table-mp)
$\wedge$ (i = mat)
$\Rightarrow$ O = FALSE

GénèreHeure : D-BASE x INTEGER $\rightarrow$ INTEGER ;

| Le but consiste à extraire pour un enregistrement, identifié par *mat* dans la base de données *I*, le nombre
| d'heures qui sont contenues dans Temps-cumulé-global.
| *On traite maintenant le fait qu'un matricule puisse être faux, par conséquent si cela est le cas on renvoie la*
| *solution avec la valeur zéro*

GénèreHeure (I, mat) = O
with
$\exists$ i : In (i , I)
$\wedge$ i = mat
$\wedge$ Temps-cumulé-global (I[i]) $\geq$ 0
$\Rightarrow$ O = [(Temps-cumulé-global (I[i]) - GénèreSeconde (I, mat)) / 60
- GénèreMinute (I, mat)] / 60

$\vee$ $\exists$ i : In (i , I)
$\wedge$ i = mat
$\wedge$ Temps-cumulé-global (I[i]) < 0
$\Rightarrow$ O = [(Temps-cumulé-global (I[i]) + GénèreSeconde (I, mat)) / 60
+ GénèreMinute (I, mat)] / 60

$\vee$ $\neg \exists i : In (i, table\text{-}mp)$
$\wedge (i = mat)$
$\Rightarrow O = 0$

GénèreMinute : D-BASE x INTEGER $\rightarrow$ INTEGER ;

| Le but consiste à extraire pour un enregistrement, identifié par *mat* dans la base de données *I*, le nombre de
| minutes qui sont contenues dans Temps-cumulé-global.
| *On traite maintenant le fait qu'un matricule puisse être faux, par conséquent si cela est le cas on renvoie la*
| *solution avec la valeur zéro*

GénèreMinute (I, mat) = O
with
$\exists$ i : In (i , I)
$\wedge$ i = mat
$\wedge$ Temps-cumulé-global (I[i]) $\geq$ 0
$\Rightarrow$ O = [(Temps-cumulé-global (I[i])

- GénèreSeconde (I, mat)) / 60] mod 60

∨ ∃ i : In (i , I)
∧ i = mat
∧ Temps-cumulé-global (I[i]) < 0
⇒ O = [(Temps-cumulé-global (I[i])
+ GénèreSeconde (I, mat)) / 60] mod 60

∨ ¬ ∃ *i : In (i, table-mp)*
∧ (i = mat)
⇒ O = 0

GénèreSeconde : D-BASE x INTEGER → INTEGER ;

| Le but consiste à extraire pour un enregistrement, identifié par *mat* dans la base de données *I*, le nombre de
| secondes qui sont contenues dans Temps-cumulé-global si l'on exprime cette valeur de Temps-cumulé-global
| en heures/minutes/secondes.
| *On traite maintenant le fait qu'un matricule puisse être faux, par conséquent si cela est le cas on renvoie la*
| *solution avec la valeur zéro*

GénèreSeconde (I, mat) = O
with
∃ i : In (i , I)
∧ i = mat
⇒ O = Temps-cumulé-global (I[i]) mod 60

∨ ¬ ∃ *i : In (i, table-mp)*
∧ (i = mat)
⇒ O = 0

InMiss : D-BASE x INTEGER x INTEGER x INTEGER x INTEGER → D-BASE ;

| Pour une instance de la base de données *I* déterminée par le matricule *mat* de la personne concernée et une
| heure *h*, minute *m*, seconde *s*, le but est d'enregistrer ce temps comme moment d'arrivée de mission.
| *On traite maintenant le fait qu'un matricule puisse être faux, par conséquent si cela est le cas on ne*
| *modifie pas la base de données*

InMiss (I, mat, h, m, s) = O
with
∃ i : In (i, I)
∧ i = mat
⇒ O = Modify (I, mat, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = FALSE

∧ Mouvement (rens')
= Append (Mouvement (rens), < <h, m, s>, In-miss >)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hendomadaire (rens)
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
∧ rens = I[mat]

∨ $\neg \exists i : In(i, table\text{-}mp) \wedge (i = mat) \Rightarrow O = I$

InNormal : D-BASE x INTEGER x INTEGER x INTEGER x INTEGER → D-BASE ;

| Pour une instance de la base de données *I* déterminée par le matricule *mat* de la personne concernée et une
| heure *h*, minute *m*, seconde *s*, le but est d'enregistrer ce temps comme moment d'arrivée normale.
| *On traite maintenant le fait qu'un matricule puisse être faux, par conséquent si cela est le cas on ne*
| *modifie pas la base de données*

InNormal (I, mat, h, m, s) = O
with
∃ i : In (i, I)
∧ i = mat
⇒ O = Modify (I, mat, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = Est-en-mission (rens)
∧ Mouvement (rens')
= Append (Mouvement (rens), < <h, m, s>, In >)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hendomadaire (rens)
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
∧ rens = I[mat]

$\vee \quad \neg \exists i : In(i, table\text{-}mp) \wedge (i = mat) \Rightarrow O = I$

OutMiss : D-BASE x INTEGER x INTEGER x INTEGER x INTEGER → D-BASE ;

| Pour une instance de la base de données *I* déterminée par le matricule *mat* de la personne concernée et une
| heure *h*, minute *m*, seconde *s*, le but est d'enregistrer ce temps comme moment de sortie mission.
| *On traite maintenant le fait qu'un matricule puisse être faux, par conséquent si cela est le cas on ne*
| *modifie pas la base de données*

OutMiss (I, mat, h, m, s) = O
with
∃ i : In (i, I)
∧ i = mat
⇒ O = Modify (I, mat, rens')
∧ Nom (rens') = Nom (rens)
∧ Prénom (rens') = Prénom (rens)
∧ Est-en-mission (rens') = TRUE
∧ Mouvement (rens')
= Append (Mouvement (rens), < <h, m, s>, Out-miss >)
∧ Congé-date (rens') = Congé-date (rens)
∧ Absence-date (rens') = Absence-date (rens)
∧ Récup-date (rens') = Récup-date (rens)
∧ Temps-total-hebdomadaire (rens')
= Temps-total-hendomadaire (rens)
∧ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
∧ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
∧ rens = I[mat]

$\vee \quad \neg \exists i : In(i, table\text{-}mp) \wedge (i = mat) \Rightarrow O = I$

OutNormal : D-BASE x INTEGER x INTEGER x INTEGER x INTEGER → D-BASE ;

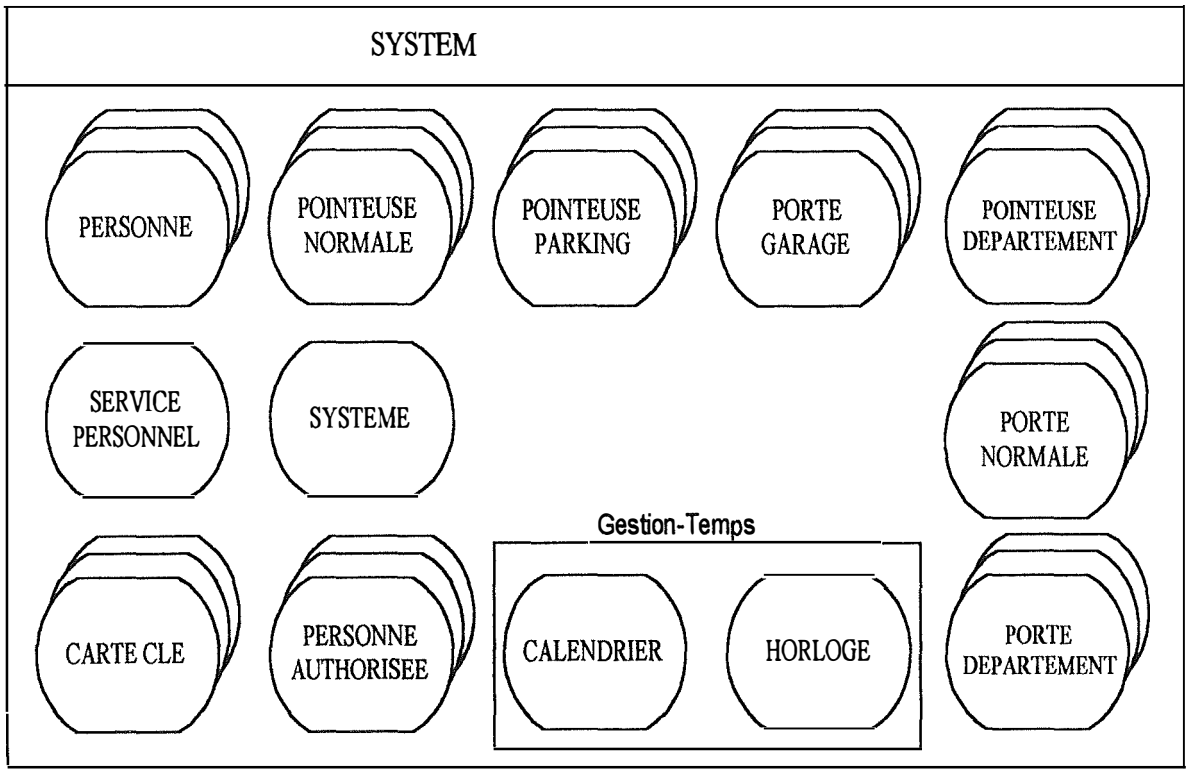
| Pour une instance de la base de données *I* déterminée par le matricule *mat* de la personne concernée et une
| heure *h*, minute *m*, seconde *s*, le but est d'enregistrer ce temps comme moment de sortie normale.
| *On traite maintenant le fait qu'un matricule puisse être faux, par conséquent si cela est le cas on ne*
| *modifie pas la base de données*

OutNormal (I, mat, h, m, s) = O
with

$\exists$ i : In (i, I)
$\wedge$ i = mat
$\Rightarrow$ O = Modify (I, mat, rens')
$\wedge$ Nom (rens') = Nom (rens)
$\wedge$ Prénom (rens') = Prénom (rens)
$\wedge$ Est-en-mission (rens') = Est-en-mission (rens)
$\wedge$ Mouvement (rens')
= Append (Mouvement (rens), < <h, m, s>, Out >)
$\wedge$ Congé-date (rens') = Congé-date (rens)
$\wedge$ Absence-date (rens') = Absence-date (rens)
$\wedge$ Récup-date (rens') = Récup-date (rens)
$\wedge$ Temps-total-hebdomadaire (rens')
= Temps-total-hendomadaire (rens)
$\wedge$ Temps-cumulé-mois-antérieur (rens')
= Temps-cumulé-mois-antérieur (rens)
$\wedge$ Temps-cumulé-global (rens')
= Temps-cumulé-global (rens)
$\wedge$ rens = I[mat]

$\vee$ $\neg \exists i : In (i, table\text{-}mp)$
$\wedge (i = mat)$
$\Rightarrow O = I$

c. Déclaration de société :

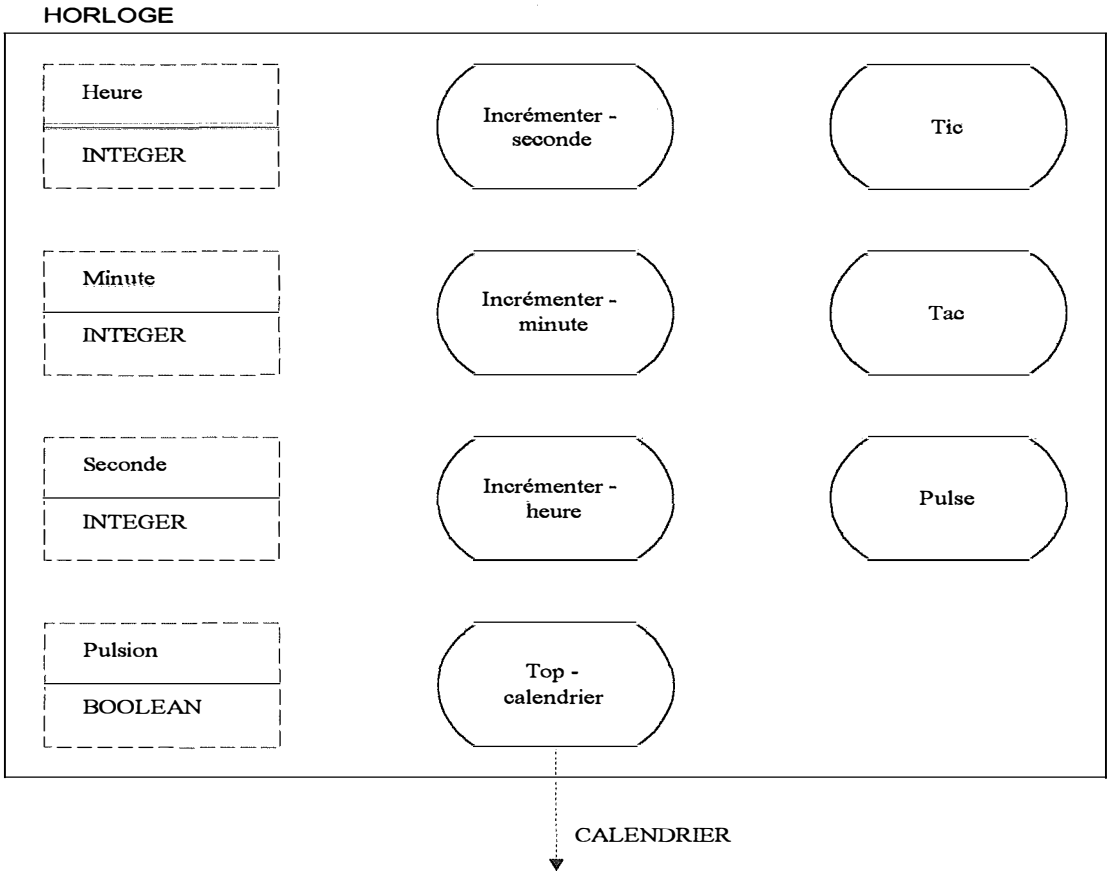


Remarques générales:

- Les modifications ou les nouveaux éléments de la spécification ont été globalement mis en italiques.
- Un mécanisme de vérification de validité des matricules et des mots de passe a été introduit dans l'agent système.
- Afin de garantir la sécurité d'accès telle que raffinée dans le point précédent, on a ajouté les agents porte normale et porte département qui figure les portes liées aux agents pointeuse normale et département.
- La carte clé a été spécifiée et reçoit le matricule comme composant d'état.
- La table des mots de passe a été également spécifiée afin de prendre en compte les besoins de performance d'accès.
- Un verrou a été ajouté au différents agents porte afin de prendre en compte la solution au problème de sécurité du personnel et de performance d'évacuation.
- Toutefois, le mécanisme de sauvegarde de la base de données n'a pas été spécifié afin de ne pas alourdir de trop l'illustration.

d. Spécification des agents :

HORLOGE



Déclaration de la structure de l'agent

State Components :

Heure	instance-of	INTEGER
Minute	instance-of	INTEGER
Seconde	instance-of	INTEGER
Pulsion	instance-of	BOOLEAN

Actions :

Tic
Incrémenter-heure
Tac
Incrémenter-minute

Pulse
Incrémenter-seconde
Top-calendrier → CALENDRIER

Contraintes sur l'agent

Initial Valuation :

Heure = 0
Minute = 0
Seconde = 0
Pulsion = FALSE

State Behaviour :

Heure ≥ 0 ∧ Heure ≤ 23
Minute ≥ 0 ∧ Minute ≤ 59
Seconde ≥ 0 ∧ Seconde ≤ 59

Effects of Actions :

Pulse	:	Pulsion : = TRUE
Incrémenter-seconde	:	Pulsion : = FALSE ; Seconde : = Seconde + 1
Incrémenter-minute	:	Pulsion : = FALSE ; Seconde : = 0 ; Minute : = Minute + 1
Incrémenter-heure	:	Pulsion : = FALSE ; Seconde : = 0 ; Minute : = 0 ; Heure : = Heure + 1
Top-calendrier	:	Pulsion : = FALSE ; Seconde : = 0 ; Minute : = 0 ; Heure : = 0

Capability :

XO (Incrémenter-seconde / Pulsion = TRUE ∧
Seconde < 59)

XO (Incrémenter-minute / Pulsion = TRUE ∧
Seconde = 59 ∧
Minute < 59)

XO (Incrémenter-heure / Pulsion = TRUE ∧
Seconde = 59 ∧
Minute = 59 ∧
Heure < 23)

XO (Top-calendrier / Pulsion = TRUE ∧
Seconde = 59 ∧
Minute = 59 ∧
Heure = 23)

Action Composition :

Tic <-> Pulse ; Tac

Tac <-> Pulse ; Tic

Action Duration :

| Pulse | = 1 sec

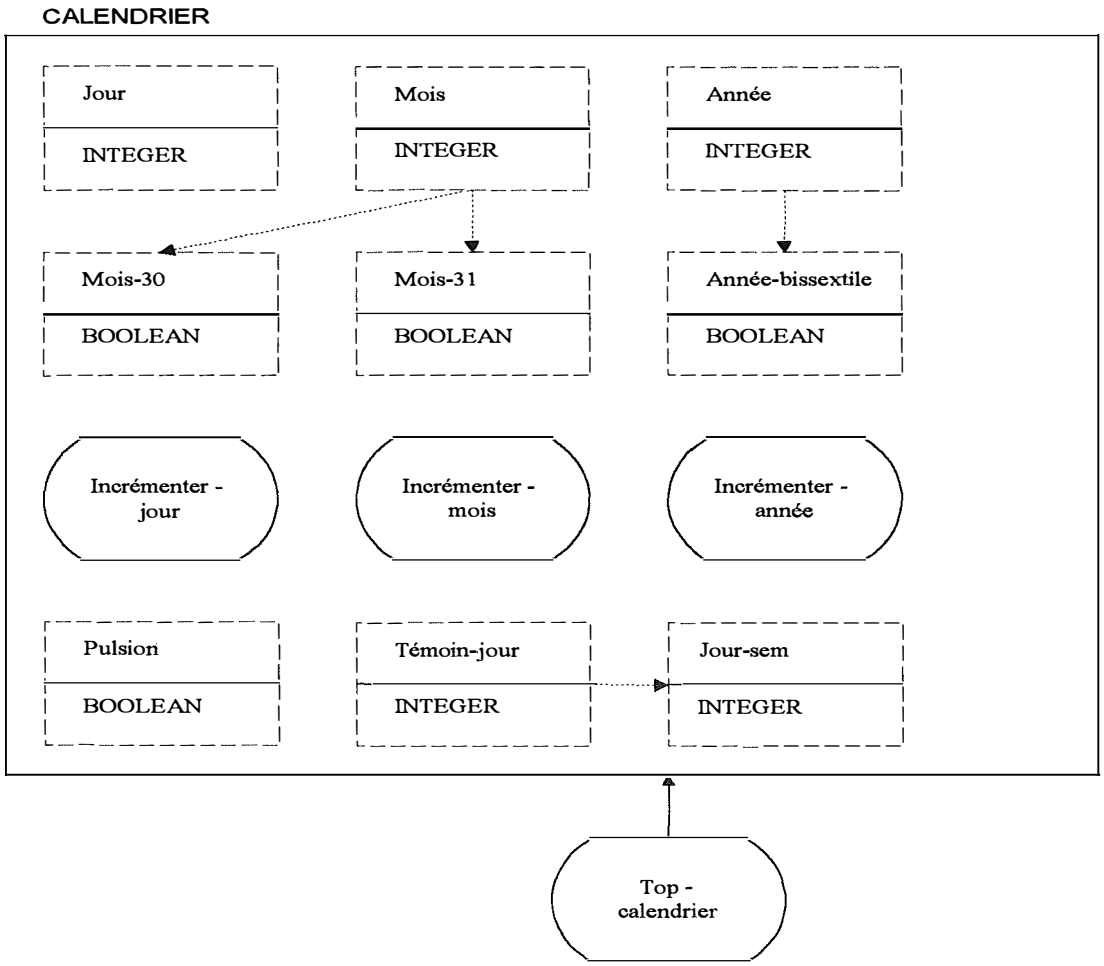
Action Information :

XK (Top-calendrier.Cal / TRUE)

State Information :

XK (Heure.Syst / TRUE)
XK (Minute.Syst / TRUE)
XK (Seconde.Syst / TRUE)

CALENDRIER



Déclaration de la structure de l'agent

State Composents :

Jour	instance-of	INTEGER
Mois	instance-of	INTEGER
Année	instance-of	INTEGER
Mois-30	instance-of	BOOLEAN
Mois-31	instance-of	BOOLEAN
Année-bissextile	instance-of	BOOLEAN
Jour-sem	instance-of	INTEGER
Témoin-jour	instance-of	INTEGER
Pulsion	instance-of	BOOLEAN

Actions :

Incrémenter-jour
Incrémenter-mois
Incrémenter-année

Contraintes sur l'agent

Derived Components :

Mois-31 $\triangleq$ Mois = 1 $\vee$ Mois = 3 $\vee$ Mois = 5 $\vee$ Mois = 7 $\vee$ Mois = 8 $\vee$ Mois = 10 $\vee$ Mois = 12

Mois-30 $\triangleq$ Mois = 4 $\vee$ Mois = 6 $\vee$ Mois = 9 $\vee$ Mois = 11

Année-bissextile $\triangleq$ Année = 1996 + (4 * valeur)

Jour-sem $\triangleq$ (Témoin-jour mod 7) + 1

Initial Valuation :

Jour = 1
Mois = 1
Année = 1996
Jour-sem = Lundi
Témoin-jour = 1
Pulsion = FALSE

State Behaviour :

Jour $\geq$ 1 $\wedge$ Jour $\leq$ 31
Mois $\geq$ 1 $\wedge$ Mois $\leq$ 12
Jour-sem $\geq$ 1 $\wedge$ Jour-sem $\leq$ 7
Année $\geq$ 1996

Effects of Actions :

Hor.Top-calendrier	:	Pulsion : = TRUE
Incrémenter-jour	:	Pulsion : = FALSE ; Jour : = Jour + 1 ; Témoin-jour : = Témoin-jour + 1
Incrémenter-mois	:	Pulsion : = FALSE ; Jour : = 1 ; Mois : = Mois + 1 Témoin-jour : = Témoin-jour + 1

Incrémenter-année : Pulsion : = FALSE ; Jour : = 1 ; Mois : = 1 ;
Année : = Année + 1
Témoin-jour : = Témoin-jour + 1

Capability :

XO (Incrémenter-jour / (Pulsion = TRUE ∧ Mois-31 = TRUE ∧ Jour < 31)
∨ (Pulsion = TRUE ∧ Mois-30 = TRUE ∧ Jour < 30)
∨ (Pulsion = TRUE ∧ Mois-31 = FALSE ∧
Mois-30 = FALSE ∧
Année-bissextile = FALSE ∧ Jour < 28)
∨ (Pulsion = TRUE ∧ Mois-31 = FALSE ∧
Mois-30 = FALSE ∧
Année-bissextile = TRUE ∧ Jour < 29))

XO (Incrémenter-mois / Pulsion = TRUE ∧ Mois ≠ 12 ∧
((Mois-30 = TRUE ∧ Jour = 30)
∨ (Mois-31 = TRUE ∧ Jour = 31)
∨ (Année-bissextile = TRUE ∧ Mois = 2 ∧
Jour = 29)
∨ (Année-bissextile = FALSE ∧ Mois = 2 ∧
Jour = 28))

XO (Incrémenter-année / Pulsion = TRUE ∧ Mois = 12 ∧ Jour = 31)

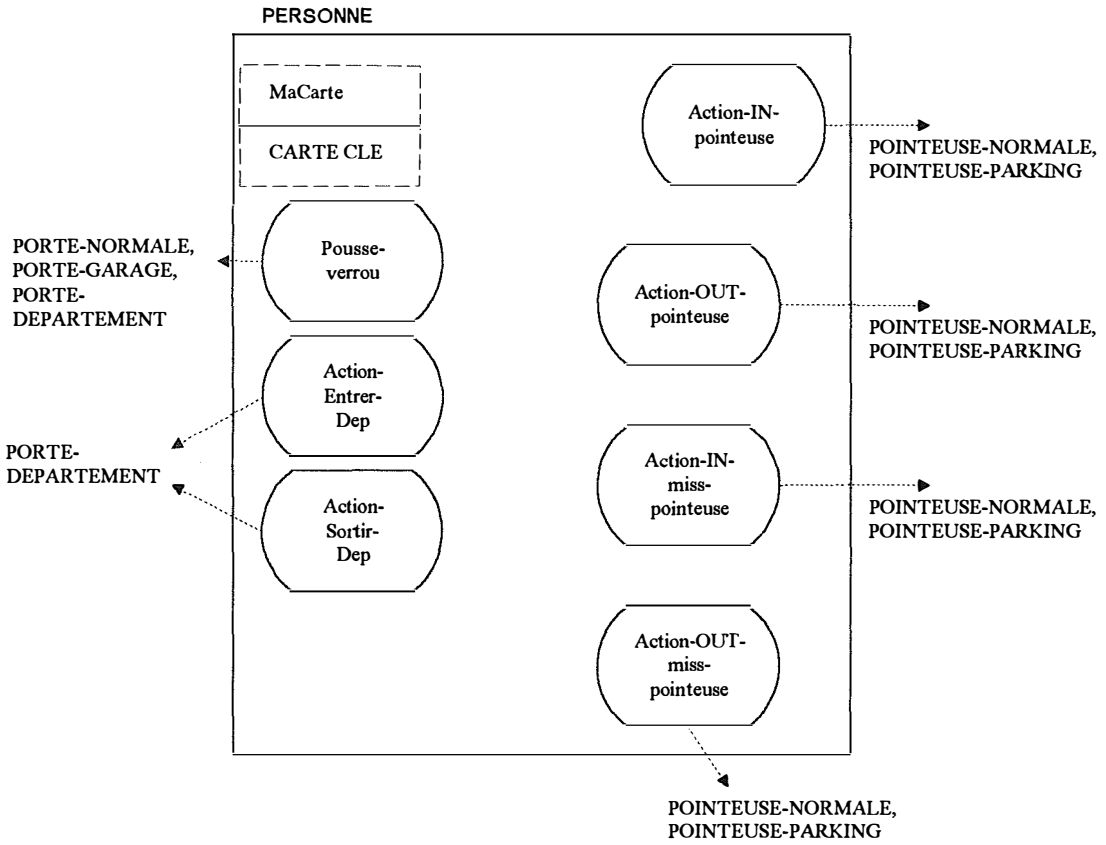
Action Perception :

XK (Hor.Top-calendrier / Pulsion)

State Information :

XK (Jour-sem.Syst / TRUE)
XK (Jour.Syst / TRUE)
XK (Mois.Syst / TRUE)
XK (Année.Syst / T RUE)

PERSONNE



Déclaration de la structure de l'agent

State Components :

MaCarte *instance-of* *CARTE-CLE*

Actions :

Pousse-verrou → *PORTE-NORMALE,*
PORTE-GARAGE, PORTE-DEPARTEMENT
Action-Entrer-Dep (CARTE-CLE, INTEGER)
→ *POINTEUSE-DEPARTEMENT**
Action-Sortie-Dep (CARTE-CLE) → *POINTEUSE-DEPARTEMENT**
Action-IN-pointeuse (CARTE-CLE)
→ *POINTEUSE-NORMALE**, *POINTEUSE-PARKING**
Action-IN-miss-pointeuse (CARTE-CLE)
→ *POINTEUSE-NORMALE**, *POINTEUSE-PARKING**
Action-OUT-miss-pointeuse (CARTE-CLE)
→ *POINTEUSE-NORMALE**, *POINTEUSE-PARKING**

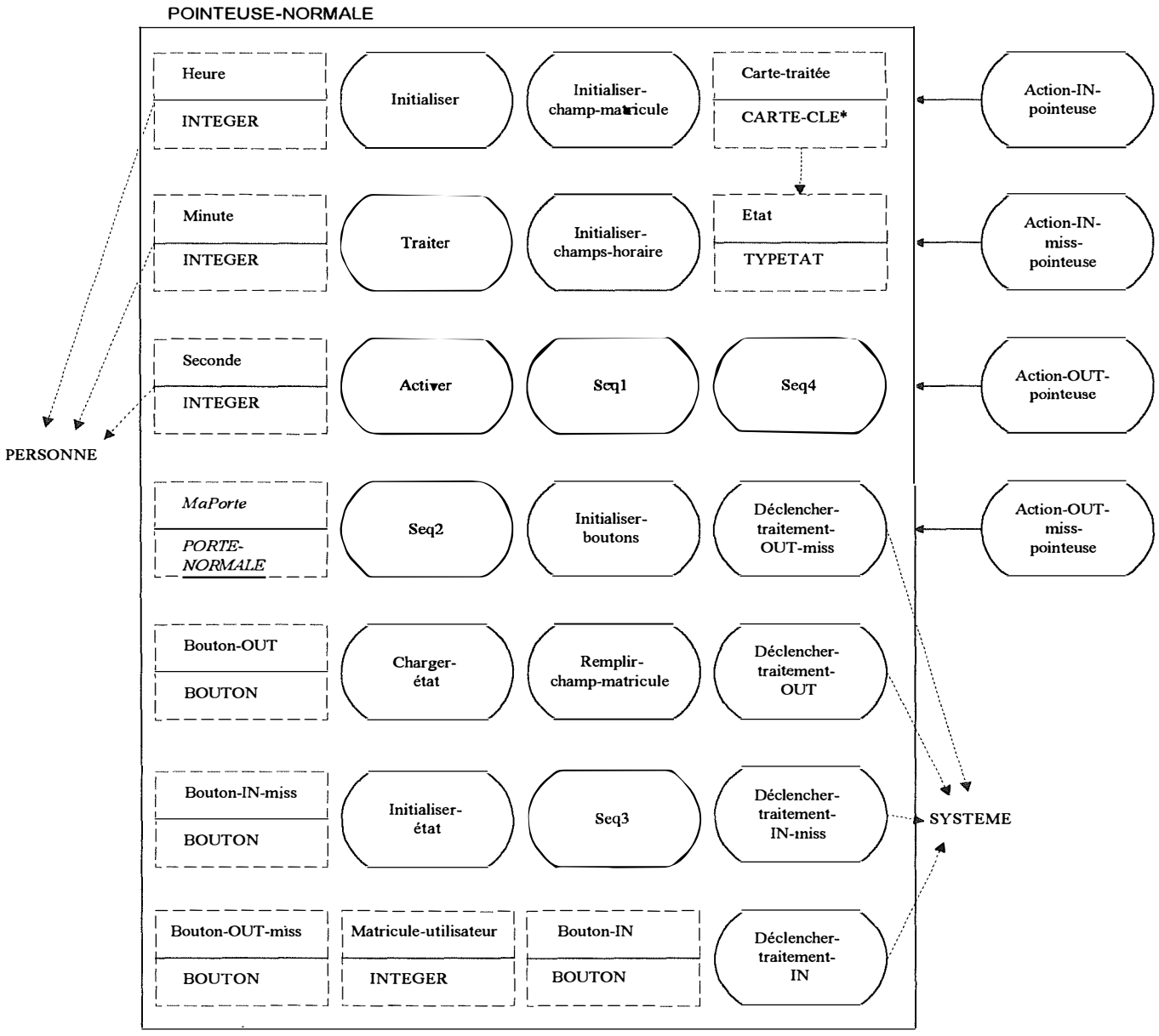
Action-OUT-pointeuse (CARTE-CLE)
→ POINTEUSE-NORMALE , POINTEUSE-PARKING**

Contraintes sur l'agent

Action Information :

XK (Action-IN-pointeuse*(MaCarte)*.Pn / TRUE)
{ Pn : POINTEUSE-NORMALE }
XK (Action-OUT-pointeuse*(MaCarte)*.Pn / TRUE)
{ Pn : POINTEUSE-NORMALE }
XK (Action-IN-miss-pointeuse*(MaCarte)*.Pn / TRUE)
{ Pn : POINTEUSE-NORMALE }
XK (Action-OUT-miss-pointeuse*(MaCarte)*.Pn / TRUE)
{ Pn : POINTEUSE-NORMALE }
XK (Action-IN-pointeuse*(MaCarte)*.Pp / TRUE)
{ Pp : POINTEUSE-PARKING }
XK (Action-OUT-pointeuse*(MaCarte)*.Pp / TRUE)
{ Pp : POINTEUSE-PARKING }
XK (Action-IN-miss-pointeuse*(MaCarte)*.Pp / TRUE)
{ Pp : POINTEUSE-PARKING }
XK (Action-OUT-miss-pointeuse*(MaCarte)*.Pp / TRUE)
{ Pp : POINTEUSE-PARKING }
XK (Action-Entrer-Dep (MaCarte, X).Point / TRUE)
XK (Action-Sortir-Dep (MaCarte).Point / TRUE)
XK (Pousse-verrou.Pg / TRUE) { Pg : PORTE-GARAGE }
XK (Pousse-verrou.Pdp / TRUE) { Pdp : PORTE-DEPARTEMENT }
XK (Pousse-verrou.Pn / TRUE) { Pn : PORTE-NORMALE }

POINTEUSE-NORMALE



Déclaration de la structure de l'agent

State Components :

Heure	instance-of	INTEGER
Minute	instance-of	INTEGER
Seconde	instance-of	INTEGER
MaPorte	*instance-of*	*PORTE-NORMALE*

Carte-traitée	*instance-of*	*CARTE-CLE**
Matricule-utilisateur	instance-of	INTEGER
Etat	instance-of	TYPETAT
Bouton-IN	instance-of	BOUTON
Bouton-IN-miss	instance-of	BOUTON
Bouton-OUT	instance-of	BOUTON
Bouton-OUT-miss	instance-of	BOUTON

Actions :

Traiter
Initialiser-champs-horaire
Activer (CARTE-CLE)
Initialiser-champ-matricule
Initialiser-état
Changer-état (CARTE-CLE)
Initialiser-boutons
Initialiser
Remplir-champ-matricule (CARTE-CLE)
Seq1
Seq2
Seq3
Seq4
Déclencher-traitement-IN (INTEGER) → SYSTEME*
Déclencher-traitement-IN-miss (INTEGER) → SYSTEME*
Déclencher-traitement-OUT (INTEGER) → SYSTEME*
Déclencher-traitement-OUT-miss (INTEGER) → SYSTEME*

Contraintes sur l'agent

Derived components :

Etat ≙ *Carte-traitée = UNDEF*

Initial Valuation :

Heure = 0	Matricule-utilisateur = 0
Minute = 0	Bouton-IN = Relaché
Seconde = 0	Bouton-IN-miss = Relaché
Carte-traitée = UNDEF	Bouton-OUT = Relaché
Bouton-OUT-miss = Relaché	

State Behaviour :

Minute ≥ 0 ∧ Minute ≤ 59
Seconde ≥ 0 ∧ Seconde ≤ 59

Effects of Actions :

Pers.Action-IN-pointeuse : Bouton-IN : = Poussé

Pers.Action-IN-miss-pointeuse : Bouton-IN-miss : = Poussé

Pers.Action-OUT-pointeuse : Bouton-OUT : = Poussé

Pers.Action-OUT-miss-pointeuse : Bouton-OUT-miss : = Poussé

Syst.AfficherTemps (h, m, s, point) : Heure : = h ;
Minute : = m ;
Seconde : = s

Initialiser-champs-horaire : Heure : = 0 ;
Minute : = 0 ;
Seconde : = 0

Initialiser-champ-matricule : Matricule-utilisateur : = 0

Initialiser-état : Carte-traitée : = UNDEF

Remplir-champ-matricule *(carte)* : Matricule-utilisateur : = *carte*.Matricule

Initialiser-bouton : Bouton-IN : = Relaché ;
Bouton-IN-miss : = Relaché ;
Bouton-OUT : = Relaché ;
Bouton-OUT-miss : = Relaché

Changer-état (carte) : Carte-traitée : = carte

Action Composition :

Seq1 < - > pers.Action-IN-pointeuse *(carte)* ; Activer *(carte)* ;
Déclencher-traitement-IN (matricule-utilisateur) ; Initialiser

Seq2 < - > pers.Action-IN-miss-pointeuse *(carte)* ; Activer *(carte)* ;
Déclencher-traitement-IN-miss (matricule-utilisateur) ; Initialiser

Seq3 < - > pers.Action-OUT-pointeuse *(carte)* ; Activer *(carte)* ;
Déclencher-traitement-OUT (matricule-utilisateur) ; Initialiser

Seq4 < - > pers.Action-OUT-miss-pointeuse *(carte)* ; Activer *(carte)* ;
Déclencher-traitement-OUT-miss (matricule-utilisateur) ; Initialiser

Activer (carte) <- > Changer-état (carte) ; Remplir-champ-matricule (carte)

Initialiser < - > Initialiser-champs-horaire ;
Initialiser-champ-matricule ;
Initialiser-boutons ; Initialiser-état

Action Perception :

XK (Pers.Action-IN-pointeuse *(carte)* / Etat = Libre)
XK (Pers.Action-IN-miss-pointeuse *(carte)* / Etat = Libre)
XK (Pers.Action-OUT-pointeuse *(carte)* / Etat = Libre)
XK (Pers.Action-OUT-miss-pointeuse *(carte)* / Etat = Libre)
XK (Syst.AfficherTemps (h, m, s, point) / TRUE)

State Information :

XK (Etat.Pers / TRUE)
XK (Seconde.Pers / TRUE)
XK (Minute.Pers / TRUE)
XK (Heure.Pers / TRUE)

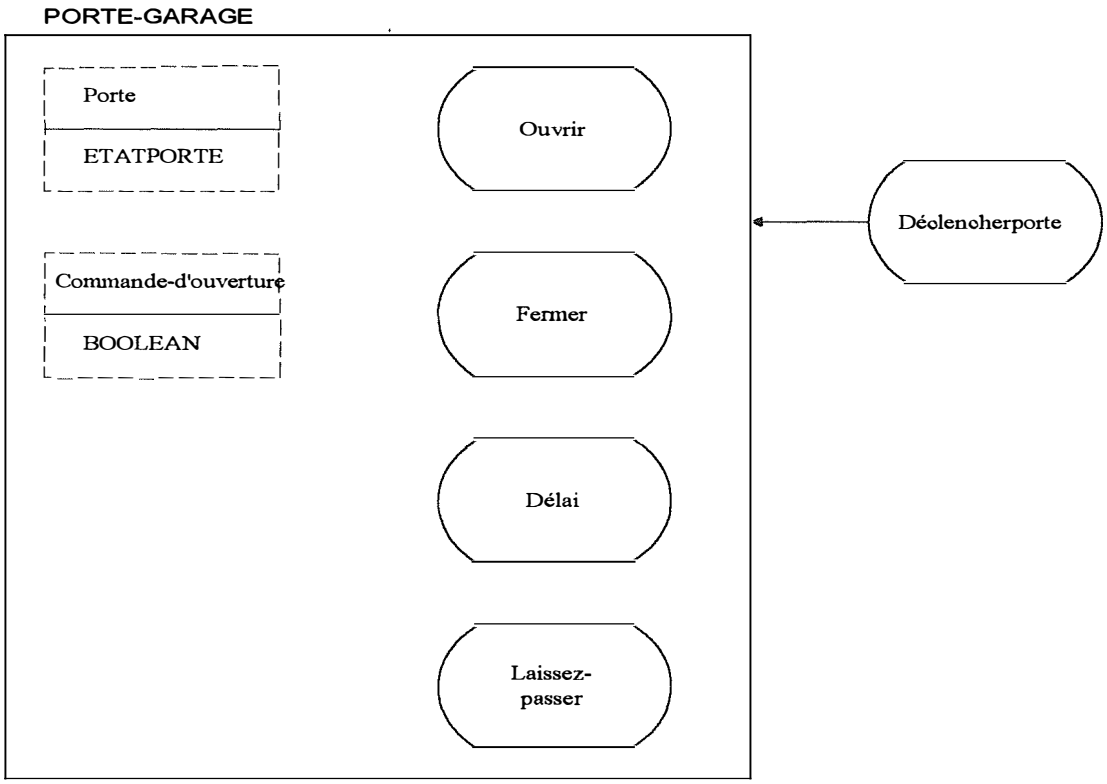
Action Information :

XK (Déclencher-traitement-IN (matricule-utilisateur).Syst / TRUE)
XK (Déclencher-traitement-IN-miss (matricule-utilisateur).Syst / TRUE)
XK (Déclencher-traitement-OUT (matricule-utilisateur).Syst / TRUE)
XK (Déclencher-traitement-OUT-miss (matricule-utilisateur).Syst / TRUE)

State Perception :

XK (carte.Matricule / Carte-traitée = carte)

PORTE-GARAGE



Déclaration de la structure de l'agent

State Components :

Commande-d'ouverture	instance-of	BOOLEAN
Porte	instance-of	ETATPORTE

Actions :

Ouvrir
Fermer
Délai
Laissez-passer

Contraintes sur l'agent

Initial Valuation :

Commande-d'ouverture = FALSE

Porte = Fermée

Effects of Actions :

Syst.Déclencheporte (prt) : Commande-d'ouverture : = TRUE

Pers.Pousse-verrour : Commande-d'ouverture : = TRUE

Ouvrir : Porte : = Ouverte ; Commande-d'ouverture : = FALSE

Fermer: Porte : = Fermée

Capability :

XO (Laissez-passer / Commande-d'ouverture = TRUE)

Action Composition :

Laissez-passer < - > Ouvrir ; Délai ; Fermer

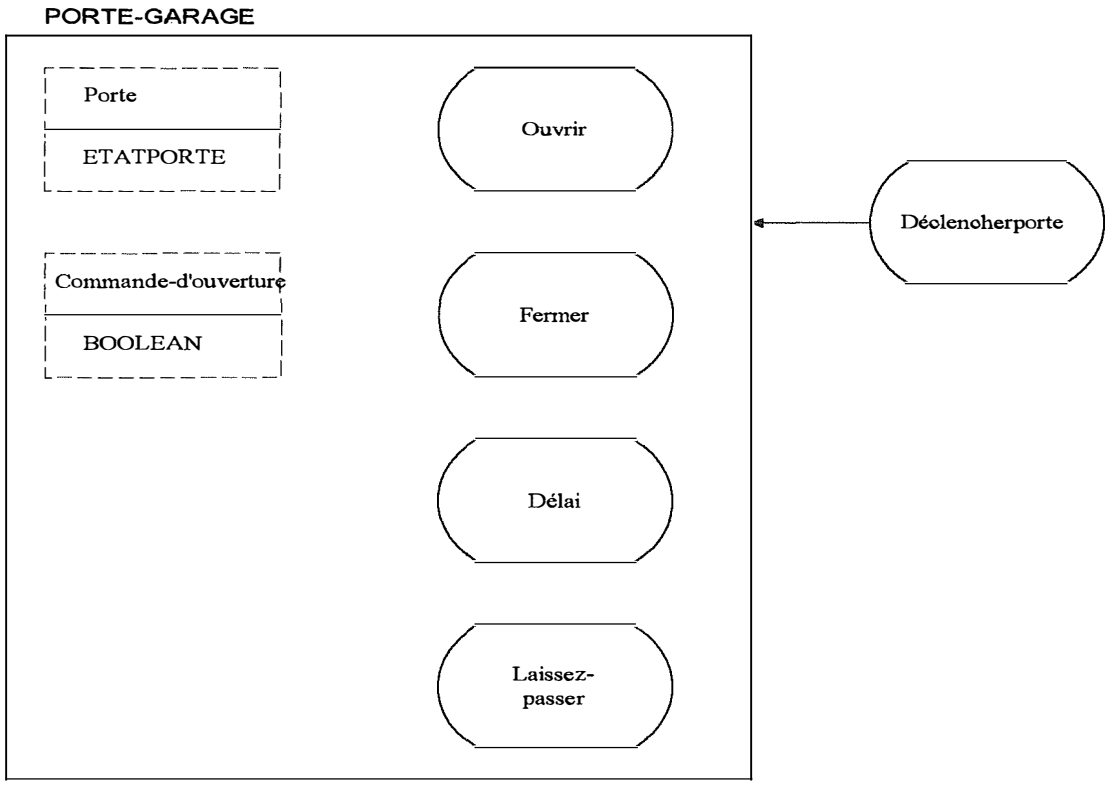
Action Duration :

| Délai | = 30 sec

Action Perception :

XK (Syst.Déclencheporte (prt) / TRUE)
XK (Pers.Pousse-verrou / TRUE)

PORTE-NORMALE



Déclaration de la structure de l'agent

State Components :

Commande-d'ouverture	*instance-of*	*BOOLEAN*
Porte	*instance-of*	*ETATPORTE*

Actions :

Ouvrir
Fermer
Délai
Laissez-passer

Contraintes sur l'agent

Initial Valuation :

Commande-d'ouverture =FALSE

Porte = Fermée

Effects of Actions :

Syst.Déclencheporte (prt) : Commande-d'ouverture : = TRUE

Pers.Pousse-verrour : Commande-d'ouverture : = TRUE

Ouvrir : Porte : = Ouverte ; Commande-d'ouverture : = FALSE

Fermer : Porte : = Fermée

Capability :

XO (Laissez-passer / Commande-d'ouverture = TRUE)

Action Composition :

Laissez-passer <- > Ouvrir ; Délai ; Fermer

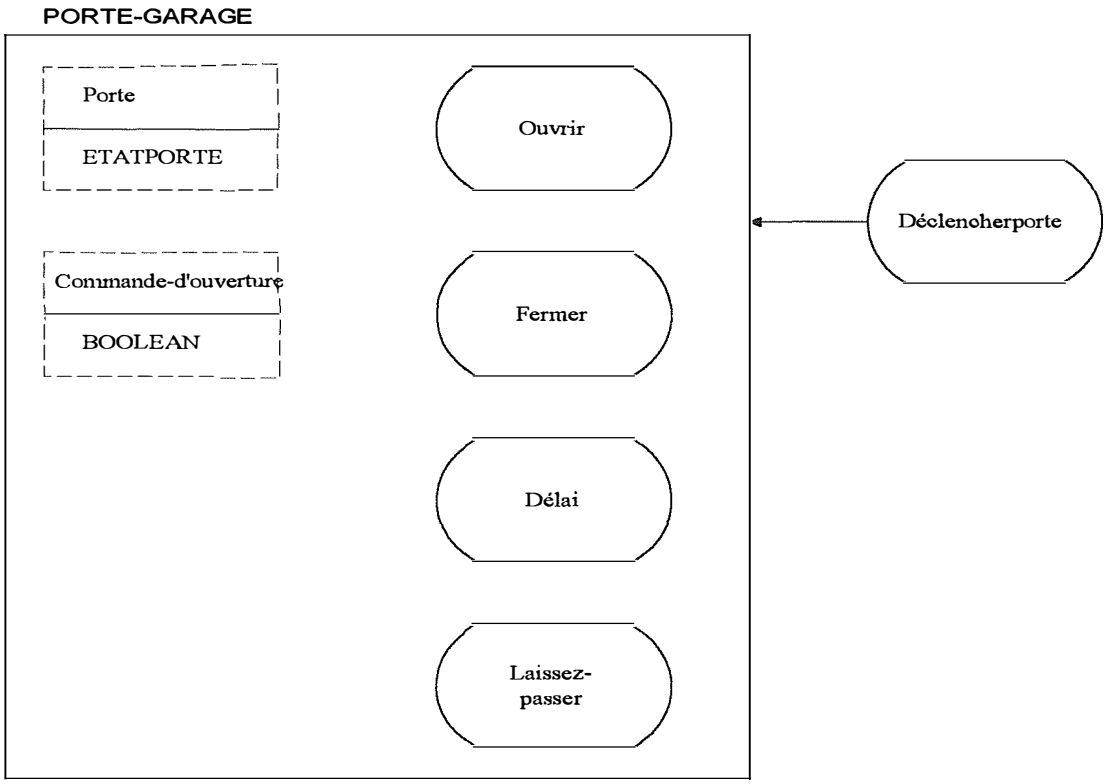
Action Duration :

| Délai | = 10 sec

Action Perception :

XK (Syst.Déclencheporte (prt) / TRUE)
XK (Pers.Pousse-verrou / TRUE)

PORTE-DEPARTEMENT



Déclaration de la structure de l'agent

State Components :

Commande-d'ouverture	*instance-of*	*BOOLEAN*
Porte	*instance-of*	*ETATPORTE*

Actions :

Ouvrir
Fermer
Délai
Laissez-passer

Contraintes sur l'agent

Initial Valuation :

Commande-d'ouverture =FALSE

Porte = Fermée

Effects of Actions :

Syst.Déclencheporte (prt) : Commande-d'ouverture : = TRUE

Pdp.Déclencher-Ouverture (prt) : Commande-d'ouverture : = TRUE

Pers.Pousse-verrour : Commande-d'ouverture : = TRUE

Ouvrir : Porte : = Ouverte ; Commande-d'ouverture : = FALSE

Fermer : Porte : = Fermée

Capability :

XO (Laissez-passer / Commande-d'ouverture = TRUE)

Action Composition :

Laissez-passer <- > Ouvrir ; Délai ; Fermer

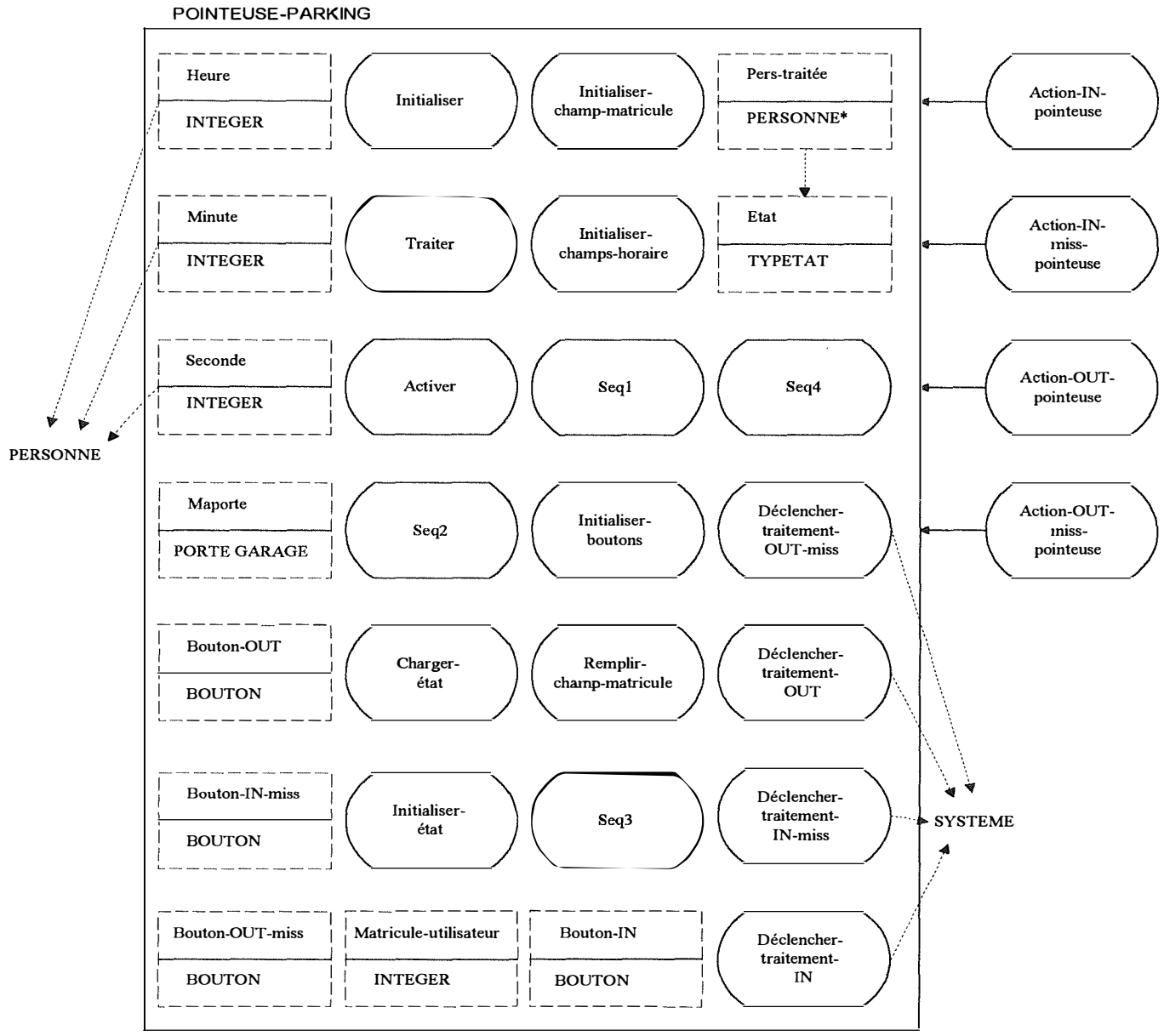
Action Duration :

| Délai | = 10 sec

Action Perception :

XK (Syst.Déclencheporte (prt) / TRUE)
XK (Pdp.Déclencher-Ouverture (prt) / TRUE)
XK (Pers.Pousse-verrou / TRUE)

POINTEUSE-PARKING



Déclaration de la structure de l'agent

State Components :

Heure	instance-of	INTEGER
Minute	instance-of	INTEGER
Seconde	instance-of	INTEGER
Matricule-utilisateur	instance-of	INTEGER
Carte-traitée	*instance-of*	*CARTE-CLE**

Etat	instance-of	TYPETAT
Maporte	instance-of	PORTE GARAGE
Bouton-IN	instance-of	BOUTON
Bouton-IN-miss	instance-of	BOUTON
Bouton-OUT	instance-of	BOUTON
Bouton-OUT-miss	instance-of	BOUTON

Actions :

Traiter
Initialiser-champs-horaire
Activer (CARTE-CLE)
Initialiser-champ-matricule
Initialiser-état
Changer-état (CARTE-CLE)
Initialiser-boutons
Initialiser
Remplir-champ-matricule (CARTE-CLE)
Seq1
Seq2
Seq3
Seq4
Déclencher-traitement-IN (INTEGER) → SYSTEME*
Déclencher-traitement-IN-miss (INTEGER) → SYSTEME*
Déclencher-traitement-OUT (INTEGER) → SYSTEME*
Déclencher-traitement-OUT-miss (INTEGER) → SYSTEME*

Contraintes sur l'agent

L'agent Pointeuse Parking reprend les mêmes actions et composants d'états que l'agent Pointeuse normale qui est muni maintenant aussi d'une porte. La seule différence vient du type de la porte qui est liée à la pointeuse.

Action Composition :

Maporte instance-of PORTE-GARAGE

SYSTEME

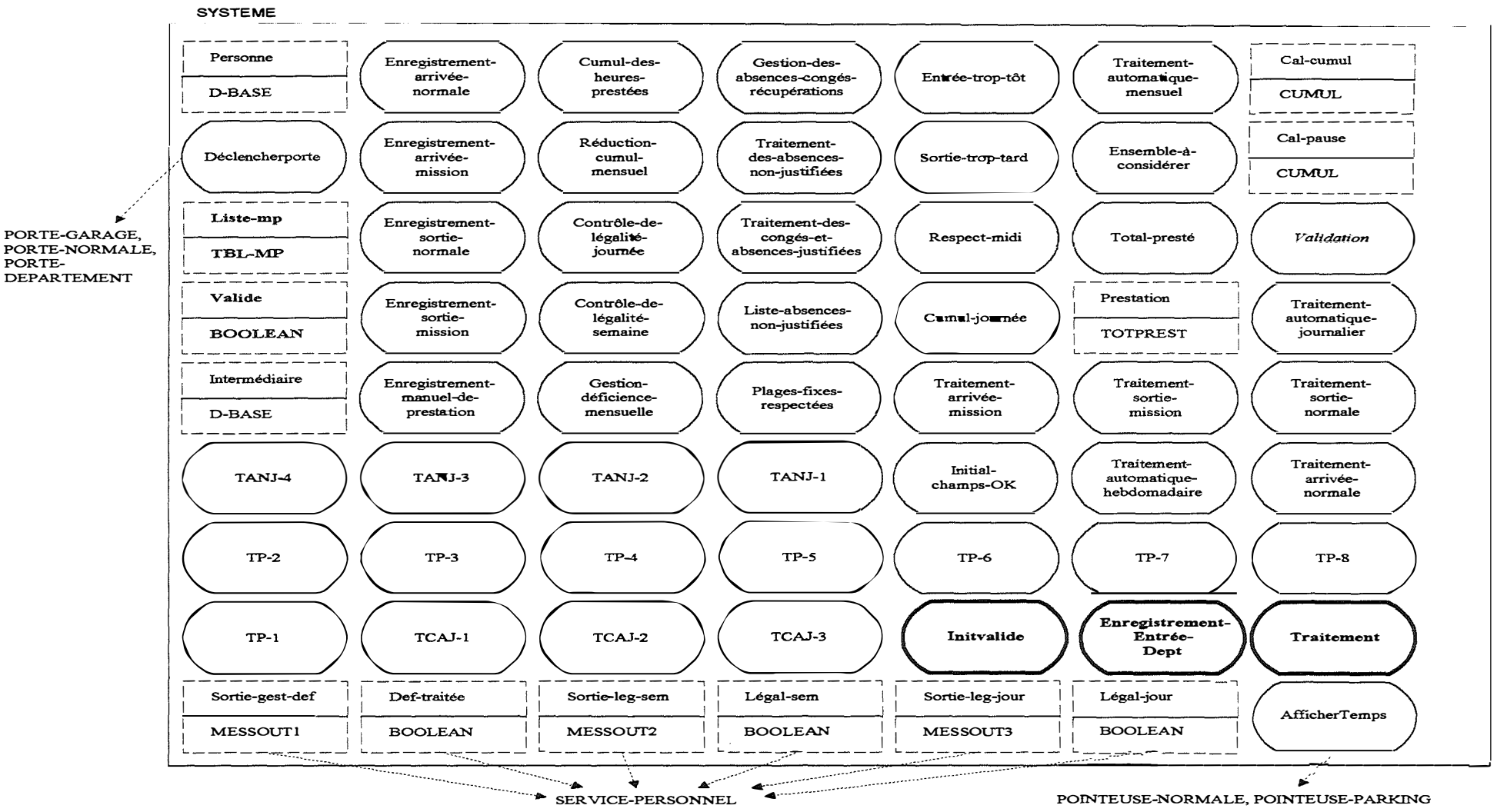
Déclaration de la structure de l'agent

State Components :

Personne	instance-of	D-BASE
Intermédiaire	instance-of	D-BASE
Sortie-gest-def	instance-of	MESSOUT1
Def-traitée	instance-of	BOOLEAN
Valide	*instance-of*	*BOOLEAN*
Liste-mp	*instance-of*	*TBL-MP*
Sortie-leg-sem	instance-of	MESSOUT2
Légal-sem	instance-of	BOOLEAN
Sortie-leg-jour	instance-of	MESSOUT3
Légal-jour	instance-of	BOOLEAN
Prestation	instance-of	TOTPREST
Cal-cumul	instance-of	CUMUL
Cal-pause	instance-of	CUMUL

Actions :

Enregistrement-arrivée-normale
Enregistrement-arrivée-mission
Enregistrement-sortie-normale
Enregistrement-sortie-mission
*Traitement-arrivée-normale (UNION [POINTEUSE-NORMALE, POINTEUSE-PARKING])
*Traitement-arrivée-mission (UNION [POINTEUSE-NORMALE, POINTEUSE-PARKING])
*Traitement-sortie-normale (UNION [POINTEUSE-NORMALE, POINTEUSE-PARKING])
*Traitement-sortie-mission (UNION [POINTEUSE-NORMALE, POINTEUSE-PARKING])
Enregistrement-manuel-de-prestation → SERVICE-PERSONNEL
Cumul-des-heures-prestées
Réduction-cumul-mensuel
Contrôle-de-légalité-journée → SERVICE-PERSONNEL
Contrôle-de-légalité-semaine → SERVICE-PERSONNEL
Gestion-déficience-mensuelle → SERVICE-PERSONNEL
Gestion-absences-congés-récupérations → SERVICE-PERSONNEL
Traitement-des-absences-non-justifiées
Liste-absences-non-justifiées
Plages-fixes-respectées
Entrée-trop-tôt
Sortie-trop-tard


SYSTEME
Personne
D-BASE
Enregistrement-arrivée-normale
Cumul-des-heures-prestées
Gestion-des-absences-congés-récupérations
Entrée-trop-tôt
Traitement-automatique-mensuel
Cal-cumul
CUMUL
Déclencherporte
Enregistrement-arrivée-mission
Réduction-cumul-mensuel
Traitement-des-absences-non-justifiées
Sortie-trop-tard
Ensemble-à-considérer
Cal-pause
CUMUL
PORTE-GARAGE, PORTE-NORMALE, PORTE-DEPARTEMENT
Liste-mp
TBL-MP
Enregistrement-sortie-normale
Contrôle-de-légalité-journée
Traitement-des-congés-et-absences-justifiées
Respect-midi
Total-presté
Validation
Valide
BOOLEAN
Enregistrement-sortie-mission
Contrôle-de-légalité-semaine
Liste-absences-non-justifiées
Cumul-journée
Prestation
TOTPREST
Traitement-automatique-journalier
Intermédiaire
D-BASE
Enregistrement-manuel-de-prestation
Gestion-déficience-mensuelle
Plages-fixes-respectées
Traitement-arrivée-mission
Traitement-sortie-mission
Traitement-sortie-normale
TANJ-4
TANJ-3
TANJ-2
TANJ-1
Initial-champs-OK
Traitement-automatique-hebdomadaire
Traitement-arrivée-normale
TP-2
TP-3
TP-4
TP-5
TP-6
TP-7
TP-8
TP-1
TCAJ-1
TCAJ-2
TCAJ-3
Initvalide
Enregistrement-Entrée-Dept
Traitement
Sortie-gest-def
MESSOUT1
Def-traitée
BOOLEAN
Sortie-leg-sem
MESSOUT2
Légal-sem
BOOLEAN
Sortie-leg-jour
MESSOUT3
Légal-jour
BOOLEAN
AfficherTemps
SERVICE-PERSONNEL
POINTEUSE-NORMALE, POINTEUSE-PARKING

Traitement (INTEGER, INTEGER)
Initvalide
Enregistrement-Entrée-Dept
*Déclencherporte (*UNION [PORTE-GARAGE,*
PORTE-DEPARTEMENT,
PORTE-NORMALE])
→ PORTE-GARAGE, *PORTE-DEPARTEMENT, PORTE-NORMALE*
*AfficherTemps (INTEGER, INTEGER, INTEGER,
UNION [POINTEUSE-NORMALE, POINTEUSE-PARKING])
→ POINTEUSE-NORMALE, POINTEUSE-PARKING
Respect-midi
Cumul-journée
Traitement-automatique-journalier
Traitement-automatique-mensuel
Ensemble-à-considérer
Total-presté
Traitement-des-congés-et-absences-justifiées
TP-1
TP-2
TP-3
TP-4
TP-5
TP-6
TP-7
TP-8
TANJ-1
TANJ-2
TANJ-3
TANJ-4
TCAJ-1
TCAJ-2
TCAJ-3
Validation (INTEGER)
Initial-champs-OK
Traitement-automatique-hebdomadaire

Contraintes sur l'agent

Initial Valuation :

Def-traitée = FALSE
Valide = FALSE
Légal-sem = FALSE
Légal-jour = FALSE

Effects of Actions :

Traitement-arrivée-normale (mat) :

Personnes : = InNormal (Personnes, mat,
Horloge.Heure, Horloge.Minute,
Horloge.Seconde)

Traitement-arrivée-mission (mat) :

Personnes : = InMiss (Personnes, mat,
Horloge.Heure, Horloge.Minute,
Horloge.Seconde)

Traitement-sortie-normale (mat) :

Personnes : = OutNormal (Personnes, mat,
Horloge.Heure, Horloge.Minute,
Horloge.Seconde)

Traitement-sortie-mission (mat) :

Personnes : = OutMiss (Personnes, mat,
Horloge.Heure, Horloge.Minute,
Horloge.Seconde)

Enregistrement-manuel-de-prestation :

Personnes : = EnregistreManuel(Personne, Service-personnel.Matricule,
Service-personnel.Heure,
Service-personnel.Minute,
Service-personnel.Seconde)

Traitement (mat, mp, porte) :

Valide : = Est-valide (mat, mp, Liste-mp, porte)

Initvalide :

Valide : =FALSE

Validation (mat) :

Valide : = Est-val (mat, Liste-mp)

Réduction-cumul-mensuel :

Personnes : = RédCumul (Personnes)

Gestion-déficience-mensuelle :

Sortie-gest-def : = GestDef (Personnes) ;
Def-traitée : = TRUE

TANJ-1 :

| On met dans Intermédiaire tous les membres du personnel qui n'ont pas fait de mouvement en
| ce jour et qui ne sont ni en mission ni en congé ni en absence justifiée.

Intermédiaire : = CréeInter1 (Personnes,
Calendrier.Jour,
Calendrier.Mois,
Calendrier.Année)

TANJ-2 :

| On traite le "Temps total hebdomadaire" et le "Temps cumulé mois antérieur" pour les
| membres du personnel qui n'ont pas fait de mouvement en ce jour et qui ne sont ni en mission
| ni en congé ni en absence justifiée. Ces personnes sont enregistrées dans Intermédiaire.

Personnes : = TraiteTTH1 (Personnes,
Intermédiaire ,
Calendrier.Jour-sem) ;
Personnes : = TraiteTCMA (Personnes,
Intermédiaire,
Calendrier.Jour,
Calendrier.Jour-sem)

TANJ-3 :

| On met à jour le "Temps cumulé global" pour les membres du personnel qui n'ont pas fait de
| mouvement en ce jour et qui ne sont ni en mission ni en congé ni en absence justifiée. Ces
| personnes sont enregistrées dans Intermédiaire.

Personnes : = TraiteTCG1 (Personnes,
Intermédiaire ,
Calendrier.Jour-sem)

TANJ-4 :

| On indique dans la base de données l'absence injustifiée pour les membres du personnel qui

| n'ont pas fait de mouvement en ce jour et qui ne sont ni en mission ni en congé ni en absence
| justifiée. Ces personnes sont enregistrées dans Intermédiaire.

.

Personnes : = InscrireAInJ (Personnes,
Intermédiaire,
Calendrier.Jour,
Calendrier.Mois
Calendrier.Année)

Gestion-des-absences-congés-récupérations :

Personnes : = GestACR (Personnes,
Service-personnel.Matricule,
Service-presonnel.Motif,
Service-personnel.Dates)

Contrôle-de-légalité-semaine :

Sortie-leg-sem : = GestCLS (Personnes) ;
Légal-sem : = TRUE

Liste-absences-non-justifiées :

Sortie-leg-jour : = TraiteLANJ (Personnes,
Calendrier.Jour,
Calendrier.Mois
Calendrier.Année)

Plages-fixes-respectées :

Sortie-leg-jour : = FaireUnion (Sortie-leg-jour, TraitePFR (Personnes,
Calendrier.Jour-sem))

Entrée-trop-tôt :

Sortie-leg-jour : = FaireUnion (Sortie-leg-jour, TraiteETT (Personnes))

Sortie-trop-tard :

Sortie-leg-jour : = FaireUnion (Sortie-leg-jour, TraiteSTT (Personnes,
Calendrier.Jour-sem))

Respect-midi :

Sortie-leg-jour : = FaireUnion(Sortie-leg-jour, TraiteRM (Cal-pause))

Cumul-journée :

Sortie-leg-jour : = FaireUnion (Sortie-leg-jour, TraiteCJ (Cal-cumul) ;
Légal-jour : = TRUE

Ensemble-à-considérer :

| On met dans intermédiaire les membres du personnel qui on soit fait un mouvement, soit sont
| en mission.

Intermédiaire : = CréeInter2 (Personnes)

TP-1 :

| On traite le champ Temps-cumulé-mois-antérieur pour les membres du personnel qui on soit
| fait un mouvement, soit sont en mission.

Personnes : = TraiteTCMA (Personnes,
Intermédiaire,
Calendrier.Jour,
Calendrier.Jour-sem)

TP-2 :

| On met à jour le temps-total-hebdomadaire pour les membres du personnel partis en mission
| toute la journée.

Personnes : = TraiteTTH2 (Personnes,
Intermédiaire,
Calendrier.Jour-sem)

TP-3 :

| Pour les personnes qui ont soit fait un mouvement et sont inscris dans intermédiaire, on retire
| les entrées sorties pointées mission.

Intermédiaire : = RetraitMiss (Intermédiaire, Personnes)

TP-4 :

| On met à jour la variable Cal-pause. C'est-à-dire que l'on enregistre dans Cal-pause toutes les
| personnes qui ont faits un mouvement pendant la journée et dont le temps de pause de midi
| minimum (45 minutes) n'a pas été respecté.

Cal-pause : = TraiteCP (Intermédiaire, Personnes)

TP-5 :

| On calcule le temps presté par les membres du personnel ayant pointés.

Prestation : = TraiteTempsPresté (Personnes, Intermédiaire)

TP-6 :

| On traite le Cal-cumul. C'est-à-dire que l'on enregistre dans Cal-cumul toutes les
| personnes qui ont faits un mouvement pendant la journée et dont le temps de travail
| dépasse le maximum autorisé (9 heures).

Cal-cumul : = TraiteCC (Prestation, Calendrier.Jour-sem)

TP-7 :

| On calcule le Temps-total-hebdomadaire pour les personnes ayant pointés.

Personnes : = TraiteTTH3 (Personnes,
Prestation,
Calendrier.Jour-sem)

TP-8 :

| On met à jour le Temps-cumulé-global pour les personnes qui étaient présentes dans la
| société.

Personnes : = TraiteTCG2 (Personnes,
Prestation,
Calendrier.Jour-sem)

TCAJ-1 :

| On met dans Intermédiaire toutes les personnes en congé ou en absence justifiée.

Intermédiaire : = CréeInter3 (Personnes,
Calendrier.Jour,
Calendrier.Mois,
Calendrier.Année)

TCAJ-2 :

| On met à jour le Temps-cumulé-mois-antérieur pour toutes les personnes en congé ou en
| absence justifiée.

Personnes : = TraiteTCMA (Personnes,
Intermédiaire,
Calendrier.Jour,
Calendrier.Jour-sem)

TCAJ-3 :

| On met à jour le Temps-total-hebdomadaire pour toutes les personnes en congé ou en
| absence justifiée.

Personnes : = TraiteTTH4 (Personnes,
Intermédiaire,
Calendrier.Jour-sem)

SERVICE PERSONNEL.Lecture-de-traitement-journalier :

Légal-jour : = FALSE

SERVICE PERSONNEL.Lecture-de-traitement-hebdomadaire :

Légal-sem : = FALSE

SERVICE PERSONNEL.Lecture-de-traitement-mensuel :

Def-traitée : = FALSE

Capability :

XO (Traitement-automatique-journalier / Service-personnel.OK-jour)
XO (Traitement-automatique-hebdomadaire / Service-personnel.OK-hebd)
XO (Traitement-automatique-mensuel / Service-personnel.OK-mens)

Action Composition :

Cumul-des-heures-prestées	< - >	Ensemble-à-considérer ; Total-presté
Contrôle-de-légalité-journée	< - >	Liste-absences-non-justifiées ; Plages-fixes-respectées ; Entrée-trop-tôt ; Sortie-trop-tard ; Respect-midi ; Cumul-journée

Traitement-automatique-journalier
< - > Cumul-des-heures-prestées ;
Traitement-des-congés-et-absences-justifiées ;
Traitement-des-absences-non-justifiées ;
Contrôle-de-légalité-journée

Traitement-automatique-mensuel
< - > Réduction-cumul-mensuel ;
Gestion-déficience-mensuelle

Traitement-automatique-hebdomadaire
< - > Contrôle-de-légalité-semaine

| *D'une manière générale, ce qui va changer pour le pointage d'arrivée, de sortie, de sortie-*
| *mission et d'arrivée-mission consiste à vérifier si la carte clé contient bien une information*
| *de matricule. Dans le cas contraire, il n'y a pas de mise à jour de la base de données, pas*
| *d'affichage de temps presté et pas d'ouverture de porte. Nous introduisons par conséquent*
| *une action Validation qui permet de regarder si le matricule existe et permettra par la mise*
| *à jour d'un champ de validation d'autoriser l'information des actions AfficherTemps et*
| *Déclencherporte.*

Enregistrement-arrivée-normale < - > point.Déclencher-traitement-IN (mat) ;
Validation (mat) ;
Traitement-arrivée-normale (mat) ;
AfficherTemps (h, m, s, point) ;
Déclencherporte (porte)
Initvalide
{ point : POINTEUSE-PARKING }
with porte = AssignerPorteParking (point)
∧ h = GénèreHeure (Personnes, mat)
∧ m = GénèreMinute (Personnes, mat)
∧ s = GénèreSeconde (Personnes, mat)

Enregistrement-arrivée-normale < - > point.Déclencher-traitement-IN (mat) ;
Validation (mat) ;
Traitement-arrivée-normale (mat) ;
AfficherTemps (h, m, s, point)
Initvalide
{ point : POINTEUSE-NORMALE }
with h = GénèreHeure (Personnes, mat)
∧ m = GénèreMinute (Personnes, mat)
∧ s = GénèreSeconde (Personnes, mat)

Enregistrement-arrivée-mission < - > point.Déclencher-traitement-IN-miss (mat) ;
Validation (mat) ;
Traitement-arrivée-mission (mat) ;
AfficherTemps (h, m, s, point) ;

Déclencherporte (porte)
Initvalide
{ point : POINTEUSE-PARKING }
with porte = AssignerPorteParking (point)
∧ h = GénèreHeure (Personnes, mat)
∧ m = GénèreMinute (Personnes, mat)
∧ s = GénèreSeconde (Personnes, mat)

Enregistrement-arrivée-mission < - > point.Déclencher-traitement-IN-miss (mat) ;
Validation (mat) ;
Traitement-arrivée-mission (mat) ;
AfficherTemps (h, m, s, point)
Initvalide
{ point : POINTEUSE-NORMALE }
with h = GénèreHeure (Personnes, mat)
∧ m = GénèreMinute (Personnes, mat)
∧ s = GénèreSeconde (Personnes, mat)

Enregistrement-sortie-normale < - > point.Déclencher-traitement-OUT (mat) ;
Validation (mat) ;
Traitement-sortie-normale (mat) ;
AfficherTemps (h, m, s, point) ;
Déclencherporte (porte)
Initvalide
{ point : POINTEUSE-PARKING }
with porte = AssignerPorteParking (point)
∧ h = GénèreHeure (Personnes, mat)
∧ m = GénèreMinute (Personnes, mat)
∧ s = GénèreSeconde (Personnes, mat)

Enregistrement-sortie-normale < - > point.Déclencher-traitement-OUT (mat) ;
Validation (mat) ;
Traitement-sortie-normale (mat) ;
AfficherTemps (h, m, s, point)
Initvalide
{ point : POINTEUSE-NORMALE }
with h = GénèreHeure (Personnes, mat)
∧ m = GénèreMinute (Personnes, mat)
∧ s = GénèreSeconde (Personnes, mat)

Enregistrement-sortie-mission < - > point.Déclencher-traitement-OUT-miss (mat) ;
Validation (mat) ;
Traitement-sortie-mission (mat) ;
AfficherTemps (h, m, s, point) ;
Déclencherporte (porte)
Initvalide
{ point : POINTEUSE-PARKING }
with porte = AssignerPorteParking (point)

∧ h = GénèreHeure (Personnes, mat)
∧ m = GénèreMinute (Personnes, mat)
∧ s = GénèreSeconde (Personnes, mat)

Enregistrement-sortie-mission < - > point.Déclencher-traitement-OUT-miss (mat) ;
Validation (mat) ;
Traitement-sortie-mission (mat) ;
AfficherTemps (h, m, s, point)
Initvalide
{ point : POINTEUSE-NORMALE }
with h = GénèreHeure (Personnes, mat)
∧ m = GénèreMinute (Personnes, mat)
∧ s = GénèreSeconde (Personnes, mat)

| *On ajoute une séquence destinée à gérer le traitement de la volonté affichée par une personne*
| *de pénétrer dans un département protégé d'accès. Il s'agit avant tout de vérifier que le mot de*
| *passe introduit correspond bien au matricule par l'action Traitement. Un composant valide*
| *est mis à jour et permet ou non selon la validité et autorise ou pas l'information de l'action*
| *déclencherporte.*

Enregistrement-Entrée-Dep <- > Point.Déclencher-Validation (mat, mp) ;
Traitement (mat, mp, porte) ;
Déclencherporte (porte) ;
Initvalide
with porte = AssignerPorteDep (P)

Traitement-des-congés-et-absences-justifiées < - > TCAJ-1 ;
TCAJ-2 ;
TCAJ-3

Total-presté < - > TP-1 ; TP-2 ; TP-3 ; TP-4 ;
TP-5 ; TP-6 ; TP-7 ; TP-8

Traitement-des-absences-non-justifiées < - > TANJ-1 ; TANJ-2 ;
TANJ-3 ; TANJ-4

State Perception :

XK (Calendrier.Jour / TRUE)
XK (Calendrier.Mois / TRUE)
XK (Calendrier.Année / TRUE)
XK (Horloge.Heure / TRUE)
XK (Horloge.Minute / TRUE)
XK (Horloge.Seconde / TRUE)
XK (Calendrier.Jour-sem / TRUE)

XK (Service-personnel.OK-mens / TRUE)
XK (Service-personnel.OK-jour / TRUE)
XK (Service-personnel.OK-hebd / TRUE)
XK (Service-personnel.OK-acr / TRUE)
XK (Service-personnel.OK-manuel / TRUE)

State Information :

XK (Sortie-gest-def.Service-personnel / TRUE)
XK (Def-traitée.Service-personnel / TRUE)
XK (Sortie-leg-jour.Service-personnel / TRUE)
XK (Légal-sem.Service-personnel / TRUE)
XK (Sortie-leg-sem.Service-personnel / TRUE)
XK (Légal-jour.Service-personnel / TRUE)

Action perception :

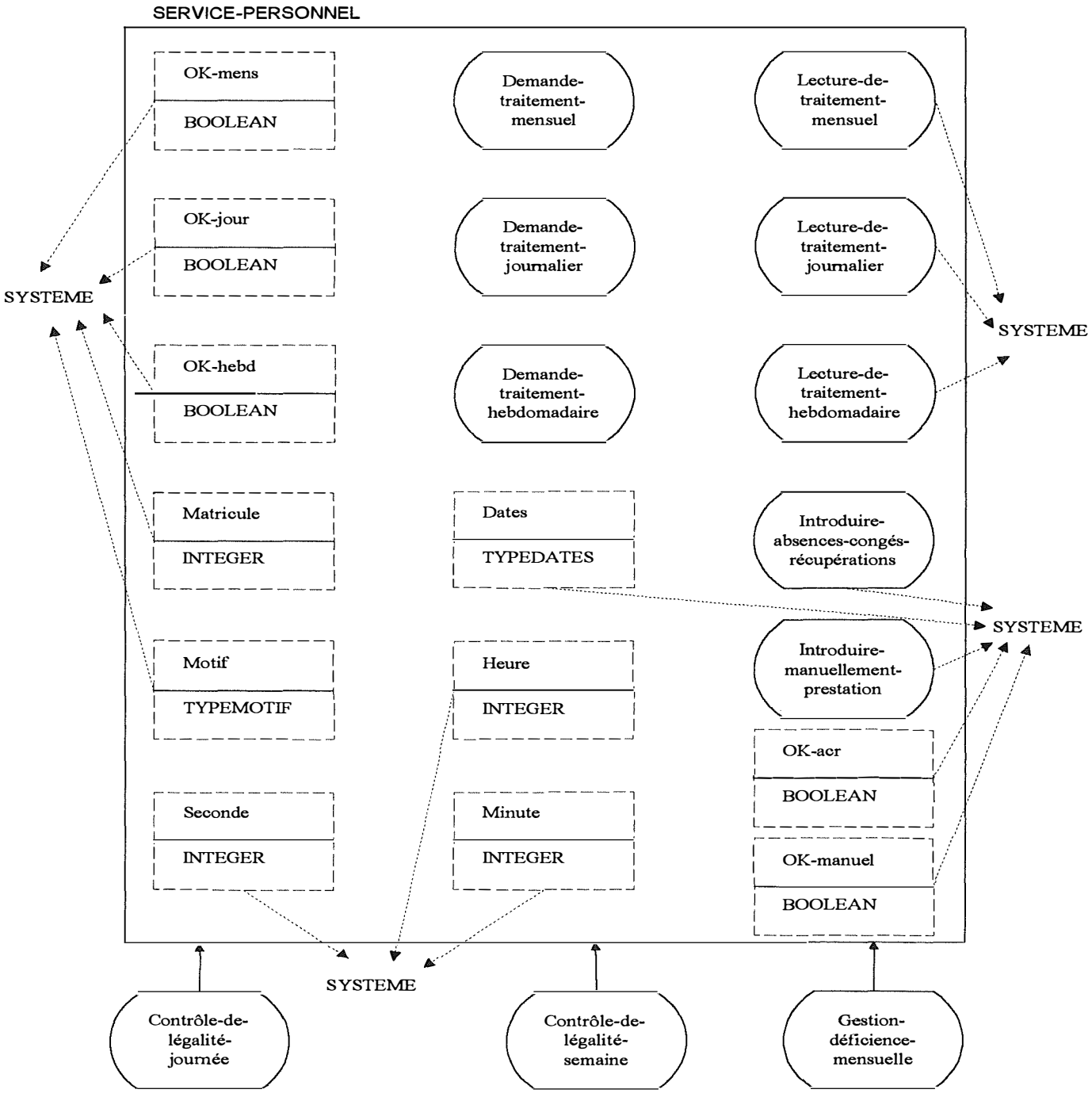
XK (Service-personnel.Lecture-de-traitement-journalier / TRUE)
XK (Service-personnel.Lecture-de-traitement-hebdomadaire / TRUE)
XK (Service-personnel.Lecture-de-traitement-mensuel / TRUE)
XK (Pdp.Déclencher-Validation (mat, mp) / TRUE)
XK (Pn.Déclencher-traitement-IN (mat) / TRUE) { Pn : POINTEUSE-NORMALE }
XK (Pn.Déclencher-traitement-IN-miss (mat) / TRUE) { Pn : POINTEUSE-NORMALE }
XK (Pn.Déclencher-traitement-OUT (mat) / TRUE) { Pn : POINTEUSE-NORMALE }
XK (Pn.Déclencher-traitement-OUT-miss (mat) / TRUE) { Pn : POINTEUSE-NORMALE }
XK (Pp.Déclencher-traitement-IN (mat) / TRUE) { Pp : POINTEUSE-PARKING }
XK (Pp.Déclencher-traitement-IN-miss (mat) / TRUE) { Pp : POINTEUSE-PARKING }
XK (Pp.Déclencher-traitement-OUT (mat) / TRUE) { Pp : POINTEUSE-PARKING }
XK (Pp.Déclencher-traitement-OUT-miss (mat) / TRUE) { Pp : POINTEUSE-PARKING }

Action information :

XK (Gestion-déficience-mensuelle.Service-personnel / TRUE)
XK (Contrôle-de-légalité-journée. Service-personnel / TRUE)
XK (Contrôle-de-légalité-semaine.Service-personnel / TRUE)
XK (Gestion-des-absences-congés-récupérations.Service-personnel / TRUE)
XK (AfficherTemps (h, m, s, point).point' / point = point')
∧ Valide = TRUE)
{ point : POINTEUSE-NORMALE, point' : POINTEUSE-NORMALE }
XK (AfficherTemps (h, m, s, point).point' / point = point')
∧ Valide = TRUE)
{ point : POINTEUSE-PARKING, point' : POINTEUSE-PARKING }
XK (Déclencherporte (porte).p / porte = p)
∧ Valide = TRUE)
{ P : PORTE GARAGE }

XK (Déclencherporte (porte).p / porte = p)
∧ Valide = TRUE)
{ P : PORTE NORMALE }
XK (Déclencherporte (porte).p / porte = p
∧ Valide = TRUE)
{ P : PORTE DEPARTEMENT }
XK (Enregistrement-manuel-de-prestation.Service-personnel / TRUE)

SERVICE-PERSONNEL



Déclaration de la structure de l'agent

State Components :

OK-mens instance-of BOOLEAN

OK-jour	instance-of	BOOLEAN
OK-hebd	instance-of	BOOLEAN
OK-acr	instance-of	BOOLEAN
OK-manuel	instance-of	BOOLEAN
Heure	instance-of	INTEGER
Minute	instance-of	INTEGER
Seconde	instance-of	INTEGER
Dates	instance-of	TYPEDATES
Motif	instance-of	TYPEMOTIF
Matricule	intance-of	INTEGER

Actions :

Demande-traitement-mensuel	
Introduire-manuellement-prestation	→ SYSTEME
Demande-traitement-journalier	
Introduire-absences-congés-récupérations	→ SYSTEME
Demande-traitement-hebdomadaire	
Lecture-de-traitement-journalier	→ SYSTEME
Lecture-de-traitement-mensuel	→ SYSTEME
Lecture-de-traitement-hebdomadaire	→ SYSTEME

Contraintes sur l'agent

Initial Valuation :

OK-mens = FALSE
OK-jour = FALSE
OK-manuel = FALSE
OK-acr = FALSE
OK-hebd = FALSE

Effects of Actions :

Syst.Gestion-déficience-mensuelle	:	OK-mens : = FALSE
Syst.Contrôle-de-légalité-journée	:	OK-jour : = FALSE
Syst.Contrôle-de-légalité-semaine	:	OK-hebd : = FALSE
Demande-traitement-mensuel	:	OK-mens : = TRUE
Demande-traitement-journalier	:	OK-jour : = TRUE
Demande-traitement-hebdomadaire	:	OK-hebd : = TRUE

Introduire-absences-congés-récupérations : Motif : = X ; Matricule : = Y ;
Dates : = Z ; OK-acr : = TRUE

Introduire-manuellement-prestation : Matricule : = Y ; Heure : = X ;
Minute : = Z ; Seconde : = W :
OK-manuel : = TRUE

Syst.Gestion-des-absences-congés-récupérations : OK-acr : = FALSE

Syst.Enregistrement-manuel-de-prestation : OK-manuel : = FALSE

Capability :

XO (Lecture-de-traitement-journalier / Syst.Légal-jour)
XO (Lecture-de-traitement-hebdomadaire / Syst.Légal-sem)
XO (Lecture-de-traitement-mensuel / Syst.Def-traitée)

Action perception :

XK (Syst.Gestion-déficience-mensuelle / TRUE)
XK (Syst.Contrôle-de-légalité-journée / TRUE)
XK (Syst.Contrôle-de-légalité-semaine / TRUE)
XK (Syst.Gestion-des-absences-congés-récupérations / TRUE)
XK (Syst.Enregistrement-manuel-de-prestation / TRUE)

State perception :

XK (Syst.Légal-jour / TRUE)
XK (Syst.Légal-sem / TRUE)
XK (Syst.Def-traitée / TRUE)
XK (Syst.Sortie-gest-def / TRUE)
XK (Syst.Sortie-leg-sem / TRUE)
XK (Syst.Sortie-leg-jour / TRUE)

Action Information :

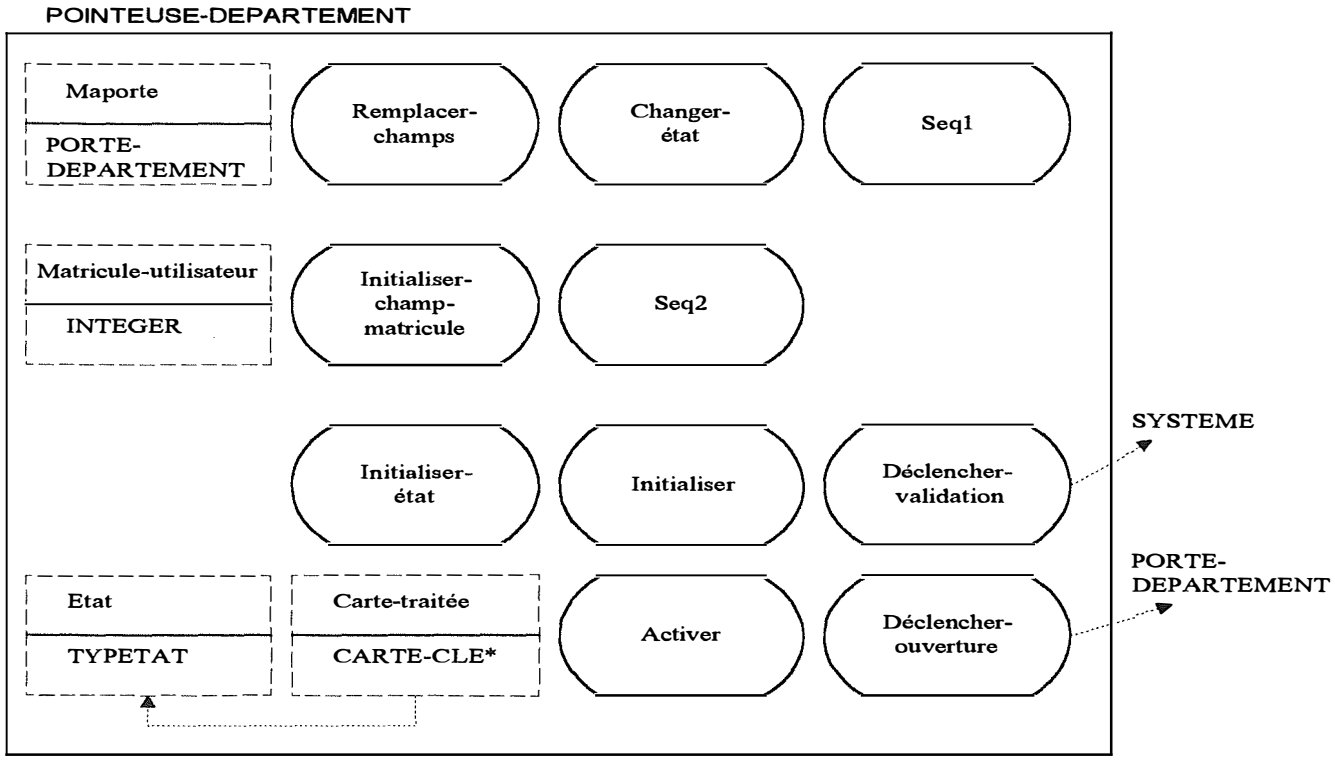
XK (Lecture-de-traitement-journalier.Syst / TRUE)
XK (Lecture-de-traitement-hebdomadaire.Syst / TRUE)
XK (Lecture-de-traitement-mensuel.Syst / TRUE)

State Information :

XK (OK-mens.Syst / TRUE)
XK (OK-jour.Syst / TRUE)
XK (OK-hebd.Syst / TRUE)
XK (Matricule.Syst / TRUE)
XK (Heure.Syst / TRUE)
XK (Minute.Syst / TRUE)

XK (Seconde.Syst / TRUE)
XK (Motif.Syst / TRUE)
XK (Dates.Syst / TRUE)
XK (OK-acr.Syst / TRUE)
XK (OK-manuel.Syst / TRUE)

POINTEUSE-DEPARTEMENT



Déclaration de la structure de l'agent

State Components :

Etat	*instance-of*	*TYPETAT*
Carte-traitée	*instance-of*	*CARTE-CLE**
Matricule-utilisateur	*instance-of*	*INTEGER*
Maporte	*instance-of*	*PORTE-DEPARTEMENT*

Actions :

Activer (CARTE CLE)
Initialiser-champ-matricule
Initialiser-état
Changer-état
Initialiser
Remplir-champs (CARTE CLE)
Seq1
Seq2
**Déclencher-ouverture (PORTE-DEPARTEMENT)* → *PORTE-DEPARTEMENT*
Déclencher-validation (INTEGER, INTEGER) → *SYSTEME**

Contraintes sur l'agent

Derived Components :

Etat ≙ *Carte-traitée = UNDEF*

Initial Valuation :

Matricule-utilisateur = 0
Carte-traitée = UNDEF

Effects of Actions :

Initialiser-champ-matricule : Matricule-utilisateur : = 0

Initialiser-état : Carte-traitée : = UNDEF

Remplir-champs (carte) : Matricule-utilisateur : = carte.Matricule ;

Changer-état (carte) : Carte-traitée : = carte

Action composition :

Seq1 <- > Pers.Action-Entrer-Dep (carte, motp); Activer (carte) ;
Déclencher-Validation (matricule-utilisateur, motp) ; Initialiser

Seq2 <- > Pers.Action-Sortir-Dep (carte) ; Activer (carte) ;
Déclencher-Ouverture (Maporte) ; Initialiser

Activer (carte) <- > Changer-état (carte) ; Remplir-champs (carte)

Initialiser <- > Initialiser-champ-matricule ;
Initialiser-état

Action perception :

XK (Pers.Action-Entrer-Dep (carte, motp) / Etat = Libre)
XK (Pers.Action-Sortie-Dep (carte) / Etat = Libre)

State perception :

XK (carte.Matricule / Carte-traitée = carte)

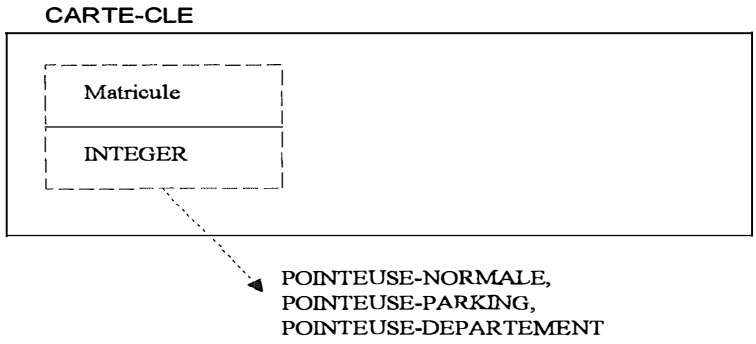
Action Information :

XK (Déclencher-Validation (matricule-utilisateur, mot-de-passe).Syst / TRUE)
XK (Déclencher-Ouverture (Maporte).P / Maporte = P)

State Information :

XK (Etat.Pers / TRUE)

CARTE-CLE



Déclaration de la structure de l'agent

State Components :

Matricule instance-of INTEGER

Contraintes sur l'agent

State Information :

XK (Matricule.Pdp / TRUE) { Pdp : POINTEUSE-DEPARTEMENT }
XK (Matricule.Pn / TRUE) { Pn : POINTEUSE-NORMALE }
XK (Matricule.Pp / TRUE) { Pp : POINTEUSE-PARKING }