FACULTÉ DES SCIENCES ET DE GÉNIE

Accès logique et compression des données de petite taille dans un dossier médical géré par carte à microprocesseur

Rachid Osmar ZAÏANE

Mémoire présenté pour l'obtention
du grade de maître ès science (M.Sc.)

ÉCOLE DES GRADUÉS
UNIVERSITÉ LAVAL

Avril 1992

RÉSUMÉ

La carte à microprocesseur est un support informatique innovateur qui marque un nouveau stade dans l'informatisation et la décentralisation des données sensibles. En Europe et au Japon, plusieurs applications utilisent déjà cette nouvelle technologie prometteuse et le succès de la carte à mémoire devient de plus en plus apparent.

L'intégration de la carte intelligente dans un système de gestion des données de santé pose plusieurs problèmes tant sur le plan de la structuration des données que sur celui de l'espace physique requis pour stocker un dossier médical.

Des expériences utilisant des cartes à mémoire conduites en Europe soulignent l'importance des problèmes de saturation des cartes et de l'accès logique sécurisé aux données via une interface de qualité. Nous proposons des scénarios de gestion des cartes de bénéficiaires et présentons des solutions à la saturation de celles-ci à travers la conception et la réalistation d'un dossier médical portable.

Mots-clé: carte à microprocesseur, carte à mémoire, carte intelligente, dossier médical portable, compression des données, simulation, système médical informatisé, informatique médicale.

Étudiant : Rachid Osmar ZAÏANE

Directeur: André GAMACHE

AVANT- PROPOS

À ceux qui collaborent avec amour.
À ceux qui oeuvrent pour améliorer la qualité de notre vie.
À mes parents, mes frères et ma soeur.

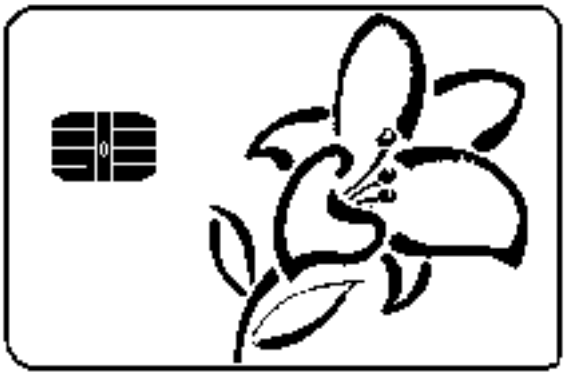
Que tous ceux qui ont contribué d'une manière ou d'une autre et qui m'ont encouragé durant la réalisation de ce mémoire trouvent ici l'expression de ma vive reconnaissance.

Je tiens à exprimer ma sincère gratitude envers mon directeur, le professeur André Gamache, pour son encadrement et sa permanente disponibilité. Je lui suis reconnaissant de m'avoir invité au Québec et de m'avoir permis de participer à des recherches dans des domaines de pointe. Je le remercie particulièrement pour sa sympathie et pour son amitié qui m'ont été d'un grand support pour mener à terme cette étude.

Je remercie très vivement la mission universitaire à Washington pour m'avoir permis de venir au Canada et pour m'avoir supporter financièrement. Que ce travail soit l'expression de toute ma profonde gratitude.

J'aimerais également exprimer ma reconnaissance à mes parents et à mes amis pour les sacrifices qu'ils n'ont cessé de faire pour moi et pour leur support moral.

Enfin, je remercie mes amis du laboratoire LIGE pour leurs encouragements au cours de moments difficiles ainsi que mes collègues de l'équipe Carte Santé du Département de médecine sociale et préventive pour leur aide précieuse dans ce travail.



Bien qu'il puisse être jugé discriminatoire, dans le présent mémoire, l'usage du genre masculin et féminin se conforme aux appellations qui ont été élaborées par les différents intervenants du projet. Le générique masculin utilisé dans le texte vaut également pour le féminin et vice-versa.

TABLE DES MATIÈRES

RÉSUMÉ i
AVANT-PROPOS ii
TABLE DES MATIÈRES iii

INTRODUCTION 1

CHAPITRE I: LES CARTES INTELLIGENTES 4

1.1 Les premières cartes à mémoire 4
1.1.1 Cartes estampées 4
1.1.2 Cartes optiques perforées 5
1.1.3 Cartes magnétiques 5
1.2 Les cartes à puce 5
1.2.1 Les cartes non intelligentes et les cartes intelligentes 6
1.2.2 L'architecture des cartes à puce et les standards ISO 6
1.2.3 Les types de mémoire 7
1.2.4 Encartage 8
1.2.5 Cycle de vie d'une carte à puce 9
1.3 Les applications de la carte à microprocesseur 10
1.3.1 Les cartes à données déductives 10
1.3.2 Les cartes à données cumulatives 11
1.3.3 Les cartes-clés électroniques 12
1.4 Les générations de cartes intelligentes 12
1.4.1 Première génération (mono-application, monoprestataire) 13
1.4.2 Deuxième génération (multi-application, monoprestataire) 14
1.4.3 Troisième génération (multi-application, multiprestataire) 15
1.5 Le choix d'une carte 17
1.6 Gestion de la mémoire 18
1.6.1 Politiques d'allocation 18
1.6.2 Politiques de récupération 23

1.7 Mécanismes de sécurité 27
1.7.1 L'identification 27
1.7.2 L'authentification 28
1.7.3 Le chiffrement 28
1.7.4 Mécanismes d'authentification 28
1.7.5 Reconnaissance biométrique 30
1.7.6 Sécurité dans un environnement réparti 30
1.8 Les technologies du futur 31
1.8.1 Carte magnétique à grande capacité 31
1.8.2 Carte optique à laser 31
1.8.3 La «super-carte intelligente» 32
1.8.4 Les cartes intelligentes sans contact 32
1.8.5 Une nouvelle carte intelligente? 32

CHAPITRE II: LE DOSSIER MÉDICAL PORTABLE (DMP) 34

2.1 Les types de cartes 35
2.1.1 Les acteurs 36
2.1.2 Les catégories de bénéficiaires 37
2.2 Les zones et les informations 38
2.2.1 Les zones logiques 38
2.2.2 Les zones physiques 40
2.2.3 Les types d'opérations sur les zones de données 41
2.2.4 Le contenu des cartes 42
2.2.4.1 La carte d'habilitation 42
2.2.4.2 La Carte Santé 45
2.3 Le dossier médical portable 48
2.3.1 Les zones logiques du DMP 49
2.3.1.1 Les zones médicales d'ordre général 49
2.3.1.2 Les zones pédiatriques 52
2.3.1.3 Les zones obstétricales 52
2.3.2 Les catégories de bénéficiaires et les zones logiques 53

2.4 L'accès à l'information et les vues 54
2.4.1 Les acteurs et leurs besoins 54
2.4.2 Les profils d'accès 55
2.5 Les types de données manipulées 56
2.6 La structure physique du dossier médical 58
2.6.1 Les zones physiques et leur contenu 59
2.6.2 Les catégories de bénéficiaires et les zones physiques 60
2.6.3 La représentation physique des données dans les zones physiques ... 61
2.7 La correspondance logique-physique 62
2.7.1 La carte d'habilitation 63
2.7.2 La Carte Santé 63
2.7.2.1 Les zones permanentes 63
2.7.2.2 Les zones du DMP 64

CHAPITRE III: SIMULATION 66

3.1 Les problèmes de saturation et de perte de cartes 66
3.2 Les buts de la simulation 67
3.3 Description des données du problème 67
3.3.1 Populations visées 68
3.3.2 Le dossier-patient utilisé 68
3.4 Le modèle de simulation 69
3.4.1 Le modèle 69
3.4.2 L'implantation du simulateur 72
3.5 Les résultats de la simulation 74
3.6 Les améliorations possibles à apporter 77

CHAPITRE IV: COMPACTAGE ET COMPRESSION DES DONNÉES 78

4.1 Généralités 78
4.1.1 Avantages et inconvénients de la compression des données 79
4.1.2 Les types de redondances 82

4.1.3 Notions sur la théorie de l'information 83
4.2 Survol des techniques de compression 87
4.2.1 Quelques définitions 87
4.2.2 Les techniques de compression de données 88
4.2.2.1 Compression logique 89
4.2.2.2 Compression physique 91
4.2.3 Les techniques utilisables ou à utiliser pour la Carte Santé 94
4.3 Algorithmes de compression et implantations 94
4.3.1 Substitution par des patrons adaptatifs 95
4.3.2 Algorithme de Huffman 97
4.3.3 Algorithme de Lempel, Ziv et Welch 103
4.3.4 Le codage arithmétique 109
4.3.5 Algorithme LZHUF 114
4.4 Les résultats de l'étude comparative 118
4.4.1 Les données manipulées 119
4.4.2 Les facteurs de mesure 123
4.4.3 Les expériences de compression 125
4.4.3.1 Description 125
4.4.3.2 Comparaison 126
4.5 Réflexions sur les résultats 133

CHAPITRE V: IMPLANTATION D'UNE CARTE SANTÉ 136

5.1 Présentation du projet Carte Santé 136
5.1.1 Le projet pilote 137
5.1.2 Les applications impliquées 137
5.1.3 Indépendance des applications vis-à-vis des technologies 138
5.1.4 Les technologies possibles 139
5.1.5 La Coquille 139
5.1.5.1 L'Interface Applicative 141
5.1.5.2 Le Pilote 142
5.2 Manipulation des données 142

5.2.1 Le format des données manipulées 142
5.2.2 Intégrité des données sur la carte 146
5.2.2.1 Atomicité logique 146
5.2.2.2 Atomicité physique 147
5.2.3 Annulation d'une information sur la carte 148
5.3 Données logiques vs données physiques 149
5.3.1 La table des correspondances 150
5.3.2 Le mécanisme de correspondance 151
5.3.3 La structure des champs et les données physiques 151
5.3.3.1 La table des structures des zones 152
5.3.3.2 La table des codifications. 152
5.3.4 Les structures de données utilisées pour le «mapping» et la codification .. 153
5.4 Le Pilote 155
5.4.1 Le «Super Driver» ou pilote universel 155
5.4.2 Les fonctionnalités 156
5.4.2.1 Formatage et analyse des données 156
5.4.2.2 Compression des données 157
5.4.2.3 Génération d'ordres pour la carte 157
5.4.2.3.1 Ordres communs de gestion 158
5.4.2.3.2 Ordres de maintenance 161
5.4.2.4 Reconnaissance des différentes technologies 162
5.5 Intégration d'un micro-serveur 164
5.5.1 Présentation du micro-serveur du Centre d'automatisation et de paiement des services de santé (CAPSS) 164
5.5.2 Proposition d'un protocole de communication 167
5.6 Les statistiques pour l'évaluation du projet 173
5.7 Perspectives pour le Pilote 173

CONCLUSION 175

BIBLIOGRAPHIE ET RÉFÉRENCES BIBLIOGRAPHIQUES 177

ANNEXES 181

A- Les zones physiques et leur contenu

B- La représentation physique des données dans les zones physiques

C- Diagramme EPAS du système Carte Santé et du Pilote

D- Classement des algorithmes par taux de compression

INTRODUCTION

Au moment où l'ère de l'information prend son essor, un nouveau média fait son apparition: la carte à puce. Cette carte que l'on dit «intelligente» est pourvue d'un microprocesseur lui accordant une certaine autonomie de gestion. Rares sont les technologies et les innovations qui exercent une aussi profonde influence sur autant de concepts comme la communication, la sécurité et la décentralisation de l'information. Depuis son invention, par Roland Moreno dans les années 70, la carte intelligente a vu son utilisation proliférer dans plusieurs domaines, notamment comme clé logique d'accès à des locaux protégés ou clé d'identification et authentification sur des réseaux informatiques, comme dossier portable d'informations confidentielles, comme portefeuille bancaire ou accumulateur de services anticipés. Son utilisation n'est réellement limitée que par l'imagination humaine.

Problématique

Dans le cadre d'une politique d'innovation, la Régie de l'assurance-maladie du Québec (RAMQ) met de l'avant un projet-pilote d'intégration d'un dossier médical portable dans une carte à microprocesseur.

Le projet de la RAMQ est l'un des plus ambitieux projets médicaux utilisant la carte intélligente jamais réalisés à nos jours. En effet, avec la richesse et la complexité des données à intégrer dans la carte à puce, les différents acteurs dans le système et la diversité des applications qui interagissent dans le projet, l'implantation du système Carte Santé est complexe et fait appel à des ressources importantes.

L'expérience est menée dans la région de Rimouski (au Québec) auprès d'enfants de 24 mois et moins, de personnes âgées de 60 ans et plus, de femmes enceintes et de quelques volontaires du village de Saint-Fabien. Plus de 9 000 cartes intelligentes seront distribuées durant le projet qui durera 18 mois.

La carte contient plusieurs informations sur différents suivis médicaux comme la

vaccination, le suivi pédiatrique, le suivi de grossesse, le suivi cardiologique, etc. On retrouve aussi des informations sur les allergies, les antécédents personnels et familiaux ainsi que les médicaments pris par le bénéficiaire. Ces informations sont adressées aux professionnels de la santé en général. Les médecins, les pharmaciens, les ambulanciers et les infirmières peuvent donc accéder à ces données. Bien entendu, l'accès à l'information n'est pas le même pour tous les professionnels de la santé. Ainsi, l'ambulancier ne peut pas accéder à toutes les données de la carte. Il en va de même pour l'infirmière ou le pharmacien. C'est par la carte intelligente que la confidentialité des renseignements est assurée en tenant compte de l'éthique médicale et la réglementation provinciale sur l'accès à l'information. L'approche que nous proposons tient compte de ces contraintes et elle est fondée sur une architecture à trois niveaux: Applications, Coquille et Cartes intelligentes (Carte Santé).

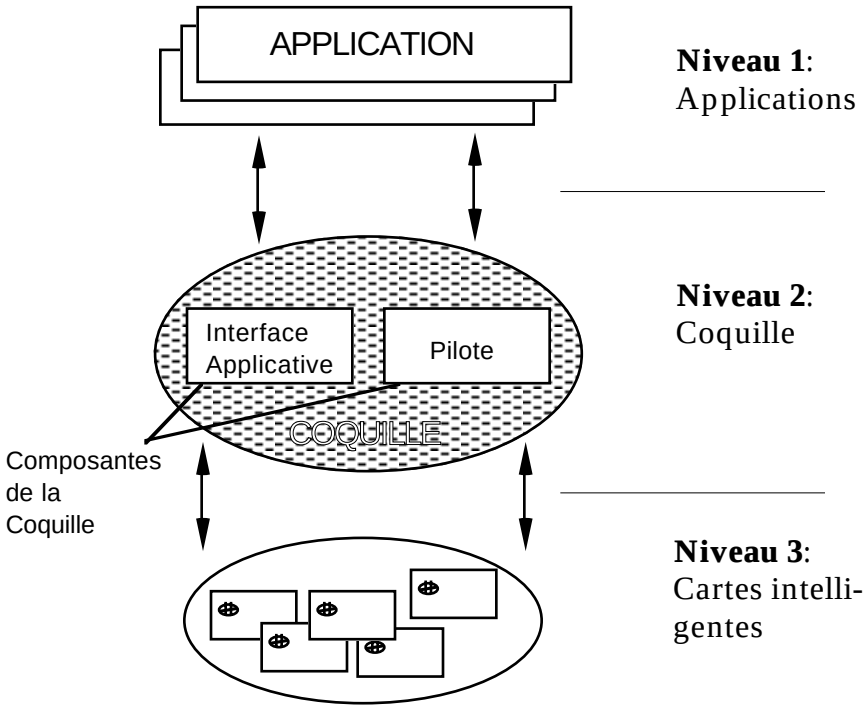


fig I.1 : Architecture à trois niveaux

La Coquille est elle-même séparée en deux modules: l'Interface Applicative qui interagit avec les applications et le Pilote qui gère les différentes cartes intelligentes. Le développement de la Coquille permet de relever plusieurs défis techniques comme une transparence des transactions des applications vers la Carte Santé, une indépendance des applications vis-à-vis de la structure des données dans la carte et la possibilité d'interagir simultanément avec plusieurs technologies de cartes intelligentes. Cependant plusieurs autres problèmes restent à résoudre, comme le problème de la saturation des zones de la carte ou la gestion de la mémoire. Nous proposons, dans ce mémoire, des scénarios possibles pour la résolution du problème de perte de la carte ainsi que des scénarios pour le retardement de la saturation de celle-ci. Nous proposons aussi plusieurs stratégies de gestion de la mémoire qui permettent de rallonger la vie d'une carte comportant des données cumulatives.

Plan

Le premier chapitre de cet essai est consacré aux cartes à mémoire et, plus particulièrement, au concept de carte intelligente. Nous présentons une étude comparative de différentes technologies de cartes à mémoire et l'évolution, à travers des générations, des cartes intégrant un microcircuit électronique. Nous décrivons les mécanismes d'authentification et les politiques de gestion de mémoire.

Après avoir présenté les informations manipulées dans le dossier médical portable (DMP), nous analysons dans le second chapitre la compléxité du DMP à travers une structure logique, telle que perçue par les utilisateurs, et une structure physique, telle que emmagasinée sur la carte à microprocesseur. Nous présentons, dans ce même chapitre, les différents acteurs exploitant le DMP ainsi que leurs droits d'accès aux données et le contenu de leur carte d'habilitation.

Le troisième chapitre présente la problématique de la saturation des cartes qui est l'un des obstacles majeurs dans le projet pilote. Nous y présentons un modèle et des études de simulation.

Le quatrième chapitre traite de la compression des données. Nous analysons différentes techniques de compression sur différents types de données et nous étudions la possibilité et l'impact sur le DMP de l'intégration d'un module de compression dans la gestion des données médicales sur une carte à mémoire.

Dans le cinquième chapitre, nous présentons le projet Carte Santé. Un bref aperçu est fait sur les différentes composantes du système Carte Santé. L'emphase est ensuite mise sur la Coquille qui permet aux applications médicales et pharmaceutiques d'interagir avec la carte intelligente. Nous expliquons le rôle des deux modules de la Coquille: l'Interface Applicative et le Pilote. Nous analysons toutes les fonctionnalités du Pilote et la possibilité d'intégrer un micro-serveur. Pour des fins d'évaluation du projet et d'amélioration du système avant une généralisation future à un niveau provincial, nous introduisons dans ce même chapitre les mesures que nous préconisons et le module de statistiques intégré au système.

CHAPITRE I

Les cartes intelligentes

Il suffit d'ouvrir son portefeuille pour y trouver de nombreuses cartes, dont des cartes de crédit, des cartes de paiement, des cartes de clubs, des cartes d'accès aux parkings, etc. En effet, l'usage des cartes est de plus en plus courant et pratique. Leur prix est relativement bas et leurs utilités pratiques. De plus, elles sont facile d'usage et leur sécurité accrue font qu'elles prolifèrent rapidement dans plusieurs domaines d'application. De nos jours qui ne possède pas de carte?

Dans ce chapitre, nous faisons le point sur l'état de l'art de la carte à mémoire. Nous présentons l'évolution des cartes jusqu'aux cartes dites intelligentes. Nous mettons ensuite l'accent sur les caractéristiques, les mécanismes et les utilisations diverses des cartes intelligentes.

1.1 Les premières cartes à mémoire

Une carte à mémoire est une carte destinée à mémoriser de l'information restituable pour des utilisations ultérieures [PARADINAS 88]. Au sens de cette définition, un permis de conduire est une carte à mémoire mémorisant par écrit des informations sur le conducteur. Dans la suite du texte, nous utilisons le terme carte à mémoire pour désigner une carte en plastique de la forme d'une carte de crédit courante et mémorisant des données de quelque façon que ce soit. Ensuite, après avoir introduit les cartes intelligentes, nous utilisons indistinctement les termes carte à mémoire, carte à puce, carte à microprocesseur ou carte intelligente.

1.1.1 Cartes estampées

C'est en 1914 que les premières cartes estampées sont apparues. Une carte estampée (ou embossée) est une carte en plastique sur laquelle des données sont inscrites en relief, directement sur le plastique, par des déformations ayant la forme de caractères alphanumériques. Ces cartes sont encore utilisées et la plus populaire au Québec est la fameuse carte d'assurance-maladie communément appelée «Carte Soleil».

Les normes concernant les polices de caractères ainsi que leurs positions sur la carte limitent à environ 127 le nombre de caractères inscriptibles sur le support plastique. La seule manière de valider l'identité du détenteur de la carte est de comparer visuellement sa signature avec celle apparaissant au verso de la carte estampée. L'information estampée est figée et non effaçable.

1.1.2 Cartes optiques perforées

fig 1.1: Carte perforée des restaurants Cesfo

Ce sont des cartes perforées en plastique rigide. Les perforations se situent sur des lignes et des colonnes précises de la carte. Leurs positions codifient des informations. Ces cartes sont lues par des lecteurs optiques qui mettent une source lumineuse sur une face de la carte et décodent les petits faisceaux de lumière générés sur la seconde face par les perforations. L'information codée sur de telles cartes est limitée et non modifiable. Le clonage de ces cartes est relativement facile.

1.1.3 Cartes magnétiques

Dans les années 70, les cartes ont commencé à être pourvues d'un support magnétique collé au verso. Aujourd'hui, des normes internationales règlementent le nombre de pistes magnétiques, leur position ainsi que les modes d'enregistrement. Contrairement aux cartes estampées et aux cartes perforées, qui ne présentent aucune sécurité contre les utilisations frauduleuses et les duplications, les cartes magnétiques offrent un certain niveau de sécurité. Les applications utilisant ces cartes attribuent à chaque porteur un NIP (Numéro d'Identification Personnelle). Le NIP est un code qui est inscrit chiffré sur la carte. Pour être reconnu comme titulaire de la carte, le porteur doit, à chaque utilisation, s'identifier en introduisant son NIP. Selon les applications, l'information enregistrée sur les pistes magnétiques peut être effacée et modifiée. Il n'y a malheureusement aucun mécanisme permettant la protection de la carte contre des modifications illicites.

Les cartes magnétiques peuvent aussi être estampées pour permettre l'impression de certaines informations sur du papier carbone. Ce genre de carte mixte est très utilisé de nos jours dans toutes nos transactions quotidiennes comme les cartes de crédit pour nos paiements à crédit et les cartes bancaires utilisées dans le réseau de guichets automatiques.

1.2 Les cartes à puce

Le premier brevet sur les objets mémoire sécurisés fût déposé en 1970 par le japonais Kanitaka Arimura. En 1974, Roland Moreno, un inventeur français, dépose un brevet sur un procédé de carte à mémoire autoprotégée. Au début des années 80, M. Ugon eu l'idée de rajouter un microprocesseur (microcalculateur programmable) au microcircuit. Aujourd'hui, Moreno est considéré comme le père de la carte à puce.

Cette invention géniale incorpore sur une carte en plastique une mémoire et un microcircuit destiné à gérer et à protéger celle-ci. Cette carte est venue pallier les nombreux inconvénients des autres cartes à mémoire. Sa duplication est quasi impossible et sa protection contre les contrefaçons est excellente. La carte ressemble aux cartes de crédit standards à la différence qu'elle porte une puce, d'où son nom *carte à puce.*

1.2.1 Les cartes non intelligentes et les cartes intelligentes

On distingue deux familles de cartes à puce; les cartes non intelligentes et les cartes intelligentes. Les cartes non intelligentes sont les cartes à microcircuit simple et les cartes à logique câblée. Ces deux types de carte ont la particularité de ne pas pouvoir assurer la protection en lecture et en écriture. On verra dans le paragraphe 1.3 que celles-ci sont souvent utilisées pour les applications de pré-paiement. Les cartes à logique câblée peuvent avoir un accès sécurisé mais la sécurité est rudimentaire et figée. Les cartes à logique câblée ne sont pas programmables et ne peuvent être utilisées que pour les opérations pour lesquelles elles ont été conçues.

Les cartes intelligentes sont dotées d'un microprocesseur leur permettant d'effectuer des calculs complexes. Il est possible, avec ces cartes, d'avoir un mécanisme de sécurité très sophistiqué et modifiable par les applications elles-mêmes. Ainsi, une carte peut accepter ou refuser une écriture ou une opération de lecture sur ses informations. Les microprocesseurs encartés sont qualifiés de MAM (Microcalculateur Autoprogrammable Monolithique) ou en anglais SPOM (Self-Programmable One chip Microcomputer).

1.2.2 L'architecture des cartes à puce et les standards ISO

La puce est un composant monolithique, c'est-à-dire qu'elle contient dans un même substrat de silicium l'ensemble des éléments qui la composent [PARADINAS 88]. On dit aussi que le microprocesseur est 'mono-chip'. Ceci permet de sécuriser la communication entre les composants du microprocesseur puisqu'aucune interception ne peut être faite. On retrouve, dans une même puce de silicium, une unité de calcul CPU (Central Processing Unit), une mémoire réservée au système, une mémoire pour l'utilisateur (dite mémoire auxiliaire) et un bus pour connecter les différents composants.

L'organisation internationale de la standardisation (ISO) a défini des normes concernant les caractéristiques internes et externes de la carte pour assurer une certaine uniformité entre les différentes technologies de carte à microprocesseur. Ces normes ont été adoptées progressivement et fixent des standards reconnus aux niveaux suivants:

- des caractéristiques physiques: Le support plastique est en PVC ou ABS et son format est d'une longueur de 85,60 mm (± 0,12), d'une largeur de 53,98 mm (± 0,05) et d'une épaisseur de 0,76 mm (± 0,02).

- de la position et des affectations des contacts: Il y a huit contacts différents dont six sont actuellement affectés et utilisés. Ils sont regroupés sur une plaque (pastille) qui chapeaute le microprocesseur. La position de cette pastille tient compte des contraintes physiques et permet l'ajout d'une piste magnétique.

- des signaux électriques et du protocole de communication: Les tensions électriques aux bornes, la fréquence de l'horloge et la vitesse de communication ainsi que la structure de la réponse à la remise à zéro (RESET) sont régie par des règles strictes. Le mode d'échange standard est série asynchrone avec une vitesse de transfert de 9600 Bauds et le format des données transmises est :

1 bit START	8 bits de données	1 bit de parité	1 bit STOP

Certains désaccords existent encore entre l'ISO et l'AFNOR (Association Française de Normalisation) quant à la position de la puce. Il existe donc une position haute, dite française, reconnue par l'AFNOR et une position basse, dite japonaise. Au niveau de l'ISO, la position haute est transitoire. En général, les lecteurs de cartes actuels acceptent indifféremment les deux positions de la puce. Il existe d'autres aspects importants encore en discussion pour une normalisation du protocole de communication et même du type de communication synchrone ou asynchrone. Le protocole T0, reconnu par ISO, transmet des caractères. Certaines technologies utilisent un autre protocole (T14) qui transmet des paquets. Ce protocole de transmission par paquet est actuellement proposé pour être standardisé (T1).

1.2.3 Les types de mémoire

Comme nous l'avons vu dans le paragraphe précédent, il existe dans la carte une mémoire du système et une mémoire auxiliaire. Pour la mémoire auxiliaire, on distingue deux technologies de mémoire; **EPROM** et **EEPROM**. La technologie EPROM (Erasable Programmable Read Only Memory) permet d'avoir une mémoire programmable et effaçable uniquement par rayon ultra-violet. Elle est considérée comme non réutilisable puisque l'effacement n'est pas possible. Une carte dotée d'une mémoire EPROM ne peut être utilisée en écriture qu'une seule fois à une même adresse. La technologie EEPROM (Electricly Erasable Programmable Read Only Memory) permet d'avoir une mémoire réutilisable. On peut écrire et réécrire indéfiniment[1] à une même adresse. Une carte dotée d'un tel type de mémoire peut être effacée et réutilisée. Néanmoins, certaines précautions sont à prendre quant à la cohérence des données inscrites puisque des écritures et des effacements partiels dûs à des interruptions peuvent être très dangereux.

La mémoire du système est composée d'une mémoire morte ou **ROM** (Read Only Memory), d'une mémoire vive ou **RAM** (Random Acces Memory) et parfois d'une mémoire programmable ou **PROM** (Programmable Read Only Memory). La ROM n'est accessible qu'en lecture et contient le système d'exploitation du

1. 200 000 à 500 000 fois

microprocesseur (masque) ainsi que les opérations de base. La RAM est une mémoire volatile dont le contenu s'efface automatiquement après une mise hors-tension du circuit de la puce. Elle sert de mémoire temporaire au système d'exploitation pour effectuer des calculs et stocker temporairement des résultats intermédiaires. Quant à la PROM, c'est une mémoire programmable une seule fois permettant une extension à la ROM en rajoutant des nouvelles fonctionnalités au système d'exploitation.

Lorsque le microprocesseur est doté d'une mémoire du type EEPROM, les parties de la mémoire système ROM et PROM peuvent être comprises dans la mémoire EEPROM et protégées pour interdire à l'usager d'y écrire.

La capacité de la mémoire EEPROM ou EPROM accessible à l'utilisateur, détermine celle de la carte à puce.

1.2.4 Encartage

L'encartage désigne le processus qui permet de mettre le microprocesseur sur son support plastique. Le microcircuit est collé ou incrusté dans du plastique puis une plaque métallique encapsule le composant. Cette plaque est une pastille de contact qui protège le composant et permet d'établir les contacts.

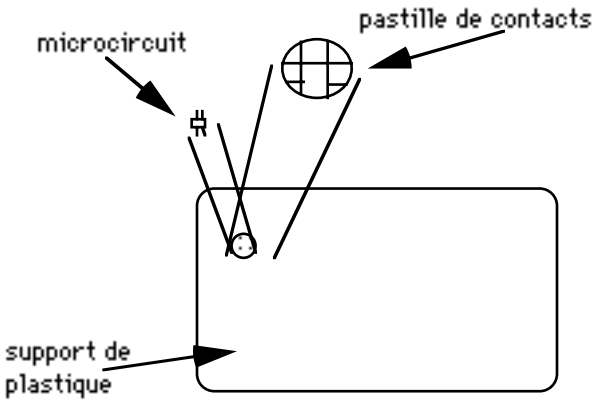


fig 1.2: Encartage de la puce

L'encartage est très important quand on parle de la qualité d'une carte à microprocesseur. Il existe plusieurs techniques d'encartage. Le support plastique peut être coupé à partir de grandes plaques ou moulé individuellement. Dans le premier cas, on ne peut que coller le microcircuit après avoir appliqué un alézage dans le plastique. Il est important de noter les enjeux dûs à la colle qui doit tenir compte des déformations, des contraintes mécaniques (flèche et torsion) et de la chaleur dégagée par le composant électronique.

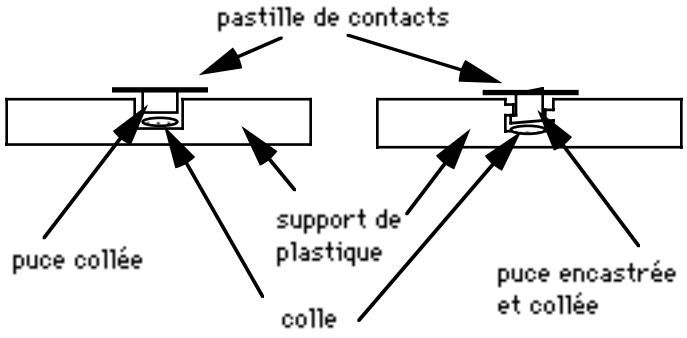


fig 1.3: Collage et incrustage de la puce

1.2.5 Cycle de vie d'une carte à puce

Le cycle de vie d'une carte à puce comprend six étapes: la fabrication, l'initialisation, la personnalisation, l'exploitation, le blocage et l'annulation.

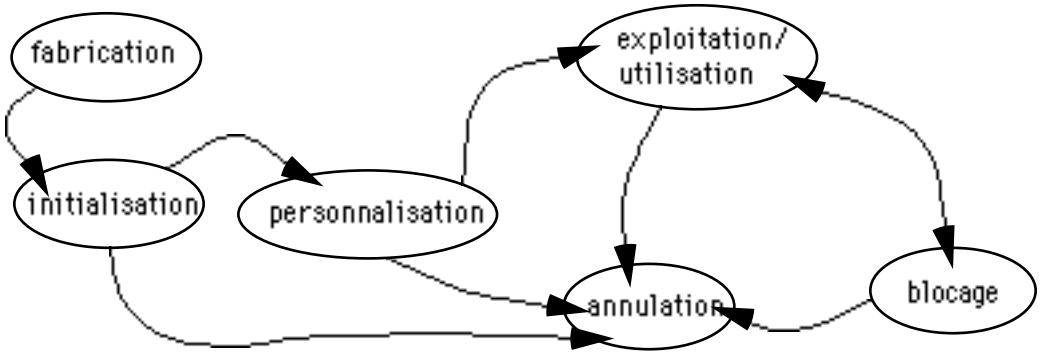


fig 1.4: Le cycle de vie de la carte

La fabrication: C'est la phase de création. Les composantes électroniques de la puce sont réalisées et testées. Le système d'exploitation (masque) et les fonctions de base de la puce sont enregistrés dans la mémoire ROM du microprocesseur. Cette phase se termine par l'encartage de la puce sur un support plastique.

L'initialisation: Elle consiste à diviser la mémoire de la puce en zones distinctes en tenant compte des besoins de l'application qui va utiliser la carte. À cette étape, certaines informations propres à l'application utilisant la carte et à l'émetteur sont inscrites dans la mémoire. Cette phase est souvent confondue et englobée dans la phase de personnalisation. En effet, les deux phases peuvent techniquement se réaliser en même temps.

La personnalisation: Elle consiste à identifier et à attribuer la carte à un porteur bien déterminé en enregistrant les données propres à ce dernier comme son identification et ses codes secrets. Cette phase est cruciale dans le cycle de vie de la carte et sa fin est signalée au masque. Suivant les technologies des cartes, plusieurs opérations ne sont plus permises après cette phase.

L'exploitation: Cette phase, appelée aussi «utilisation», est la phase la plus importante durant laquelle la carte est utilisée pour inscrire des données ou en lire. Cette phase, qui constitue la partie principale du cycle de vie, peut être interrompue par un blocage de la carte ou une annulation à la suite d'une saturation.

Le blocage: Appelée aussi 'hibernation', cette phase est provisoire et est provoquée à la suite d'un blocage des fonctions de la carte. Lorsqu'une tentative frauduleuse est détectée par la carte, celle-ci s'auto-verrouille pour interdire tout accès et devient inactive. Cette phase est interrompue par un déverrouillage qui nécessite l'intervention de l'émetteur. Certains blocages dûs à des erreurs de manipulation graves peuvent entraîner l'annulation définitive de la carte.

L'annulation: C'est la dernière phase du cycle de vie de la carte. Une carte annulée est dite morte et ne revient plus à l'état d'exploitation. Elle se traduit, suivant les applications, par une destruction de la carte ou son renouvellement.

Mis à part la fabrication, toutes les phases peuvent engendrer l'annulation de la carte pour des raisons diverses. Les annulations à la phase de fabrication sont considérées comme des rebuts industriels.

1.3 Les applications de la carte à microprocesseur

Grâce à son support actif, la carte à puce peut entamer des dialogues intelligents avec les systèmes dans lesquels elle s'insère. C'est ce qui fait que de nombreux champs d'application s'offrent aujourd'hui à la carte intelligente dans des domaines aussi variés que les services de paiement et de pré-paiement (monétique), la sécurité physique et logique, ainsi que les dossiers portables. D'autres applications plus localisées mais non moins considérables, comme les loisirs (cinéma, club ...), laissent entrevoir l'ouverture d'un marché non négligeable pour la carte à mémoire intelligente.

Nous allons présenter quelques applications originales autour de trois axes principaux: le paiement, le stockage d'informations et la sécurité d'accès.

1.3.1 Les cartes à données déductives

Ce sont généralement des cartes à microcircuit simple ou à logique câblée comme pour les pré-paiements ou à microprocesseur comme pour les cartes de paiement bancaire.

Cartes de pré-paiement: Ce sont des cartes achetées par le porteur directement auprès du prestataire de services. Elles contiennent des unités de consommation. Ces unités pré-payées sont «grillées» au fur et à mesure des consommations effectuées par le porteur. L'application la plus connue de ces cartes de pré-paiement est la télécarte qui, en France, a eu un grand succès commercial après que France Télécom ait vendu plus de 45 millions de cartes. La télécarte est dite jetable après usage car elle n'est pas recyclable après épuisement de ses unités de consommation. Avec certaines applications, il est possible de recharger la carte auprès de l'organisme émetteur après l'épuisement d'une partie ou de la totalité des unités chargées. C'est le cas de la cinécarte qui a été présentée en France en 1987 par la chaîne Pathé Cinéma pour fidéliser sa clientèle. La carte contient

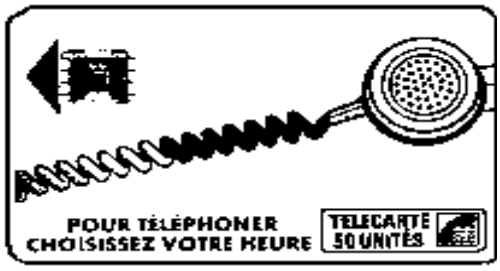


fig 1.5: la télécarte de France Télécom

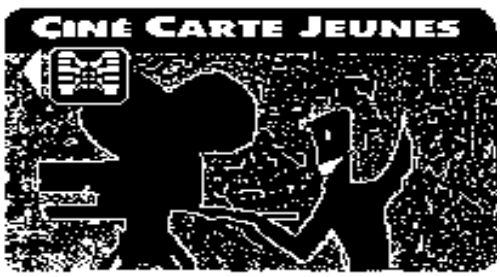


fig 1.6: la cinécarte de Pathé cinéma

initialement un nombre de séances pré-payées. Ce nombre est décrémenté à chaque utilisation et peut être rechargé six fois.

Cartes de services: Ce sont des cartes d'abonnement aux services. La carte est chargée de droits d'accès à des services pour une période limitée. On peut retrouver ces cartes dans des centres de loisir, par exemple. Les abonnés détiennent une carte nominative leur permettant d'accéder aux prestations offertes tout en comptabilisant les services utilisés. Les opérations réalisées par le porteur lui sont alors facturées ultérieurement. Les visiteurs occasionnels disposent à l'entrée d'une carte chargée d'un certain crédit. La carte est pré-payée. À la sortie, en introduisant la carte dans un appareil automatique de remboursement, le visiteur se fait rembourser le solde du crédit non consommé. Ces cartes sont ensuite ré-utilisées pour d'autres visiteurs.

Cartes de paiement: Ce sont des cartes intelligentes distribuées aux clients d'une banque et chargée chacune d'un plafond mensuel d'achats variant en fonction de la solvabilité de chaque client. À chaque achat, le montant de règlement effectué par le client ainsi que le nouveau solde et la date sont inscrits sur la carte. Le plafond mensuel est alors décrémenté et seules les transactions effectuées dans la limite de ce nouveau plafond sont autorisées. La sécurité offerte par le microprocesseur permet au commerçant d'accepter en toute confiance les paiements effectués avec ce porte-monnaie électronique.

fig 1.7: la carte de paiement Visa/ BNP

1.3.2 Les cartes à données cumulatives

Grâce à sa capacité-mémoire, la carte intelligente peut répondre d'une façon idéale aux besoins de stockage de données évolutives liées à une personne ou à un produit. La carte à données cumulatives est une carte qui se remplit au fur et à mesure que de nouvelles données privatives se présentent. C'est un dossier portable. Les informations contenues dans le dossier peuvent être, par exemple, des données concernant la fabrication d'un produit pour son suivi de production, le dossier scolaire ou universitaire ou le dossier de santé d'un porteur.

Cartes d'étudiants: La carte mémorise toute information utile à l'étudiant dans ses démarches administratives et pédagogiques. Elle contient des informations sur la scolarité de l'étudiant porteur, comme les matières choisies, les notes obtenues, ses diplômes, etc. Plusieurs projets de dossier scolaire portable ont été fais en Europe comme à l'université de Rome en Italie et à l'université de Lille en France. Certaines cartes

fig 1.8: la carte étudiant de Lille

proposent d'autres services à l'étudiant; l'accès aux bibliothèques, les paiements à la cafétéria...

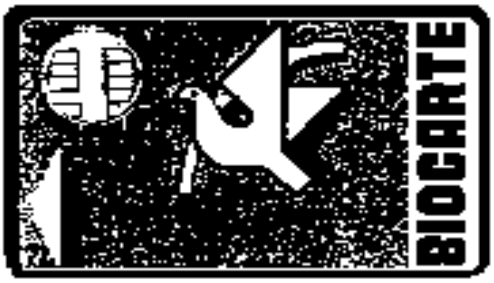


fig 1.9: la carte patient BIOCARTE

Cartes de santé: La carte mémorise, pour un bénéficiaire, des informations médicales variées concernant son état de santé. D'un côté, la carte rassemble des informations qui sont normalement disséminées géographiquement entre les médecins consultés, les laboratoires d'analyses et les centres hospitaliers fréquentés par le bénéficiaire et, d'un autre côté, elle redistribue vers les traitants l'information qui est centralisée. La carte à mémoire est utilisée pour améliorer les soins apportés à l'individu. Plusieurs expériences utilisant la carte intelligente dans le domaine de la santé ont été conduites, dont notamment six en France comme Biocarte, Santal, la carte Transvie et la carte santé. Cette carte peut aussi être utile pour des fonctions de gestion en matière de santé comme l'admission dans un centre hospitalier, la facturation etc.

Livrets de bords: Ce sont des cartes intelligentes associées à un véhicule d'une compagnie de transport ou un véhicule en location. Elle prélève des informations concernant le fonctionnement du véhicule et constitue un carnet de bord. Les informations recueillies dans le livret portable servent au garage d'entretien ou à la compagnie de location pour facturer le client.

1.3.3 Les cartes-clés électroniques

La carte intelligente peut être utilisée pour sécuriser l'accès logique à des informations confidentielles ou à des réseaux informatiques. Elle peut aussi être utilisée pour sécuriser l'accès physique à des sites ou locaux protégés. Ses mécanismes d'identification et d'authentification font d'elle une véritable clé électronique. Selon les applications, la carte peut aussi être utilisée pour contrôler les fréquences d'accès, par exemple.

Plusieurs autres applications sont possibles. On note notamment la location de logiciels où la carte sert non seulement de clé d'accès logique mais comme compteur d'utilisation. La carte, qui est initialement chargée avec des unités de consommation par la compagnie locatrice, décompte les unités au fur et à mesure que le logiciel associé à la carte est utilisé. Lorsque la carte est épuisée, elle peut être téléchargée par l'émetteur.

1.4 Les générations de cartes intelligentes

De la carte à unique zone de transaction avec simple mécanisme de sécurité à la carte multiservice, multiprestataire avec une sécurité sophistiquée, il y a eu plusieurs générations de cartes intelligentes. La carte a évolué en répondant à la croissance des besoins et des exigences des applications. Nous avons classé l'évolution de la carte intelligente en trois générations. Une première concerne la carte simple avec

quelques zones, dont une seule pour les transactions. La deuxième génération est celle des cartes avec une possibilité de créer plus d'une zone. À la troisième génération, les cartes sont devenues multiservice, multiprestataire. Certains experts considèrent la «super smart card» comme étant la dernière génération des cartes (voir 1.8: Les technologies du futur). Nous ne la traitons pas ici car, pour l'instant, elle n'est encore qu'un gadget à l'état expérimental.

1.4.1 Première génération (mono-application, monoprestataire)

La première génération de carte disposait d'un microcalculateur simple ayant des fonctionnalités rudimentaires pour les entrées-sorties, la gestion de la mémoire et les procédures de sécurité. Les cartes de cette génération n'ont qu'une seule zone pour les transactions, ce qui les limite à une seule application et à un seul émetteur d'où leur qualificatif de mono-application et de mono-prestataire. Pour illustrer ces fonctionnalités, nous présentons dans ce qui suit les cartes Bull CP8 masque 4 et masque 9. Les cartes fonctionnent essentiellement de la même façon sauf que la carte masque 9 dispose d'une mémoire EEPROM lui permettant une réécriture. La mémoire est de 8K bits et se décompose en 256 mots de 32 bits. Elle est divisée en six zones distinctes:

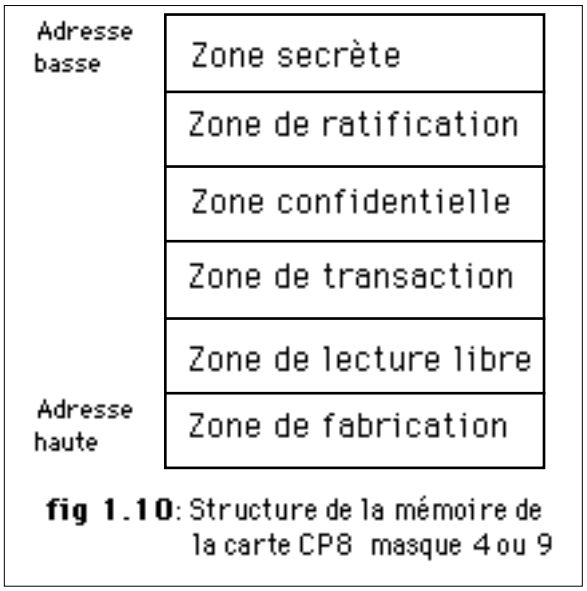


fig 1.10: Structure de la mémoire de la carte CP8 masque 4 ou 9

-**Zone secrète**: qui contient les clés d'accès telles que le NIP, la clé de l'émetteur et la clé du gestionnaire. Elle n'est accessible qu'au microprocesseur.
-**Zone de ratification**: qui contient l'historique des présentations de clés. Elle est accessible en lecture mais nul ne peut y écrire sauf le microprocesseur. Après trois échecs de présentation de la même clé, la carte se verrouille et ne peut être remise en fonction qu'avec la soumission de la clé de l'émetteur et du NIP.
-**Zone confidentielle**: qui contient des informations confidentielles permanentes. Elle n'est accessible en lecture qu'après avoir présenté une clé.
-**Zone de transaction**: elle est destinée à la réception des inscriptions successives de l'application. Elle peut être protégée par une clé. Cette zone est la seule dans laquelle on peut écrire après la phase de personnalisation.
-**Zone de lecture libre**: qui contient des informations permanentes qui peuvent être lues librement.
-**Zone de fabrication**: qui contient les adresses des zones précédentes ainsi que les caractéristiques de protection de la zone de transaction. Cette zone, qui a une taille fixe, contient donc les tailles des cinq autres zones.

Avant la mise en service de la carte, toutes les zones, à l'exception de la zone de transaction, sont initialisées et leurs adresses respectives sont inscrites dans la zone

de fabrication. Après la personnalisation, l'écriture n'est permise que dans la zone de transaction. La zone de transaction, qui est l'unique zone de l'application, est composée d'un certain nombre de mots de 32 bits. Trois bits de chaque mot sont réservés par le système. Le premier est destiné à la validation du mot (les 29 bits restant). C'est-à-dire qu'après l'écriture de ce bit, aucune modification du mot n'est permise. Les deux bits suivants identifient le type de clé à soumettre avant la validation du mot.

Ce type de carte dispose aussi de fonctions de calcul cryptographique se basant sur des paramètres externes comme un nombre aléatoire et des clés secrètes propres à la carte pour assurer des opérations comme l'authentification, le chiffrement ou la télé-écriture.

L'inconvénient majeur de cette génération de carte est que la manipulation de l'information au sein de la mémoire se fait par adressage absolu aux mots ou du moins par adressage relatif en se référant à l'adresse du début d'une zone. Ceci rend la programmation et la manipulation de l'information plus délicate.

1.4.2 Deuxième génération (multi-application, monoprestataire)

Les cartes de la deuxième génération ont une plus grande mémoire que les cartes de la première et disposent de plusieurs zones pour les transactions ainsi qu'un mécanisme de sécurité permettant d'attribuer à chaque zone un profil d'accès différent et autonome. Ceci permet à différentes applications de se partager la mémoire de la carte. Ainsi avec la carte IBM, on peut avoir quatre applications concurrentes, alors qu'avec la carte TOSMART-A de Toshiba, cinq applications différentes peuvent se partager l'espace de données de la mémoire.

La carte TOSMART-A est bi-chip et elle est équipée d'une mémoire de 64K bits. La mémoire est divisée en deux parties; l'une pour les données, composée de zones appelées «aires», et l'autre pour le système et contient les paramètres des cinq applications. Chaque application est autonome et possède ses propres clés d'accès et son niveau de sécurité. Les aires et les droits d'accès associés à une application sont décrits dans un descripteur de répertoire. Chaque application possède un répertoire et une aire peut être associée à plusieurs applications différentes.

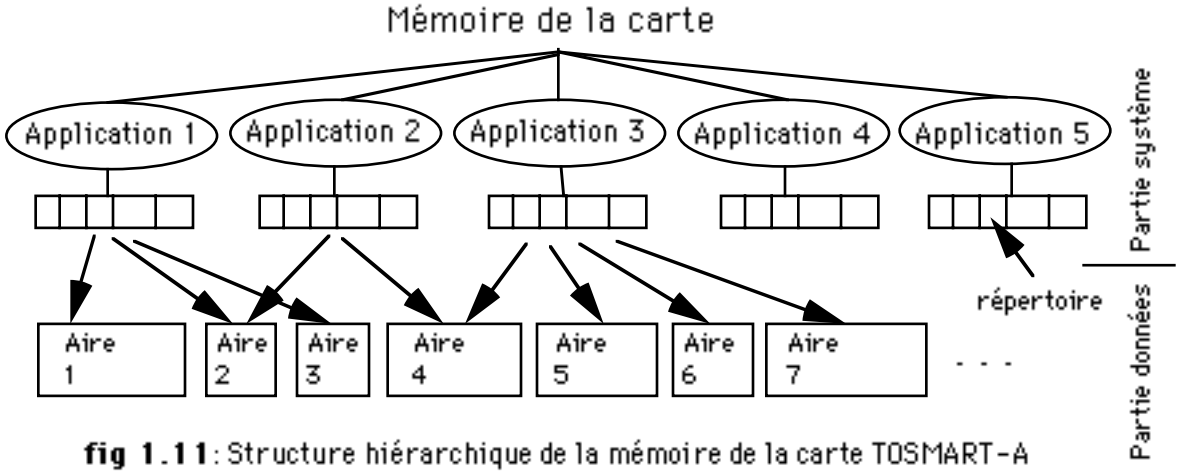


fig 1.11: Structure hiérarchique de la mémoire de la carte TOSMART-A

La carte possède une clé pour l'émetteur, un NIP pour le porteur et cinq clés d'authentification pour les applications. Un mécanisme permet de compter les échecs de soumission de clés et verrouille la carte si ce nombre dépasse un seuil établi au préalable.

De son côté, la carte IBM possède une mémoire EEPROM de 64K bits composée de zones appelées «bloc». Chaque bloc est autonome et possède une partie système appelée «en-tête» et une seconde partie pour les données. L'en-tête comporte entre autres, l'identification du bloc, la clé du bloc, et le profil d'accès des quatre applications appelé «indicateur d'autorisations». L'indicateur d'autorisations est un octet où chaque paire de bits indique pour une application s'il lui est autorisé de lire ou d'écrire. C'est ce qui limite le nombre total d'applications à quatre.

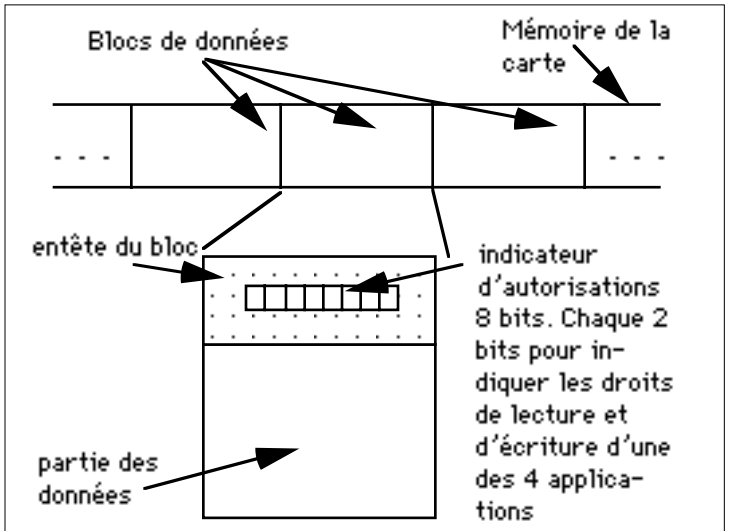


fig 1.12: structure de la mémoire de la carte IBM

De cette façon, les blocs de données peuvent être partagés par les quatre applications ou réservés à une seule application. Chaque application a son propre profil d'accès qui contient non seulement une clé mais spécifie pour chaque application, les commandes du microprocesseur autorisées pour celle-ci. La carte IBM possède aussi un mécanisme de sécurité très sophistiqué permettant par exemple de contrôler le fonctionnement de la carte à des horaires donnés et à des jours particuliers de l'année.

1.4.3 Troisième génération (multi-application, multiprestataire)

Les cartes de la seconde génération permettent pour un même prestataire d'avoir plusieurs applications concurrentes partageant les mêmes données sur la carte, comme pour une carte étudiant où l'on retrouverait un dossier scolaire pour une application administrative, les prêts de livres pour l'application de la bibliothèque et des unités de consommation pour l'application monétique de la cafétéria. Elles ne permettent pas à plusieurs prestataires de partager la mémoire d'une même carte, du moins d'une façon sécuritaire. C'est ce qui est offert par la troisième génération de cartes. La mémoire de ces cartes est subdivisée en répertoires et sous-répertoires possédant chacun son propre bloc de sécurité. Cette hiérarchisation sécurisée des zones permet de protéger adéquatement les données et tolère la cohabitation d'informations de plusieurs émetteurs. Ainsi, dans une carte bancaire, on peut retrouver des unités de consommation pour le téléphone ainsi que des informations pour l'accès à des locaux, par exemple. Ce qui fait la différence entre les générations de cartes, c'est le masque et les possibilités de partage sécurisé de la mémoire. Pour

illustrer le mécanisme de gestion de la mémoire offert par la troisième génération, nous présentons deux cartes différentes la carte MCOS de Gemplus et la carte MP de Bull.

La carte MCOS: La carte MCOS et une carte EEPROM de 32K bits. Elle est structurée comme un disque d'un micro-ordinateur. La mémoire commence par une zone d'identification et une zone de contrôle. Le reste de la mémoire est partagé entre une zone pour les données et leurs codes secrets, et une zone pour la table d'allocation. Dans la zone des données, l'information est emmagasinée en fichiers. Les fichiers sont rassemblés en répertoires comme les répertoires de fichier sous le système d'exploitation MS-DOS. Chaque répertoire a son propre bloc de sécurité et chaque fichier est protégé en lecture, en écriture et en réécriture. L'ensemble des répertoires de la carte est inclu dans un répertoire principal protégé par un bloc principal de sécurité. Ce bloc contient le NIP du porteur de la carte. Chaque bloc de sécurité, qu'il soit celui du répertoire principal ou de tout autre sous-répertoire, contient huit mots de passe. Le premier est réservé à l'émetteur ou le prestataire intéressé par le répertoire, et les sept autres servent comme clés pour protéger les accès aux zones. Théoriquement, on peut avoir 31 répertoires et sous-répertoires différents, ce qui permet à un maximum de 31 prestataires de partager la mémoire de la carte d'une façon sécuritaire. Chaque prestataire peut avoir jusqu'à sept applications concurrentes, vu les codes secrets disponibles. La zone pour la table d'allocation est comme la FAT (File Allocation Table) sur un disque dur. Elle contient les descripteurs des fichiers de l'ensemble des répertoires.

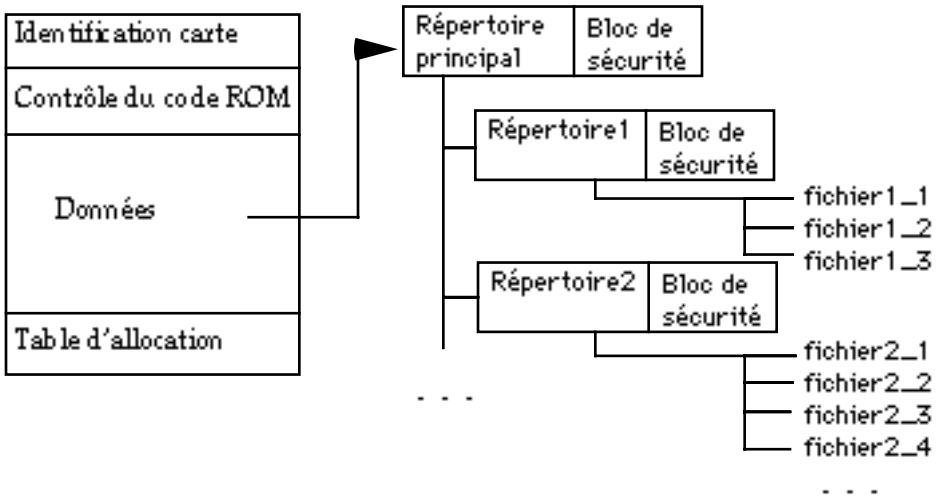


fig 1.13: Schéma de la structure de la mémoire d'une carte MCOS

Cette carte a la particularité de pouvoir contenir des programmes exécutables que l'on peut ajouter lors de la personnalisation et qui accroissent les fonctionnalités du masque de la carte.

La carte MP: La carte MP de Bull dispose d'une mémoire EPROM de 64K bits. Elle a une structure hiérarchique à trois niveaux: carte, application et service. L'ensemble de la mémoire de la carte constitue le premier niveau. Ce niveau peut être partagé

ou contenir des blocs autonomes; ce sont les applications. Chaque application peut être divisée en sous-blocs; ce sont les services. Le niveau carte contient des informations relatives à l'émetteur et au porteur de la carte. Quant au niveau application, il contient des entités allouées par l'émetteur à différents prestataires. Le nombre de prestataires avec cette carte n'est limité que par la taille de la mémoire. Le niveau service contient des entités assignées aux applications. Chaque application peut partager la mémoire lui étant allouée à différents services. La carte, comme toute application et tout service, peut être contrôlée par un jeu de clés qui lui est propre. Lorsqu'une application est créée, on lui alloue un espace mémoire qui reste fixe. De même, on alloue un espace mémoire fixe à un service. À n'importe quel niveau des trois niveaux de la hiérarchie, on peut rajouter des zones (ou fichiers élémentaires). La taille d'une zone (ainsi que le nombre des zones) n'est limitée que par la taille de la mémoire réservée à l'entité à laquelle la zone est rattachée. Il y a quatre types de zones différentes: les zones secrètes pour recevoir les données de contrôle exploitées par le microprocesseur comme les clés et les codes; les zones d'accès pour recevoir l'historique des présentations de clés; les zones publiques pour recevoir les données à lecture libre de l'usager; et enfin, les zones de transactions pour recevoir les transactions à accès paramétrable.

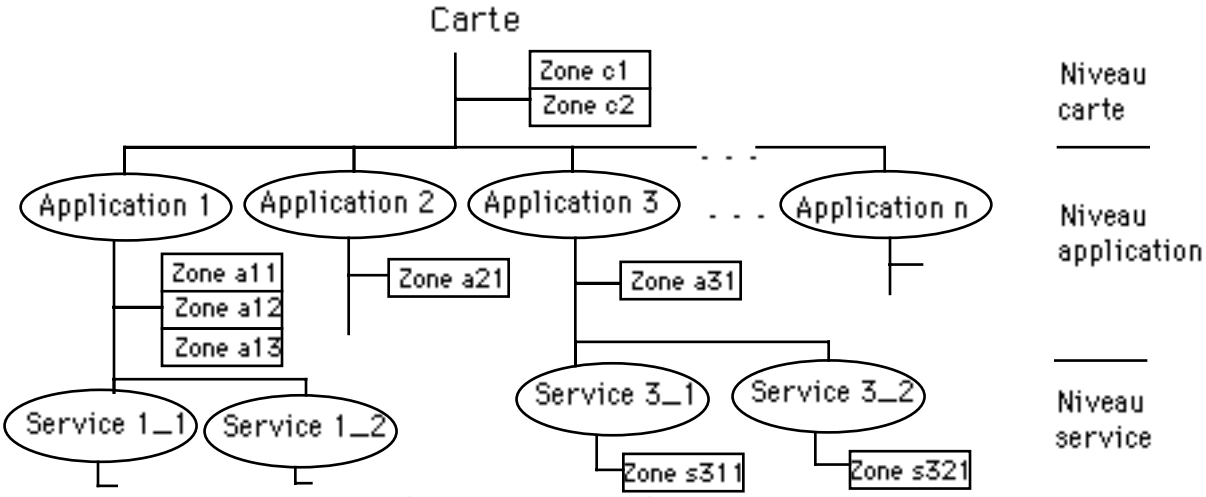


fig 1.14: Structure hiérarchique de la mémoire de la carte MP de Bull

1.5 Le choix d'une carte

Le nombre et la diversité des technologies de carte à mémoire rendent le choix d'une carte parfois très délicat. Chaque technologie comporte des avantages ainsi que des inconvénients. Le choix n'est certainement pas évident, mais il dépend nécessairement des objectifs que veut atteindre l'organisation émettrice avec l'introduction des cartes intelligentes. Lors d'une telle prise de décision, il faut considérer les points cruciaux suivants:

- la capacité mémoire nécessaire pour l'application;
- le niveau de sécurité exigé par l'application;
- le niveau de segmentation de la mémoire exigé par l'application;
- la pertinence et la possibilité d'effacer des données sur la carte;

- la ré-utilisabilité de la carte après son cycle de vie (carte jetable ou recyclable);
- l'utilisation monoservice ou multiservice de la carte et;
- l'utilisation monoprestataire ou multiprestataire.

1.6 Gestion de la mémoire

Les techniques de gestion de mémoire diffèrent d'une technologie à l'autre. Ces techniques cherchent à optimiser l'utilisation de l'espace mémoire de la carte. Elles sont tributaires du type de mémoire de la carte ainsi que de son masque, et peuvent varier de la plus simple, avec une unique zone séquentielle, à la plus complexe, avec plusieurs zones dynamiques. Une mémoire de type EPROM et une mémoire de type EEPROM ne permettent pas les mêmes techniques de gestion de mémoire. En effet, aucune ré-utilisation d'espace déjà alloué n'est possible avec les mémoires EPROM, par exemple.

Optimiser l'utilisation de la mémoire implique, pour une carte à microprocesseur, de minimiser les pertes de mémoire inutiles et ainsi reporter à plus tard l'échéance de la saturation de la mémoire. Cette optimisation se fait lors de l'allocation des blocs de mémoire ainsi que par des récupérations d'espace déjà alloué et inutilisé pour une nouvelle utilisation.

Dans ce qui suit, nous présentons différentes techniques d'allocation de mémoire et différentes techniques de récupération de mémoire déjà utilisée. Nous expliquons leurs contraintes, leurs avantages et leurs défauts.

1.6.1 Politiques d'allocation

Une politique d'allocation de mémoire consiste à préciser la stratégie de division de la mémoire et de son association aux différentes zones.

Nous distinguons deux classes de politiques d'allocation de mémoire. Les politiques de la première classe sont dites statiques et consistent à subdiviser définitivement toute la mémoire en une seule étape. Les politiques de la seconde, qui allouent des espaces mémoire au fur et à mesure des besoins, sont dites dynamiques. Elles ne sont possibles qu'avec une mémoire EEPROM. La seconde classe est aussi composée de deux sous-classes: une pour les politiques qui ne peuvent pas varier la taille d'un bloc une fois alloué et une autre pour celles qui varient la taille d'un bloc après son allocation.

Nous désignons par bloc un quantum d'espace mémoire. Une zone est un ensemble de données sur la carte, liées sémantiquement par des droits d'accès. Une zone peut être composée de plusieurs blocs sur la carte.

Allocation Statique Taille Fixe (ASTF): Toute la mémoire est divisée en blocs de taille fixe et identique. Cette division se fait en une seule étape avant l'exploitation, généralement au cours de la personnalisation ou de l'initialisation de la carte. La portion de la mémoire qui n'est pas allouée est perdue et ne peut être allouée

ultérieurement au cours de l'exploitation de la carte. Il est alors recommandé de diviser la totalité de la mémoire disponible.

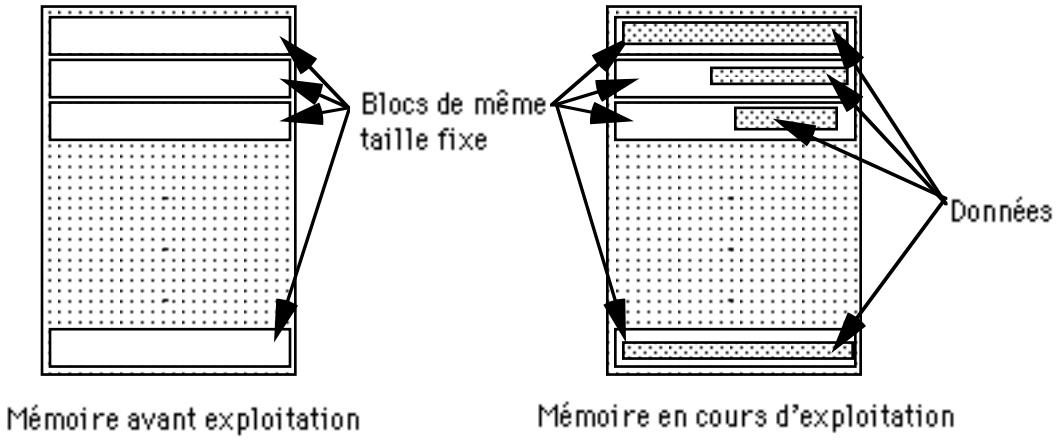


fig 1.15: Allocation statique taille fixe

Étant donné que sur une carte, chaque bloc consomme un certain espace pour sa gestion, il est plus intéressant de minimiser le nombre de blocs. Chaque zone doit donc en avoir un nombre minimum. Ceci se fait en maximisant la taille des blocs tout en sachant qu'il existe une limite pour cette taille dictée par la technologie utilisée. Cette taille, le nombre total des blocs associés aux différentes zones et la mémoire restante (ou perdue) sont calculés de la façon suivante:

Dans cet algorithme, nous supposons que la taille du descripteur, nécessaire pour la gestion d'un bloc sur la carte, soit incluse dans la taille totale du bloc. Une fois le nombre total de blocs déterminé et ceux-ci alloués, ils sont ensuite attribués aux différentes zones. Toutes les zones n'ont pas nécessairement le même nombre de blocs.

```
Nombre_de_Blocs = Nombre_de_Zones
Taille_Bloc = 0
Tant que Taille_Bloc = 0
faire
 Taille = Taille_Disponible / Nombre_de_Blocs
 si Taille > TAILLE_BLOC_MAXIMUM
 alors
   Nombre_de_Blocs = Nombre_de_Blocs + 1
 sinon
   Taille_Bloc = Taille
   Taille_Disponible = Taille_Disponible -(Taille_Bloc * Nombre_de_Blocs)
 finsi
fait
```

Cette politique a l'avantage d'être facile à implanter et à gérer et peut être réalisée avec tout type de mémoire. Néanmoins, la mémoire est mal distribuée entre les zones. En effet, les différentes zones n'ont pas les mêmes besoins en espace. De ce fait, lorsque la carte se sature, il peut encore exister de l'espace non utilisé, alloué à

des zones peu actives. Il y a donc perte d'espace mémoire. Cette perte peut même être dénombrée en nombre de blocs. Plusieurs blocs entièrement vides peuvent être perdus.

Allocation Statique Taille Variable (ASTV): Toute la mémoire est divisée en blocs de taille variable. Tous les blocs ont des tailles différentes et celles-ci sont fonction de la zone à laquelle ils vont être associés. Chaque zone est caractérisée par une fréquence de transactions. La taille d'un bloc attribué à une zone est donc proportionnelle à la fréquence des transactions dans la zone et à l'espace moyen nécessaire par transaction. La division se fait en une seule étape avant l'exploitation, généralement au cours de la personnalisation ou l'initialisation de la carte. La portion de la mémoire qui n'est pas allouée est perdue et ne peut être allouée ultérieurement au cours de l'exploitation de la carte.

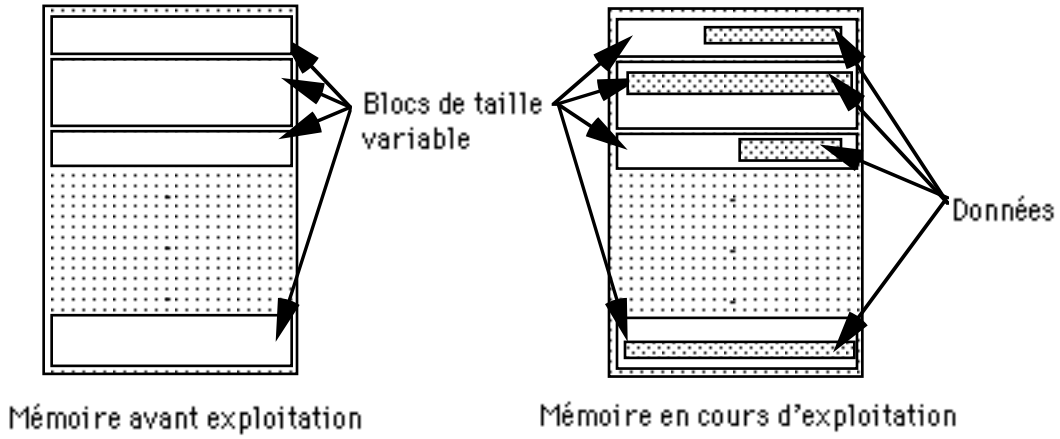


fig 1.16: Allocation statique taille variable

Tous les blocs attribués à une même zone ont la même taille. La taille des blocs de chaque zone est fixée au début avant de répartir la mémoire. Le nombre de blocs alloués à chaque zone est calculé dans le tableau Nombre_de_Blocs[1..Nombre_de_Zones] de la façon suivante:

```
initialiser Taille_Bloc[1..Nombre_de_Zones]
Nombre_de_Blocs[1.Nombre_de_Zones] = 0
Zones_Complètes = 0
Tant que Zones_Complètes < Nombre_de_Zones
faire
 Pour Zone = 1 à Nombre_de_Zones
 faire
  Si Taille_Disponible >= Taille_Bloc[Zone]
  alors
   Taille_Disponible = Taille_Disponible - Taille_Bloc[Zone]
   Nombre_de_Blocs[Zone] = Nombre_de_Blocs[Zone] + 1
  sinon
   Zones_Complètes = Zones_Complètes + 1;
  finsi
 fait
fait
```

Cette politique rationalise mieux l'espace mémoire que la politique précédente.

Elle a aussi l'avantage d'être facilement réalisable sur les mémoires de type EPROM ou EEPROM. Étant donné que la taille des blocs varie d'une zone à l'autre en considérant la fréquence des transactions en écriture de la zone, les pertes d'espace non utilisé sont inférieures à celles faites avec ASTF. Cependant, les pertes sont encore importantes et lorsque la carte se sature, il peut encore exister de l'espace non utilisé se dénombrant parfois aussi en nombre de blocs. Des zones peu actives peuvent encore avoir plusieurs blocs entièrement vides.

Allocation Dynamique Taille Fixe (ADTF): Cette politique ressemble à ASTF en ce sens que les blocs ont tous une taille identique et fixe. Cependant, la mémoire n'est pas allouée dans sa totalité lors d'une première phase. Les blocs de mémoire sont seulement alloués au besoin. À chaque fois qu'une zone manque d'espace mémoire, un bloc lui est alloué à partir de la mémoire disponible. Initialement, toutes les zones sont constituées chacune d'un bloc unique. Le reste de la mémoire demeure disponible. C'est seulement lorsqu'une zone est pleine que l'on alloue un nouveau bloc. Avec cette politique, un deuxième scénario est possible à la phase initiale. Au lieu d'attribuer un bloc à chaque zone, on peut ne créer aucune zone et allouer le premier bloc d'une zone lorsque celle-ci est utilisée la première fois. Ensuite, l'allocation des blocs suivants se fait dynamiquement au besoin.

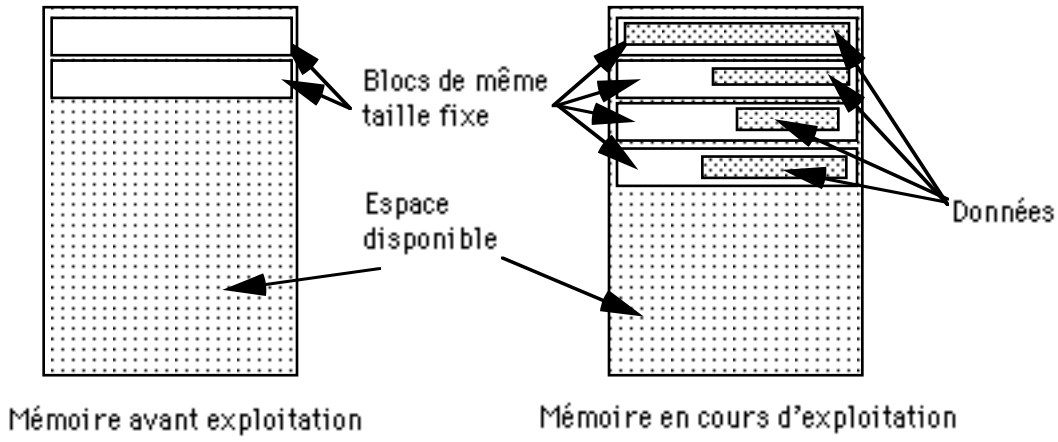


fig 1.17: Allocation dynamique taille fixe

Étant donné qu'avec cette politique les blocs sont alloués dynamiquement au besoin, il n'y a pas de risque de perdre, lors de la saturation de la carte, des blocs entièrement vides comme avec les deux précédentes politiques d'allocation. Les risques de perte de mémoire persistent cependant, parce que la taille des blocs ne tient pas compte de l'activité de la zone à laquelle ils sont attribués. Cette politique ne peut être implantée qu'avec les technologies qui permettent une allocation dynamique au cours de l'exploitation de la carte. Certaines technologies ne permettent d'allouer de l'espace qu'avant la phase de personnalisation de la carte.

Allocation Dynamique Taille Variable (ADTV): Cette politique est identique à ADTF à la différence près que les blocs ont des tailles différentes. La taille d'un bloc dépend de la zone à laquelle il est attribué. Tous les blocs d'une même zone ont la même taille. Mais d'une zone à l'autre, la taille varie en fonction de la fréquence des

transactions et de leur masse informationnelle.

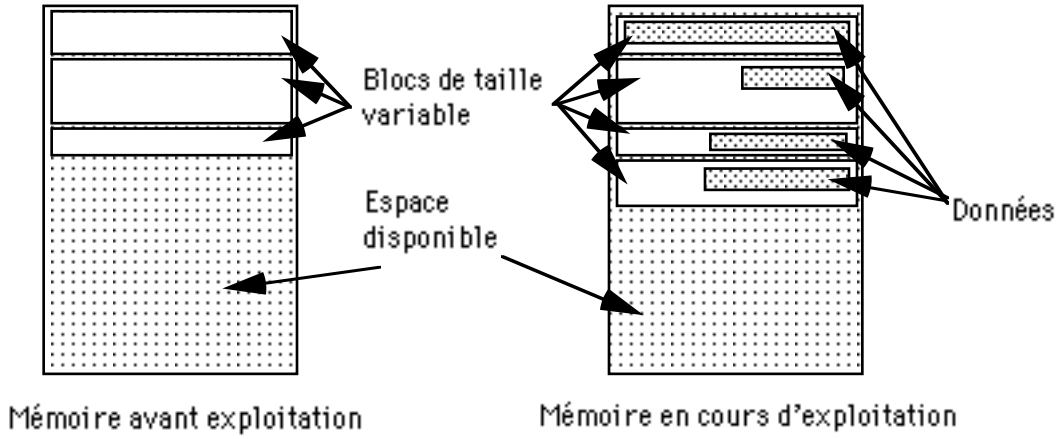


fig 1.18: Allocation dynamique taille variable

Cette politique enregistre moins de perte de mémoire non utilisée lors de la saturation de la carte que les politiques précédentes. Elle ne peut être implantée qu'avec les technologies qui permettent d'allouer dynamiquement de l'espace mémoire après la personnalisation de la carte.

Allocation Dynamique Taille Dynamique (ADTD): Cette politique ressemble à la précédente, sauf que la taille des blocs peut varier au sein d'une même zone. Initialement, le premier bloc de chaque zone a une taille donnée et bien précise. Lors des allocations suivantes, la taille du bloc est évaluée et calculée à partir de la fréquence de remplissage de la zone. Une zone dont les blocs se remplissent souvent, a tendance à voir ses nouveaux blocs augmenter en taille. Les zones qui se remplissent lentement ont par contre tendance à avoir une diminution dans la taille de leurs nouveaux blocs. La décision d'augmenter ou de diminuer la taille d'un nouveau bloc d'une zone dépend du nombre de blocs attribués à cette zone par rapport à la moyenne du nombre de blocs par zone. La décision est prise de la façon suivante:

```
/* soit Zone la zone concernée par l'allocation de bloc */
moyenne = Nombre_Total_de_Blocs / Nombre_de_Zones
si      (Nombre_de_Blocs[Zone] > moyenne) alors AUGMENTATION
sinon si (Nombre_de_Blocs[Zone] = moyenne) alors PAS_DE _CHANGEMENT
sinon si (Nombre_de_Blocs[Zone] < moyenne) alors DIMINUTION
```

Le calcul de la taille du nouveau bloc pour une augmentation ou une diminution se fait par une même opération:

Nouvelle_Taille = Ancienne_Taille * (Nombre_de_Blocs[Zone]/moyenne)

En effet, (Nombre_de_Blocs[Zone]/moyenne) est supérieur à 1 en cas d'augmentation, inférieur à 1 en cas de diminution, et égale à 1 dans le cas d'une décision de non changement.

Malgré sa légère complexité par rapport à ADTV, cette politique permet d'avoir

moins de perte d'espace mémoire. Elle peut être réalisée comme les politiques précédentes sur une mémoire EPROM ou EEPROM. L'ADTD ne peut être implantée qu'avec les technologies qui permettent d'allouer dynamiquement de l'espace mémoire après la personnalisation de la carte.

Allocation dynamique avec gonflage des zones (ADGZ): Avec les politiques d'allocation vues précédemment, un bloc est alloué pour accommoder plusieurs transactions. Le bloc est suffisamment grand et son espace est utilisé au fur et à mesure, jusqu'à épuisement. Avec la politique de gonflage, chaque zone a initialement un bloc de taille minimale. À l'ajout de nouvelles informations, le bloc de la zone augmente d'une taille égale à l'espace requis pour les nouvelles données. Un nouveau bloc est attribué à la zone lorsque le premier bloc atteint la taille maximale limitée par la technologie utilisée. De ce fait, l'espace mémoire utilisé sur la carte est toujours exactement égal à l'espace requis par l'information inscrite dans celle-ci et l'espace est équitablement distribué au fur et à mesure et aux besoins des zones. Lorsque la carte est saturée, il n'y a aucune perte d'espace non utilisé et on est sûr que toutes les zones sont aussi saturées.

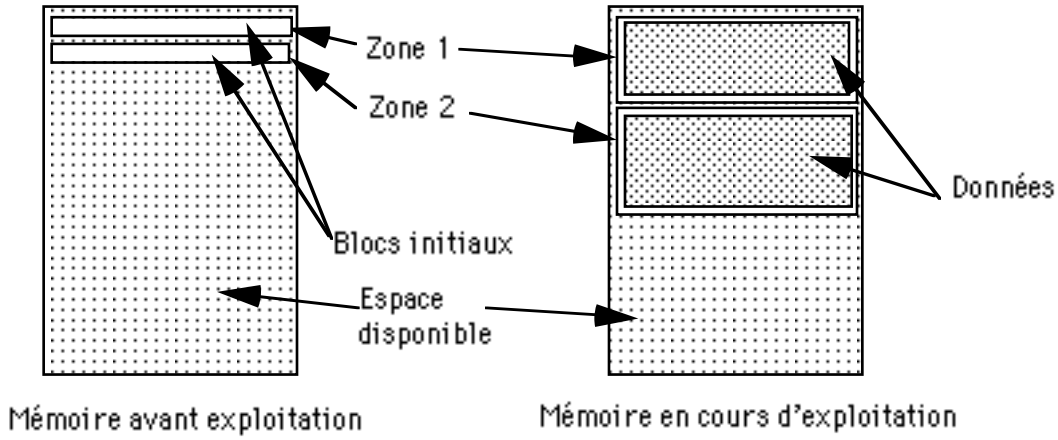


fig 1.19: Allocation dynamique avec gonflage de zones

Cette politique ne peut se réaliser qu'avec une mémoire EEPROM permettant l'effacement et avec une technologie permettant l'allocation d'espace après la personnalisation. La possibilité de désallocation d'espace est aussi requise si la technologie ne permet pas le gonflage de bloc physique.

1.6.2 Politiques de récupération

Lorsqu'une zone est saturée et que l'on ne peut pas y inscrire de nouvelles données, l'algorithme d'allocation de mémoire est mis en route pour résoudre cette saturation. Si une allocation supplémentaire est impossible, la carte est alors dite saturée et la nouvelle information ne peut pas y être insérée. Cependant, dépendamment de la politique d'allocation, d'autres zones dans la carte peuvent encore avoir de l'espace libre récupérable. Des techniques de récupération de cet espace peuvent être réalisées. Ce sont les récupérations automatiques. Les récupérations automatiques peuvent aussi, suivant l'application, récupérer de

l'espace utilisé par des données obsolètes en se basant sur des dates ou la technique LRU (Least Recently Used). D'autres techniques, dites manuelles, requièrent l'intervention d'un utilisateur humain décideur. Celui-ci a ou peut avoir l'autorisation d'effacer des informations qu'il juge non pertinentes dans une ou plusieurs zones de la carte intelligente et conserver les informations jugées essentielles pour les activités décisionnelles en cours ou prévisibles. De toute évidence, l'effacement est une partie de la résolution des problèmes de saturation et est en fait distincte de la récupération. La récupération d'espace non utilisé suit séquentiellement dans le temps la phase d'effacement de données si celle-ci est autorisée et possible. Les politiques de récupération que nous énumérons ci-dessous sont en fait des techniques de résolution de la saturation et englobe les techniques d'effacement et les techniques de récupération proprement dites.

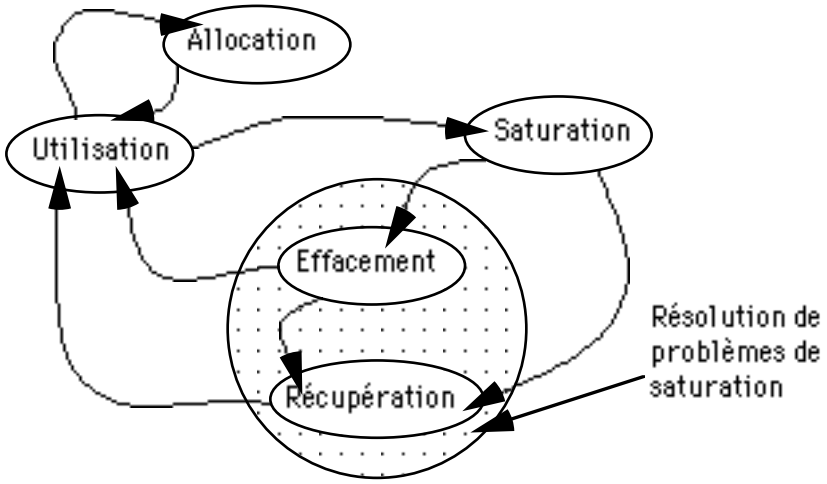


fig 1.20: Cycle chronologique des phases de gestion de la mémoire

Les politiques de récupération d'espace que nous présentons sont parfois liées et dépendantes de la politique d'allocation utilisée. Elles peuvent aussi perturber l'homogénéité des tailles des blocs d'une même zone dictée par certaines politiques d'allocation.

Politique de l'autruche (PE): Cette politique consiste à ne rien faire lors de la saturation de la mémoire de la carte et à confirmer la fin de la vie de la carte. Elle se traduit par l'absence totale de techniques de résolution de la saturation de la carte. Il n'existe alors aucune restriction sur le type de la mémoire, ni sur la politique d'allocation utilisée.

Effacement dans une même zone (EMZ): C'est une politique d'effacement qui limite l'élimination, manuelle ou automatique, des données périmées dans une seule zone. L'unique zone touchée par l'effacement est celle qui a été saturée. L'effacement s'effectue avec un pourcentage maximum qui limite la suppression des données jugées périmées. En se basant sur l'heuristique voulant que la péremption des données dans une zone diminue avec le nombre d'effacements effectués dans la zone, on peut décrémenter le pourcentage d'effacement permis, d'un pas fixe après chaque phase d'effacement de données dans la zone. EMZ nécessite une mémoire de

type EEPROM pour être implantée.

Effacement dans plusieurs zones (EPZ): Cette politique permet un effacement dans plusieurs zones lors de la saturation d'une zone donnée. L'ensemble des zones touchées par l'effacement est tributaire du droit d'accès à ces différentes zones et de l'acteur responsable de la saturation. Pour définir cet ensemble, on utilise aussi une matrice où l'on retrouve l'ensemble des zones pouvant être épurées lors d'une saturation d'une zone donnée.

Zone saturée \ Zones d'effacement	Z1	Z2	Z3	Z4	Z5	...
Z1	0ou1	0ou1	0ou1	0ou1	0ou1	...
Z2	0ou1	0ou1	0ou1	0ou1	0ou1	...
Z3	0ou1	0ou1	0ou1	0ou1	0ou1	...
Z4	0ou1	0ou1	0ou1	0ou1	0ou1	...
...			...			

où
0 = effacement interdit
1 = effacement autorisé

Comme pour l'EMZ, cette politique limite l'effacement dans chaque zone par un pourcentage propre à cette même zone. Tous les pourcentages sont définis dans un vecteur et dépendent de la pertinence des données de chaque zone. Ils peuvent aussi être décrémentés par un pas de décrémentation après chaque effacement.

Cette politique permet d'effacer plus de données périmées et donc, de récupérer davantage d'espace. Mais elle nécessite plus d'efforts d'implantation, de gestion et de contrôle et ne peut être implantée avec une mémoire de type EPROM.

Désallocation d'espace libre (DEL): C'est une technique de récupération proprement dite. Elle consiste à rechercher un espace non utilisé dans une zone, de le désallouer et de le réallouer à la zone saturée. La recherche d'espace libre est faite systématiquement dans toutes les zones de la carte. La quantité d'espace libre désalloué et récupéré dans chaque zone est dictée par des pourcentages de récupération qui dépendent de l'activité de chaque zone. L'espace récupéré et alloué à la zone saturée a une taille sensiblement égale à la taille de l'espace manquant à la transaction responsable de la saturation. L'algorithme de recherche et de récupération d'espace est le suivant:

```
initialiser Taille_Bloc[1..Nombre_de_Zones]
Nombre_de_Blocs[1..Nombre_de_Zones] = 0
Zones_Complètes = 0
Tant que Zones_Complètes < Nombre_de_Zones
faire
 Pour Zone = 1 à Nombre_de_Zones
 faire
  Si Taille_Disponible >= Taille_Bloc[Zone]
  alors
   Taille_Disponible = Taille_Disponible - Taille_Bloc[Zone]
   Nombre_de_Blocs[Zone] = Nombre_de_Blocs[Zone] + 1
  sinon
   Zones_Complètes = Zones_Complètes + 1;
  finsi
 fait
fait
```

Cette politique nécessite la possibilité de désallocation d'espace et la possibilité d'allocation dynamique après la phase de personnalisation. Seules les mémoires de type EEPROM peuvent supporter DEL.

La désallocation d'espace libre est beaucoup plus avantageuse lorsque le masque de la carte dispose d'un «ramasse-miettes». Dans le cas contraire, cette politique exige une gestion des morceaux d'espace récupérés, en faisant un décalage des zones, par exemple. Cette gestion peut alourdir considérablement l'algorithme.

Dégonflage de zone (DGZ): C'est une politique de récupération qui va de pair avec la politique d'allocation dynamique par gonflage de zone. Elle consiste à réduire automatiquement la taille de la zone (ou «dégonfler» la zone) après avoir effectué l'effacement dans celle-ci. Une zone dans laquelle une certaine quantité d'information est effacée voit sa taille diminuer automatiquement et proportionnellement à la taille de l'information éliminée. L'espace récupéré est rajouté à la mémoire disponible de la carte. Elle est alors réutilisée au prochain appel à l'algorithme d'allocation.

Cette politique, qui permet une gestion optimale de l'espace mémoire, nécessite une mémoire EEPROM et les possibilités d'effacement, de désallocation et d'allocation dynamique.

Le schéma suivant montre les compatibilités et les dépendances entre les politiques d'allocation et les politiques de résolution de la saturation.

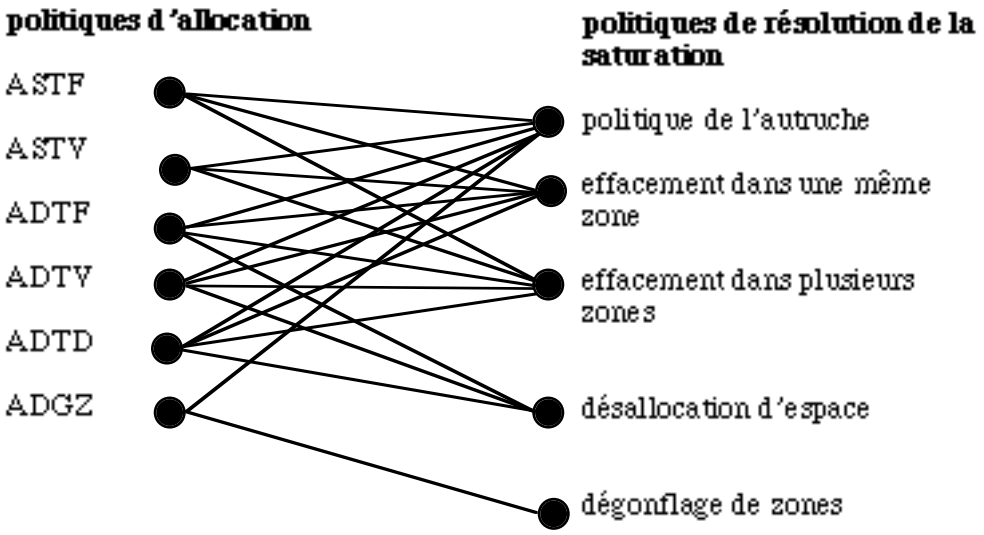


fig 1.21: Compatibilité des politiques

1.7 Mécanismes de sécurité

Ce qui a fait le succès de la carte intelligente, c'est l'aspect sécurité qu'elle offre. En effet, plusieurs fonctionnalités de la carte sont étroitement liées à la sécurité. On peut structurer et hiérarchiser l'information dans une carte en subdivisant la mémoire de la carte en zones. On peut attribuer une clé à chaque niveau de la hiérarchie. Ainsi, toute donnée peut être protégée en lecture, en écriture et même, en réécriture. Une clé principale appelée NIP (Numéro d'Identification Personnelle) verrouille l'accès à la carte comme le cadenas d'un coffre. En plus, la vérification de toute soumission de clé se fait à l'interne par le microprocesseur de la carte, contrairement aux autres cartes à mémoire qui laissent le soin au lecteur ou à l'application de faire la comparaison des clés. Ce mécanisme de sécurité, qui contrôle les accès aux données et qui permet la gestion et la combinaison des codes d'accès, est offert par la majorité des technologies de cartes intelligentes. Cependant, il diffère légèrement d'une technologie à l'autre en fonction de la structuration permise des données.

Nous présentons dans ce qui suit d'autres aspects de la sécurité offerte par les cartes intelligentes.

1.7.1 Identification

Elle consiste, pour un porteur, à délivrer un identifiant, c'est-à-dire une information en sa possession qui lui est propre, lui permettant ainsi de se faire reconnaître. Cet identifiant est un code unique, infalsifiable et, de préférence, secret. Le numéro d'identification personnelle (NIP) est le cas classique d'identifiant. Il permet au porteur de s'identifier à la carte. La carte contrôle elle-même l'identité du porteur en vérifiant le code confidentiel qui lui est présenté. Ce code peut être en clair ou chiffré. La carte est capable de détecter et d'éliminer les tentatives d'accès il-

licites. En effet, après un nombre pré-établi d'essais infructueux, la carte se verrouille. Un identifiant d'une application, mémorisé dans la carte, est un autre cas classique d'identification où les terminaux, en consultant cet identifiant, reconnaissent la carte et l'application pour laquelle elle est dédiée. Pour éviter les simulations d'identifiant, on utilise un processus d'authentification.

1.7.2 Authentification

Ce processus complète souvent l'identification. C'est un moyen par lequel une entité prouve qu'elle est bien l'entité qu'elle affirme être. L'authentification peut se faire entre un porteur et une carte, entre un terminal et une carte ou entre deux cartes. L'idée repose sur le partage d'une information secrète (authentifiant) connue uniquement des deux entités. L'authentification se fait généralement automatiquement par un mécanisme interne à la carte. Le mécanisme de l'authentification est expliqué plus bas.

1.7.3 Chiffrement

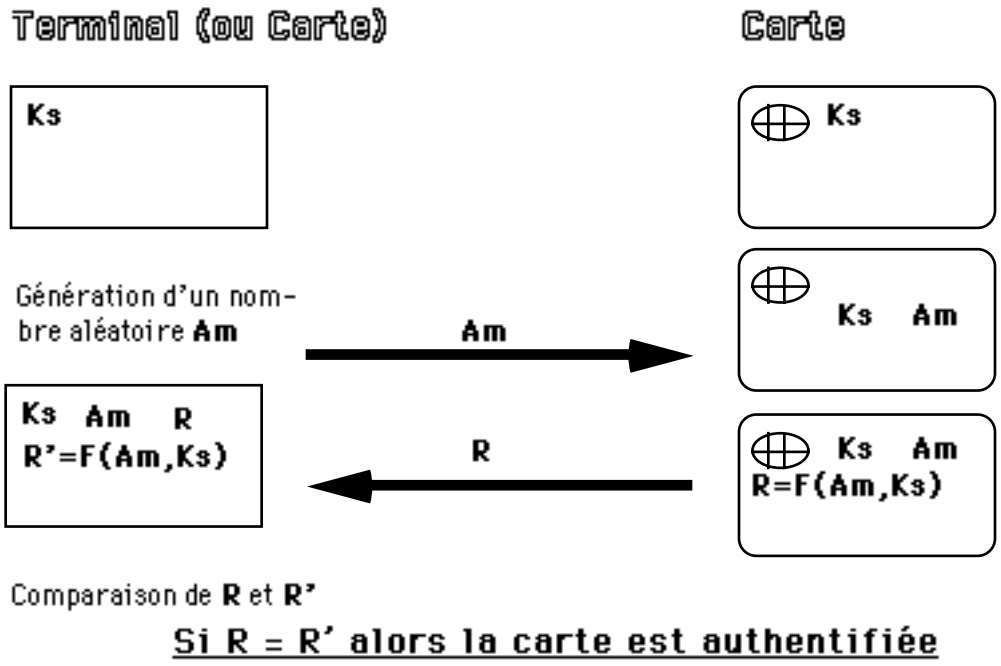
Cette fonction permet de chiffrer un texte initialement en clair, à l'aide d'un algorithme cryptographique. Le déchiffrement est l'opération inverse. Le chiffrement (ou la cryptographie) permet de transformer à l'aide d'une fonction **F** le contenu d'un message **M** au moyen d'une clé **K** pour donner un nouveau message **R=F(M,K)** appelé cryptogramme. Les fonctions de chiffrement sont des fonctions à sens unique. C'est-à-dire que, connaissant la fonction et le cryptogramme, on ne peut pas, ou du moins difficilement, au moyen de la fonction inverse, trouver le message en clair. Si $y = f(x)$, il est très difficile de retrouver la fonction inverse f^{-1}, tel que $x = f^{-1}(y)$. La difficulté vient des moyens actuels dont on dispose, comme le temps et les ressources de calcul. Pour pouvoir retrouver x à partir de y, les fonctions de déchiffrement doivent avoir une autre information, appelée information compromettante. C'est la clé de chiffrement et de déchiffrement. C'est la raison pour laquelle la clé doit être connue par les deux parties: l'émetteur et le récepteur. Si **R=F(M,K)**, alors on peut retrouver **M** avec $\mathbf{M=F^{-1}(R,K)}$. La sécurité ne repose pas sur la connaissance de l'algorithme de chiffrement et de déchiffrement mais sur celle de la clé utilisée.

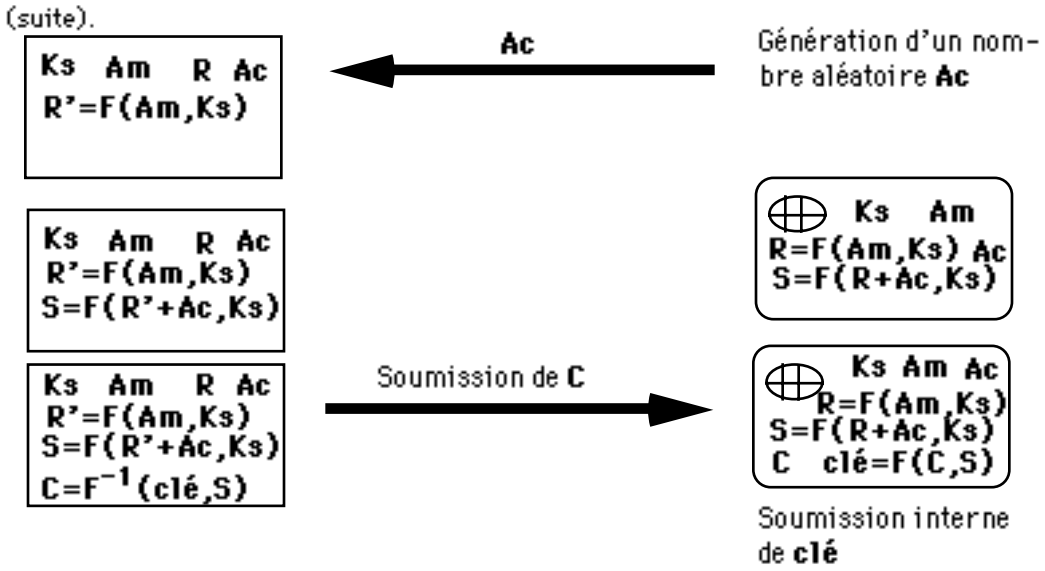
Il existe plusieurs systèmes de chiffrement utilisés par les cartes intelligentes dont les plus connus sont DES (Data Encryption Standards) publié par le National Bureau of Standards en 1977, RSA (Rivest Shamir Adleman) publié en 1978, et TELEPASS, un algorithme secret utilisé par la compagnie Bull.

1.7.4 Mécanisme d'authentification

L'authentification est mise en oeuvre grâce au chiffrement. Cette reconnaissance bilatérale sécurisée repose sur le partage d'un secret, le code secret. Pour avoir un mécanisme d'authentification sûr, il est important d'avoir une authentification non duplicable, c'est-à-dire qu'il est impossible de rééditer un processus d'authentification. On crée alors une clé pour chaque session. Cette clé, appelée clé

de session, est basée sur la clé secrète et un nombre aléatoire qui varie d'une session à l'autre. Tous les messages dans une session sont cryptés à partir de la clé de session. Le mécanisme d'authentification entre un terminal (ou une carte) et une autre carte est schématisé plus bas. Initialement, les deux parties partagent un secret; la clé **Ks**. Au départ, l'entité qui veut communiquer avec la carte génère un nombre aléatoire **Am** et le transmet à la carte. Celle-ci l'encrypte avec la clé **Ks** pour obtenir le cryptogramme **R=F(Am,Ks)** et le retransmet au terminal requérant. Le terminal (ou la carte) de l'autre côté fait de même en calculant le cryptogramme **R'=F(Am,Ks)** puis le compare au message reçu **R**. Si **R**=**R'**, alors la carte est authentifiée, sinon la session est interrompue. Dans le cas positif, l'authentification se poursuit par une soumission de clé chiffrée. Pour ce faire, la carte génère un nombre aléatoire **Ac** et le transmet au terminal (ou à l'autre carte) voulant établir la communication. Cette autre partie calcule une clé de session **S=F(R+Ac,Ks)** puis soumet une clé chiffrée **$C=F^{-1}$(clé,S)**. La carte qui a calculé de la même façon la clé de session **S** décrypte la clé chiffrée soumise **clé=F(C,S)** et effectue sa vérification. Si la clé est bonne, alors le terminal ou la carte est authentifiée.





1.7.5 Reconnaissance biométrique

La reconnaissance biométrique est la reconnaissance d'empreintes digitales, de rétines, de voix, de signatures dynamiques ou de toutes autres caractéristiques physiques. La carte intelligente peut être associée à un système de reconnaissance biométrique externe. Les paramètres biométriques de référence sont alors emmagasinés dans la carte et transmis codés au système de reconnaissance qui effectue lui même la vérification. Ce processus ne peut se faire qu'après une authentification entre la carte et le système de reconnaissance. Ce système d'identification enrichit les méthodes de contrôle d'accès et les possibilités d'authentification déjà offertes par la carte intelligente.

1.7.6 Sécurité dans un environnement réparti

En plus du chiffrement et du déchiffrement que l'on a vu plus haut, deux autres problèmes se posent dans un environnement réparti: l'écriture dans une carte à distance et la certification des données.

La télé-écriture: appelée aussi écriture sécurisée, la télé-écriture est un processus qui permet à la carte de recevoir à distance un ordre d'écriture sans révéler les codes d'accès. L'ordre d'écriture est reçu chiffré avec des données chiffrées. La carte est capable d'interpréter le message et de déterminer si le message provient effectivement d'un émetteur habilité en mettant en oeuvre un algorithme cryptographique réversible. Dans le cas positif, l'écriture est automatique. Un système espion recevant le message ne peut d'aucune façon interpréter son contenu, ni connaître la clé de la carte ou reconstruire un message susceptible d'ouvrir frauduleusement des sessions similaires avec la même carte ou une autre.
La certification: C'est une signature électronique qui permet de s'assurer qu'une télé-écriture a bien été faite à une adresse précise et qu'elle n'a pas été simulée. Avec

un mécanisme qui rappelle celui de l'authentification, la carte calcule un certificat qui prouve la présence de la donnée à l'adresse indiquée.

Autres sécurités: Il existe aussi un autre niveau de sécurité dit physique, par opposition à la sécurité logique vue plus haut. En effet, certains constructeurs offrent des sécurités permettant au microprocesseur de s'auto-verrouiller ou même, d'annuler définitivement l'utilisation de la carte en cas de violations graves. Ainsi, il existe des bits témoins (ou mines) parsemés dans la mémoire. Ces bits ne sont jamais accessibles normalement par une application et ne font pas partie logiquement des zones de données, bien qu'ils soient physiquement mêlés aux autres bits de la mémoire. Le microprocesseur vérifie régulièrement ces bits témoins et se verrouille dans le cas où ceux-ci ont été modifiés. De même, il existe aussi des processus sophistiqués qui permettent au microprocesseur de reconnaître une tentative de lecture après l'ouverture de la puce. Dans le cas d'une telle «effraction» le microprocesseur se verrouille définitivement.

Certaines technologies offrent des fusibles appelés bits de validation, lesquels sont associés à des données sur la carte et qui, lorsqu'ils sont grillés, empêchent toute nouvelle écriture sur ces données.

1.8 Les technologies du futur

1.8.1 Carte magnétique de grande capacité

Contrairement aux cartes magnétiques classiques qui n'ont qu'une étroite bande magnétique, les cartes magnétiques de grande capacité sont entièrement recouvertes d'une couche magnétique à haut champ coersitif. Cette couche magnétique en ferrite de baryum peut atteindre plus de 5000 Oersteds. La capacité d'une telle carte peut atteindre quatre mégabits. Elle est lue longitudinalement comme toute bande magnétique. Il existe, néanmoins, une technique d'écriture et de lecture originale traitant la carte comme une disquette. En effet, la longueur de la diagonale d'une carte ISO coïncide avec les dimensions d'une disquette trois pouces et demi. Ceci permet d'adapter un lecteur de disquette standard pour formater le support magnétique de la carte, lire et écrire l'information comme avec toute autre disquette standard.

1.8.2 Carte optique à laser

Dans son principe général, elle est similaire à la carte magnétique à grande capacité. Elle emprunte la technologie des disques compacts (CD ROM) en remplaçant la couche magnétique par un support optique qui est un film à haute résolution (crevasses de 25 microns) inscriptible par photolithographie à l'aide d'un rayon laser. L'avantage est que de telles cartes disposent d'une énorme capacité de mémorisation pouvant atteindre deux méga-octets. Pour l'instant, ces supports ne sont pas encore effaçables comme une piste magnétique et n'offrent pas de sécurité, puisque la lecture et la duplication se font simplement par projection de faisceaux laser sur la carte. On peut s'attendre à voir apparaître, dans un proche avenir, des

interfaces (via le lecteur) entre un tel support et un microprocesseur qui chiffrerait automatiquement l'information sur la carte.

1.8.3 La «super-carte intelligente»

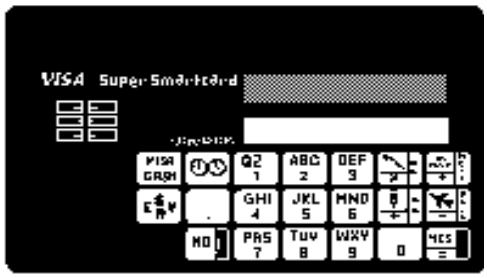


fig 1.22: La «Super Smart Card»

Connue sous le nom de *super_smart_card*, cette carte a le même format qu'une carte standard mais avec une épaisseur légèrement supérieure. Elle a la particularité de présenter des périphériques sur son support comme un écran d'affichage à cristaux liquides de 16 caractères, un clavier de 20 touches alphanumériques et une piste magnétique, en plus du microprocesseur. Cette carte contient des batteries pour l'alimenter et permettre une utilisation sans terminal d'exploitation. Les codes secrets peuvent être directement introduits par le clavier de la carte et certaines informations peuvent être consultées sur l'écran. Cette carte est présentée par Toshiba et Casio. Elle n'est présentement pas encore monolithique mais «bi-chip».

1.8.4 Les cartes intelligentes sans contact

Le principal défaut de la carte à microprocesseur concerne les problèmes engendrés par frottement lors d'insertion de la carte dans les systèmes de lecture, la corrosion aux contacts et certains accidents liés au relief des contacts, comme la torsion ou la déformation de ces derniers qui peuvent causer un mal fonctionnement de la carte ou un rejet de celle-ci. Sans compter les problèmes liés à l'encartage. De plus, la sensibilité du microcircuit aux surcharges électriques accidentelles des lecteurs de cartes entraîne des rejets parfois importants de la carte à microprocesseur.

Ces reproches adressés aux cartes à puce conduisent les industriels à rechercher des solutions de remplacement. Les cartes intelligentes sans contacts ont alors commencé à voir le jour. L'alimentation électrique et la transmission des données avec ces cartes se basent sur l'induction électrique et l'utilisation des micro-ondes. Ces cartes n'ont pas encore atteint un stade de développement important aujourd'hui et doivent encore voir leur taille diminuer pour être intégrées dans une carte au format ISO. Le développement des cartes intelligentes sans contacts aura des retombées dans plusieurs domaines, comme le suivi de production, le payage routier, le transport en commun et même le domaine militaire.

1.8.5 Une nouvelle carte intelligente?

En dix ans, les micro-ordinateurs ont évolué comme on ne pouvait jamais l'imaginer au début des années 80. Dans un avenir pas si lointain, la carte va doubler ou tripler de mémoire et d'intelligence. On peut prévoir que la carte aura l'intelligence et les capacités d'un micro-ordinateur d'aujourd'hui et même plus.

Sur le même support plastique, on peut facilement mettre deux et même quatre puces distinctes et autonomes. De cette façon, on obtient plusieurs cartes à puce sur le même support. Il suffit de retourner la carte pour profiter du second microprocesseur. La communication sécurisée entre les deux puces est une autre paire de manches. De la même façon, on peut concevoir une carte optique à laser ou une carte magnétique à grande capacité avec un microprocesseur incorporé. Une interface de communication sécuritaire entre les deux techniques de stockage et permettant au microprocesseur de gérer la mémoire de masse reste à réaliser.

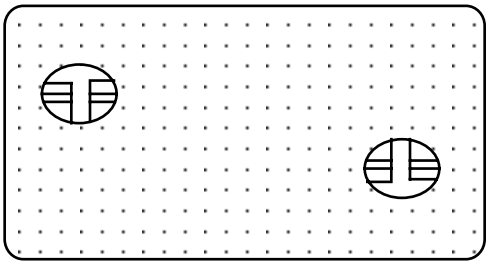


fig 1.23: Carte intelligente bi-microprocesseur

Les cartes intelligentes évoluent si rapidement qu'on peut imaginer des lendemains dignes d'un vrai film de science-fiction. En effet, on peut très bien imaginer une puce sans contact incrustée dans le lobe de l'oreille d'un bébé et qui l'accompagnerait durant toute sa vie pour contenir son dossier médical complet. La capacité de cette puce serait suffisamment importante pour contenir les digitalisations de radiographies. On peut aussi imaginer une carte à puce moins fantastique mais plus facilement réalisable (et acceptable) dans un avenir plus proche. Cette carte engloberait dans sa mémoire des données partagées par plusieurs applications. Cette carte multi-applications pourrait remplacer toutes les cartes bancaires et de crédit dans notre portefeuille et, pourquoi pas, même notre portefeuille.

Les interfaces SQL pour cartes intelligentes, qui sont en cours de développement en France et ailleurs, ainsi que la «super smart card» peuvent nous laisser entrevoir la création d'une carte avec des interfaces usagers de communication très conviviales et sophistiquées comme une reconnaissance de la parole et des requêtes avancées. Un émetteur-récepteur à infra-rouge ou micro-ondes pour émettre et recevoir des informations à distance d'une façon autonome et, pourquoi pas, télécommander à distance certains appareils pourrait aussi être un jour ajouté à notre carte intelligente.

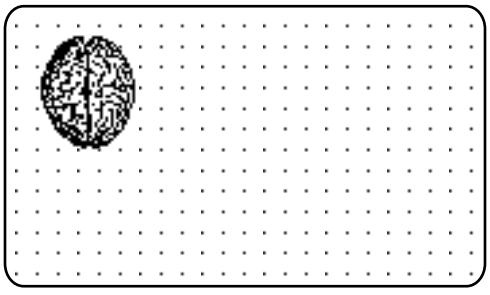


fig 1.24: Une carte à mémoire intelligente?

CHAPITRE II

Le Dossier Médical Portable

Plusieurs expériences menées dans divers pays, principalement en Europe, démontrent l'intérêt de la carte intelligente dans un système médical informatisé. La carte intelligente devient un outil très intéressant et fiable pour la collecte et la diffusion rapide de l'information médicale ainsi que pour son partage sécurisé. Dans un tel système, la carte devient un support pour «porter» l'information médicale d'un intervenant à l'autre. C'est dans ce cadre que s'inscrit le projet-pilote Carte Santé.

Le Dossier Médical Portable (DMP) est un extrait du dossier médical d'un patient, hospitalier ou de clinique, emmagasiné dans une carte intelligente. Il se veut un aide-mémoire pour les médecins ou les professionnels de la santé et ne remplace à aucun moment le vrai dossier hospitalier. Son contenu reflète les besoins des intervenants de la santé qui l'ont eux-mêmes conçu.

Le DMP est implanté sur une carte à microprocesseur sécuritaire pour garantir une confidentialité importante des données. Il est uniquement utilisé à des fins cliniques pour améliorer la qualité des soins en se basant sur le partage d'informations fiables et n'est en aucun cas un outil de centralisation des faits de santé concernant les citoyens.

Le DMP se veut un élément actif dans un réseau virtuel d'aide à la décision thérapeutique pour les intervenants dans le domaine de la santé. Les connections, dans ce réseau, sont hautement sécurisées au moyen de la technologie de la carte à puce. Ce réseau améliore la qualité des services médicaux et pharmaceutiques en permettant le partage fiable et rapide de faits médicaux pertinents et intègres [GAMACHE 90].

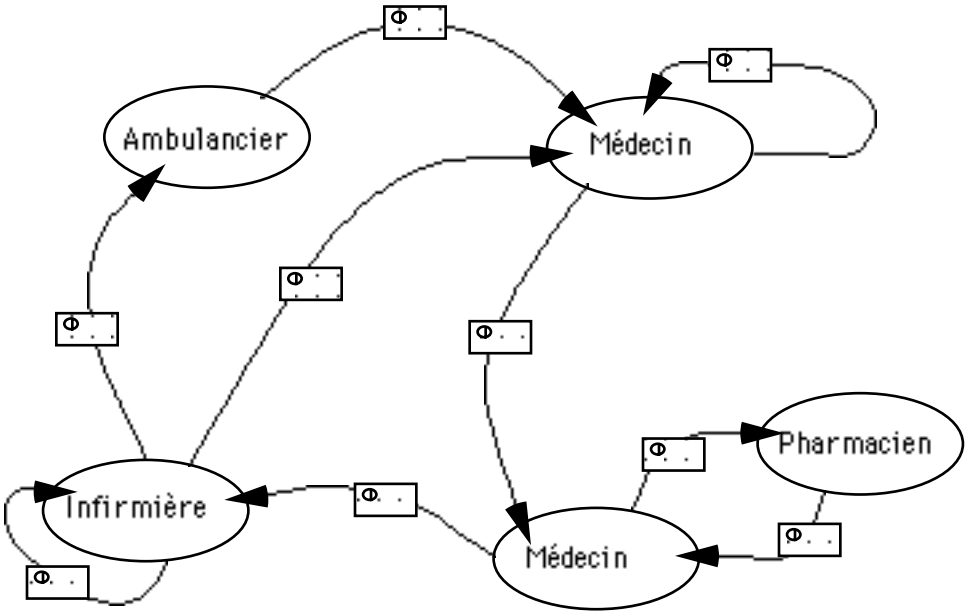


fig 2.1: Réseau virtuel de professionnels de santé connecté par la Carte Santé

Le DMP que nous allons présenter est complexe et comporte plusieurs regroupements d'informations très variées. Avant d'énumérer son contenu, nous allons présenter dans ce qui suit les diverses cartes utilisées dans le système, les intervenants y agissant et utilisant les cartes et les catégories de personnes visées par ce projet.

2.1 Les types de cartes

Le système Carte Santé gère deux familles de cartes à mémoire: les cartes d'habilitation et les cartes des bénéficiaires appelées aussi cartes santé. Ces deux familles sont essentiellement distinguées par leurs fonctions respectives et le type de données emmagasinées dans chacune d'elle. Les cartes des bénéficiaires contiennent les informations médicales, donc le DMP des bénéficiaires des services de soins. Les cartes d'habilitation servent aux prestataires de soins à s'identifier et à être habilités à accéder aux informations du DMP qui leur sont nécessaires pour mener à bien les soins qu'ils donnent aux usagers. C'est en quelque sorte une clef pour les titulaires de carte d'habilitation leur permettant l'accès aux informations conformément à leur profil d'accès.

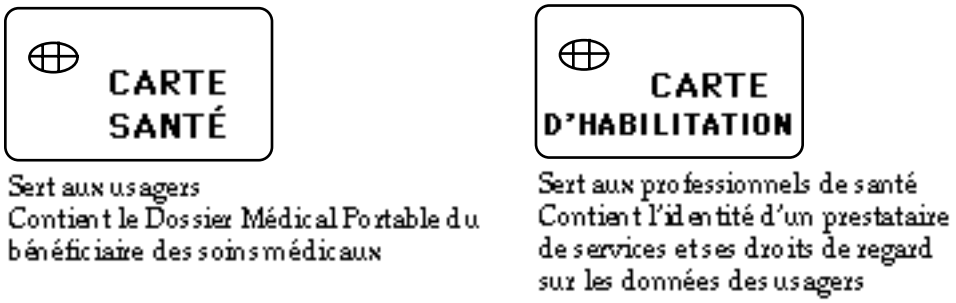


fig 2.2: Carte Santé et carte d'habilitation

2.1.1 Les acteurs

Il y a différentes catégories de professionnels de santé qui peuvent avoir une carte d'habilitation et utiliser le système Carte Santé, comme les médecins, les pharmaciens, les infirmières, les infirmières en chef et les ambulanciers. D'autres intervenants dans le système peuvent aussi avoir des cartes d'habilitation pour intervenir à certains niveaux du système, comme l'émetteur des cartes et les préposés à la maintenance du système. Avec le bénéficiaire, tous les membres de ces catégories sont appelés des acteurs dans le système Carte Santé.

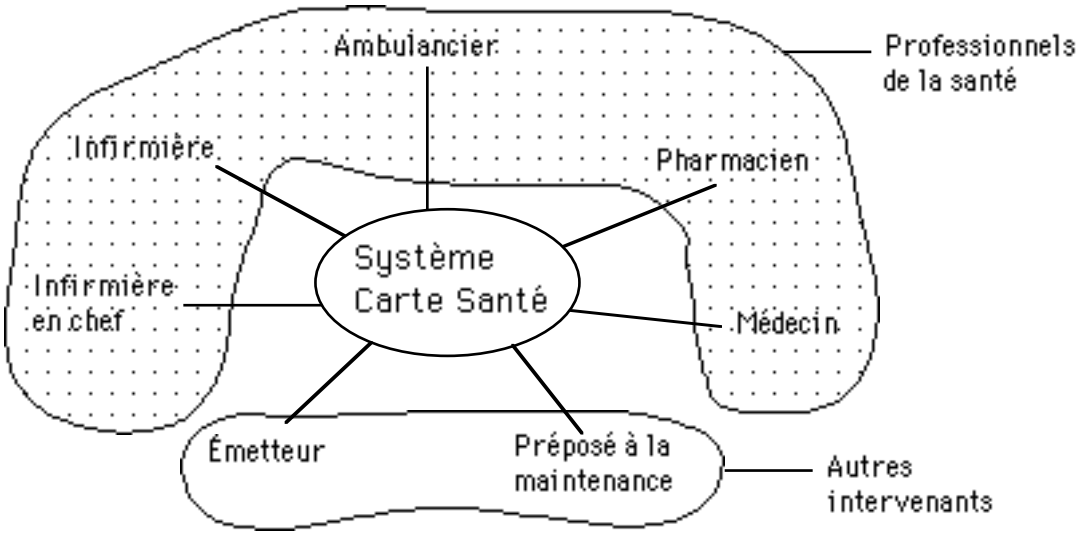


fig 2.3: Les acteurs qui interréagissent avec le système Carte Santé

Chaque acteur a un type de carte différent et accède aux informations avec des droits qui lui sont propres. La Carte Santé appartient à l'unique acteur «bénéficiaire», alors que les autres acteurs possèdent tous une carte d'habilitation.

Les acteurs sont reconnus par le système Carte Santé par leur numéro respectif codé de la façon suivante:

Numéro	Acteur	Code binaire
0	Bénéficiaire (Usager)	0000
1	Médecin	0001
2	Pharmacien	0010
3	Ambulancier	0011
4	Infirmière	0100
5	Infirmière en chef	0101
6	Émetteur	0110
7	Préposé à la maintenance	0111
de 8 à 15	libre	

fig 2.4: La codification des différents acteurs

Les numéros de 8 à 15 sont libres pour permettre d'introduire de nouveaux acteurs en cours d'exploitation du projet, comme les résidants par exemple, s'ils doivent avoir des droits de regard sur le DMP différents de ceux des autres acteurs.

Le préposé à la maintenance est un acteur qui a les mêmes droits que l'émetteur, mais en est distingué ici pour pouvoir reconnaître les écritures faites dans la carte à des fins de maintenance.

2.1.1 Les catégories des bénéficiaires

Le projet Carte Santé sert essentiellement comme référence pour évaluer la possibilité de généralisation de l'utilisation de la Carte Santé au niveau de toute la province du Québec. C'est pourquoi des catégories de bénéficiaires de soins de santé bien spécifiques sont visées. Ces catégories sont celles qui sont normalement les plus génératrices d'informations médicales dans leur dossier de soins de santé. La première catégorie est celle des nouveau-nés de 0 à 24 mois dont le suivi pédiatrique et vaccinal contient souvent beaucoup d'informations. La deuxième catégorie est celle des personnes âgées de 60 ans et plus. Cette catégorie présente généralement des cas importants de pathologies. La troisième est celle des femmes enceintes dont le suivi obstétrical, important pour les prestataires de soins, est générateur d'une masse d'informations importante. Ces trois ensembles d'usagers sont complétés par une quatrième catégorie comprenant des volontaires n'appartenant pas à ces trois ensembles. Ces volontaires sont des habitants de la municipalité de Saint-Fabien dans les environs de Rimouski. En fait, tous les usagers porteurs d'une Carte Santé sont des volontaires. Le nom de la catégorie des volontaires est donné par abus de langage. Les différentes catégories n'ont théoriquement pas d'intersection. Un volontaire âgé de plus de 60 ans appartient automatiquement à la catégorie des personnes âgées et une volontaire enceinte appartient automatiquement à la catégorie des femmes enceintes. La probabilité d'une intersection entre la catégorie des nouveau-nés et des autres catégories est nulle. Alors que la probabilité d'une intersection entre les personnes âgées et les femmes enceintes est extrêmement faible. Nous l'avons donc négligée.

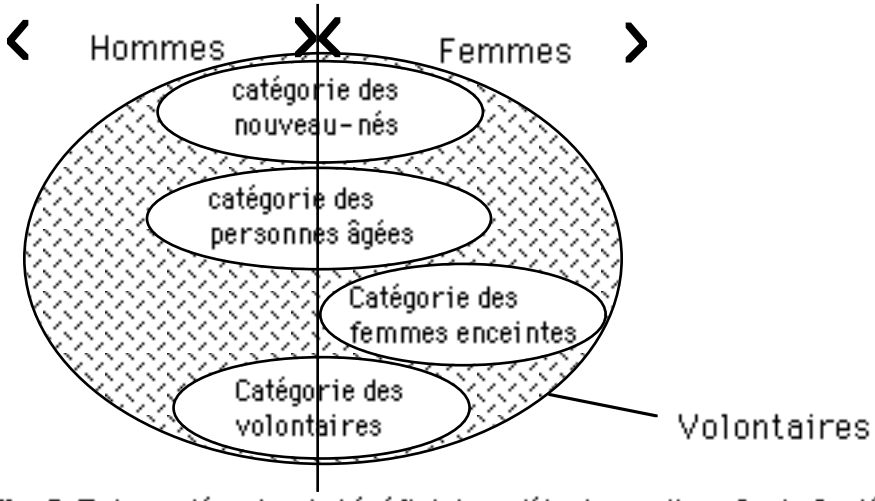


fig 2.5: Les catégories de bénéficiaires détenteurs d'une Carte Santé

2.2 Les zones et les informations

Les données inscrites sur les cartes diffèrent suivant le type de carte utilisée. Ainsi, la carte Santé qui sert de média comporte des informations médicales, alors que la carte d'habilitation contient une identification de son propriétaire et ses droits d'accès. Quel que soit le type de la carte, l'information est toujours regroupée par ensemble de données sémantiquement liées. Une application qui manipule des données dans une carte traite toujours l'information en groupe telle que regroupée sémantiquement dans les fenêtres et les écrans de l'application. Chaque ensemble est regroupé en zone, appelée zone logique. Cette zone logique ne fait aucunement référence à des droits d'accès mais correspond à des champs ou des tables apparaissant dans les fenêtres ou les écrans d'une application. Une zone telle que manipulée par une application n'est pas stockée telle quelle dans la carte, mais est restructurée pour être emmagasinée, d'une façon optimale dans d'autres ensembles. Dans la mémoire de la carte, l'information est aussi regroupée en zones. Ces zones sont dites physiques. Elles regroupent des données sur lesquelles les acteurs ont les mêmes droits d'accès. Les zones logiques et les zones physiques ne sont pas les mêmes et une correspondance doit se faire avant une écriture physique et après une lecture physique sur la carte (voir 2.7: la correspondance Logique-Physique).

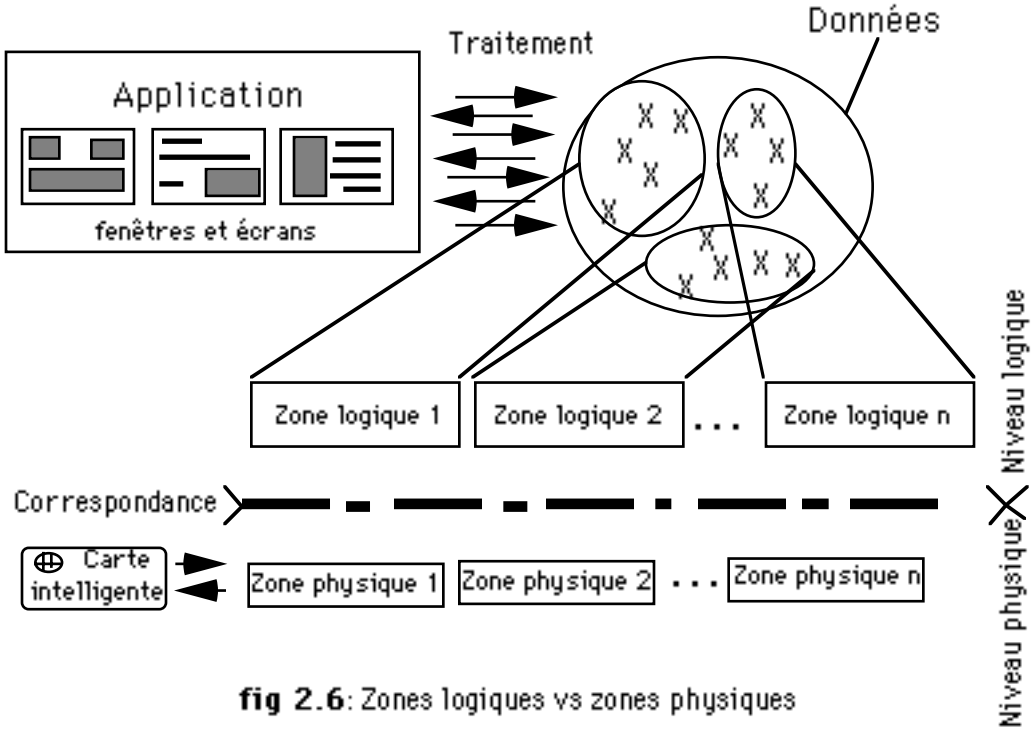


fig 2.6: Zones logiques vs zones physiques

2.2.1 Les zones logiques

Pour traiter les données, l'application utilise donc un niveau d'abstraction, dit logique, qui lui permet de traiter les données sans connaître la structure du contenu d'une carte et d'utiliser des transactions logiques permettant une indépendance des applications vis-à-vis de la structure des données sur les cartes intelligentes (voir chapitre V).

Comme mentionné plus haut, une zone logique est donc un regroupement d'éléments d'information liés sémantiquement par leur appartenance à des fenêtres ou écrans précis de l'application qui les manipule. Une zone peut alors être issue d'un ensemble de champs ou de fenêtres d'information venant de un ou de plusieurs écrans de l'application. Elle peut aussi être constituée d'éléments d'information venant d'un écran entier, qui sont écrits dans la carte ou lus de la carte et reliés ensemble.

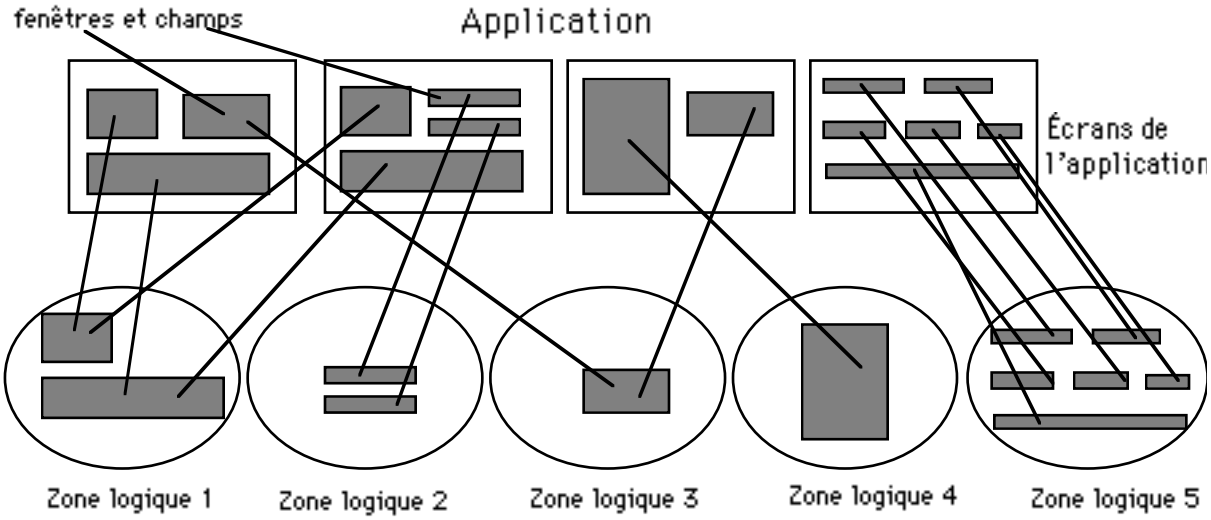


fig 2.7: Les zones logiques

Les zones logiques, qui sont donc théoriques, peuvent être au nombre de 256. Elles sont numérotées de 0 à 255 pour être identifiées lors des transactions logiques.

Une zone logique correspond à une table constituée d'enregistrements comme une relation d'une base de données relationnelle. Chaque enregistrement (ou tuple) est constitué de champs (ou attributs). Il y a une possibilité d'avoir jusqu'à 256 champs différents. Chaque champ est une variable logique et a un type précis (se définit sur un domaine).

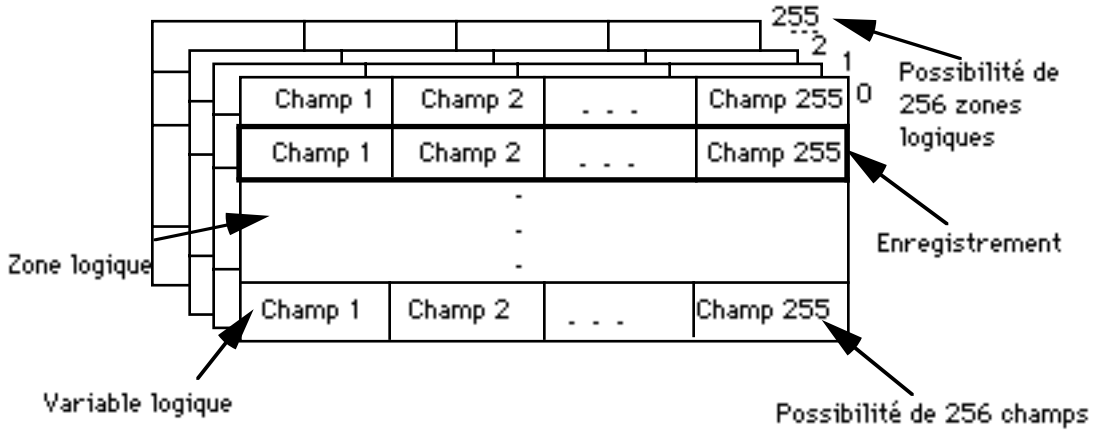


fig 2.8: Ensemble de zones logiques avec leurs enregistrements

La particularité des enregistrements d'une zone est qu'ils peuvent avoir une

longueur variable. Un même champ dans deux enregistrements peut avoir deux tailles différentes. L'information relative à un champ peut aussi être absente dans certains enregistrements. Grâce au fait que les variables logiques sont identifiées par des numéros (entre 0 et 255), l'absence ou la présence d'un champ peut facilement être décelée. Cette variabilité des enregistrements est dûe à la structure possiblement complexe des champs. Chaque variable logique a un type précis. Un des types de données possible est la liste. Une liste est une suite d'informations de taille variable. Une variable de type liste pourrait avoir trois éléments dans sa suite alors que la même variable dans un autre enregistrement pourrait n'en avoir que deux.

2.2.2 Les zones physiques

Contrairement aux zones logiques, les zones physiques sont très liées aux droits d'accès à l'information. Une zone physique regroupe des éléments d'information sur lesquels les acteurs ont un même droit d'accès. Ces éléments d'information peuvent venir d'une ou de plusieurs zones logiques.

Les zones physiques sont implantées physiquement sur la carte en allouant des blocs de mémoire dans la puce de la carte.

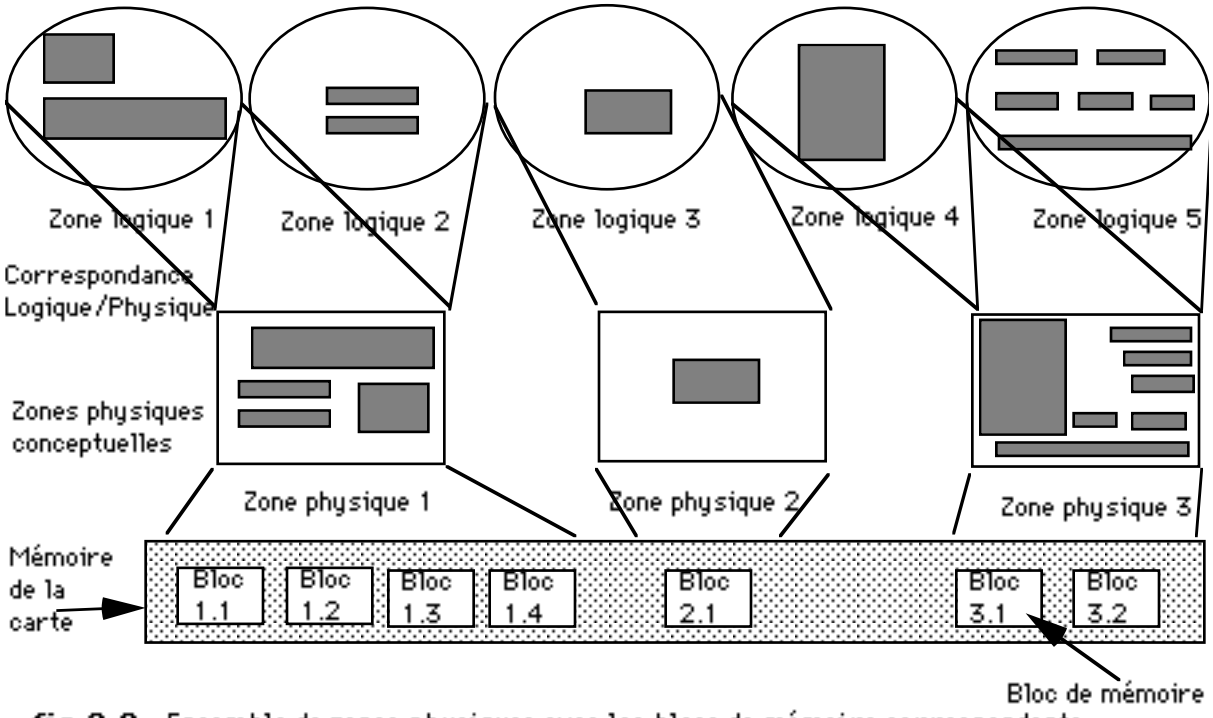


fig 2.9: Ensemble de zones physiques avec les blocs de mémoire correspondants

Théoriquement, les zones physiques peuvent aussi être au nombre de 256. Mais certaines technologies de cartes intelligentes imposent une limite inférieure. Les champs au sein de ces zones diffèrent cependant d'une zone à l'autre.

2.2.3 Les types d'opérations sur les zones de données

Aucune modification ni suppression n'est permise sur les données médicales du DMP. Cependant, certaines rectifications de l'information sont possibles en cas d'écriture de données erronées sur la carte. La rectification ne se traduit alors pas par une modification ou une suppression, mais simplement par une annulation de l'information erronée et l'écriture de la nouvelle information. L'annulation est elle-même une écriture d'une information indiquant qu'une donnée particulière sur la carte est annulée (voir 5.2.3). Il y a donc essentiellement deux types d'opérations sur les zones d'information du DMP: l'écriture des données et la lecture des données. L'information du DMP est donc cumulative; on ne fait que rajouter des données sans jamais en supprimer. Ce choix a été pris pour assurer une meilleure sécurité pour le contenu informationnel du DMP. Cependant dans la mémoire de la carte intelligente, d'autres zones d'information indépendantes du DMP peuvent résider, comme les zones de statistiques ou les zones de gestion. Les zones de gestion sont gérées par la Coquille et parfois même par l'application. Elles ne nécessitent pas d'être cumulatives car l'ajout d'une information dans la zone rend souvent l'ancienne information périmée. Pour ces zones, un second type d'écriture est nécessaire: l'écriture de mise à jour par opposition à l'écriture cumulative. Comme l'annulation de l'information permise indirectement avec l'opération d'écriture, l'écriture de mise à jour permet la suppression des données. En effet, dans certaines zones de gestion, l'opération de suppression peut être indispensable pour effacer, par exemple, des caractères de contrôle.

La figure 2.10 schématise les types d'opération possibles sur les zones de données. L'opération d'annulation et l'opération de suppression sont indirectement possibles grâce aux deux types d'écriture.

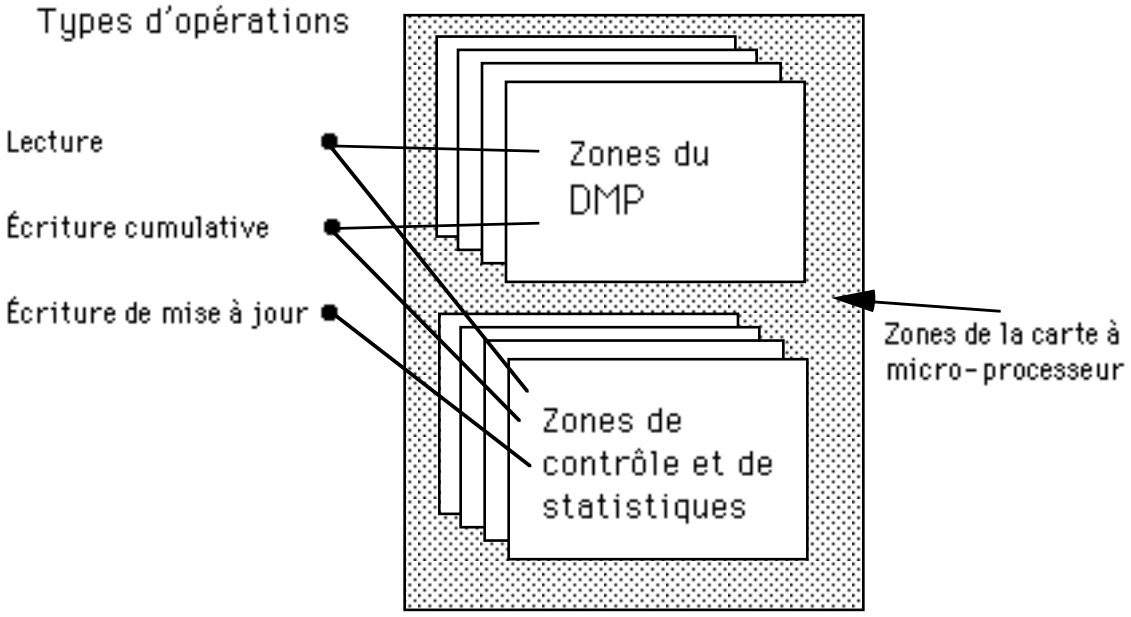


fig 2.10: Les types d'opérations possibles sur les zones

Chaque zone est caractérisée par le type d'écriture que l'on peut effectuer sur ses données. Il faut noter que toutes les zones du DMP ne sont accessibles que par l'écriture cumulative.

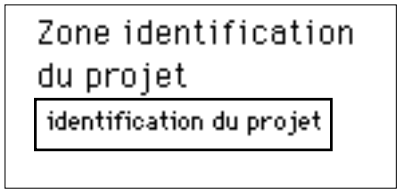
2.2.4 Le contenu des cartes

2.2.4.1 La carte d'habilitation

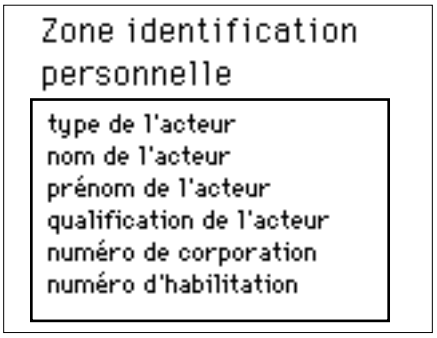
La carte d'habilitation permet à l'acteur porteur de celle-ci d'accéder à l'information d'une Carte Santé après s'être identifié et au besoin s'être authentifier. Elle contient donc des informations nominatives pour identifier son propriétaire ainsi que des informations secrètes habilitant le porteur à accéder aux informations du DMP auxquelles il a droit.

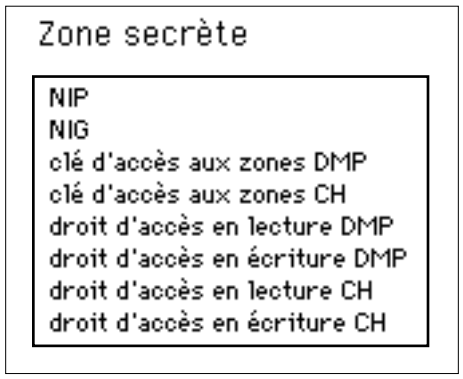
Dans une carte d'habilitation, il y a huit zones logiques: une zone d'identification du projet, une zone d'identification du détenteur de la carte, une zone secrète, une zone pour la liste des paramétrisations, une zone pour les paramétrisations annulées, une zone pour la dernière paramétrisation choisie, une zone pour les statistiques et une zone de contrôle.

Zone identification du projet (zone logique H1): Cette zone contient un identifiant permettant à une application de savoir si la carte appartient au projet-pilote de Carte Santé ou est une quelconque autre carte. L'écriture de mise à jour est permise sur cette zone.

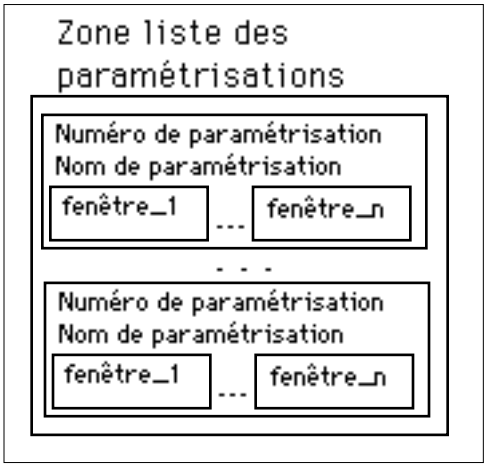


Zone d'identification du détenteur de la carte (zone logique H2): Cette zone contient le type de l'acteur tel que défini à la figure 2.4, le nom et le prénom de l'acteur, sa qualification (M, Mme, Melle, Dr, Pr ...), son numéro à la corporation à laquelle il adhère et son numéro d'habilitation dans le projet-pilote. Le numéro d'habilitation est un numéro séquentiel permettant d'identifier une carte d'habilitation du projet sans avoir de lien avec l'identification du titulaire de la carte. Il est entre autres utile pour relever des statistiques. L'écriture de mise à jour est permise sur cette zone.

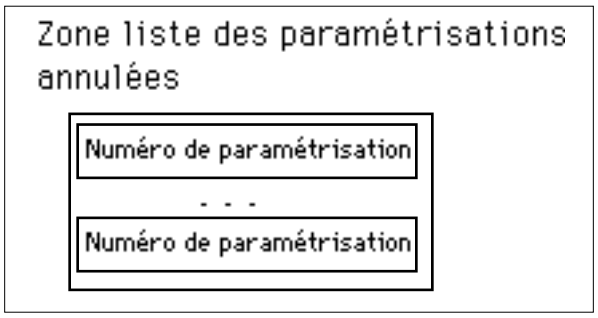




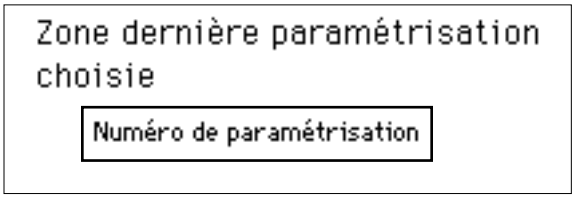
Zone secrète (zone logique H3): Cette zone contient les informations secrètes comme le numéro d'identification personnel de l'acteur détenteur de la carte (NIP), le numéro d'identification général (NIG), la clé d'accès aux zones du DMP, la clé d'accès aux zones d'une carte d'habilitation (CH) ainsi que le profil d'accès aux deux types de cartes. Le NIG est un numéro permettant d'accéder à toutes les Cartes Santé. Il remplace le NIP de chaque Carte Santé qui, en fait, n'en est pas pourvue. La clé d'accès aux zones du DMP permet au masque de la carte intelligente comportant le DMP de connaître les zones physiques auxquelles l'acteur a accès. Cette clé définit, d'une façon intrinsèque à la Carte Santé, les droits de lecture et d'écriture que l'acteur peut effectuer sur chaque zone physique du DMP auquelle il a droit. De même, la clé d'accès aux zones CH permet au masque d'une carte d'habilitation quelconque de connaître les zones physiques auxquelles l'acteur a accès. Le profil d'accès aux deux types de cartes, carte d'habilitation et Carte Santé, est composé de quatre champs. Les deux premiers indiquent les droits d'accès en lecture puis en écriture de l'acteur dans toutes les zones logiques d'une Carte Santé et les deux derniers indiquent les droits d'accès en lecture puis en écriture de l'acteur dans les zones logiques d'une carte d'habilitation. Le profil d'accès logique permet à la Coquille de contrôler et d'intercepter, à un premier niveau de sécurité, les tentatives frauduleuses ou erronées d'accès à une information interdite. L'écriture de mise à jour est permise sur cette zone.



Zone pour la liste des paramétrisations (zone logique H4): Cette zone contient les paramétrisations que le détenteur de la carte d'habilitation a créées lui-même. La paramétrisation est une représentation de la configuration de l'application que le détenteur de la carte a choisi d'avoir à chaque fois qu'il accède à une Carte Santé. En effet, l'application médicale peut être configurée à volonté pour permettre, à un médecin par exemple, d'avoir certaines fenêtres particulières ouvertes automatiquement à chaque fois qu'il insère une Carte Santé. Cette configuration personnelle à un médecin, par exemple, peut différer suivant le type de pratique du médecin envers ses patients. C'est la raison pour laquelle on permet d'emmagasiner plusieurs paramétrisations. Une paramétrisation est composée d'un numéro séquentiel l'identifiant, d'un nom servant d'identifiant dans un menu et d'une suite de numéros de fenêtres de l'application que le porteur veut automatiquement ouvertes à l'insertion d'une Carte Santé. L'écriture de mise à jour n'est pas permise sur cette zone.



Zone pour les paramétrisations annulées (zone logique H5): Cette zone contient la liste des paramétrisations que le détenteur de la carte d'habilitation a annulées et ne veut plus voir apparaître sur le menu de choix de paramétrisation. L'écriture de mise à jour est permise sur cette zone. Lorsque le détenteur de la carte rajoute une nouvelle paramétrisation dans la zone précédante, le système choisit d'abord un numéro de paramétrisation annulé dans la zone H5 pour l'attribuer à la nouvelle paramétrisation. C'est seulement lorsqu'il n'y a aucune paramétrisation annulée dans cette zone que le système attribue un nouveau numéro à la nouvelle paramétrisation de la zone H4. Ceci est fait dans le but d'optimiser la gestion des numéros de paramétrisations.



Zone pour la dernière paramétrisation choisie (zone logique H6): Cette zone contient le numéro de la dernière paramétrisation choisie. Connaissant ce numéro, l'application peut mettre en évidence le dernier choix dans le menu des paramétrisations quel que soit le site où le détenteur de la carte d'habilitation utilise sa carte. Ceci évite des manipulations inutiles de la part du détenteur de la carte et rend l'application beaucoup plus conviviale. L'écriture de mise à jour est permise sur cette zone.

Zone pour les statistiques (zone logique H7): Cette zone contient différentes variables recensées pour les statistiques nécessaires pour une évaluation du projet pilote (voir 5.6: les statistiques pour l'évaluation du projet). L'écriture de mise à jour n'est pas permise sur cette zone.

Zone de contrôle (zone logique H8): Cette zone contient des variables de gestion permettant l'intégrité des données en contrôlant l'atomicité des données logiques et physiques (voir 5.2.2: intégrité des données sur la carte). L'écriture de mise à jour est permise sur cette zone.

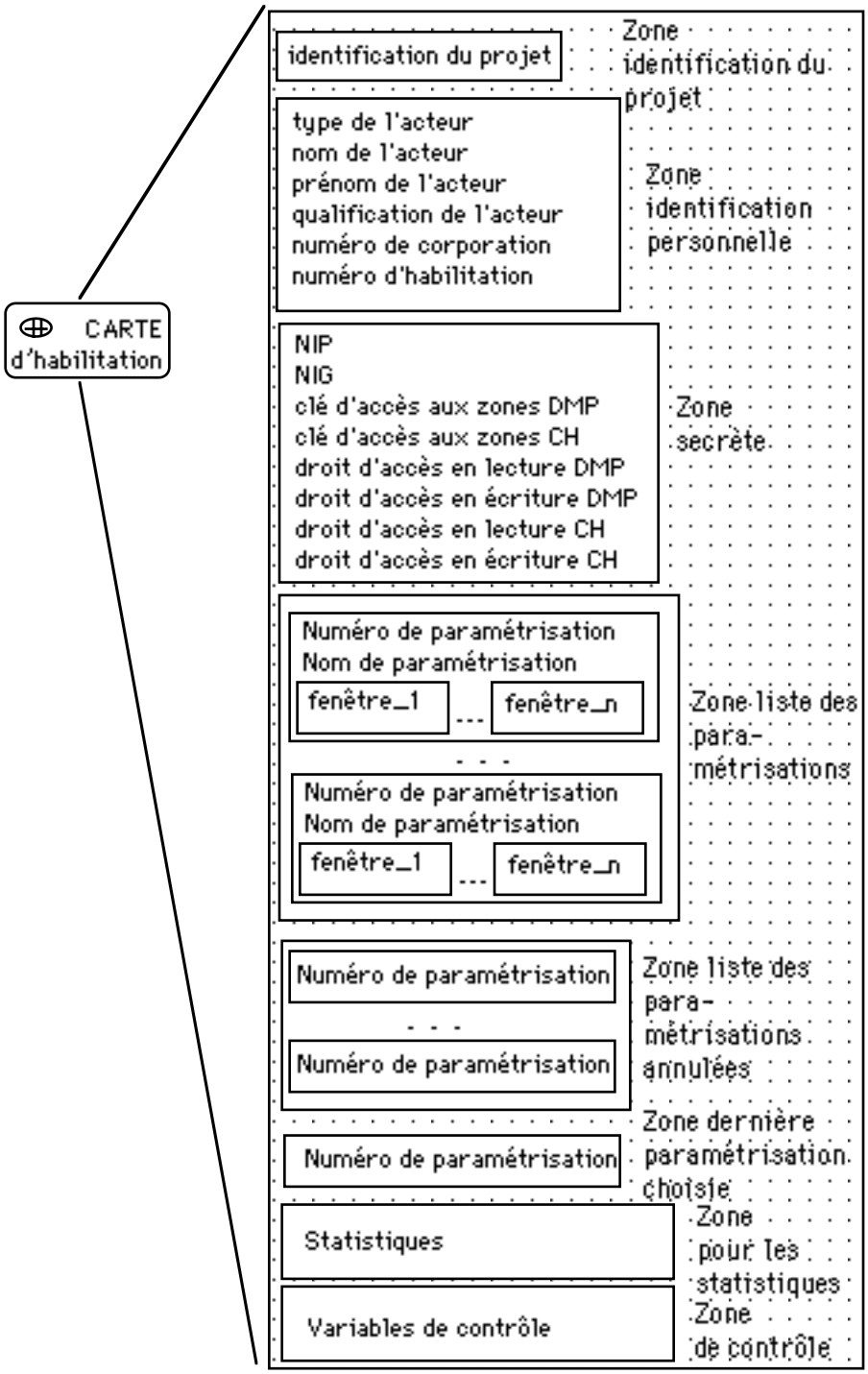


fig 2.11 : Le contenu d'une carte d'habilitation

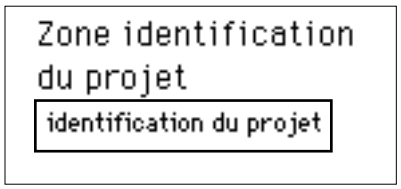
2.2.4.2 La Carte Santé

La Carte Santé est la carte qui contient le DMP d'un bénéficiaire des soins médicaux. Chaque carte contient un DMP identifié au détenteur de la carte. Dans une Carte Santé, il y a six zones logiques permanentes et un ensemble de zones médicales constituant le véritable DMP. Les zones permanentes sont une zone

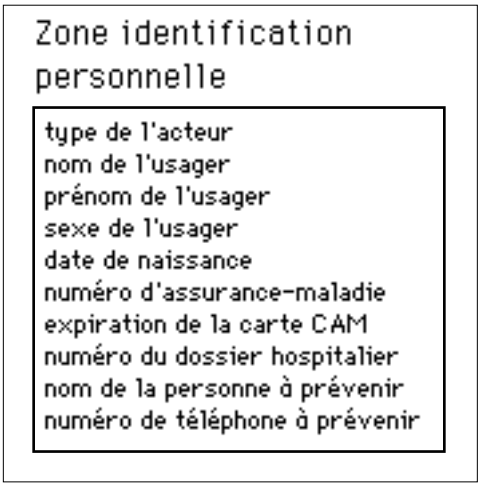
d'identification du projet, une zone d'identification du détenteur de la carte, une zone des signatures des prestataires, une zone pour les statistiques, une zone de contrôle et une zone de structure.

Dans ce chapitre, la zone d'identification personnelle fait parfois partie des zones médicales. En fait, dans plusieurs documents elle fait partie du DMP pour permettre l'identification de celui-ci et d'avoir des informations nominatives. Cette zone est néanmoins permanente. C'est pourquoi nous l'explicitons avec les zones permanentes. Cependant, nous lui avons donné un numéro appartenant à la numérotation des zones du DMP et nous l'avons incluse dans l'explication du contenu et de la codification des champs des zones du DMP.

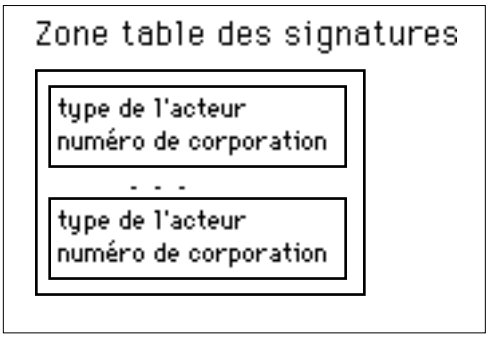
Zone identification du projet (zone logique S1): Cette zone contient un identifiant permettant à une application de savoir si la carte appartient au projet-pilote de Carte Santé ou est une quelconque autre carte. L'écriture de mise à jour est permise sur cette zone.



Zone d'identification du détenteur de la carte (zone logique 1): Cette zone contient le type de l'acteur, dans ce cas 0, le nom et le prénom ainsi que le sexe et la date de naissance du détenteur de la carte, son numéro d'assurance-maladie, la date d'expiration de sa carte d'assurance-maladie, le numéro de son dossier hospitalier à l'hôpital de Rimouski, le nom, le prénom, le numéro de téléphone de la personne à prévenir en cas d'urgence et un numéro d'usager. Le numéro d'usager est un numéro séquentiel permettant d'identifier une Carte Santé du projet sans avoir de lien avec l'identification du titulaire de la carte. Il est entre autres utile pour relever des statistiques. L'écriture de mise à jour n'est pas permise sur cette zone.



Zone des signatures des prestataires (zone logique S2): Cette zone est une table contenant la signature électronique de chaque prestataire de soins qui a écrit et signé une information dans la carte. Cette table peut contenir jusqu'à 256 signatures numérotées de 0 à 255. Le numéro dans la table constitue un index pour identifier chaque signature figurant dans le DMP. La signature électronique est composée du type de l'acteur ayant effectué la signature suivi de son numéro corporatif. L'écriture de mise à jour n'est pas permise sur cette zone.



Zone pour les statistiques (zone logique S3): Cette zone contient différentes vari-

ables recensées pour les statistiques nécessaires pour une évaluation du projet-pilote (voir 5.6: les statistiques pour l'évaluation du projet). L'écriture de mise à jour n'est pas permise sur cette zone.

Zone de contrôle (zone logique S4): Cette zone contient des variables de gestion permettant l'intégrité des données en contrôlant l'atomicité des données logiques et physiques (voir 5.2.2: intégrité des données sur la carte). L'écriture de mise à jour est permise sur cette zone.

Zone de structure (zone logique S5): Cette zone contient une représentation du DMP. Pour chaque zone logique médicale du DMP, une variable indique si cette dernière est vide ou contient de l'information. Cette zone, qui donne en fait la structure du DMP du porteur de la carte, sert à l'application pour accélérer les affichages sur écran sans être obligé de lire le dossier en entier sur la carte.

Zones médicales (zones logiques 2 à 40): Ces zones constituent le DMP. Elles sont explicitées en détail dans les sections qui suivent.

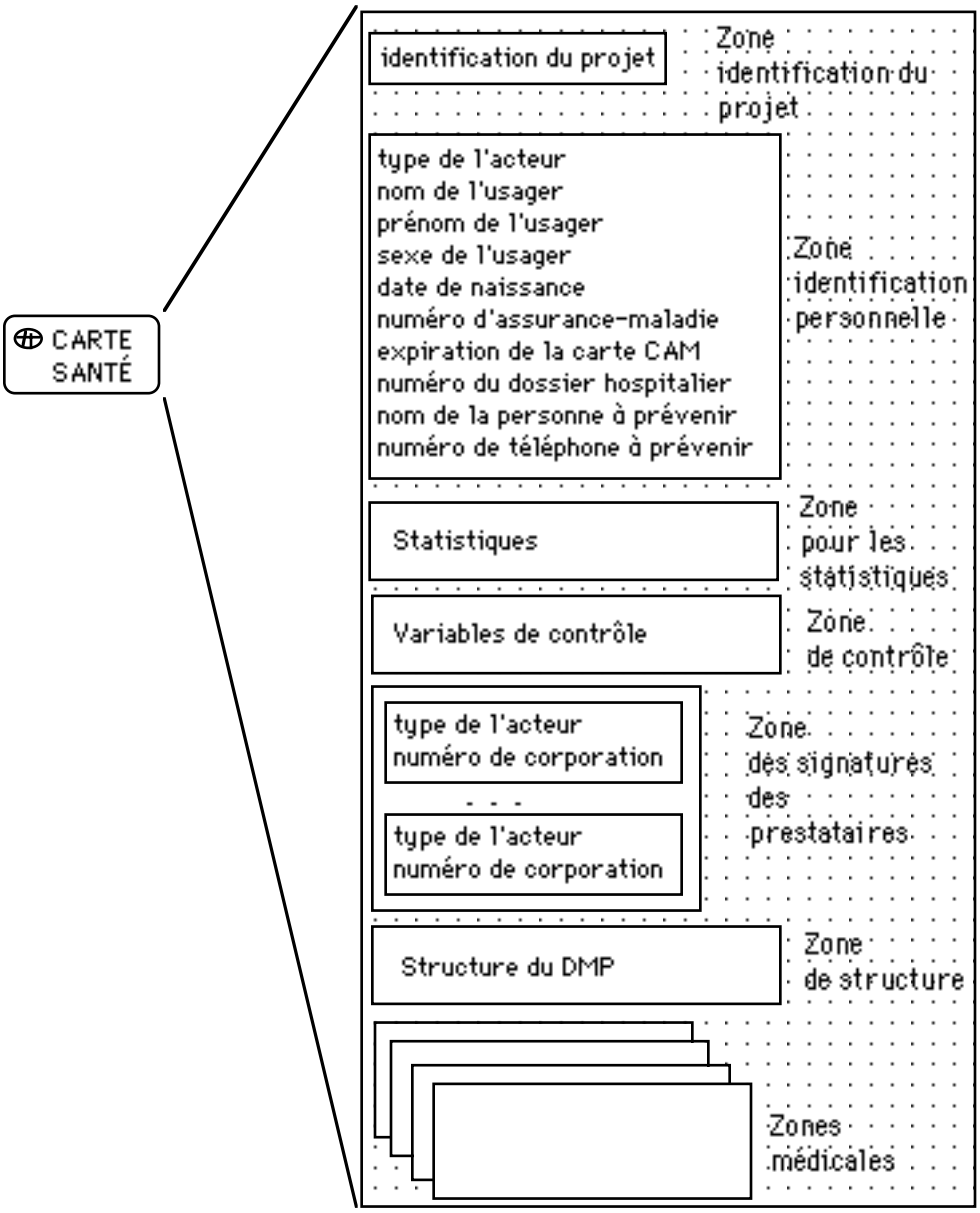


fig 2.12: Le contenu d'une Carte Santé

2.3 Le dossier médical portable logique

Le DMP est un simple support pour un ensemble restreint d'informations médicales pertinentes et utiles pour appuyer toute action ou décision concernant la santé d'une personne. Les informations jugées utiles et retenues pour être inscrites dans le DMP sont regroupées par ensemble. Nous retrouvons en premier l'ensemble des informations d'identification du porteur du DMP (ces informations sont dans la zone d'identification personnelle explicitée avec les zones permanentes). Les ensembles contenant les faits médicaux proprement dits sont au nombre de 13. Nous avons l'ensemble des informations d'urgence, des allergies, des vaccinations, des médications prises par le détenteur de la Carte Santé, ainsi que celui des antécédents familiaux puis personnels et les données ophtalmologiques regroupées aussi dans

un module. Nous retrouvons aussi des ensembles d'informations permettant certains suivis spécifiques comme les informations concernant les facteurs de risques cardio-vasculaires, les informations concernant le diabète, les fonctions respiratoires ou les informations concernant les insuffisances rénales. Un autre groupe d'informations concerne les suivis généraux. Enfin, un module témoin permet d'avoir des informations sur les examens de prévention. Pour des catégories de personnes bien spécifiques, on retrouve aussi un ensemble d'informations pédiatriques et un ensemble d'informations obstétricales permettant le suivie de grossesse. Tous ces modules d'informations sont regroupés en zones logiques.

2.3.1 Les zones logiques du DMP

Il y a 40 zones logiques dans le DMP. Une zone logique concerne l'identification, deux autres zones sont pour le suivi pédiatrique, 15 pour le suivi obstétrical et 22 contiennent des informations médicales d'ordre général.

Le contenu de la zone logique d'identification personnelle (zone logique 1) a été présenté dans le paragraphe 2.2.4.2. Nous présentons, dans ce qui suit, le contenu des différentes zones médicales proprement dites du DMP.

2.3.1.1 Les zones médicales d'ordre général

Le groupe sanguin (zone logique 2): Elle contient le groupe sanguin et le rhésus. Huit combinaisons sont possibles:

Numéro	Groupe sanguin	Numéro	Groupe sanguin
0	A+	4	A-
1	B+	5	B-
2	AB+	6	AB-
3	0+	7	0-

Urgence (zone logique 3): Elle contient les pathologies importantes à connaître en cas d'urgence et les prothèses que porte le bénéficiaire. Les pathologies et les prothèses sont codées ensembles pour représenter 16 variantes.

Informations d'urgence (zone logique 4): Ce sont des informations pertinentes en cas d'urgence écrites en texte libre.

Médication (zone logique 5): Elle contient tous les médicaments que le titulaire de la Carte Santé a pris. Cette zone contient aussi les médicaments en vente libre que le bénéficiaire s'est procuré et dont il a permis l'inscription sur sa carte. Pour chaque médication, on retrouve le numéro DIN, la quantité délivrée, la durée du traitement, le nombre de renouvellement permis et le Pro-ré-nata (PRN). Le PRN indique si le médicament est pris au besoin ou régulièrement lors du traitement.

Intolérances médicamenteuses (zone logique 6): Elle contient la liste des

médicaments que le détenteur de la Carte Santé ne tolère pas.

Allergies médicamenteuses (zone logique 7): Elle contient la liste des médicaments auxquels le détenteur de la Carte Santé est allergique.

Allergies autres que médicamenteuses (zone logique 8): Les allergies du détenteur de la Carte Santé sont inscrites dans cette zone logique en texte libre.

Vaccination (zone logique 9): Elle contient toutes les vaccinations du titulaire de la Carte Santé.

Suivi du poids (zone logique 10): Cette zone contient toutes les mesures de poids du titulaire de la Carte Santé

Occupation (zone logique 11): Elle contient l'occupation professionnelle du porteur de la carte. Elle permet d'évaluer les risques courrus en milieu de travail. Les occupations différentes sont codées pour accommoder 128 possibilités.

Antécédents familiaux (zone logique 12): Elle contient les antécédents familiaux comme le cancer, l'asthme, les infarctus... La liste de ces antécédents (16) se trouve à l'annexe A.

Antécédents familiaux texte (zone logique 13): Cette zone contient, en texte libre, les antécédents familiaux qui n'ont pas pu être codés dans la zone précédente.

Antécédents personnels (zone logique 14): Elle contient tous les antécédents personnels; médicaux, chirurgicaux ainsi qu'obstétricaux. Chaque antécédent est codé par une table et on mémorise l'année de son intervention ou de son diagnostic. Plusieurs tables sont possibles comme CIM-9 (Classification internationale des maladies neuvième révision), ICPC (International classification of primaru care), Read-Code... On retrouve donc dans cette zone pour chaque antécédent l'indice de la table utilisée, le code de l'antécédent et l'année concernée.

Suivi général (zone logique 15): Elle contient tout ce qui a été diagnostiqué chez le porteur de la Carte Santé. Chaque diagnostic peut être codé par une table différente comme ICPC ou ReadCode et est accompagné d'une information permettant de savoir si le diagnostic a été fait avec certitude. Pour chaque diagnostic, nous retrouvons aussi une liste de conduites qui lui sont liées. C'est une gestion orientée par problèmes de santé. Les conduites peuvent être codées ou introduites en texte libre.

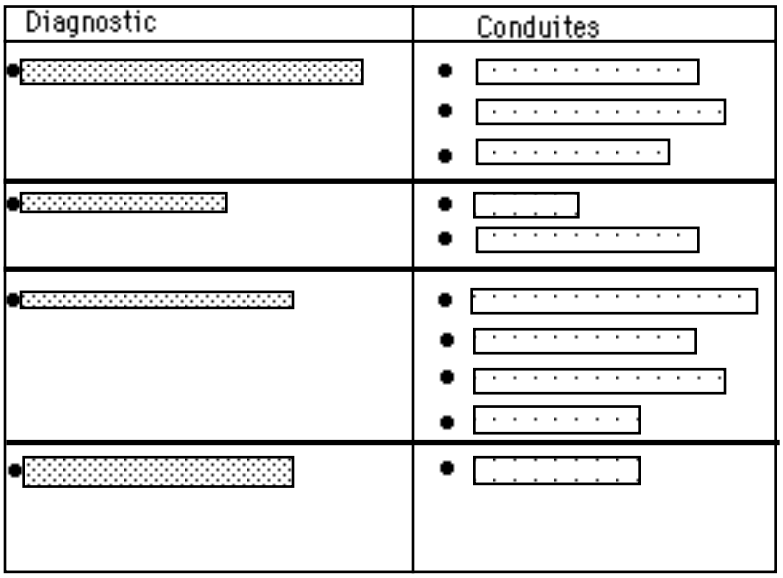


fig 2.13: «Approche par problème»

Prévention (zone logique 16): Cette zone contient les témoins des examens de prévention.

Ophtalmologie (zone logique 17): Elle contient des informations ophtalmologiques. On y retrouve l'acuité visuelle pour les deux yeux avec et sans lunette, l'acuité visuelle de chaque oeil avec et sans lunette, la papille droite et gauche, la tension oculaire droite et gauche et les différentes réfractions.

Suivi cardio-vasculaire (zone logique 18): Cette zone contient certaines informations sur les facteurs de risque cardio-vasculaire. Les variables suivies comme les taux de cholestérol et de triglycéride se trouvent aussi dans cette zone.

Suivi diabète (zone logique 19): Elle contient des informations pertinentes à suivre chez les diabétiques (voir annexe A).

Créatinine (zone logique 20): Cette zone contient deux variables: le BUN (urée) et la créatinine qui sont suivis avec le diabète, de même qu'au moment des insuffisances rénales.

Suivi Insuffisance rénale (zone logique 21): Cette zone contient des informations pertinentes à suivre lors des insuffisances rénales (voir annexe A).

Suivi fonction respiratoire (zone logique 22): Cette zone contient des données sur les fonctions respiratoires comme le débit de pointe, l'indice de saturation et des mesures de gaz carbonique et d'oxygène.

Pression artérielle (zone logique 23): Cette zone contient la fréquence cardiaque et les différentes pressions artérielles. Ces variables sont suivies lors des insuffisances

rénales et des suivis cardio-vasculaires.

2.3.1.2 Les zones pédiatriques

Suivi pédiatrique (zone logique 24): Cette zone contient l'évolution de la taille du titulaire de la carte ainsi que l'évolution de son périmètre crânien.

Antécédents pédiatriques (zone logique 25): Cette zone contient des informations pertinentes concernant la gestation et l'accouchement du bébé. On y retrouve le taux d'éveil du bébé (apgar) ainsi que des informations sur les éventuelles complications survenues lors de l'accouchement (voir annexe A).

2.3.1.3 Les zones obstétricales

GPA (zone logique 26): Cette zone contient le nombre de grossesses antérieures (Gravida), le nombre de grossesses prématurées (Para), le nombre d'avortements subis par la détentrice de la Carte Santé (Aborta) ainsi que le nombre d'enfants vivants.

Accouchement (zone logique 27): Cette zone contient des informations relatives à l'accouchement comme la date d'accouchement, les suites de couches (normales ou anormales) et les hémorragies puerpérales. S'il y a eu fièvre, on y trouve aussi les causes de celle-ci.

Résumés des grossesses antérieures (zone logique 28): Cette zone contient des informations concernant les grossesses antérieures. On y trouve les dates d'accouchement, les durées de grossesse et des informations et particularités sur les accouchements antérieurs(voir annexe A).

Amniocentèse (zone logique 29): Cette zone contient les résultats de l'amniocentèse.

Résultat de l'échographie (zone logique 30): Cette zone contient les résultats d'une échographie comme l'âge gestationnel et la morphologie (voir annexe A).

Immunoglobine (zone logique 31): Cette zone contient des informations sur l'immunoglobine anti D.

Grossesse à risque (zone logique 32) : Elle contient une liste des facteurs de risque chez la femme enceinte porteuse de la Carte Santé.

Résumés des visites prénatales (zone logique 33): Cette zone contient des informations concernant les visites prénatales tout au long de la grossesse. On y retrouve des informations sur l'évolution de la grossesse et du foetus (voir annexe A).

Examens de laboratoire (zone logique 34): Cette zone contient des résultats d'examens de laboratoires effectués lors d'un suivi de grossesse (voir annexe A).

Tests sanguins (zone logique 35): Cette zone contient des résultats de tests sanguins effectués lors d'un suivi de grossesse (voir annexe A).

Une dernière zone appelée **Suivi spécifique (zone logique 36)** contient des variables de suivi pouvant appartenir à différentes zones logiques de suivi, que ce soient des zones médicales d'ordre général, des zones obstétricales ou pédiatriques. Cette zone permet au praticien de suivre des variables spécifiques qu'il définit dynamiquement.

Le contenu précis et détaillé de chaque zone logique se trouve à l'annexe A explicité vis-à-vis des variables physiques.

2.3.2 Les catégories de bénéficiaires et les zones logiques

Toutes les zones logiques vues précédemment ne se trouvent pas nécessairement dans toutes les Cartes Santé, toutes catégories de bénéficiaires confondues. En effet, les zones pédiatriques ne sont présentes que dans les cartes des nouveau-nés. De même, les zones obstétricales ne se trouvent que dans les cartes des femmes enceintes. Dans une Carte Santé d'une catégorie de bénéficiaire donnée, il y a des zones logiques qui s'y trouvent nécessairement, d'autres probablement et certaines zones qui ne s'y trouvent jamais.

Dans ce qui suit, nous présentons les catégories de bénéficiaires et les zones logiques présentes nécessairement ou probablement dans leurs cartes.

Les nouveau-nés (0 à 24 mois):
Zones logiques obligatoires: 1, 2, 3, 4, 5, 6 ,7, 8, 9, 10, 12, 13, 24 et 25.
Zones logiques probables : 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 et 36.

Les personnes âgées (60 ans et plus):
Zones logiques obligatoires: 1, 2, 3, 4, 5, 6, 7, 8, 12, 13, 14, 15, 16, 18, 19, 20, 21, 22, 23 et 36.
Zones logiques probables : 9, 10, 11 et 17.

Les femmes enceintes:
Zones logiques obligatoires: 1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 15, 16, 26, 27, 28, 29, 30, 31, 32, 33, 34 et 35.
Zones logiques probables : 9, 10, 17, 18, 19, 20, 21, 22, 23 et 36.

Les autres volontaires:
Zones logiques obligatoires: 1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 15 et 16.
Zones logiques probables : 9, 10, 17, 18, 19, 20, 21, 22, 23 et 36.

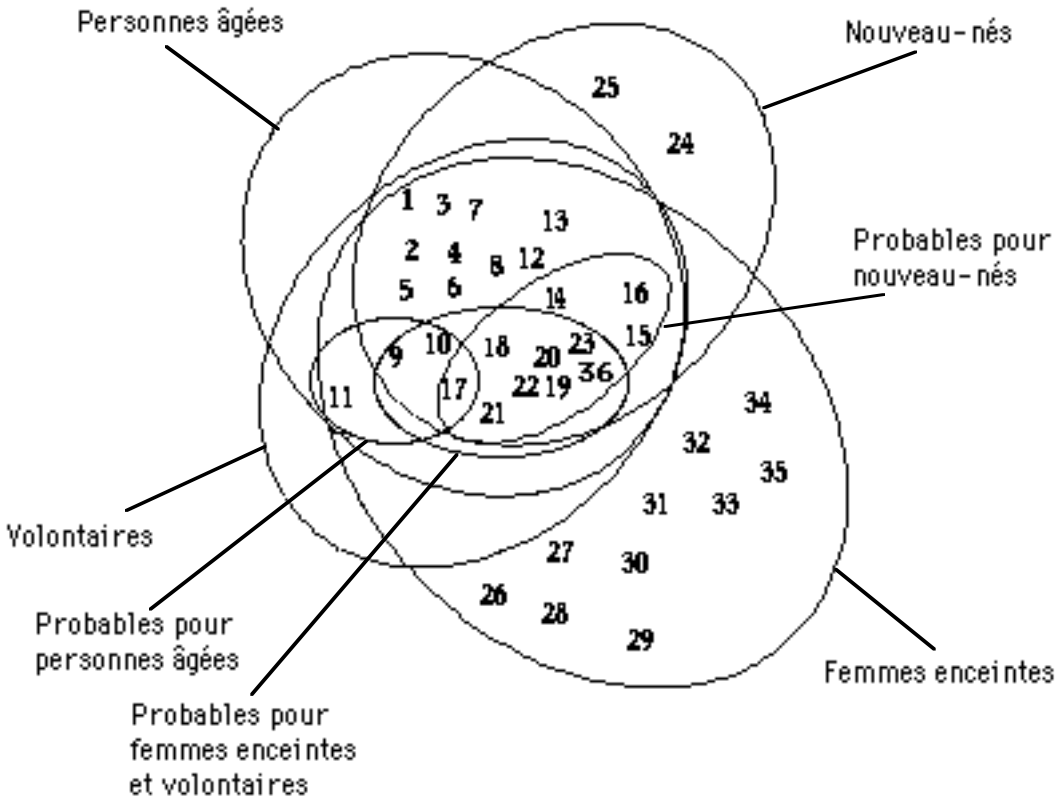


fig 2.14: Les catégories de bénéficiaires et les zones logiques

2.4 L'accès à l'information et les vues

Comme nous l'avons présenté au paragraphe 2.1.1, il y a plusieurs intervenants qui sont habiletés à utiliser la carte. Il leur est possible de lire l'information déjà inscrite comme d'écrire de nouvelles données.

2.4.1 Les acteurs et leurs besoins

La carte intelligente, de part la sécurité qu'elle offre, permet d'imposer un contrôle d'accès aux données. Un intervenant n'est autorisé à accéder qu'aux informations pertinentes à son champ de pratique. En tenant compte des règles de pratique et de déontologie, des contraintes légales et techniques, il est possible d'établir les droits de regard à l'information médicale personnelle d'un patient de chaque intervenant dans le système Carte Santé. En fait la carte intelligente garantit des «filtres» qui, posés au-dessus de l'information du DMP, ne permettent aux acteurs que de voir les informations qui les intéressent dans le cadre de leur pratique. Cette interprétation figurée permet d'illustrer le concept des vues. Une vue, pour un acteur, est l'unique portion d'informations à laquelle il a accès. Le reste des données qui ne figurent pas dans la vue sont comme inexistantes pour l'acteur.

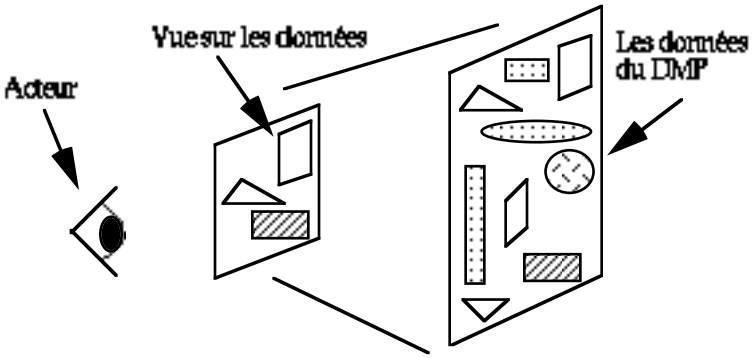


fig 2.15: La vue d'un acteur sur les données

2.4.2 Les profils d'accès

L'accès à l'information peut se faire pour une lecture (accès en lecture) ou pour inscrire une nouvelle donnée (accès en écriture). Les droits d'accès en lecture et en écriture d'un acteur dans l'ensemble des zones du DMP sont appelés le profil d'accès de l'acteur au DMP. Ce profil d'accès est automatiquement pris en compte par le masque de la carte à micro-processeur. On peut aussi définir d'autres types d'accès comme l'accès en réécriture, l'accès en suppression ou accès en impression. L'accès en réécriture ou en suppression, c'est-à-dire l'accès en mise à jour, est normalement impliqué, dans le cadre du projet Carte Santé, dans le droit d'écriture. Ce droit de mise à jour n'est pas supporté par toutes les technologies de cartes intelligentes. Il est cependant contrôlé par la Coquille, c'est à dire le module qui prend en charge toutes les transactions destinées à la carte intelligente. Le droit d'impression, pour sa part, n'est pas contrôlable par la carte intelligente. Il est cependant possiblement traité par l'application.

Toutes les zones logiques d'une carte d'habilitation, mis à part la zone secrète et les deux zones d'identification, sont accessibles en lecture et écriture en tout temps. Les trois autres zones sont accessibles par l'émetteur en lecture et écriture et par le porteur de la carte en lecture uniquement.

Les zones permanentes de la Carte Santé sont aussi accessibles en lecture et écriture en tout temps. L'écriture n'est cependant permise que pour l'émetteur dans la zone d'identification du projet, alors que l'accès en écriture à la zone d'identification personnelle est contrôlé et dépend de l'acteur (voir tableau suivant).

Nous présentons dans le tableau qui suit les profils d'accès aux zones logiques du DMP de tous les acteurs dans le système Carte Santé.

Zones logiques	Médecin	Pharmacien	Ambulancier	Infirmière	Infirmière en chef	Emetteur	Maintenance
1	LE	LE	L	LE	LE	LE	LE
2	LE	L	L	L	L	LE	LE
3	LE	L	L	L	L	LE	LE
4	LE	L	L	L	L	LE	LE
5	LE	LE	—	—	L	LE	LE
6	LE	LE	L	L	L	LE	LE
7	LE	LE	L	L	L	LE	LE
8	LE	LE	L	L	L	LE	LE
9	LE	L	L	LE	LE	LE	LE
10	LE	L	—	LE	LE	LE	LE
11	LE	—	—	—	—	LE	LE
12	LE	—	—	—	—	LE	LE
13	LE	—	—	—	—	LE	LE
14	LE	—	—	—	—	LE	LE
15	LE	—	—	—	—	LE	LE
16	LE	—	—	—	—	LE	LE
17	LE	—	—	—	—	LE	LE
18	LE	—	—	—	—	LE	LE
19	LE	—	—	—	—	LE	LE
20	LE	—	—	—	—	LE	LE
21	LE	—	—	—	—	LE	LE
22	LE	—	—	—	—	LE	LE
23	LE	—	—	—	—	LE	LE
24	LE	—	—	LE	LE	LE	LE
25	LE	—	—	LE	LE	LE	LE
26	LE	—	—	—	L	LE	LE
27	LE	—	—	—	L	LE	LE
28	LE	—	—	—	L	LE	LE
29	LE	—	—	—	L	LE	LE
30	LE	—	—	—	L	LE	LE
31	LE	—	—	—	L	LE	LE
32	LE	—	—	—	L	LE	LE
33	LE	—	—	—	L	LE	LE
34	LE	—	—	—	L	LE	LE
35	LE	—	—	—	L	LE	LE
36	LE	—	—	—	L	LE	LE

Table des profils d'accès aux zones logiques

2.5 Les types de données manipulées

Dans l'information manipulée dans le DMP, il y a six types de données: les booléens, les entiers, les réels, les caractères, les chaînes de caractères et les dates.

Les booléens: On distingue les booléens unaires et les chaînes de bits. Le booléen unaire est un bit qui a deux valeurs possibles: 1 ou 0. La chaîne de bits (bits stream) est une suite de bits unaires qui ont chacun une signification particulière.

Les entiers: Ils existent 17 entiers différents, chacun ayant une précision particulière qui dépend de sa taille en bits. Tous ces entiers sont positifs. Le tableau suivant résume tous les entiers avec leur taille en bits et leurs limites en précision.

Types d'entier	Nombre de bits	Entier maximum codé
entier_1	2	4
entier_2	3	8
entier_3	4	16
entier_4	5	32
entier_5	6	64
entier_6	7	128
entier_7	8	256
entier_8	9	512
entier_9	10	1 024
entier_10	11	2 048
entier_11	12	4 046
entier_12	13	8 192
entier_13	14	16 384
entier_14	15	32 768
entier_15	16	65 536
entier_16	17	131 072
entier_17	18	621 442

Les réels: Il existe dix réels différents. Ils sont tous positifs. Chaque réel a une précision particulière. La différence entre ces réels est le nombre de chiffres permis avant et après la virgule. Les réels positifs et négatifs sont codifiés de la même façon. Dans le cas des champs de type réel pouvant être positifs comme négatifs, le signe constitue alors un autre champ de type booléen. Le tableau suivant résume tous les réels avec leur taille en bits avant et après la virgule, ainsi que leurs limites en précision.

Types de réels	Nombre de chiffres avant la virgule	Nombre de chiffres après la virgule	Nombre de bits avant la virgule	Nombre de bits après la virgule	Nombre total de bits	Réel maximum codé
réel_1	1	1	4 b	4 b	8 b	± 9.9
réel_2	1	2	4 b	7 b	11 b	± 9.99
réel_3	1	3	4 b	10 b	14 b	± 9.999
réel_4	2	1	7 b	4 b	11 b	± 99.9
réel_5	2	2	7 b	7 b	14 b	± 99.99
réel_6	2	3	7 b	10 b	17 b	± 99.999
réel_7	3	1	10 b	4 b	14 b	± 999.9
réel_8	3	2	10 b	7 b	17 b	± 999.99
réel_9	3	3	10 b	10 b	20 b	± 999.999
réel_10	5	5	16 b	16 b	32 b	±65536.65536

Les caractères: Il y a quatre types de caractères différents, le premier est le BCD. Sur quatre bits, il permet de coder les chiffres de 0 à 9 et quelques autres caractères spéciaux. Le second est le ISO5 qui, sur cinq bits, permet de coder toutes les lettres

de l'alphabet en majuscule. Le troisième est le ISO6 qui code sur six bits les chiffres et les lettres de l'alphabet en majuscule ainsi que quelques caractères spéciaux. Le dernier type est le ASCII7 qui permet de coder toutes les lettres de l'alphabet en majuscule et en minuscule, les chiffres, ainsi que quelques caractères spéciaux sur sept bits. Les tables illustrant la codification de ces quatre types de caractères sont présentées à l'annexe B.

Les chaînes de caractères: On distingue les chaînes de taille fixe et les chaînes de taille variable. Les chaînes de taille fixe sont composées uniquement de caractères. Leur nombre est fixe et connu lors de la manipulation. Les chaînes de taille variable sont, par contre, précédées par un entier indiquant le nombre de caractères qui composent la chaîne. Le type de l'entier est généralement entier_5 codé sur six bits et permettant une taille de 64 caractères pour la chaîne.

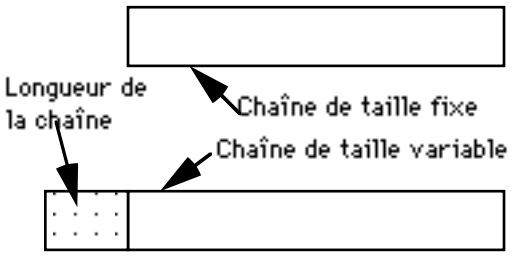


fig 2.16: Les chaînes de caractères

Les dates: On distingue les dates comportant uniquement le mois et l'année, celles comportant aussi le jour, les dates qui tiennent compte du siècle et enfin celles qui sont codées en nombre de jours après une date fixe. Sachant que 'j' est un bit pour coder le jour, 'm' un bit pour le mois, 'a' un bit pour l'année et 's' un bit pour le siècle, le tableau suivant résume les types de date avec leur codification.

Types de dates	Nombre de bits		Dates extrêmes codées
date_1	17	ssaaaaaammmmmjjjjj	du 1-01-1800 au 31-12-2027
date_2	16	aaaaaaaammmmmjjjjj	du 1-01-1900 au 31-12-2027
date_3	11	aaaaaaaammmm	du 01-1900 au 12-2027
date_4	10	jjjjjjjjjj	1024 jours à partir d'une date fixe

2.6 La structure physique du dossier médical

Pour minimiser le nombre de zones physiques possiblement présentes dans la mémoire de la carte, nous avons regroupé les données suivant les profils d'accès des zones logiques qui les contiennent. Cependant, pour ne pas avoir trop de données différentes à distinguer au sein d'une même zone physique, nous avons ménagé la minimisation du nombre de zones physiques. À partir des 40 zones logiques, nous avons relevé neuf profils d'accès au DMP différents. Ces profils d'accès sont des combinaisons des droits de lecture et des droits d'écriture des différents acteurs dans les zones logiques. Pour reconnaître et distinguer les différentes données dans une zone physique, nous avons rajouté à chaque groupe d'informations, liées sémantiquement, un identificateur. Cet identificateur a une taille qui dépend essentiellement du nombre de groupes de données dans la zone physique qui les contient. Pour ne pas avoir des identificateurs de taille trop importante, nous avons parfois limité le nombre de groupes de données dans une zone physique. De ces

contraintes et de ces neuf combinaisons de profils d'accès, nous avons défini 16 zones physiques pour le DMP.

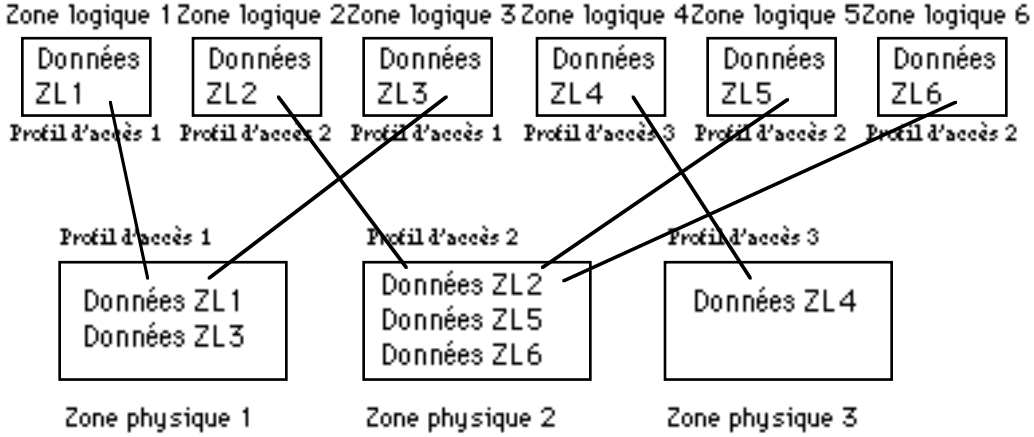


fig 2.17: Exemple de regroupement des données de zones logiques dans des zones physiques en tenant compte des profils d'accès

2.6.1 Les zones physiques et leur contenu

Dans ce qui suit, nous présentons les 15 zones physiques du DMP. Leur contenu respectif est divisé par groupe de données. Les données d'un même groupe se trouvent toujours ensemble sur la carte et sont toujours inscrites par la même transaction. Un groupe de données est inscrit dans la carte avec une date et une signature indiquant la date de l'inscription et l'identification de l'acteur ayant inscrit ces informations. La signature est en fait un indice d'une signature dans la table contenue dans la zone S2 (zone des signatures des prestataires). Nous ne présentons que les zones physiques avec la zones logique d'origine et l'ensemble de groupes qui les composent. Le profil d'accès aux zones physiques ainsi que le contenu détaillé de ces derniers se trouvent à l'annexe A.

Identification (zone physique 1): Cette zone comprend huit groupes d'informations distincts qui proviennent tous de la zone logique 1.

Urgence (zone physique 2): Cette zone comprend trois groupes d'informations distincts. Ils proviennent des zones logiques 2, 3 et 4.

Médication (zone physique 3): Cette zone ne comprend qu'un seul groupe d'informations. Il provient de la zone logique 5.

Intolérances médicamenteuses et allergies (zone physique 4): Cette zone comprend trois groupes d'informations distincts. Ils proviennent des zones logiques 6, 7 et 8.

Vaccination (zone physique 5): Cette zone ne comprend qu'un seul groupe d'informations. Il provient de la zone logique 9.

Suivi du poids (zone physique 6): Cette zone ne comprend qu'un seul groupe d'information. Il provient de la zone logique 10.

Suivi pédiatrique (zone physique 7): Cette zone comprend quatre groupes d'informations. Ils proviennent des zones logiques 24 et 25.

Suivi général et prévention (zone physique 8): Cette zone comprend quatre groupes d'informations. Il proviennent des zones logiques 11, 15, 16 et 23.

Autre suivis (zone physique 9): Cette zone comprend quatre groupes d'informations. Ils proviennent respectivement des zones logiques 18, 19, 21 et 22.

Variables de suivi (zone physique 10): Cette zone comprend des variables de différents types. L'information peut provenir de différentes zones logiques, plus particulièrement des zones logiques 18, 19, 21 et 22. On peut aussi y mettre des variables n'appartenant actuellement à aucune zone.

Antécédents et créatinine (zone physique 11): Cette zone comprend quatre groupes d'informations distincts. Ils proviennent des zones logiques 12, 13, 14 et 20.

Ophtalmologie (zone physique 12): Cette zone comprend quatre groupes d'information. Ils proviennent tous de la zone logique 17.

Résumé des grossesses antérieures (zone physique 13): Cette zone comprend quatre groupes d'informations. Ils proviennent des zones logiques 26, 28 et 32.

Suivi de grossesse et visites prénatales (zone physique 14): Cette zone comprend deux groupes d'informations. Ils proviennent des zones logiques 27 et 33.

Résultats de tests pour grossesse (zone logique 15): Cette zone comprend sept groupes d'informations. Ils proviennent des zones logiques 29, 30, 31, 34 et 35.

La description détaillée des ce zones est donnée à l'annexe A.

2.6.2 Les catégories de bénéficiaires et les zones physiques

Comme les zones logiques ne se trouvent pas nécessairement toutes dans l'ensemble des Cartes Santé, les zones physiques ne se trouvent pas aussi toutes, par conséquent, dans toutes les Cartes Santé quelles que soient les catégories de bénéficiaire correspondantes.

Dans ce qui suit, nous présentons les catégories de bénéficiaires et les zones physiques présentes nécessairement ou probablement dans leurs cartes.

Les nouveau-nés (0 à 24 mois):
Zones physiques obligatoires: 1, 2, 3, 4, 5, 6 ,7 et 11.
Zones physiques probables : 8, 9, 10 et 12.

Les personnes âgées (60 ans et plus):
Zones physiques obligatoires: 1, 2, 3, 4, 8, 9, 10 et 11..
Zones physiques probables : 5, 6 et 12.

Les femmes enceintes:
Zones physiques obligatoires: 1, 2, 3, 4, 8, 11, 13, 14, 15 et 16.
Zonesphysiques probables : 5, 6, 9, 10 et 12.

Les autres volontaires:
Zones physiques obligatoires: 1, 2, 3, 4, 8 et 11.
Zones physiques probables : 5, 6, 9, 10 et 12.

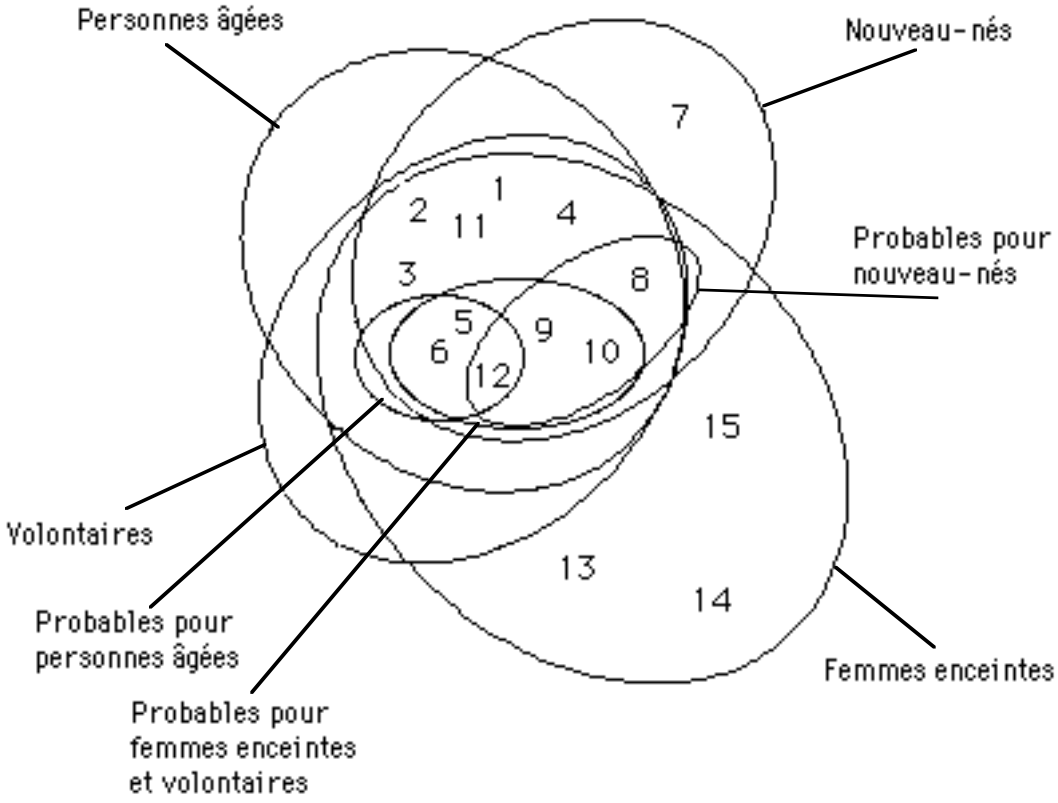


fig 2.18: Les catégories de bénéficiaires et les zones physiques

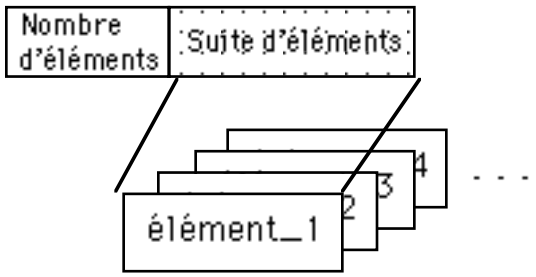
2.6.3 La représentation physique des données dans les zones physiques

Chaque groupe d'informations codé est accompagné d'une signature et d'une date. Le groupe est identifié au sein d'une zone au moyen d'un identificateur.

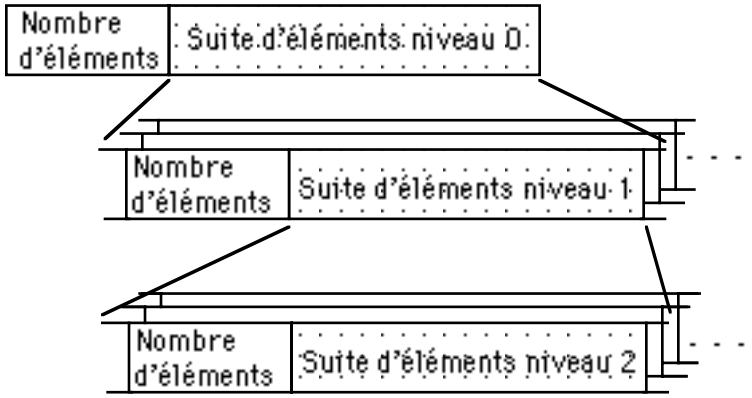
Identif-icateur	Groupe d'informations	Signature	Date

Le groupe d'informations est codé en utilisant une codification optimale (voir annexe B). Chaque groupe d'informations est constitué de différents champs d'information.

Un champ a un type de données bien précis et est codifié pour utiliser le moins d'espace possible (voir 2.5: les types de données manipulées). Un champ peut aussi représenter une suite d'éléments de même nature, comme une suite de médicaments ou une suite de vaccins. Dans ces cas, le champ est constitué de deux parties: l'une pour indiquer le nombre d'éléments de la suite et le second pour représenter les éléments.



De la même façon un élément d'une suite peut être constitué de différents champs d'information. Un champ d'un élément de suite peut aussi être une suite à son tour. Dans le DMP, on retrouve une récursion de ce type à trois niveaux de profondeur.



La description détaillée de la représentation des champs dans les zones physiques ainsi que la taille nécessaire pour le stockage de chacun de ces champs sont données à l'annexe B.

2.7 La correspondance logique-physique

Il est important de faire la correspondance entre les zones logiques et les zones physique. Les zones logiques reflètent les vues que peuvent avoir les applications sur les données d'une carte, alors que les zones physiques sont la vraie représentation physique des données sur la carte. Puisque les applications n'ont pas d'informations sur les zones physiques et n'en connaissent même pas l'existence, c'est la Coquille qui se charge de faire le lien entre les données logiques et leurs représentations physiques sur la carte. Ce lien se réalise au moyen d'une table de correspondance où l'on retrouve toutes les variables, leur zone logique d'origine et leur zone physique de destination. C'est ce qui est appelé «Mapping» ou correspondance Logique-Physique.

Le contenu d'une zone physique peut provenir d'une ou plusieurs zones logiques. Par contre, le contenu d'une zone logique n'est inscrit que dans une seule zone physique et n'est jamais divisé pour aller dans plusieurs zones (voir figure 2.9).

Dans ce qui suit, nous présentons des tables de correspondance où l'on ne retrouve que les zones logiques et physiques. Pour des raisons de simplification, les variables logiques ne figurent pas dans ces tables. En fait, toutes les variables d'une zone logique vont automatiquement dans la zone physique qui correspond à cette dernière, même si ces variables sont subdivisées en groupes distincts.

2.7.1 La carte d'habilitation

Comme nous l'avons présenté au paragraphe 2.2.4.1, il y a huit zones logiques dans une carte d'habilitation. Elles sont représentées par six zones physiques sur la carte. La correspondance est donnée par la table suivante:

Physique	Logique
Zone physique **H1**: Identification	Zone logique **H1**: Identification du projet
	Zone logique **H2**: Identification personnelle
Zone physique **H2**: Secret	Zone logique **H3**: Zone secrète
Zone physique **H3**: Paramétrisations	Zone logique **H4**: Liste des paramétrisations
Zone physique **H4**: Gestion de Paramétrisations	Zone logique **H5**: Paramétrisations annulées
	Zone logique **H6**: Dernière paramétrisation
Zone physique **H5**: Statistiques	Zone logique **H7**: Statistiques
Zone physique **H6**: Contrôles	Zone logique **H8**: Zone de contrôle

2.7.2 La Carte Santé

2.7.2.1 Les zones permanentes

Les six zones logiques permanentes de la Carte Santé sont réduites à quatre zones physiques tel que le montre la table suivante:

Physique	Logique
Zone physique **1**: Identification	Zone logique **S1**: Identification du projet
	Zone logique **1**: Identification personnelle
Zone physique **S2**: Table des signatures	Zone logique **S2**: Signatures des prestataires
Zone physique **S3**: Statistiques	Zone logique **S3**: Statistiques
Zone physique **S4**: Contrôles	Zone logique **S4**: Zone de contrôle
	Zone logique **S5**: Structure

2.7.2.2 Les zones du DMP

La correspondance entre les zones logiques du DMP et ses zones physiques est donnée dans la table suivante:

Physique	Logique
Zone Physique 1	L1
Zone physique 2	L2, L3 et L4
Zone physique 3	L5
Zone physique 4	L6, L7 et L8
Zone physique 5	L9
Zone physique 6	L10
Zone physique 7	L24 et L25
Zone physique 8	L11, L15, L16 et L23
Zone physique 9	L18, L19, L21et L22
Zone physique 10	L36
Zone physique 11	L12, L13, L14 et L20
Zone physique 12	L17
Zone physique 13	L26, L28 et L32
Zone physique 14	L27 et L33
Zone physique 15	L29, L30, L31, L34 et L35

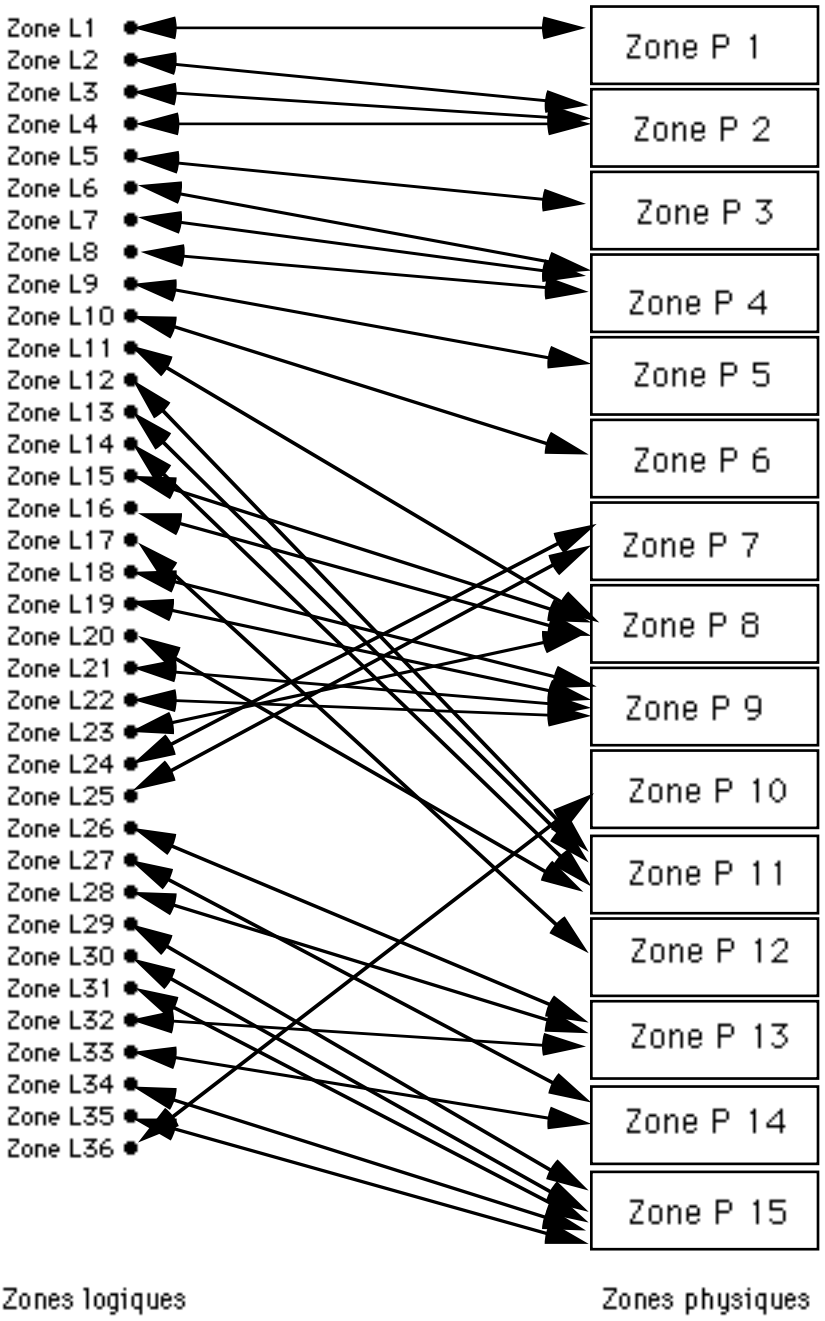


Schéma des correspondances
entre les zones logiques et les zones physiques

Le mécanisme de correspondance qui utilise ces tables, appelé «mapping», est présenté au paragraphe 5.3.2 du chapitre V.

CHAPITRE III

Simulation

L'implantation de la Carte Santé est associée à divers problèmes comme la perte de la carte et la saturation. L'un des facteurs critiques de cette implantation est une gestion optimale de la mémoire pour résoudre, prévenir ou alléger le problème de la saturation qui apparaît comme inévitable. Les politiques de gestion de mémoire (voir chapitre I) sont tributaires des possibilités offertes par la technologie utilisée. Cette technologie influence directement le phénomène de saturation par la limite de capacité de la mémoire qu'elle offre.

3.1 Les problèmes de saturation et de perte de carte

La perte d'une carte est un problème qui se traduit par un recouvrement des informations contenues dans l'ancienne carte. Lorsqu'une carte est perdue, une nouvelle est réémise et est soit initialisée par l'information élémentaire détenue par l'émetteur, soit chargée par l'information détenue par les prestataires de services de santé. Dans le second cas, plusieurs scénarios sont possibles. On peut citer, par exemple, la sauvegarde systématique du contenu de la carte par chaque prestataire qui utilise la Carte Santé. La dernière version de la carte se trouve ainsi toujours chez le dernier prestataire de santé l'ayant utilisée. Un autre scénario consiste en une sauvegarde des transactions locales pertinentes pour un prestataire et une reconstitution du dossier à travers la suite des prestataires visités par le porteur, cette dernière étant obtenue par le système de facturation de l'émetteur.

La saturation est un problème inévitable découlant directement de la capacité de stockage limitée de la carte intelligente. Ce problème est posé à cause de l'accumulation continue des transactions médicales. Lorsqu'une saturation survient, la carte peut être épurée. L'épuration de l'information de la carte consiste à éliminer les

informations obsolètes et récupérer l'espace libéré pour d'autres transactions (voir 1.6.2: politiques de récupération). Un autre scénario consiste à réémettre une nouvelle carte vide et à la joindre à la première comme suite de celle-ci. Un bénéficiaire serait alors porteur de plusieurs cartes santé. La nouvelle carte peut aussi contenir un sommaire de la précédente et remplacer cette dernière. Le bénéficiaire n'aurait alors qu'une seule carte santé.

La simulation que nous avons réalisée s'intéresse essentiellement à la saturation et l'évalue à l'aide d'un modèle théorique.

3.2 Les buts de la simulation

La simulation ne s'intéresse qu'au phénomène de saturation et vise essentiellement quatre buts. Le premier but consiste à évaluer la longévité d'une carte. L'objectif est de déterminer la durée de vie moyenne de la phase d'exploitation dans le cycle de vie de la carte. On s'intéresse alors à savoir combien de temps met une carte, avec une configuration donnée, pour se remplir. Ceci permet de savoir le pourcentage de cartes, dans un échantillon donné, qui peuvent survivre durant la durée du projet (18 mois). Le second but est d'étudier l'impact du choix d'une politique d'allocation de mémoire et d'une politique de résolution du problème de la saturation. Le troisième but consiste à comparer les différentes technologies et mettre en évidence celle qui permet d'avoir le moins de situations de saturations possibles, puisque chaque technologie offre des caractéristiques différentes permettant des longévités distinctes. Enfin, le quatrième but vise à déterminer la taille optimale des quantum d'espace alloués aux différentes zones. En effet, et surtout avec les allocations statiques, lorsqu'une zone est saturée, il peut arriver que la carte entière soit saturée bien qu'il y ait encore de l'espace libre alloué pour une ou plusieurs autres zones. La simulation permet alors d'estimer la meilleure taille à allouer pour les zones relativement au niveau d'activité de chacune d'elles.

Avec la simulation, on peut déceler des problèmes de saturation d'une ampleur insoupçonnée. Une fois ces problèmes dévoilés, on peut être amené à modifier certains éléments de la conception du DMP physique et aller jusqu'à remettre en cause la pertinence de certaines informations portées sur la carte. C'est ce qui fait d'ailleurs l'importance de la simulation avant l'implantation.

3.3 Description des données du problème

Les données du problème proviennent de différentes sphères. Les technologies de cartes intelligentes disposent de caractéristiques variées et réagissent différemment à la saturation. Les DMP des différentes catégories de bénéficiaires ne reçoivent pas les mêmes transactions avec la même fréquence et dans les mêmes zones. Les acteurs n'accèdent pas aux mêmes données. Les politiques de gestion de mémoire et de récupération d'espace ne résistent pas et ne luttent pas de la même façon contre le phénomène de saturation. Toutes ces variables sont prises en compte dans la simulation.

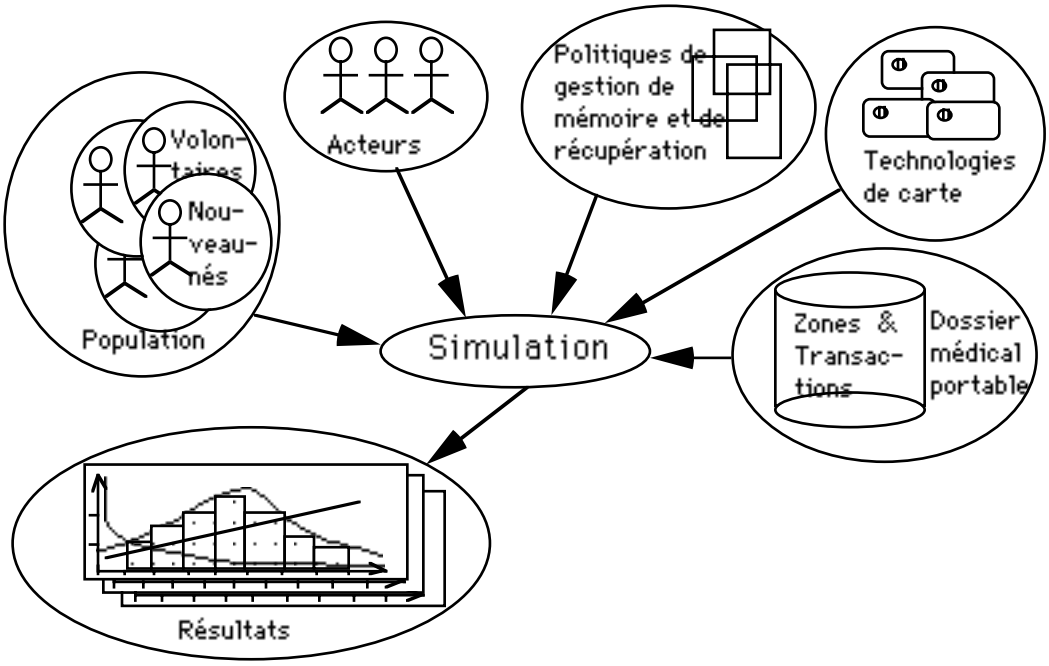


fig 3.1: Système de simulation du problème de saturation

3.3.1 Populations visées

On distingue les populations de bénéficiaires et les populations de technologies de cartes intelligentes. Les bénéficiaires étant constitués de populations hétérogènes, nous avons départagé les populations dans des simulations indépendantes. Ainsi, les nouveau-nés, les personnes âgées, les femmes enceintes et les volontaires de St-Fabien sont distingués par la taille de leurs transactions respectives et leurs fréquences. Une simulation pour chaque population est faite séparément. Les résultats que nous présentons dans le paragraphe 3.5 sont cependant ceux obtenus avec les volontaires de St-Fabien.

Les technologies testées sont au nombre de quatre (4): la carte IBM, la carte MP de Bull, la carte MCOS de Gemplus et la carte TOSMART-A de Toshiba. L'utilisation de chacune a été simulée séparément puisque chacune d'elles a des caractéristiques propres, comme la taille du descripteur de bloc, la taille réellement utilisable et les possibilités de gestion.

3.3.2 Le dossier-patient utilisé

Le dossier-patient utilisé dans cette simulation est très limité et ne comporte que sept (7) zones:
- zone identité;
- zone urgence;
- zone allergies;
- zone vaccins;
- zone antécédents;
- zone médication et
- zone suivie.

Trois (3) acteurs agissent sur ces zones à savoir le médecin, le pharmacien et le professionnel en urgence. Il y donc trois (3) transactions possibles, chacune pouvant toucher à une ou plusieurs zones à la fois. La taille des données inscrites par une transaction est connue et fixée à une moyenne. Cette restriction, par rapport au vrai dossier et au vrai nombre d'acteurs décrits au chapitre II, vient du fait que la simulation a été faite bien avant la définition finale du contenu du DMP. Le DMP a beaucoup évolué pendant et après la simulation. Les résultats peuvent néanmoins nous donner une idée globale sur le phénomène de saturation et sur la réaction des technologies.

3.4 Le modèle de simulation

Le modèle est spécifié sans référence explicite à une technologie de carte intelligente en particulier, ni à un nombre de zones, de transactions ou d'acteurs. Il est paramétré afin de permettre son adaptation facile à diverses situations.

3.4.1 Le modèle

Le modèle traite une carte donnée sur une durée de temps divisée en intervalles identiques appelés «unité de temps» (Δt). Chaque carte est associée à une catégorie qui est caractérisée par un intervalle noté $[NT_{min}, NT_{max}]$. Cet intervalle, délimité par un nombre minimum et un nombre maximum de transactions possibles au cours d'une unité de temps, «catégorise» en quelque sorte le bénéficiaire de la carte. On suppose qu'une carte reste toujours dans la même catégorie, c'est-à-dire que le bénéficiaire de la carte ne change pas d'habitude en ce qui concerne les visites qu'il effectue chez les professionnels de la santé. L'intervalle $[NT_{min}, NT_{max}]$ est calculé pour chaque carte à partir d'une moyenne $\boldsymbol{\mu}$ d'une distribution poissonnienne de la façon suivante:

u_1=uniforme(0,1)
$\lambda 1$= -Log(1-u_1) * μ
u_2=uniforme(0,1)
$\lambda 2$= -Log(1-u_2) * μ
NT_{min} = min($\lambda 1$,$\lambda 2$)
NT_{max} = max($\lambda 1$,$\lambda 2$)

La moyenne μ de la loi exponentielle utilisée, l'unité de temps, le nombre de transactions différentes (**NBT**), le nombre de zones (**NBZ**) ainsi que le pas de décrémentation des pourcentages de récupération (**PDP**) (voir 1.6.2: politiques de récupération) sont des paramètres du modèle donnés aux simulateurs. Les paramètres sont regroupés en trois ensembles: les caractéristiques de la carte, le dossier médical portable et les données générales de la simulation.

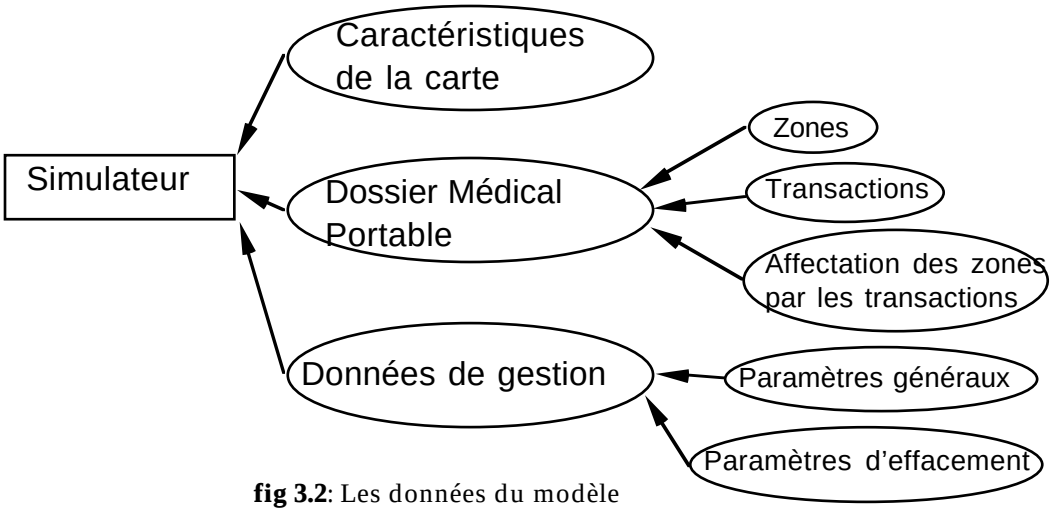


fig 3.2: Les données du modèle

Les caractéristiques de la carte: Elles englobent la taille de la mémoire de la carte utilisable (**TUC**), la taille d'un descripteur de zone (**TD**), la taille maximum d'un bloc physique (**TMBP**), l'unité de mémoire minimale manipulable (**UM**), le type de la mémoire EEPROM ou EPROM et des informations sur les possibilités d'allocation dynamique, d'effacement d'informations et de récupération de blocs.

Le dossier médical portable: Il est composé de trois sous-ensembles: un premier pour décrire les transactions, un second pour décrire les zones et un dernier pour faire la correspondance entre les transactions et les zones qu'elles touchent. Chaque transaction T_i est décrite avec un pourcentage indiquant la probabilité P_i que cette transaction se produise dans l'intervalle Δt. P_i est en fait une probabilité relative indiquant le nombre d'occurrences de T_i sur un ensemble de 100 transactions produites dans l'intervalle Δt. La procédure permettant de choisir l'identité de chaque transaction dans un ensemble de **T** transactions différentes est la suivante:

```
/*Trier dans le sens décroissant l'ensemble des probabilités des transactions*/
Probabilités=trier(décroissant,{P1,P2,...,Pi,...,PNBT})
Pour i de 1 à T
faire
 /*Générer avec une distribution uniforme un nombre entre 0 et 100%*/
 u=uniforme(0,100)
 /*chercher dans Probabilités la première probabilité inférieure à u*/
 j=1
 tant que (u>Probabilités[j])
 faire
   j = j + 1
 fait
 /*sélectionner la transaction Tj comme étant la ième transaction*/
fait
```

Dépendamment de son type, une transaction peut affecter une ou plusieurs zones. La

transaction est alors associée à un intervalle de zones noté [$NZmin_i$, $NZmax_i$] indiquant le nombre minimum et maximum de zones affectées par la transaction T_i. Une dernière information, l'autorisation d'effacement multi-zones (AE_i), indique si l'auteur de la transaction a le droit de faire des effacements dans plusieurs zones, dans le cas où la transaction se traduit par une saturation. Les zones sont caractérisées par un intervalle noté [Bit_{min}, Bit_{max}] indiquant le nombre minimum et maximum de bits utilisés par une transaction dans cette zone. Avec la description d'une zone Z_i, on retrouve aussi la taille d'un bloc physique de cette zone (TB_i) et la taille de l'information initiale dans Z_i avant exploitation. Le dernier sous-ensemble intéresse l'affectation des zones par les transactions. On y retrouve pour chaque transaction la probabilité que cette dernière affecte chacune des zones. C'est une matrice de n lignes et k colonnes où n est le nombre de transactions et k le nombre de zones. P_{ij} est le pourcentage d'affectation de la transaction T_i à la zone Z_j, par rapport à l'ensemble des affectations de T_i sur toutes les zones. Si P_{ij}=0, alors la transaction T_i n'affecte pas la zone Z_i.

Les données générales de la simulation: Elles englobent les données générales telles que Δt,μ, NBT, NBZ, PDP, la politique d'allocation de mémoire choisie, la politique de récupération choisie, et les données de gestion de l'effacement et de récupération d'espace. Pour cela, on dispose du pourcentage d'effacement et de récupération possible en cas de saturation, pour chaque zone. Une matrice indique les zones épurables en cas de saturation de chaque zone du DMP.

Six distributions sont utilisées dans notre modèle. La consommation des soins suit une loi exponentielle. Le nombre de transactions, le nombre des zones touchées par une transaction ainsi que le nombre de bits consommés dans une zone par une transaction sont calculés selon une distribution uniforme. L'identification des transactions ainsi que l'identification des zones touchées suivent des lois de probabilités variables.

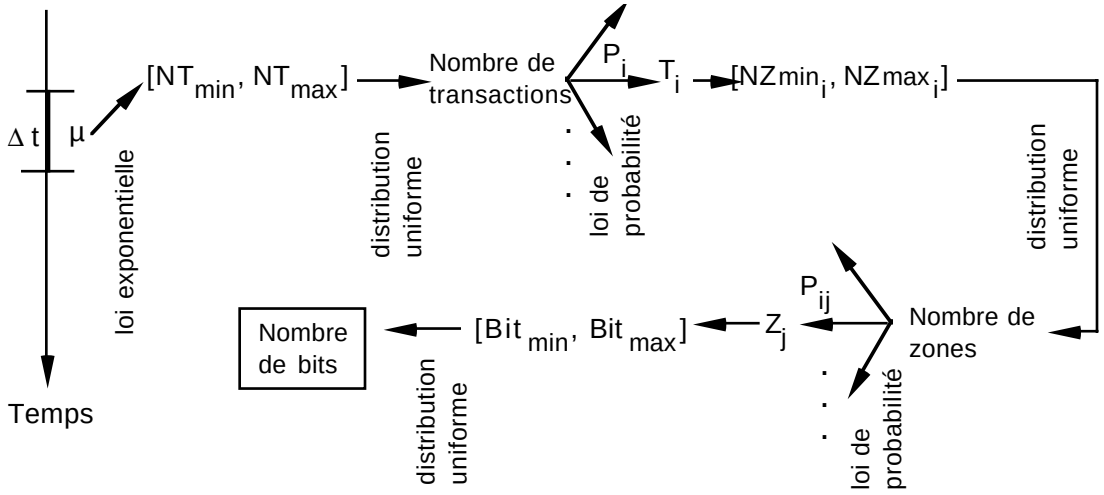


fig 3.3: Schématisation du processus de remplissage simulé d'une carte

3.4.2 L'implantation du simulateur

Le programme de simulation a été écrit en langage C. Toutes les données et les paramètres du modèle ont été mis, autant que possible, dans des fichiers externes au programme pour permettre une modification aisée des paramètres sans avoir à modifier le programme. Ces fichiers sont en ASCII, facilement lisible avec un éditeur de texte. Leur nombre est de sept (7). Le premier fichier (fichier de simulation) contient l'identificateur de la politique de gestion de mémoire désirée pour la simulation, l'identificateur de la politique de résolution de la saturation choisie, ainsi que le nom des six (6) autres fichiers concernant respectivement l'information générale, l'information sur la technologie, l'information sur les zones, l'information sur les transactions, la matrice d'affectation des zones / transactions et l'information sur les effacements et récupérations. Le second fichier (information générale) contient les variables suivantes:

- Unité de temps (Δ**t**)
- Moyenne des transactions (**μ**)
- Nombre de transactions différentes (**NBT**)
- Nombre de zones différentes (**NBZ**)
- Pas de décrémentation des pourcentages (**PDP**)

Le troisième fichier (information sur la technologie) contient les caractéristiques d'une technologie de carte intelligente donnée à savoir: son nom qui va servir de libellé pour les graphiques, la taille totale de la carte, la taille utilisable en bits (**TUC**), la taille d'un descripteur de bloc (**TD**), la taille maximale d'un bloc physique (**TMBP**) ainsi que l'unité minimale (bit, octet ou mot) (**UM**), le type de la mémoire; EEPROM ou EPROM ainsi que trois booléens indiquant la possibilité d'effacement, de récupération et d'allocation dynamique. Le quatrième fichier (information sur les zones) contient les caractéristiques des différentes zones à savoir: le libellé de la zone, la taille d'un bloc physique appartenant à la zone, la taille de l'information initiale dans la zone avant la simulation, le nombre minimum et le nombre maximum de bits utilisés par une transaction et le nombre de blocs physiques à allouer dans le cas des allocations statiques. Le cinquième fichier (information sur les transactions) contient les différentes transactions possibles avec leurs caractéristiques comme le libellé de la transaction, la probabilité préférentielle de classement dans l'ensemble des transactions, l'indicateur d'autorisation d'effacement multizones de l'acteur et le nombre minimum et maximum de zones affectées par la transaction. Le sixième fichier (affectation zones / transactions) contient une matrice indiquant pour chaque transaction la probabilité d'affecter chacune des zones et, pour chaque zone, la probabilité d'être affectée par chacune des transactions. Le dernier fichier (effacements et récupérations) contient une matrice et un vecteur. La matrice indique les zones possiblement épurées à la suite de la saturation d'une zone donnée. Lorsqu'un élément Mij de la matrice est égal à 1, cela signifie que si la zone i est saturée, il est possible de faire des effacements dans la zone j. Le vecteur, pour sa part, contient les pourcentages d'effacement et de récupération dans les zones. Lorsqu'une zone est épurée, son pourcentage d'effacement est alors décrémenté par PDP pour qu'à la prochaine épuration, moins d'informations soit effacées.

La structure générale de l'algorithme de notre simulateur est la suivante:

```
initialisation des variables globales
lecture des paramètres dans les fichiers externes
lecture de Nombre_de_Cartes
pour i de 1 à Nombre_de_Cartes
faire
    initialisation des variables locales pour une carte
    NT_min = Expon(μ)
    NT_max = Expon(μ)
    CarteSaturée = Faux
    Unité_de_Temps = 0
    Tantque (CarteSaturée = Faux)
    faire
        Nombre_de_Transactions = Uniforme(NT_min, NT_max)
        pour j de 1 à Nombre_de_Transactions
        faire
            déterminer la nature de la transaction T_j suivant {P_1,P_2,...,P_j,...,P_NBT}
            Nombre_de_Zones = Uniforme(NZmin_j, NZmax_j)
            pour k de 1 à Nombre_de_Zones
            faire
                déterminer la nature de la zone Z_k suivant {P_j1,P_j2,...,P_jk,...,P_jNBZ}
                Nombre_de_bits = Uniforme(bitmin_k, bitmax_k)
                libre = consommer_dans_Zone(Z_k,Nombre_de_bits)
                si (libre = Faux)
                alors
                    RécupérerEspace()
                    libre = consommer_dans_Zone(Z_k,Nombre_de_bits)
                finsi
                si (libre = Faux)
                alors
                    CarteSaturée = Vrai
                finsi
            fait
        fait
        Unité_de_Temps = Unité_de_Temps + 1
    fait
fait
```

La fonction *consommer_dans_Zone*(z,Nb) consomme Nb bits dans la zone z et alloue des espaces-mémoire tel que le dicte la politique de gestion de mémoire demandée. Elle remet une variable booléenne indiquant si la consommation a pu se faire.

La fonction *RécupérerEspace*() récupère de l'espace en tenant compte de la politique de récupération choisie et des paramètres d'effacement et de récupération.

3.5 Les résultats de la simulation

La simulation a été réalisée sur un micro-ordinateur de type 80386 à 20 MHz. Nous avons choisi une moyenne **µ** de 3.2 et des probabilités pour les transactions médicales, pharmaceutiques et d'urgence, respectivement égales à 30%, 65% et 5%. Nous n'avons pas permis l'écriture au personnel d'urgence. Pour les pharmaciens, seule la zone de médication leur a été laissée accessible en écriture alors que pour les médecins, une transaction touche dans 5% des cas à la zone urgence, 10% à la zone allergies, 50 % à la zone vaccins et enfin, 80% à la zone suivie. Les simulations ont été faites sur un échantillon de 10 000 cartes avec des tailles de zones et de données bien choisies.

Pour comparer les différentes politiques d'allocation de mémoire et leur répercussion sur le cycle de vie de la carte, nous avons fait des tests sur la carte IBM car c'est la seule carte acceptant toutes les politiques d'allocation et d'effacement. Dans la table qui suit, on retrouve la durée de vie moyenne pour toutes les politiques d'allocation de mémoire, sauf ADTD. Faute de temps, cette dernière n'a pas été simulée. Chaque politique d'allocation a été testée soit avec la politique de l'autruche (PE), soit avec l'effacement dans une unique zone (EMZ). La carte IBM a normalement 64 Kb de mémoire, mais nous l'avons aussi testée avec une taille de 32 Kb. Nous avons défini un ratio pour connaître le pourcentage de l'espace utilisable réellement utilisé durant le cycle de vie de la carte. Dans le tableau qui suit, le premier chiffre de chaque case est une durée moyenne en nombre de mois et le second chiffre est le pourcentage moyen de remplissage de l'espace utilisable lors de la saturation finale. Le test a chaque fois été fait sur un échantillon de 10 000 cartes.

Durée du cycle de vie de la carte en mois avec la technologie IBM

	ASTF		ASTV		ADTF		ADTV		ADGZ	
	PE	EMZ	PE	EMZ	PE	EMZ	PE	EMZ	PE	DGZ
64 Kbits	22.1 m 27.0%	50.5 m 34.2%	64.2 m 78.3%	101.5m 80.1%	63.9 m 78.2%	83.7 m 71.9%	64.7 m 80.0%	104.8m 81,7%	74.9 m 91.0%	147.4 92.2%
32 Kbits	9.3 m 25.8%	20.5 m 33.7%	25.7 m 71.9%	35.9 m 75.7%	24.9 m 69.3%	37.9 m 68.5%	26.2 m 73.6%	39.5 m 72.1%	32.0 m 88.7%	61.5 m 90.5%

Cette table montre que la durée de vie de la carte croît avec la taille de la mémoire, ce qui n'est pas surprenant. La durée de vie de la carte est proportionnelle à la taille de la mémoire disponible. On voit aussi que les politiques d'allocation dynamique donnent de meilleurs résultats que les politiques d'allocation statique. Le gonflage de zone (ADGZ) permet d'avoir la plus longue durée de vie. On obtient aussi de meilleurs résultats en faisant varier la taille des blocs associés aux différentes zones. On est donc pénalisé en fixant la taille des blocs. En effet, chaque zone a des besoins différents en espace de stockage suivant l'information qu'on y met. En jouant sur cette variation , on peut facilement déterminer la taille optimale pour les blocs de chaque zone. C'est ce qui fait que la différence entre ASTV et ADTV n'est pas très grande puisque les tailles des blocs ont été optimisées et calibrées après une suite de simulations d'essai.

Enfin, on remarque que la durée de vie est sensiblement allongée avec une politique d'effacement. Si l'effacement n'est pas possible, on peut améliorer la durée de vie de la carte en diminuant la taille moyenne des transactions, ce qui équivaut à de la compression des données.

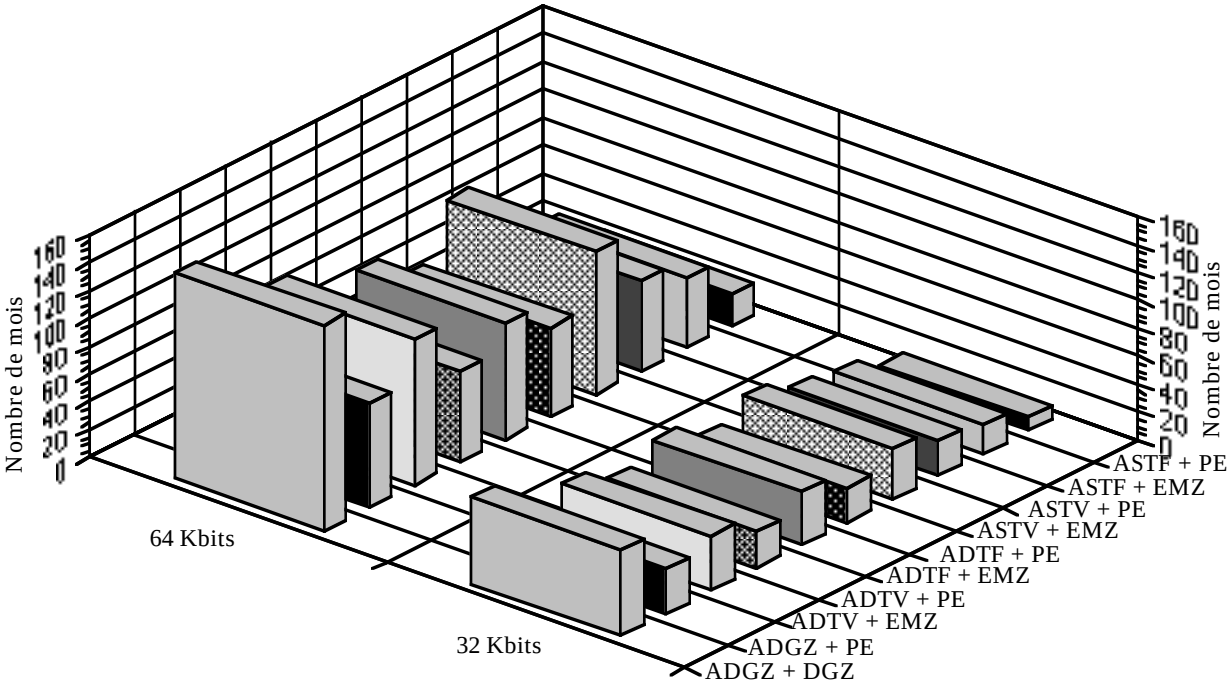


fig 3.4: Durée de vie versus la politique de gestion de mémoire et la taille de la carte

Pour comparer les différentes technologies, nous avons simulé chacune d'elles avec la meilleure politique d'allocation de mémoire permise. Le tableau suivant montre les résultats obtenus. Dans le tableau, on retrouve la moyenne du ratio indiquant le pourcentage de l'espace utilisable de la carte réellement utilisé lors de la saturation pour l'échantillon de 10 000 cartes.

Classification des technologies

				Durée de vie en mois		Ratio d'utilisation	
Carte	Taille	Mémoire	P. Allocation	Moyenne	MIN, MAX, Ecart type	Moyenne	MIN, MAX, Ecart type
IBM	64 Kb	EEPROM	ADGF	74.9 m	9, 327, 46.5	91.0%	89.5,91.6, 2.3
MP Bull	64 Kb	EPROM	ADTV	70.5 m	9, 322, 48.0	88.1%	82.4,94.1, 1.3
TOSMART-A	64 Kb	EEPROM	ASTV	63.3 m	8, 284, 46.1	83.7%	62.3,95.0, 4.8
MCOS Gemplus	32 Kb	EEPROM	ADTV	39.0 m	5, 189, 34.0	86.5%	79.6,92.0, 2.8

Handicapée par sa taille, la carte de Gemplus donne de mauvais résultats par rapport aux autres. La Tosmart-A ne donne pas de résultats intéressants malgré sa taille, à cause du fait qu'elle ne permet pas les politiques d'allocation dynamique. La carte d'IBM donne les meilleurs résultats grâce au gonflage de zones. Avec une politique

d'effacement, toutes ces technologies permettent d'avoir de meilleurs résultats que ceux de la table précédente. La carte MP de Bull ne permet cependant pas l'effacement puisqu'elle dispose d'une mémoire EPROM.

En supposant une carte EEPROM de 64 Kb d'une technologie quelconque, l'étude de saturation sur un échantillon de 10 000 individus donne les résultats suivants:

Politique	Durée de vie moyenne	Durée de vie minimum et maximum	Pourcentage de cartes saturées avant 20 mois	Nombre de transactions médecin	Nombre de transactions pharmacie	Nombre de transactions urgence
ADTV	64.7 m	9 .. 296 m	11.80%	32.5	71.1	4.9
ADGZ	74.9 m	9 .. 327 m	7.14%	36.9	80.7	5.5

Les données du tableau ci-dessus nous permettent de faire des pronostics optimistes. En effet, la durée moyenne que l'on obtient dépasse largement la durée du projet. Avec une politique ADGZ et une mémoire de 64 Kbits, moins de 8% des bénéficiaires ont besoin de plus d'une carte durant le projet. Il demeure cependant que les données utilisées ne correspondent pas au vrai DMP.

La figure 3.5 nous donne une idée de la durée de vie de l'ensemble des cartes. On remarque que dans la majorité des cas, les cartes ont une durée de vie relativement courte et que certaines ont une durée de vie qui se prolonge indéfiniment. La courbe a une allure logarithmique et montre qu'il y a des bénéficiaires qui utilisent excessivement leur carte alors que d'autres ne l'utilisent guère.

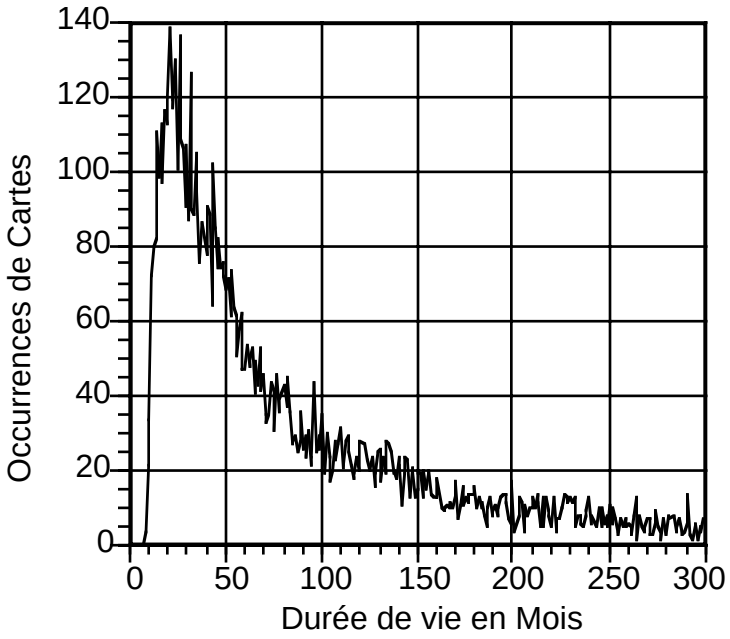


fig 3.5: Distribution des cartes en fonction de la durée de vie avec la ADTV

On peut supposer qu'avec un DMP tel que défini dans le chapitre II, la courbe ne changerait pas d'allure mais que l'axe du temps se rétrécirait.

Ces simulations montrent clairement qu'il peut y avoir des situations problématiques à prendre en considération dépendamment de l'espace et du type de mémoire de la carte, des politiques de gestion utilisées et de la structure des données emmagasinées. Bien que non basés sur des données réelles, les résultats montrent qu'avec un DMP tel qu'utilisé, le cycle de vie moyen des cartes est suffisamment long pour couvrir la période du projet estimée à 18 mois.

3.6 Les améliorations possibles à apporter

Dans notre simulateur, nous avons fixé le temps et fait varier le nombre de transactions dans une unité temporelle. Une deuxième façon de procéder serait de faire varier le temps avec une distribution estimée exponentielle. En effet, un bénéficiaire ne consulte pas à des intervalles réguliers. L'algorithme général serait:

```
temps = 0
tantque carte non saturée
faire
      durée = Expon(μ)
      temps = temps + durée
      consultation()
fait
```

Faute d'informations réelles, nous avons été contraints de faire des hypothèses quant à la répartition poissonnienne de la consommation des soins, la conservation des habitudes de consultation par un bénéficiaire, l'indépendance des transactions les unes par rapport aux autres, l'uniformité des distributions du nombre de bits consommés par zone et le nombre de zones touchées par transaction. Ces hypothèses font que notre modèle n'est pas vraiment réel. Il s'approche néanmoins de la réalité même si les probabilités, les intervalles et la moyenne de la loi exponentielle ont été choisis arbitrairement.

Pour faire de meilleures simulations, il faudrait disposer de statistiques des services professionnels de soins effectués sur le site du projet. Avec un DMP mis à jour, on pourrait alors simuler le remplissage des cartes à partir de données réelles relevées.

Le projet pilote de Rimouski peut aussi être une occasion de relever des données statistiques pour des fins de simulation avant une généralisation au niveau de toute la province. Dans ce cas, les données intéressantes à retenir sont:

- le nombre de transactions d'écriture par consultation;
- le nombre de zones touchées par chaque transaction d'écriture et;
- le nombre de bits écrits par chaque transaction dans chaque zone touchée.

Ces données nous éviteraient d'émettre des hypothèses quant aux différentes distributions.

CHAPITRE IV

Compactage et compression des données

De larges quantités d'informations sont traitées quotidiennement. Plusieurs applications informatiques traitent et emmagasinent de larges volumes d'informations qui prennent beaucoup de place sur les supports de stockage. En parallèle, la prolifération des réseaux de communication et les applications de télétraitement engendrent un transfert massif d'informations sur des liens de communication à longue distance. Ces transferts d'informations impliquent des coûts considérables pour les entreprises qui y adhèrent. L'information contient généralement des redondances significatives. Supprimer ou réduire cette redondance dans la représentation de l'information peut diminuer considérablement la taille des données. C'est ce que l'on appelle la compression des données.

4.1 Généralités

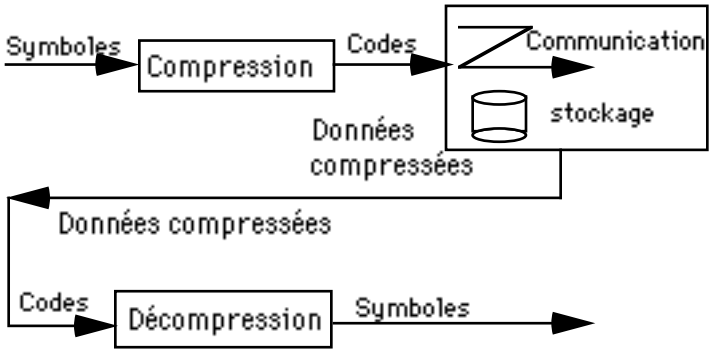


fig. 4.1: Modèle d'un système de compression et de décompression de données

La compression, cette diminution de la taille de l'information, engendre une réduction du temps de transfert de celle-ci, et par conséquent un coût de communication moindre. Elle contribue aussi à une réduction dans les exigences en stockage. La compression de données est un mécanisme qui consiste à transformer une chaine de données en une nouvelle chaine contenant la même information mais ayant une taille plus petite. Pour être efficace et utile, une technique de compression doit être en mesure de retransformer l'information

compressée en une copie identique ou en une approximation acceptable de l'origine.

4.1.1 Avantages et inconvénients de la compression de données

La compression de données comporte plusieurs avantages. Elle amène un gain d'espace et de temps. Elle améliore parfois même la sécurité de l'information. Ces avantages sont intéressants, parfois même vitaux à certaines applications informatiques.

Espace de stockage: La compression consiste à éliminer les redondances contenues dans l'information, donc à réduire la taille de cette dernière. La compression entraine une réduction dans l'espace nécessaire pour stocker l'information. On parle souvent d'un gain de 30 à 50% de l'espace requis, voire 75 ou 80% pour certains types d'informations [RUBIN 76]. En effet, comme on le verra plus loin, l'information n'est pas comprimable de la même façon et avec le même taux de compression dans tous les cas. La réduction dans les besoins en espace de stockage engendre une baisse du coût de stockage. Les entreprises sous-traitant leurs traitements informatiques dans des centres de calcul sont souvent facturées sur l'espace occupé par leurs fichiers de données. De même, avec un espace fixe, la compression de données permet de stocker davantage d'information. Ceci donne la possibilité de mettre plus de données sur un même support informatique. C'est ainsi que des applications ayant une taille supérieure à celle d'une simple disquette peuvent être contenues dans celle-ci. D'autres avantages indirectement engendrés par la réduction de la taille des données, sont aussi à noter. En effet, l'information compressée entraine une diminution de chargement et d'écriture. La réduction du temps de transfert de l'information compressée sur les canaux d'entrées/sorties d'une installation informatique est aussi non négligeable. Ceci est très appréciable au niveau d'un serveur de fichiers dans un réseau local, par exemple. La réduction du volume des données limite le nombre d'accès aux disques tandis que le codage et le décodage que suppose la compression ne se traduisent que par l'exécution de quelques instructions supplémentaires. Cela réduit considérablement le temps d'exécution des programmes puisque le temps d'exécution d'un groupe d'instructions est normalement très inférieur au temps nécessaire pour accéder aux données ou de les transférer vers un périphérique.

Temps de communication: Le temps de transmission d'une information sur une ligne de communication à longue distance se compte, suivant la taille des données transmises, en secondes et souvent en minutes. En réduisant la taille de l'information à transmettre, c'est-à-dire en la compressant avant l'émission, le temps nécessaire pour la transmission est amoindri. En effet, le temps nécessaire pour la compression, d'un côté, et la décompression, de l'autre, est négligeable par rapport au temps de transmission, ce qui donne un gain global appréciable en temps. Ce gain en temps engendre par le fait même un coût de communication plus bas. En se donnant un laps de temps déterminé, la compression de données donne la possibilité de transmettre plus d'informations. La compression des données influe sur le taux de transfert d'informations. En émettant des données compressées à 50% sur une ligne à une vitesse de transmission de 9600 bps, par exemple, on obtient un taux effectif

de transfert d'informations de 19 200 bps.

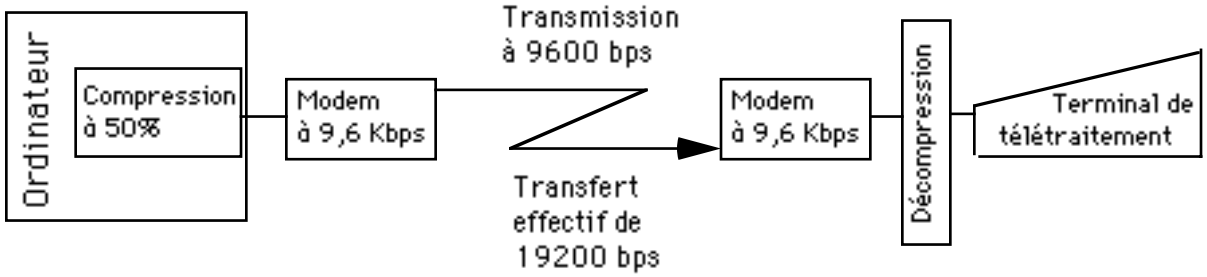


fig 4.2: Influence de la compression sur le taux de transfert d'informations

Sécurité de l'information: C'est souvent la redondance qui permet le déchiffrement de messages codés ou, à tout le moins, le facilite. Or, cette redondance, qui rend l'information plus sensible au déchiffrement est éliminée ou diminuée d'une façon importante par la compression. La compression permet donc une sécurité légèrement accrue par l'ignorance de l'algorithme. Une information compressée n'est pas lisible, ni compréhensible sauf après décompression. Or, la décompression est étroitement liée à la méthode ou à l'algorithme utilisé pour la compression. La méthode de compression se trouve donc être une clé pour la décompression. Elle démontre un autre niveau de sécurité car pour pouvoir en faire l'expansion, on doit connaitre la méthode appliquée pour la compression.

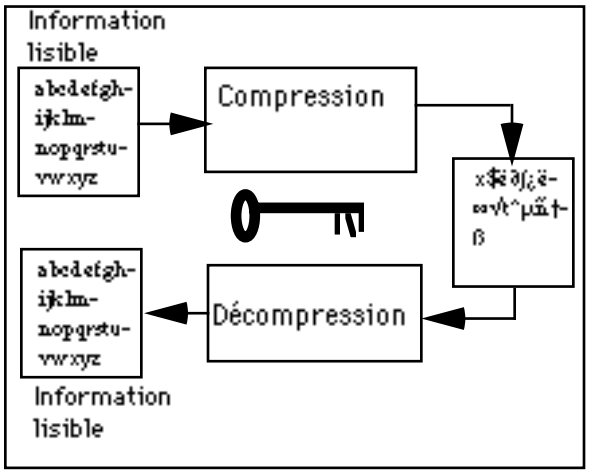


fig 4.3: Sécurité des données assurée par compression

Malgré tous ces avantages, la compression présente quelques inconvénients. En effet, depuis que l'on a pensé à la compression de données comme solution aux problèmes de manque d'espace de stockage et aux problèmes de coût de communication, plusieurs modèles et méthodes de compression ont vu le jour. Mais aucun standard n'a encore été établi. Quelques efforts de standardisation au niveau de la compression d'images commencent à peine à se faire sentir. L'absence de standard dans le domaine de la compression réduit considérablement sa portabilité. La décompression est toujours étroitement liée à la technique utilisée pour la compression. Ceci est considéré comme un inconvénient mais permet quand même d'avoir une sécurité relative plus accrue. La compression des données réduit cependant la fiabilité de ces dernières puisque les redondances, utiles pour le recouvrement d'erreurs, sont éliminées. En l'absence de redondances, plusieurs techniques de recouvrement d'erreurs ne sont plus opérationnelles. D'autres protocoles de correction d'erreurs sont alors utilisés mais sans beaucoup d'efficacité.

De nos jours il y a plusieurs techniques de compression de données qui existent. Certaines sont plus simples que d'autres. Certaines utilisent moins de ressources que d'autres. Mais aucune n'est vraiment simple à implanter. La compression des données augmente considérablement la complexité informatique d'un système, surtout si cette compression est transparente. La maintenance de systèmes comportant de la compression nécessite des considérations délicates puisqu'une modification des données manipulées peut mettre en cause l'algorithme de compression utilisé.

Malgré ces inconvénients, la compression des données peut engendrer des gains très substantiels pouvant atteindre des taux considérables. L'apport que peut apporter la compression à la Carte Santé est incontestable. Ayant une taille plus petite, l'information compressée occuperait moins d'espace de stockage sur la carte, ce qui permettrait d'emmagasiner plus d'informations médicales dans le dossier portable. Cela se traduit par un rallongement de la vie d'une carte. En comprimant de 5% l'information contenue dans la carte, on libère de l'espace mémoire pour de nouvelles transactions. On prolonge ainsi la durée vie de la carte de 5%, si le débit des transactions est régulier. Étant donné que l'information a une taille réduite, le temps de transfert entre la mémoire de la carte et l'application est diminué de façon appréciable. Ainsi, les traitements sont accélérés puisque le temps requis pour la compression et la décompression est négligeable devant le temps de transfert de l'information. En effet, la compression et la décompression sont effectuées par un programme dans la mémoire de l'ordinateur et non par le microprocesseur de la carte.

Il est à noter que la compression des données et leur décompression ne s'effectuent jamais ensembles. En effet, l'application ne fait appel à la compression qu'au moment de l'écriture d'une information dans la mémoire de la carte, alors que la décompression n'est nécessaire qu'au moment de la lecture des données.

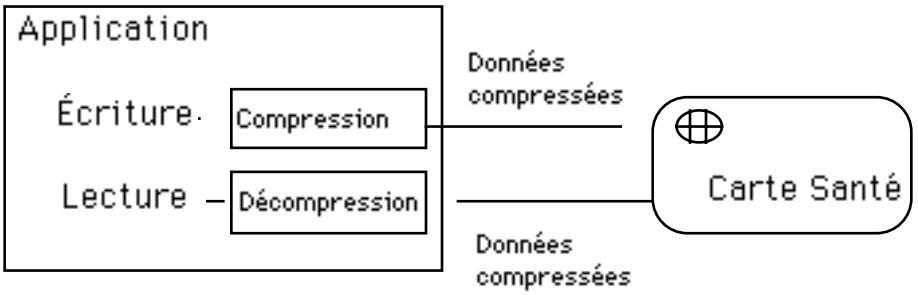


fig 4.4: Activation de la compression et de la décompression pour une carte à p

Le dernier avantage de la compression est la sécurité assurée pour les données contenues dans la carte. En effet, les données compressées, donc codées, ne sont lisibles que par l'application médicale qui connait la technique de compression utilisée. Une autre application qui tenterait de lire l'information ne pourrait pas la déchiffrer aisément. La compression des données rend l'information médicale sur la carte plus sécuritaire. Toutefois cette sécurité n'est qu'un niveau qui se rajoute à la sécurité conventionnelle. Cette dernière ne doit en aucun cas être substituée puisque

le niveau de sécurité assuré par la compression n'est pas infaillible.

4.1.2 Les types de redondances

Pour améliorer la densité des données emmagasinées ou transférées, la compression de données réduit la redondance présente dans l'information. Les techniques ou modèles de compression de données qui existent ne sont pas tous flexibles pour traiter simultanément tous les types de redondance. Elles s'intéressent à un type ou deux à la fois. C'est ce qui fait la différence entre les taux de compression réalisés par les différents modèles sur les mêmes données. On dénombre quatre types de redondances différentes dans les données commerciales: la distribution de caractères, la répétition des caractères, l'utilisation des patrons de caractères et la redondance de positionnement [WELCH 84]. Les redondances font référence à ce qui est observable sans tenir compte de l'interprétation des données.

Distribution des caractères: Dans une chaine de caractères, et par conséquent dans un fichier de données quelconque, les caractères n'apparaissent pas à une même fréquence. En effet, dans un ensemble typique de caractères, certains caractères sont utilisés plus fréquemment que d'autres. Le cas est encore plus fréquent dans des textes lorsqu'il sagit d'une représentation ASCII sur huit bits. Les trois-quarts des 256 combinaisons ne sont pas utilisées dans la majorité des cas. En français les caractères apparaissent avec une distribution bien connue. Ainsi le 'e' et l'espace ou la lettre 'a' apparaissent avec une fréquence plus élevée que celle de la lettre 'z', par exemple. Dans un texte en français, chaque lettre de l'alphabet a une probabilité d'apparition bien précise. La distribution des paires de lettres est aussi intéressante que celle des singletons. En effet, la paire 'QU' ou 'NE' a plus de chances d'apparaitre dans un texte en français que la paire 'ZK', par exemple. Cette distribution des caractères influence le nombre moyen de bits significatifs nécessaires pour le codage d'un caractère, appelé entropie [SHANNON 48]. L'introduction de valeurs numériques, de graphiques ou de caractères de contrôle peut faire varier sensiblement la distribution des caractères dans un ensemble de données. Elles peuvent faire la différence entre les statistiques d'apparition des caractères d'un fichier à l'autre. La méthode de compression de Huffman se base exclusivement sur ce type de redondance (voir 4.3.2).

Répétition des caractères: Dans un texte conventionnel, un caractère peut se répéter plusieurs fois consécutivement. C'est le cas des lignes blanches constituées d'espaces consécutifs, des tableaux ou des cadres formés de traits ainsi que des graphiques comportant des caractères spécifiques contigus. Ces chaines constituées d'une répétition d'un même caractère peuvent être représentées d'une façon plus compacte. Le codage EBCDIC traite ce genre de redondance. La technologie du télécopieur (fax) se penche aussi sur ce type de redondance. Ainsi, les suites de blanc ou les suites de noir dans un facsimilé sont codées d'une façon concise. On voit aussi cette redondance dans le traitement vocal digitalisé où une suite de silences ou un son allongé est codé d'une façon compacte. Avec l'intégration des multimédias, la fusion des images graphiques avec des tableaux de gestion et du texte, on fait face à des apparitions de plus en plus fréquentes d'ensembles de données homogènes dans

un ensemble d'informations de plus en plus hétérogènes, comme un ensemble de points d'une même couleur et de mêmes caractéristiques dans une image vidéo. Dans un tel contexte d'intégration, la répétition de caractères peut poser un défi à la compression de données en tant que redondance.

Utilisation des patrons: Dans un texte, et même dans des données binaires, certaines séquences de caractères réapparaissent assez souvent. Ces séquences, qui pouvent avoir dans certains cas, une taille importante, peuvent être codées sur moins de bits et petmettre ainsi un gain d'espace. Dans un texte, des séquences de caractères comme les articles le, la, un, les..., ou même des verbes conjugués comme est, été..., peuvent se répéter fréquemment. Les mots 'compression de données', 'information' et 'redondance', par exemple, se retrouvent avec une fréquence très élevée dans cet essai. Dans un texte français, des paires de caractères comme 'TE' ou des triplets comme 'QUE' peut avoir une plus grande probabilité d'apparition que celle de certains caractères comme 'k' ou 'y'. Dans des textes spécifiques, comme en médecine ou en science, la distribution varie et on peut remarquer une fréquence d'apparition élevée des séquences de caractères comme 'PHE' ou 'PHIE' par exemple. Ces séquences peuvent alors être codées sur moins de huit bits par octets. La méthode de compression de Lempel, Ziv et Welch (LZW) se base exclusivement sur ce type de redondance [ZIV 77; ZIV 78] (voir 4.3.3).

Redondance de positionnement: Lorsque certains caractères ou ensembles de caractères apparaissent constamment à un endroit prévisible dans chaque bloc appartenant à une suite de blocs de données, ils sont alors considérés comme partiellement redondants. On rencontre ce type de redondance dans les en-têtes de fichiers ou les blocs de contrôle. Les représentations de balayage d'images, les scanners médicaux, les images satellites ainsi que les images de télévision vidéo de haute définition (HDTV) comportent beaucoup de redondance de positionnement. Par contre, ce type de redondance n'existe quasiment pas dans du texte.

Dans une certaine mesure, ces quatre types de redondances cohabitent et se chevauchent dans un même ensemble de données. Un modèle de compression élaboré à partir de plusieurs types de redondances à la fois est susceptible d'atteindre des taux de compression importants.

D'autres types de redondances, autres que ceux énumérés ci-dessus, existent. Le traitement des signaux vocaux digitalisés en est un bon exemple. On retrouve dans la voix plusieurs redondances. Celles-ci sont traitées en fonction des fréquences, non pas en fonction du temps. En général, les méthodes utilisées pour la compression des signaux vocaux ne sont pas utilisées dans la compression de données alphanumériques et vice- versa. On retouve ces techniques dans la compression d'informations venant de capteurs électroniques, les électrocardiogrammes, etc.

4.1.3 Notions sur la théorie de l’information

Plusieurs termes et notions de base sont fondamentaux en théorie de l'information. Il est intéressant de passer en revue certaines de ces notions avant

d'entamer la présentation de la théorie sur laquelle se base toute communication numérique. En termes simples, la communication consiste à transmettre des *symboles* d'une source vers une destination à travers un *canal*. Un symbole est un élément d'un ensemble fini prédéterminé appelé *alphabet*. Une séquence de symboles, appelée *chaine*, constitue un *message*. Le processus qui génère séquentiellement des symboles est appelé *source*. On associe à chaque symbole produit par la source une probabilité qui caractérise sa fréquence d'émission. Une source est dite *discrète* si le nombre de symboles émis ou à émettre est fini et connu. Elle est *continue* si le nombre de symboles tend vers l'infini ou est inconnu. Le canal, pour sa part, est simplement un moyen de communication entre l'émetteur et le récepteur. C'est un modèle des dispositifs et milieux de transmission interposés entre la source et le destinataire.

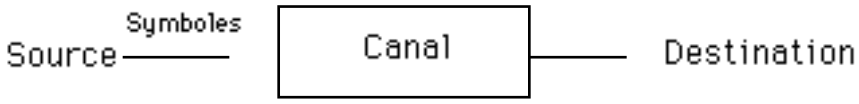


fig 4.5: Canal de transmission

La communication numérique traite essentiellement deux problèmes: la minimisation du nombre de bits qui doivent être transmis sur le canal de communication avec une fidélité prédeterminée (codage de la source), et l'assurance que les bits transmis sur le canal soient correctement reçus malgré les différents types d'interférences (codage du canal).

En 1948, Claude Shannon proposa les fondations d'une théorie de l'information donnant des solutions à ces deux problèmes [SHANNON 48]. Au cours des décennies qui suivirent, l'évolution et l'application de cette théorie de l'information eurent une influence importante dans les implantations pratiques de systèmes de communication numérique. Shannon fut parmi les premiers à essayer de quantifier l'efficacité des codages. L'un des codages qui l'intéressa en particulier fut celui du texte anglais. Il prouva qu'il était possible, en parcourant un texte lettre par lettre, de deviner le prochain caractère en se basant sur l'apparition des précédents. Ainsi, en se basant sur l'orthographe, les règles de grammaire et certains usages de l'anglais, il est possible de se donner un modèle pour deviner correctement 65% des caractères d'un texte. Ceci implique que les deux tiers des lettres soient redondantes. Shannon conclut alors que seulement deux des huit bits d'un octet d'un texte contiennent de l'information. Pour lui, le contenu informationnel moyen en anglais est de deux bits par symbole. Shannon appela cette quantité «entropie» parce que la formule qui l'exprime a la même forme que l'équation dérivée par Boltzmann pour l'entropie thermodynamique $S = k \log(w)$, où S est l'entropie, w est le nombre de façons de réamménager les parties d'un système et k est la constante fondamentale de la nature connue sous le nom de constante de Boltzmann. En théorie de l'information comme en thermodynamique, l'entropie est la mesure de liberté de choix. En communication, l'entropie est aussi connue sous le nom de degré de liberté informationnelle. L'information est définie en terme de mesure d'incertitude. Plus un message est incertain, plus important est son contenu informatif. Lorsque la probabilité d'un message augmente, l'information qu'il contient diminue. On dit que

le contenu informationel d'un message à transmettre est une fonction monotone décroissante de sa probabilité d'occurrence. Le négatif du logarithme est une telle fonction. Cette fonction résume bien les propriétés que l'on désire ici. En effet le logarithme de 1 vaut 0, ce qui symbolise le fait que l'on ne reçoit rien (ou on n'apprend rien) que l'on sait déjà. En d'autres termes, on connait le symbole émis avant de le recevoir. Avec une source sans dépendance entre les événements successifs, appelée source discrète sans mémoire, l'information d'un événement est alors formulée de la façon suivante:

$$I = - \log_2 P \quad \text{en bits} \qquad (1)$$

où P est la probabilité de l'événement ($0 \leq P \leq 1$). Le logarithme est choisi à base 2 parce qu'il se relie plus facilement au codage binaire. La fonction de la figure 4.6 montre la quantité informationnelle pour quelques valeurs discrètes de probabilités d'occurrences.

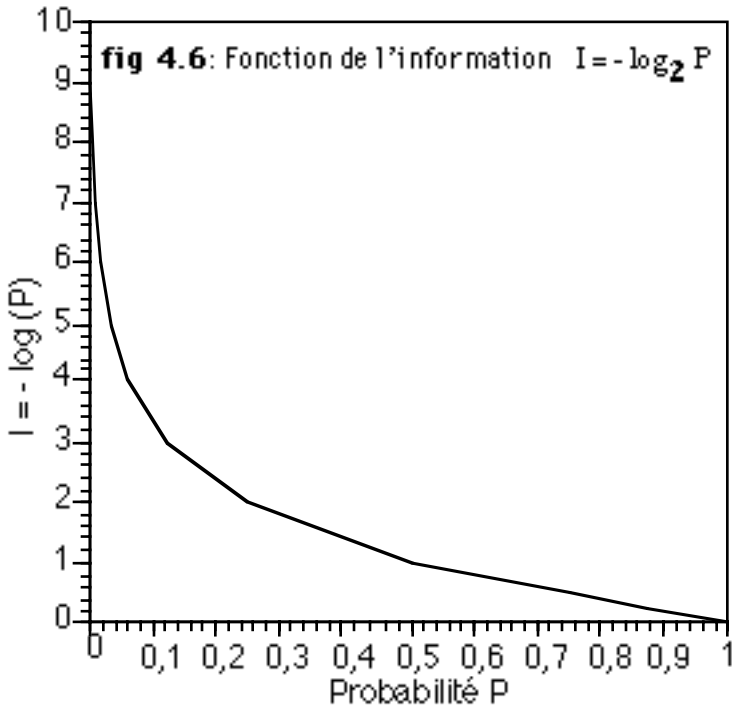


fig 4.6: Fonction de l'information $I = - \log_2 P$

En faisant la moyenne des contenus informationnels de tous les événements possibles d'une source, on obtient alors l'entropie de la source. L'entropie d'une source discrète sans mémoire, qui génère des symboles dans un ensemble de k valeurs différentes, est une simple moyenne formulée de la façon suivante:

$$E = - \sum_{i=1}^{k} (P_i \log_2 \qquad (2)$$

Dans le cas plus général d'une source avec mémoire, c'est-à-dire qui garde trace des fréquences d'apparition des anciens symboles reçus pour déterminer les symboles suivants, on permet d'avoir une certaine dépendance entre les événements

successifs. Une estimation plus concrète de l'entropie est alors obtenue en tenant compte du contexte. Pour une source continue, c'est-à-dire lorsque le nombre d'événements tend vers l'infini, l'entropie tend aussi vers l'infini. En effet, en regardant la définition de l'information d'un événement d'une source discrète (1), on voit que lorsque $i \to \infty$, $P_i \to 0$ et $-\log_2 P_i \to \infty$. On dit qu'une source continue a une information infinie. Sa formulation mathématique est la suivante:

$$E(x) = -\int_{-\infty}^{\infty} p(x) \log \qquad (3)$$

où p(x) est une densité de probabilité de la source avec:

$$\int_{-\infty}^{\infty} p(x)\, dx = 1 \qquad (4)$$

La formulation définie par (3) est appelée entropie différentielle. Elle caractérise une source continue.

Toutes les propriétés de l'entropie d'une source discrète s'appliquent pour l'entropie différentielle. Toutefois cette dernière n'est pas une mesure d'incertitude.

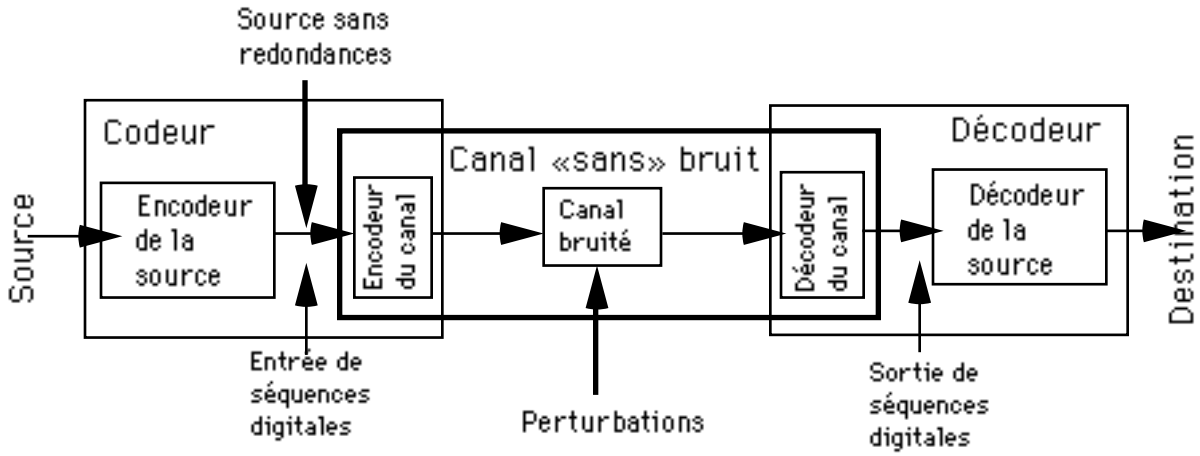


fig 4.7: Modèle de base d'un système de communication numérique

La figure 4.7 reproduit le shéma originel de Shannon. Ce schéma décrit le problème dual formulé et résolu par Shannon. On y retrouve le canal de transmission reliant la source, émettant une suite de symboles, et son destinataire. On retouve à l'entrée du canal un codeur qui transforme la suite de symboles issu de la source en une suite appliquée à l'entrée du canal, afin d'adapter les caractéristiques de la source à celles du canal. À la sortie du canal, le décodeur opère la transformation inverse [VITERBI 79] afin de reconstituer la suite à l'intention du

destinataire. Le codeur, comme le décodeur, sont constitués de deux dispositifs: codeur et décodeur de source, codeur et décodeur de canal. Grâce à l'effet conjugué du codeur et du décodeur de canal, le canal se comporte comme un canal sans bruit puisque ces derniers annulent l'effet des perturbations. La fonction du codeur de source est de réduire la redondance des messages émis. Son rôle est donc de résoudre la première problématique de la communication, à savoir: comment envoyer un message avec un minimum de bits. On sait qu'un canal est le siège de perturbations aléatoires importantes. Un symbole apparaissant à la sortie d'un canal bruyant n'est pas toujours égal à celui fourni à l'entrée. L'existence de cette perturbation entraine à la réception d'un symbole une ambiguïté sur la nature du symbole fourni à l'entrée. La fonction du codeur de canal est de réduire cette ambiguïté, c'est-à-dire qu'il permet, à la suite de la réception d'un message, d'identifier avec certitude la séquence de symboles émis. C'est la réponse à la deuxième problématique, qui est d'assurer une réception correcte des bits transmis sur le canal. Idéalement, le canal est sans bruit. Mais théoriquement, ceci n'est possible que si l'entropie de la source est inférieure à la capacité du canal. La capacité est la grandeur caractéristique du canal mesurant la plus grande quantité d'information moyenne, par symbole entrant, dont le canal peut assurer le transfert. La compression des données, donc le codage de source, suppose un canal sans bruit. De même, le codage effectué pour le canal suppose la source dépourvue de redondance. Le codeur de canal rend la source redondante en rajoutant d'une façon intelligente de la redondance aux messages entrants. La redondance artificielle rajoutée permet un meilleur contrôle d'erreur par le canal. À titre d'exemple, on peut citer le codage convolutionnel ou le codage de blocs comme codeurs de canal et l'algorithme de Viterbi ou l'algorithme M comme décodeurs de canal [VITERBI 79]. Quant au codeur de source, les paragraphes suivants font référence à une multitude de techniques et de modèles permettant une réduction de redondance.

4.2 Survol des techniques de compression

4.2.1 Quelques définitions

En général, la compression de données consiste à prendre des symboles d'une entrée, à les traiter et à sortir des codes. Ces codes sont censés occuper moins d'espace que les symboles à l'entrée. Les symboles en entrée peuvent être des octets, des pixels, des réels sur 32 ou même 80 bits, des caractères EBCDIC ou une quelconque entrée d'information. Nous allons, dans ce qui suit, introduire des termes pour définir le problème d'une façon précise. Le texte, ou l'information, à coder est appelé *chaine d'entrée*. L'unité syntaxique de la chaine d'entrée est appelée *caractère*. Le caractère est donc une entité de la chaine d'entrée pouvant ainsi être une lettre de l'alphabet, un nombre ou un signal quelconque. Le code qui représente un caractère est appelé *code d'entrée* et la correspondance entre le code et le caractère est la *représentation de l'entrée*. La compression est un simple codage qui consiste à diviser la chaine d'entrée en sous-groupes appelés *groupes d'entrée*. Chaque groupe d'entrée est remplacé par un *code de sortie*. La correspondance entre les groupes d'entrées et les codes de sortie est la *représentation de sortie* appelée plus communément *table de codage* [RUBIN 76].

La réduction de l'espace qui doit être alloué pour un message donné peut s'effec-

tuer par la compression sur un volume physique, un intervalle de temps ou une portion du spectre électromagnétique comme la largeur de bande. La largeur de bande, le temps et le volume sont interreliés par la formule suivante:

Volume = f (Temps, Largeur de bande)

Les techniques de codage modernes se basent sur un *modèle* partagé par le codeur et le décodeur pour coder et regénérer l'information. Le modèle est une distribution de probabilité d'un symbole en entrée dans n'importe quel contexte. Lorsque le contexte n'est pas impliqué, le modèle devient figé et les probabilités sont determinées par un calcul des fréquences d'apparition des symboles. Le *modèle figé* est alors partagé à l'avance par le codeur et le décodeur pour être utilisé pour plusieurs messages. Un *modèle adaptatif* peut changer à chaque transmission de symbole en se basant sur la fréquence d'apparition du symbole dans la partie de la chaine d'entrée déjà analysée. Du côté du décodeur, le modèle s'adapte aussi à chaque réception de symbole. Tous les messages sont alors traités indépendamment les uns des autres.

4.2.2 Les techniques de compression de données

Plusieurs auteurs considèrent l'invention du VOCODER (de VOice CODER), par Dudley en 1939, comme le premier exemple pratique de la compression. Le VOCODER réduit la largeur de bande nécessaire pour transmettre la parole sur une ligne téléphonique. Il partitionne l'énergie spectrale de la voix en un nombre fini de bandes de fréquence et transmet le niveau d'énergie dans chaque bande. Bien avant, au XIXème siècle, Morse inventa un alphabet pour transmettre des messages. Son alphabet, un codage à longueur variable peut être considéré comme un essai de compression de données. À cette époque, il n' y avait pas une vraie raison d'être de la compression et on ne s'y pencha que bien plus tard. Aujourd'hui, plusieurs techniques de compression existent. On retrouve leur application dans de multiples domaines comme les bases de données, le traitement de la parole, la vidéo et la télévision, les images ainsi que la télémétrie. Depuis le VOCODER, un intérêt particulier est porté sur la compression de la parole pour pouvoir la transmettre sur un canal à largeur de bande réduite. Certaines techniques de compression de la parole réduisent la redondance de celle-ci en gardant les paramètres pertinents du signal de manière à ce que le décodeur puisse synthétiser la parole. D'autres, appelées techniques de codage de forme d'ondes, réduisent la redondance du signal de manière à pouvoir reconstituer intégralement le parole à la sortie du canal. Avec l'arrivée de la télévision à haute définition (HDTV), le système vidéotex et, plus récemment, la télévision interactive, de même avec la prolifération des réseaux de télévision, le besoin de réduire la largeur de bande de transmission s'est beaucoup accru. Plusieurs techniques spécialement conçues et adaptées aux images en mouvement ont vu le jour. Ces techniques se basent essentiellement sur le codage avec transformées de Fourier, Hadamard et Haar. Elles se classent pour la plupart dans la catégorie des techniques de «compaction». Il y a eu, à l'heure actuelle, plusieurs tentatives pour regrouper les multiples techniques de compression dans des classes pour les différencier. Le codage de source, étant un domaine relativement nouveau dans la théorie de communication, sa classification n'a pas encore était normalisée.

Néanmoins, la majorité des auteurs s'accordent à dire que les techniques de compression se regroupent en deux catégories: les compressions réversibles et les compressions irréversibles. On retrouve ces classes sous différentes nominations comme la réduction de redondance ou le codage sans bruit pour la compression réversible et réduction d'entropie ou codage à réduction de fidélité pour la compression irréversible. Les techniques réversibles permettent de retrouver intégralement les données d'origine, après la décompression. Ces techniques restituent à la décompression toutes les redondances supprimées lors du codage. Les techniques irréversibles ne restituent pas intégralement les données, mais elles fournissent à la décompression une image acceptable et interprétable des données d'origine. On les appelle aussi techniques de «compaction». Elles sont souvent appliquées pour la compression des images ou des graphiques qui n'exigent pas une haute définition. Elles réduisent la représentation physique de l'information en éliminant un sous-ensemble non pertinent de celle-ci. À partir de cette nouvelle représentation, une approximation des données d'origine peut être reconstruite. Dans cet essai, nous introduisons une classification d'un niveau sémantique supérieur, à savoir la compression logique et la compression physique. Dans la classe compression physique on retrouve les regroupements classiques: réversible et irréversible. Les compressions logiques sont quant à elles toujours réversibles.

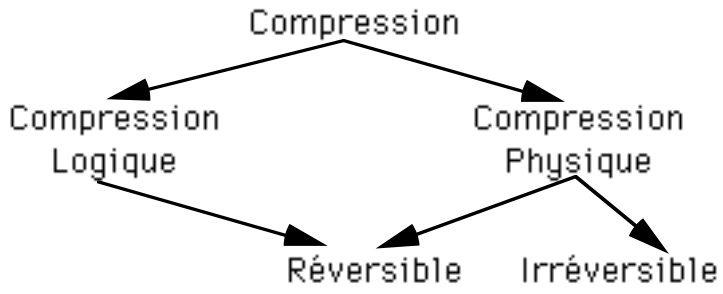


fig 4.8: Classification des techniques de compression

4.2.2.1 Compression Logique

La compression logique permet une représentation plus efficace des données. Elle est logique car elle ne procède pas à une réduction physique du volume en tenant compte des fréquences d'apparition des caractères ou des groupes de caractères dans la chaine d'entrée. Elle se soucie plutôt de créer des liens logiques pour éliminer les redondances.

Normalisation: Le cas le plus typique de la compression logique est la normalisation. La mise en formes normales des fichiers permet d'éliminer une grande partie des redondances d'information dans une base de données. La normalisation permet donc de réduire l'espace requis pour une base de données sans réduire ou modifier l'information la contenant, et cela en réarrangeant l'information d'une façon plus efficace. L'exemple suivant illustre bien le gain d'espace que peut apporter cette compression logique dans une petite base de données représentée dans une carte à puce par exemple. Cet exemple suppose qu'il y ait 1 000 transactions médicales dans un dossier médical portable et que chaque transaction

soit signée par le médecin traitant. Si une signature prend deux octets comme espace de stockage, il faut compter 2 000 octets comme espace total pour toutes les signatures. En admettant qu'un patient ne consulte pas plus de 256 médecins différents, la normalisation consisterait à créer une table contenant les signatures de tous les médecins traitant le dossier du patient, soit un espace de 256 fois 2 octets, et de ne mettre avec la transaction médicale que le rang de la signature dans la table. En sachant qu'un identificateur de signature n'a besoin que d'un octet, cela nous donne un espace total de 1 512 octets, soit un gain de 24,40%.

Codage à longueur fixe: La codification ASCII sur huit bits pour représenter un caractère, est la codification la plus répandue. La table ASCII peut codifier 256 caractères différents. Il arrive souvent que l'alphabet d'une chaine d'entrée soit composé d'un nombre de symboles inférieur à 256. C'est le cas, dans une base de données, des noms de personnes, des adresses ou du texte en général. Le nom d'une personne écrit en majuscule, par exemple, se définit dans un alphabet de 26 caractères seulement. Utiliser la table ASCII pour codifier un tel champ nous ferait perdre systématiquement trois bits par caractères. Il est plus efficace d'utiliser une table de 32 entrées comportant les 26 lettres de l'alphabet en majuscules chacune étant codée sur cinq bits. Le même raisonnement peut se faire sur d'autres champs utilisant un alphabet restreint. On obtient ainsi quatre nouvelles tables: la table BCD pour coder sur quatre bits les chiffres en caractère, la table ISO5 pour coder sur cinq bits les lettres majuscules, la table ISO6 pour coder sur six bits les chiffres et les lettres majuscules et enfin, la table ASCII7 pour tous les caractères alphanumériques codés sur sept bits.

Notation compacte: Lorsqu'un champ de données est stocké dans sa forme lisible, il contient souvent plus de caractères que nécessaire. Les dates en sont un exemple typique. Le champ des dates apparait fréquemment dans les bases de données. Au lieu d'écrire la date au complet, on utilise souvent une notation numérique plus compacte comme 910910 pour représenter le 10 septembre 1991. Bien que cette compression logique ramène à six caractères la taille du champ date, une réduction supplémentaire peut être obtenue en inscrivant la date sous forme binaire. En sachant que le jour ne dépasse jamais 31, cinq bits suffisent pour le représenter. De même, quatre bits suffisent pour coder le mois et sept bits pour l'année. Avec une base correspondante à 1900, ce système permet de couvrir la période de 1900 à 2027. Ce raisonnement peut être fait pour les champs représentant un horaire ou toute autre information similaire. La notation compacte est un outil très efficace pour minimiser la taille des champs d'une base de données [ALSBERG 75].

Substitution par des entrées dans un dictionnaire: Lorsque la valeur d'un champ dans une base de données se définit dans un ensemble limité d'attributs, il n'est pas intéressant d'orthographier cette valeur. Il est alors plus efficace d'utiliser un code pour représenter toutes les valeurs possibles [PIKE 81]. Cette stratégie est connue sous le nom de FLMB (Fixed-length minimum-bit). Un champ représentant la vaccination d'un patient est un bon exemple puisque tous les types de vaccins sont déjà connus. Ils peuvent donc être codifiés de façon concise. Les médicaments codés par un DIN, dans un dossier médical, sont aussi un exemple concret de la compression

logique de l'information. La substitution par des entrées dans un dictionnaire s'apparente beaucoup à la normalisation des données.

Stockage par différenciation: Cette méthode est très intéressante pour stocker des données successives ayant une petite variation de valeur. Elle mémorise la différence de valeurs des entrées. Ainsi, au lieu de stocker le périmètre cranien d'un bébé de façon périodique (P1, P2, ..., Pn), il est préférable de garder l'évolution du périmètre cranien en mémorisant la différence des entrées (P1, P2-P1, ..., Pn-Pn-1). La variation des entrées étant petite et souvent beaucoup plus petite que la valeur des entrées, elle nécessite moins d'espace de stockage que les données elles-mêmes. Le stockage des précipitations cumulatives quotidiennes est aussi un exemple bien pratique. Un gain d'espace considérable peut être obtenu grâce à la différence des précipitations cumulatives de deux jours consécutifs qui est inférieure à la valeur de chacune d'elles, d'autant plus que cette différence est souvent nulle. L'inconvénient de cette méthode est de devoir parcourir séquentiellement l'ensemble des données pour pouvoir les recalculer. Un accès direct à l'information est alors impossible.

4.2.2.2 Compression physique

La compression physique est un procédé qui tient compte des fréquences d'apparition des symboles ou des groupes de symboles pour réduire physiquement le volume des données avant leur entrée dans un canal de transmission et le rétablir au format initial à leur réception. Elle tire profit de la variabilité des probabilités d'apparition d'un symbole ou d'un groupe de symboles pour créer une codification plus efficace. Comme dans le cas de la compression logique, il existe de nombreuses méthodes de compression physique. Nous en énumérons certaines dans ce qui suit.

Compression du bon sens: Nous avons appelé «codages du bons sens» les codages élémentaires qui devraient être effectués en tout temps. Ces codages sont simples et permettent parfois des taux de compression appréciables. L'élimination des champs vides, la suppression des caractères répétés et la conversion des chiffres décimaux en binaires sont des exemples de compression du bon sens.

- **Élimination des champs vides**: Dans certains fichiers, des champs de taille parfois importante peuvent être vides. L'élimination physique de ces champs peut engendrer des gains d'espace. En ajoutant des bits de présence à un enregistrement de la base de données, on peut facilement indiquer par une étiquette la présence ou l'absence des champs. La figure 4.9 montre un format d'enregistrement où le champ «bits de présence» indique l'existance ou l'absence des autres champs.

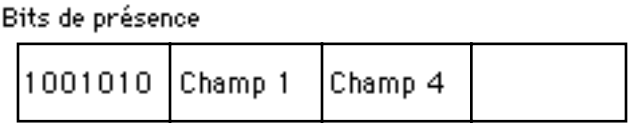


fig 4.9: Enregistrement avec bits de présence

- **Suppression des caractères répétés**: Plusieurs fichiers contiennent des blancs ou d'autres caractères qui se répètent à une fréquence importante. Leur compaction, comme celle des zéros qui se suivent dans un champ numérique, peut être très utile. La suppression des caractères qui se répètent est une technique simple par laquelle les chaines formées de caractères qui se répètent consécutivement sont remplacées par un ensemble de trois caractères, l'un indiquant la présence d'une suite de caractères, l'autre indiquant le nombre de répétition et le dernier représentant le caractère en question [RUTH 72]. Le codage EBCDIC utilise ce genre de compression.

- **Conversion des nombres décimaux en binaires**: Au lieu de sauvegarder des nombres avec une notation décimale qui demande quatre bits et parfois huit bits par chiffre, il est plus astucieux d'utiliser des représentations binaires. Ainsi, des nombre entre 0 et 255 peuvent être stockés en utilisant seulement huit bits.

Exploitation de l'ordre des données: En examinant la relation entre des données ordonnées successives, on peut souvent dériver un codage efficace permettant parfois une compression de haut niveau. On remarque, par exemple, que dans une liste ordonnée de noms de personnes, la première partie d'un nom se répète dans le nom qui le suit. Cette redondance partielle peut être éliminée en adoptant un codage particulier. En effet, seuls les caractères du nom n'apparaissant pas dans le nom précédent sont représentés. Les autres sont codifiés par un chiffre indiquant le nombre de caractères au début du nom précédent qui se répètent. La figure 4.10 montre un exemple d'une liste ordonnée de noms prise dans un annuaire téléphonique et codée pour un stockage sur support informatique. L'inconvénient de cette méthode est de devoir parcourir séquentiellement l'ensemble des données pour pouvoir les reconstituer à nouveau. Un accès direct à l'information est alors impossible.

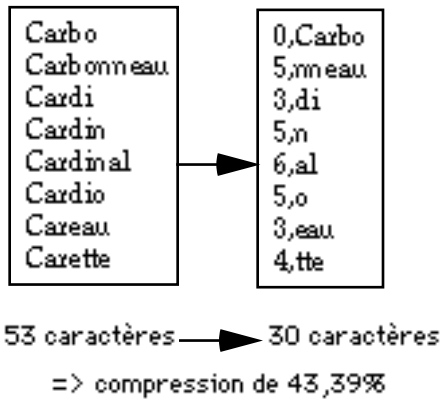


fig 4.10: Compression d'une liste ordonnée de noms

Codage à longueur variable: Normalement, le codage d'un caractère a une longueur fixe en bits comme celui de la table ASCII. Une compression importante peut s'effectuer en variant la longueur du codage par caractère en fonction de sa fréquence d'apparition dans la chaine d'entrée à compresser. Ainsi, les caractères les plus fréquents auront un code sur moins de bits que les caractères «rares». Ce type de code est utilisé par le codage Morse pour la télégraphie. L'algorithme de codage à longueur variable le plus connu est celui de Huffman [HUFFMAN 52]. Cet algorithme minimise le nombre total des bits pour les symboles qui apparaissent à une fréquence donnée dans une chaine d'entrée (voir 4.3.2). Cette technique exige un pré-traitement pour calculer les fréquences d'apparition des caractères dans un texte et construire un arbre de codage correspondant aux symboles de la chaine d'entrée.

L'arbre de codage est ensuite stocké avec la chaine de sortie, ce qui permet ensuite la décompression de l'information. La technique de codage à longueur variable n'est efficace que lorsque la fréquence d'apparition des symboles de l'entrée n'est pas la même pour tous les symboles. Lorsque les symboles de la source sont équiprobables et que leurs distributions sont équivalentes, le codage revient à un codage à longueur fixe.

Codage à variabilité de longueur restreinte: Cette technique repose sur la division de l'ensemble des caractères par deux: un sous-ensemble des caractères les plus fréquents et un deuxième contenant les caractères les moins fréquents. En utilisant les octets pour coder les symboles, on peut coder sur sept bits les 127 symboles les plus fréquents et mettre le premier bit toujours à 0. Lorsque ce bit est à 1, il indique que les 15 bits suivants (sur deux octets) codent un symbole rare. Ceci permet de coder 127 symboles fréquents et 382 rares.

Substitution par des patrons adaptatifs: Dans une chaine d'entrée, on retrouve souvent des séquences de symboles qui se répètent. Ces séquences peuvent être codifiées de manière à gagner de l'espace. Cette technique consiste à rechercher les patrons de 2, 3, 4 caractères ou plus qui se répètent souvent dans le texte. Tous les patrons fréquents sont alors substitués à chaque apparition par un codage associé [RUTH 72; REGHBATI 81] (voir 4.3.1).

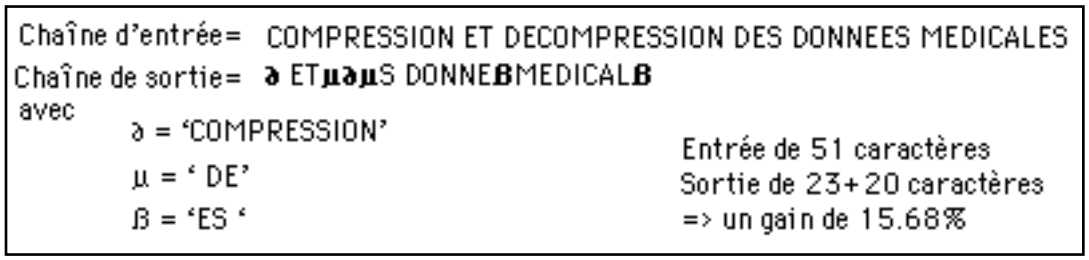


fig 4.11: Exemple de compression par substitution de patrons adaptatifs

Cette technique exige un pré-traitement pour rechercher les patrons qui se répètent et leur associer un symbole particulier. Les symboles utilisés pour le codage des patrons ne doivent pas être utilisés ailleurs dans la chaine d'entrée. Cette méthode n'est pas utilisable si tous les symboles de l'alphabet sont utilisés dans la chaine d'entrée. Les patrons substitués et leurs codages associés doivent être stockés avec la chaine de sortie pour permettre ensuite la décompression de l'information.

Codage de chaine: Le codage de chaine, comme son nom l'indique, consiste à coder les chaines de symboles. Par un assortiment de suites de symboles, l'algorithme convertit des chaines de longueur variable en un code de longueur fixe. Le codage de chaine est un codage adaptatif. Il s'adapte dynamiquement à la chaine d'entrée. Le processus de compression ne nécessite aucune statistique préalable sur les symboles à compresser et effectue le codage à la volée. L'inconvénient est que l'entrée doit être assez longue pour permettre une compression efficace.

Codage arithmétique: Au lieu de remplacer un symbole, ou un ensemble de symboles en entrée par un code spécifique, le codage arithmétique remplace une chaine

de symboles par un nombre réel. Plus la chaine d'entrée est longue et complexe, plus nombreux sont les bits dans le nombre réel en sortie [WITTEN 87; NELSON 91]. Après avoir assigné des probabilités d'apparition aux symboles d'entrée, un nombre unique entre 0 et 1 est calculé en se basant sur ces probabilités pour codifier une suite de symboles. Ce nombre réel est décodé de façon unique pour reconstituer la chaine de symboles d'origine (voir 4.3.4).

Codage mixte: Certains modèles de compression intègrent différentes techniques légèrement modifiées. Ainsi le codage de chaine, qui est un codage de longueur fixe de chaines de symboles de longueurs variables, peut être adapté pour avoir un codage à longeur variable de chaine variable [DESOKY 88; RODEH 81]. On retrouve aussi d'autres codages mixtes avec un mariage du codage à longueur fixe et le codage arithmétique ou le codage de chaine avec le codage arithmétique [RAMABADRAN 89]. Les codages mixtes profitent des avantages des modèles qui les composent pour atteindre des taux de compression importants. L'algorithme LZHUF, présenté dans le paragraphe 4.3.5, est un codage mixte.

4.2.3 Les techniques utilisables ou à utiliser pour la Carte Santé

La Carte Santé comporte un dossier médical quasi-complet. Du texte libre aux champs binaires, en passant par des valeurs numériques et des dates, on trouve des données très hétérogènes avec des structures très diversifiées. Chaque type de données est représenté de manière à utiliser le moins de bits possible. Ces représentations permettent d'utiliser la mémoire disponible de la carte de façon optimale (voir chapitre II). Les données à inscrire sur la Carte Santé sont donc déjà à un premier niveau de compression. Des tests sur un échantillon de dossiers médicaux réels ont montré qu'une telle notation compacte permettait, dans une première étape, de compresser jusqu'à 33% de la taille de l'information par rapport aux dossiers médicaux classiques en texte libre. Les données, ainsi structurées, sont constituées d'entités sémantiques de taille en bits différentes. Ainsi, on retrouve des valeurs numériques de trois bits avec des dates, des chaines de texte sur cinq bits par caractère et des séquences de bits diverses tous concaténés les uns aux autres. Pris par paquets de huit bits, ces données forment des séquences quasi aléatoires. La distribution des octets est imprévisible et variable d'un ensemble de données à l'autre, ce qui donne une information complexe, pauvre en redondance et difficile à compresser. Les algorithmes utilisables avec de telles données, et pouvant à priori donner des résultats satisfaisants, sont rares. Les techniques de codage à longueur variable, la substitution de patron adaptatif, le codage de chaine et le codage arithmétique sont néanmoins à tester. On peut alors atteindre des taux de compression bas mais satisfaisants pour de petits ensembles de données.

4.3 Algorithmes de compression et implantation

Nous avons choisi cinq algorithmes différents pour notre étude: la substitution par des patrons adaptatifs, l'algorithme de Huffman, LZW, le codage arithmétique et l'algorithme LZHUF. Dans ce qui suit, nous présentons ces algorithmes avec leur implantation.

4.3.1 Substitution par des patrons adaptatifs.

La substitution par des patrons adaptatifs consiste à rechercher des suites de symboles qui se répètent dans la chaine d'entrée et de les changer par des symboles inutilisés mais qui sont dans l'alphabet. La première difficulté est de trouver des symboles inutilisés pouvant servir de code de substitution. Cette difficulté est d'autant plus grande que les symboles utilisés dans un dossier médical portable sous forme binaire sont équiprobables. Tous les symboles de l'alphabet sont quasi susceptibles d'être utilisés. Cette difficulté peut être partiellement résolue en utilisant aussi comme code les symboles qui n'apparaissent que dans les suites qui vont être substituées. Le caractère utilisé uniquement dans des patrons qui sont appelés à être supprimés devient inutilisé dans le reste de la chaine d'entrée. Il peut donc devenir un code de substitution. Pa conséquent, la première phase consiste à rechercher des codes de substitution et des patrons substituables. La compression proprement dite se fait dans la deuxième phase. On remplace alors les patrons identifiés dans la première phase par des codes de substitution, s'il en existe. La compression se fait en un seul passage. Par contre, la première phase est plus complexe; on doit parcourir la chaine d'entrée plusieurs fois. Le procédé le plus évident est de fixer la taille des patrons à rechercher puis d'identifier toutes les suites possibles ayant cette taille et vérifier si elles se répètent dans la chaine. Cette méthode nous oblige à parcourir la chaine d'entrée plusieurs fois, ce qui n'est pas très avantageux pour la rapidité d'exécution. On aura au total $(n+1)/2$ parcours de la chaine d'entrée, n étant le nombre de symboles de la chaine d'entrée. En effet, en recherchant le premier patron, on fait n lectures de symboles, soit le parcours total de la chaine d'entrée. Lorsque l'on recherche le deuxième patron, qui chevauche de 1 caractère le premier patron, on fait n-1 lectures de symboles et ainsi de suite. Le nombre de lectures total égale la suite $n+(n-1)+(n-2)+...+2+1$. Cela nous donne $n/2 * (n+1)$ lectures. Or, dans un parcours total, on a n lectures de symboles puisque n est la longueur de la chaine d'entrée. Le résultat est bien $(n+1)/2$ parcours de la chaine d'entrée. Sans compter un premier parcours pour calculer la fréquence de tous les symboles de l'alphabet afin d'identifier les symboles inutilisés et de les traiter comme code de substitution. Une méthode plus astucieuse consiste à ne parcourir la chaine d'entrée qu'une seule fois pour faire l'analyse complète et l'identification de tous les patrons qui se répètent. Pour ce faire, on utilise trois structures: la structure 'OCCURRENCES', le tableau 'PATRONS' et la structure 'SUBSTITUTIONS'. OCCURRENCES est un tableau contenant les 256 symboles de la table ASCII avec leur fréquence d'apparition et la liste de leurs apparitions, c'est-à-dire les coordonnées dans la chaine d'entrée de leurs apparitions. SUBSTITUTIONS est une liste des coordonnées des suites de symboles à substituer avec un pointeur vers un élément de la table PATRON. PATRON est un tableau contenant tous les

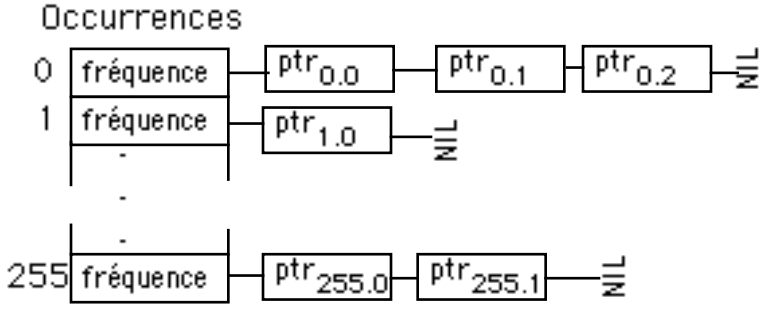


fig 4.12: Structure de données pour mémoriser les statistiques

patrons identifiés avec leur code respectif. PATRON est donc la table de codage. Une première itération calcule la fréquence d'apparition des caractères et mémorise leurs apparitions dans la structure occurences. À la suite de ce premier parcours, on peut identifier les symboles inutilisés et les réserver pour des codes. En traitant les éléments de la structure OCCURRENCES pris deux à deux on peut identifier des suites de deux symboles qui se répètent. On doit, à priori, parcourir 256^2 fois les listes de la structure OCCURRENCES. Cependant plusieurs éléments peuvent être éliminés. C'est le cas des symboles n'apparaissant qu'une seule fois, de ceux ayant une fréquence nulle ainsi que celui des symboles ayant une fréquence égale à la fréquence d'apparition des patrons dans lesquels ils figurent.

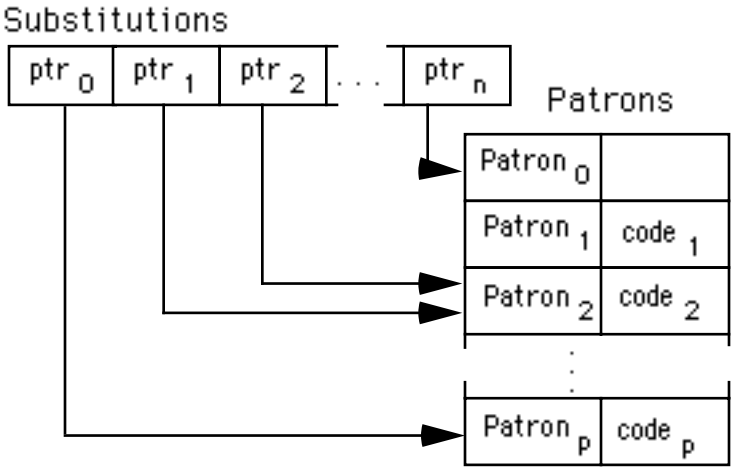


fig 4.13: structure de données pour représenter la table de codage et son utilisation

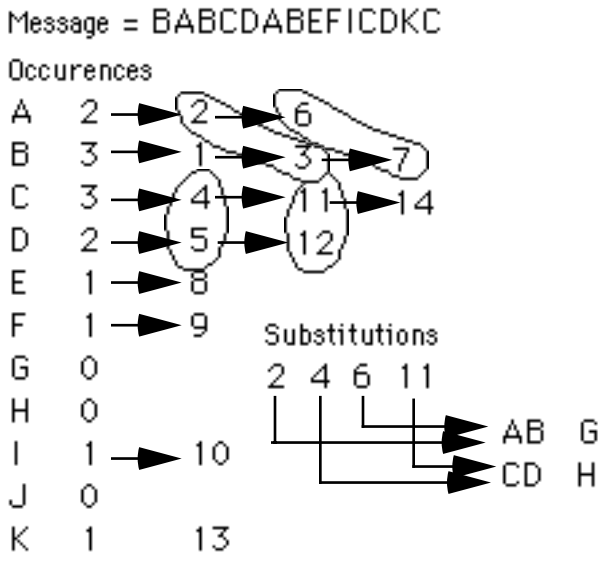


fig 4.14: illustration de l'utilisation des structures

Cette technique permet d'identifier des patrons de même taille. On peut trouver toutes les suites qui se répètent avec des tailles variables en procédant comme suit: après avoir identifier tous les patrons de deux caractères, on étudie leur contexte-avant, c'est-à-dire le caractère qui suit directement chacun de ces patrons. On trouve ainsi les patrons ayant une taille de trois caractères. En procédant de cette façon on peut déterminer les suites de symboles qui se répètent quelle que soit leur taille. Compte tenu des inconvénients en temps d'exécution de cette méthode et de sa relative complexité par rapport aux autres méthodes, nous n'avons pas jugé opportun de l'implanter sur machine. Bien qu'elle touche à plusieurs types de redondances et qu'elle puisse être jumelée aisément à une autre méthode de compression [DESOKY 88] , la substitution par des patrons adaptatifs a été abondonnée. En effet, la compression qui nous intéresse pour intégrer dans le projet Carte Santé doit avoir un temps d'exécution négligeable afin de ne pas alourdir le système pour l'utilisateur; elle doit être complètement transparente.

4.3.2 Algorithme de Huffman

La méthode de compression la plus populaire est sans doute le codage de Huffman. Pendant très longtemps, ce codage fût l'état de l'art de la compression des données. Le codage standard de Huffman [HUFFMAN 52] décrit un moyen d'associer un code de sortie à chaque symbole de la chaine d'entrée de façon telle que la taille en bits du code de sortie soit approximativement égale à log_2*(probabilité du symbole)* où la probabilité du symbole est la fréquence d'apparition relative du symbole dans la chaine d'entrée exprimée sous forme de probabilité. L'algorithme de Huffman suppose que la chaine d'entrée est une suite de symboles indépendants apparaissant à des fréquences relatives différentes. Ceci est évident pour des données textuelles en français où on utilise certaines lettres plus que d'autres. Cette caractéristique de la répartition des symboles est aussi vérifiable pour tout message à compresser. En utilisant la formule (1) de l'information décrite dans 4.1.3, Huffman code des symboles ayant une taille fixe avec un codage à longueur variable. Ainsi, si un caractère a une probabilité d'apparaitre de 0.125, il est codé sur trois bits, alors qu'un caractère ayant une probabilité d'apparaitre de 0.001 est représenté avec un code de dix bits. L'algorithme est relativement simple. En procédant par une analyse de la chaine d'entrée, la fréquence d'apparition de chaque symbole est déterminée. Avec les fréquences relatives de tous les symboles de la chaine, l'algorithme construit une structure d'arbre, appelée arbre de Huffman, permettant d'associer à chaque symbole une suite de bits représentant son code. L'arbre de Huffman est construit de façon à ce que les symboles ayant les plus grandes fréquences, aient les plus petites suites de bits comme codes et les symboles les plus rares aient de longues suites de bits comme codes. Cette manière de procéder oblige le traitement de la chaine d'entrée à être en deux passages. La première passe consiste à analyser les probabilités d'apparition des différents symboles de l'alphabet et la construction de l'arbre de Huffman avec les codes associés. La seconde remplace chaque symbole par son code et l'envoie sur le canal ou le mémorise dans le fichier compressé. Pour bien comprendre la technique de compression de Huffman, il est bien important d'observer le processus de construction de l'arbre de Huffman qui représente la table de codage. L'arbre est un arbre binaire dont la structure est composée de noeuds contenant chacun un symbole et sa fréquence d'apparition dans la chaine d'entrée. Chaque noeud est lié à un noeud père et pointe vers deux noeuds fils. En fait, comme on pourra le voir dans la figure 4.15, il y a deux types de noeuds: ceux représentant les pères et n'ayant pas d'information propre et ceux représentant les feuilles et comportant un symbole donné avec sa fréquence d'apparition. Pour un alphabet de 256 symboles comme la table ASCII, il y a 256 noeuds_feuilles et 255 noeuds_parents. Le processus de construction de l'arbre commence par l'initialisation d'une liste de noeuds avec l'ensemble des feuilles tous indépendants les uns des autres. Ensuite, l'arbre croit récursivement sur cette liste en construisant à chaque fois un sous-arbre reliant les deux noeuds de la liste ayant les fréquences les plus basses. La racine de ce sous-arbre constitue un nouveau noeud dont la liste a comme fréquence la somme des fréquences de ses deux noeuds fils. Cette itération continue jusqu'à ce qu'il n'y ait plus qu'un seul noeud sans parent dans la liste initiale. Ce noeud devient la racine de l'arbre. Voici, dans ce qui suit, l'algorithme simplifié de construction de l'arbre de Huffman:

initialiser la liste des sous-arbres L,
  avec l'ensemble des noeuds représentant tous les symboles ayant
  une probabilité non nulle
tant que L contient plus de deux noeuds indépendants
faire
    retirer de L les deux noeuds ayant les plus petites fréquences
    rattacher ces deux noeuds à un nouveau noeud ayant comme fréquence la
            somme des deux anciennes fréquences
    ajouter ce nouveau noeud à L
fait

Algorithme de construction de l'arbre de Huffman

La figure suivante illustre la construction de l'arbre de Huffman avec l'alphabet {A,B,C,D,E} et les fréquences respectives {20,4,9,15,52}.

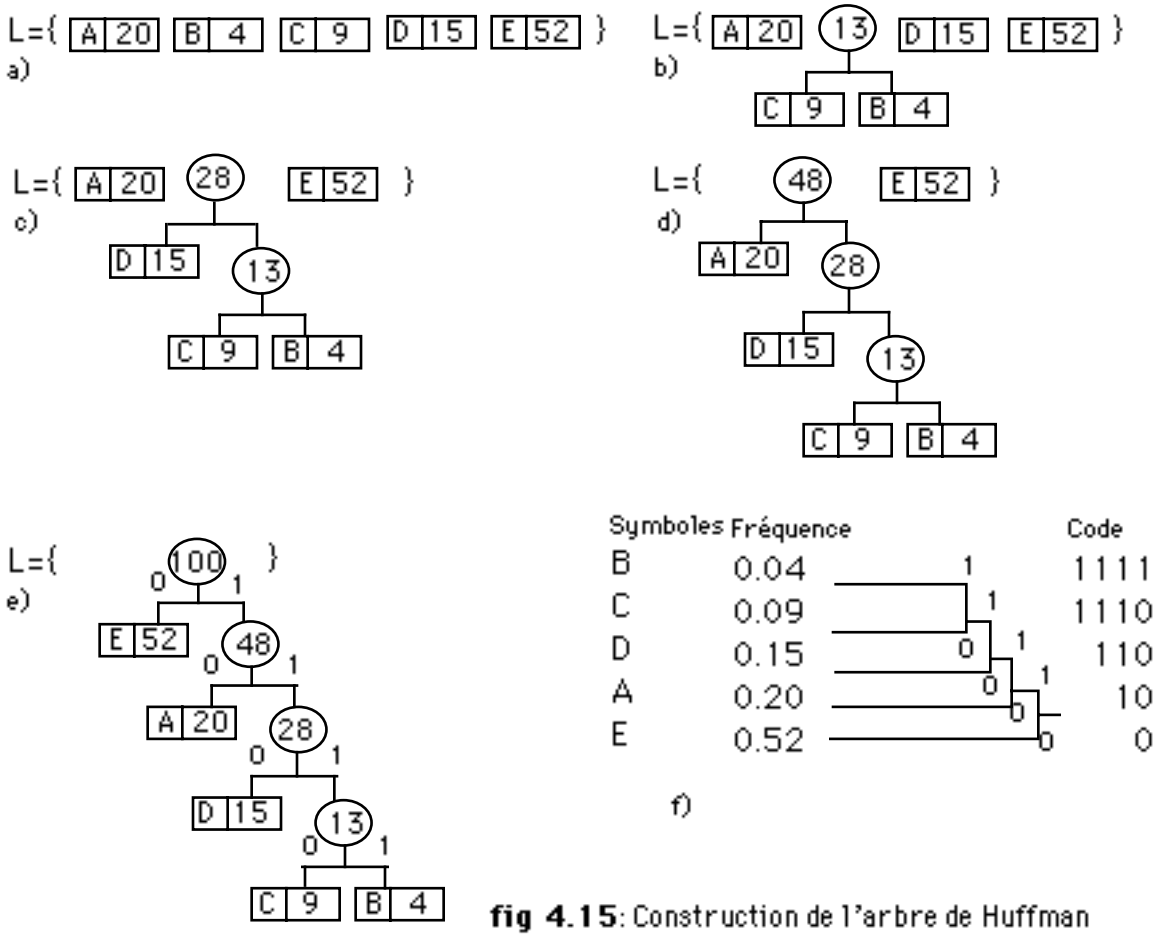


fig 4.15: Construction de l'arbre de Huffman

Dans la figure 4.15, les feuilles de l'arbre sont représentées par des rectangles, alors que les autres noeuds sont représentés pas des cercles. Notons que seules les

feuilles ont un contenu significatif. L'algorithme de compression utilise cet arborescence pour trouver la correspondance entre un symbole et son code de sortie. On peut remarquer dans la figure 4.15 (e) qu'en commençant à la racine et en parcourant le chemin vers 'E' (le caractère le plus fréquent), on a moins de noeuds à traiter que dans le chemin qui mène à 'B' (le caractère le moins fréquent). En attribuant un poids binaire (0 ou 1) à chacune des deux branches de chaque noeud, on peut faire correspondre un code binaire à chaque feuille de l'arbre. La figure 4.15 (f) nous montre le codage obtenu en attribuant le bit 1 aux branches de droite et le bit 0 aux branches de gauche de l'arbre. Cette attribution de poids est arbitraire et peut être inversée. La compression de chaque symbole reçu est donc un simple parcours de l'arbre en commençant par la feuille représentant le symbole et en allant vers la racine. Lors de ce parcours, le traitement itératif consiste à identifier le parent du noeud courant et à déterminer si le noeud courant est un fils droit ou gauche pour savoir si le prochain bit de la suite des bits, représentant le code de sortie, est un 1 ou un 0. Comme le parcours se fait des feuilles vers la racine de l'arbre, la suite des bits que l'on obtient est inversée. Le compresseur les renverse pour mettre le code à l'endroit avant de le transmettre.

Le processus de décodage est un vrai reflet de la compression. Après la réception du tableau représentant la répartition des symboles, le décodeur construit l'arbre de Huffman comme le fait le codeur. Lorsqu'une suite de bits arrive du compresseur, le décodeur parcourt l'arbre à partir de la racine. Tout dépendant de la valeur du bit traité, le parcours se dirige vers la branche de droite ou de gauche de l'arborescence. Le symbole à décoder est identifié lorsque le parcours aboutit à la feuille. Le même traitement reprend ensuite à la racine de l'arbre pour le décodage des bits suivants.

Bien que considérée comme optimale pour la compression des messages d'une source discrète sans mémoire, la technique de codage de Huffman présente beaucoup d'inconvénients. On rappelle qu'une source sans mémoire est une source d'ordre 1, c'est-à-dire une source dont les symboles sont considérés comme étant indépendants les uns des autres. La compression de Huffman ne traite que les symboles pris les uns indépendamment des autres. Autrement dit, le contexte des symboles n'est pas pris en considération. Ainsi, seule la redondance liée à la distribution des caractères est traitée et non les autres types de redondances. Un problème se pose pour le codage des sources continues. En effet, la table de codage n'est pas dynamique et elle ne devient optimale qu'après l'analyse de toute la chaine d'entrée, ce qui est impossible avec une source continue. Knuth a néanmoins présenté un procédé qui construit dynamiquement l'arbre de Huffman [KNUTH 85]. Un autre inconvénient touche les phases de compression. Il faut analyser la chaine d'entrée pour déterminer la répartition des symboles de l'alphabet afin de construire une table de codage efficace. Cette analyse se fait dans une phase distincte et ce au détriment de la vraie phase de compression. Des adaptations et améliorations de l'algorithme ont été faites pour réduire ces deux phases en une seule [GALLAGER 78]. Il est aussi à noter que la distribution des symboles est une caractéristique propre de chaque source. Pour avoir une compression optimale, il faut avoir une table de codage pour chaque source. Ceci nous contraint donc à transmettre la table avec les données compressées et, par conséquent, de réduire

l'efficacité et le taux de la compression. La nécessité de transmettre la table de codage peut être éliminée en créant différentes tables de codage figées dans le programme de compression et de décompression et de ne transmettre que l'indice de la table utilisée; en quelque sorte, l'utilisation de tables de codage statiques. Pour se rapprocher au mieux des statistiques de la chaine d'entrée, le choix de la table utilisée est alors fait à la suite d'une analyse d'un échantillon de la source. Cette approche nous évite de transmettre la table de codage qui diminue inévitablement le taux de compression final, mais elle ne nous permet pas d'avoir un taux de compression optimal puisque les arbres de codage sont créés au préalable et figés dans l'application. L'approche choisie est de créer dynamiquement la table de codage dans le but de réduire les phases de compression en une seule [GALLAGER 78]. En implantant le même algorithme de création dynamique de la table de codage pour le compresseur ainsi que pour le décompresseur, la transmission des statistiques n'est plus nécessaire. À tout instant, l'arbre de codage reflète les statistiques de la portion du message déjà traitée. Chaque fois qu'un symbole est traité, l'arbre de codage de Huffman est remis à jour. Lorsque le compresseur reçoit le «ième» symbole, ce dernier est codé en utilisant un arbre respectant les statistiques des (i-1) premiers symboles. Ensuite l'arbre est modifié pour respecter les nouvelles statistiques en tenant compte de l'occurrence du «ième» symbole. Les modifications ainsi effectuées ne sont nullement transmises au décompresseur. Celui-ci effectue les mêmes variations dans l'arbre de codage pour se synchroniser au codeur.

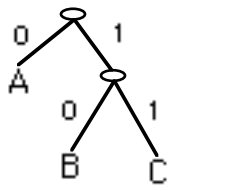
On peut améliorer la compression en traitant des paires d'octets comme entrée au lieu d'un caractère à la fois. Mais ceci nécessite une arborescence de 65 536 feuilles, ce qui n'est pas toujours possible ou du moins pratique. Le codage de Huffman fait partie de techniques de compression basées sur les statistiques. Ces méthodes sont plus efficaces si elles tiennent compte du contexte des symboles pour l'évaluation des probabilités. En effet, l'interdépendance entre les symboles d'une source engendre une redondance importante qu'il faut considérer. L'implantation que nous avons testée tient compte, en quelque sorte, de cette redondance. Pour augmenter le taux de compression, nous avons donc exploité l'interdépendance entre les caractères successifs. La probabilité de chaque symbole est calculée selon le contexte dans lequel le symbole apparait. Par exemple, si l'on ne considère pas le voisinage des symboles et qu'on les traite indépendamment les uns des autres (modèle d'ordre 0) dans un texte en français, la probabilité de voir apparaitre la lettre 'u' peut être de 7%. Par contre, si l'on sait que la lettre précédente est 'q', la probabilité de voir apparaitre la lettre 'u' augmente à 95%. Ceci montre clairement l'intérêt de tenir compte du contexte des symboles dans le calcul des probabilités. Logiquement, le taux de compression est proportionnel à l'ordre du modèle. Mais un modèle d'ordre élevé demande plus de ressources pour stocker les statistiques. Nous avons choisi un modèle d'ordre 1 pour notre implantation. Nous tenons donc uniquement compte du prédécesseur immédiat du symbole traité. Pour mémoriser les statistiques, nous avons utilisé 65 536 entiers (pour chaque symbole des 256 symboles de la table ASCII, il y a 256 prédécesseurs possibles). Ces entiers forment une table dans laquelle nous mémorisons les statistiques en terme de fréquences de

succession. La figure 4.16 est un exemple du contenu de cette table qui décrit l'interdépendance entre les caractères successifs du message AABABBCABABCA. Être capable de prédire les symboles avec une haute probabilité permet de diminuer le nombre de bits nécessaires à la codification, ce qui engendre un meilleur taux de compression. La table ci-haut permet de faire de meilleures prédictions sur l'apparition des caractères. La figure ci-dessous montre la table classique des fréquences d'apparition pour le même message. Elle contient moins d'informations que la précédente.

symbole	successeur	fréquence
A	A	1
	B	4
	C	0
B	A	2
	B	1
	C	2
C	A	2
	B	0
	C	0

fig 4.16: Table des fréquences de succession

symbole	fréquence
A	6
B	5
C	2



arbre de Huffman correspondant

fig 4.17: Table des fréquences d'apparition

Avec cette table et l'arbre correspondant, le message considéré est codé par '00100101011010010110', soit 20 bits. Par contre, avec la table des fréquences de succession, on construit un arbre pour chaque symbole. Chaque arbre représente la succession possible d'un symbole. Nous obtenons donc, pour notre exemple, les arbres représentés dans la figure 4.18. L'abre de C n'est pas représenté ici car la lettre C n'a qu'un successeur possible dans notre message, à savoir le A. Pour coder la séquence 'CA', le codeur ne code que la lettre C. Le décodeur de son côté rajoute automatiquement la lettre A après avoir décodé le C.

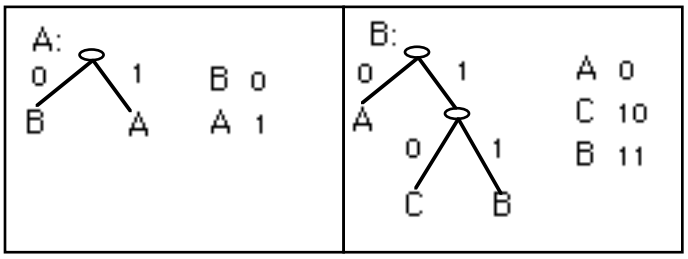


fig 4.18: Arbres de succession

La stratégie de codage consiste à indiquer dans le code de sortie l'arbre utilisé ainsi que le chemin suivi dans celui-ci. Le chemin identifie le symbole du successeur qui indique par la même le prochain arbre à utiliser. Pour connaitre le premier arbre à utiliser, le premier symbole est simplement codé par son code ASCII. Pour bien

saisir le mécanisme de cette stratégie, essayons-la sur notre exemple précédent: AA-BABBCABABCA. Le premier A est codé par son code ASCII. On se positionne sur l'arbre A. À l'arrivée du second A, on ajoute le bit 1 au code puisque dans l'arbre A, le code du symbole A est 1. À l'arrivée du B, on ajoute 0 au code et on se positionne sur l'arbre B. Lorsque A est traité, on ajoute le bit 0 au code et on se positionne sur l'arbre A. La codification se poursuit jusqu'à la fin du message pour obtenir 'a1000111000010' avec 'a' le code ASCII de la lettre A. Pour notre alphabet réduit, le code ASCII de A peut être en fait sur deux bits, ce qui nous donne un code de sortie sur 15 bits au lieu de 20. Le processus complet de codage du message peut être schématisé comme suit:

Message	Arbre utilisé	Émission	Code cumulé
A		'a'	a
A	arbre_A	1	a1
B	arbre_A	0	a10
A	arbre_B	0	a100
B	arbre_A	0	a1000
B	arbre_B	11	a100011
C	arbre_B	10	a10001110
A			
B	arbre_A	0	a100011100
A	arbre_B	0	a1000111000
B	arbre_A	0	a10001110000
C	arbre_B	10	a1000111000010
A			

fig 4.19: Codage en utilisant les arbres de succession

On remarque que la lettre A qui succède à C ne change pas le code cumulé et qu'aucune émission ne se produit. Ceci est dû au fait que le caractère C n'a que A comme seul successeur possible dans notre message. À chaque fois qu'un C est rencontré, un A le suit automatiquement. Le décodeur en tient compte aussi pour générer automatiquement un A après chaque C rencontré. Le processus de décodage de la séquence de bits a1000111000010 est décrite dans le tableau de la figure 4.20.

Code	Arbre utilisé	Décodage
'a'		A
1	arbre_A	A
0	arbre_A	B
0	arbre_B	A
0	arbre_A	B
11	arbre_B	B
10	arbre_B	CA
0	arbre_A	B
0	arbre_B	A
0	arbre_A	B
10	arbre_B	CA

fig 4.20: Décodage utilisant les arbres de succession

4.3.3 Algorithme de Lempel, Ziv et Welch

L'approche originale de cet algorithme fut publiée en 1977 par Lempel et Ziv [ZIV 77; ZIV 78]. Elle est basée sur des études sur le codage de source publié par ZIV en 1972 [ZIV 72]. Terry Welch l'a plus tard adaptée en 1984 [WELCH 84]. Le schéma de codage universel présenté dans ces articles est une technique pour compresser tout genre d'entrée de toute source discrète. L'algorithme est étrangement simple. Son principe est de remplacer des chaines de symboles par des codes de longueurs fixes sans faire d'analyse préalable de l'entrée. La chaine d'entrée est examinée en une seule passe. La compression est donc faite à la volée. L'idée de base est de mémoriser chaque nouvelle chaine de symboles recontrée dans une table de chaines et de lui attribuer un code unique. Ce code sert au codage futur de la chaine lors d'une nouvelle apparition. En effet, chaque fois que l'on traite une chaine et que cette dernière est connue (se trouve dans la table des chaines), on transmet son code existant sans en créer un nouveau. Pour avoir un codage optimal, c'est-à-dire pour coder des suites de symboles ayant la plus grande longueur possible, on tente toujours d'allonger en taille une chaine connue, en lui rajoutant un autre caractère (le prochain dans la chaine d'entrée). La table de codage, que nous avons appelée table des chaines, contient les suites de symboles rencontrées dans le message à compresser. Elles reflètent donc les statistiques de la chaine d'entrée. Comme pour l'algorithme de Huffman, cette table permet le décodage ultérieur du code reçu. Il faut, par conséquent la transmettre au décompresseur ou la lui faire connaitre. L'originalité de LZW est justement de ne pas transmettre cette dernière, mais de la reconstruire dynamiquement à la réception des codes. C'est ce que nous verrons dans l'algorithme de décompression. Pour ce faire, il faut que les codes déjà reçus permettent de créer les nouveaux codes et que chaque code reçu soit déjà décodé. C'est ce qui explique la séquence dans l'algorithme. En effet, on remarque un décalage entre le code émis et le code ajouté dans la table de codage. L'algorithme LZW dans sa forme simple est le suivant:

```
initialiser la table de codage avec les symboles de l'alphabet
lire caractère
chaine = caractère
tant qu'il y a des caractères en entrée
faire
    lire caractère
    si concaténation(chaine,caractère) dans table des chaines
    alors
        chaine = concaténation(chaine,caractère)
    sinon
        sortir code de la chaine
        ajouter concaténation (chaine,caractère) dans table des chaines
        chaine = caractère
    finsi
fait
sortir code de la chaine
```

Algorithme de compression LZW

On remarque qu'au moment du codage, le code émis est toujours un code figurant déjà dans la table et que le nouveau code placé dans la table de codage peut être reconstruit à partir des codes émis précédemment. Pour illustrer cet algorithme, prenons par exemple un alphabet de trois lettres {A,B,C} et le message AABABBCABABCA. Initialement, la table de codage contient tous les symboles de l'alphabet et leur codage respectif, à savoir (A,0), (B,1) et (C,2). La première lettre 'A' est lue. À la lecture de la deuxième lettre 'A', la chaine 'AA' est formée et cherchée dans la table. Le résultat de la recherche étant négatif, le code 0 est alors transmis et la suite 'AA' est ajoutée à la table avec le code 3. La chaine est alors initialisée à la lettre 'A' qui n'a pas été codée. À la lecture de 'B', le même processus se reproduit. La suite 'AB' est formée et cherchée dans la table. Ne s'y trouvant pas, le code 0 de 'A' est transmis, la chaine 'AB', avec le code 4, est ajoutée à la table de codage et 'B' réinitialise la chaine. Lorsque la quatrième lettre est reçue, 'BA' est ajoutée à la table avec le code 5 et le code 1 de 'B' est transmis. À la réception de 'B', la suite 'AB' est formée et recherchée dans la table. Étant déjà dans la table des chaines, 'AB' devient une sous-chaine pour la suite de symboles de l'itération suivante et un autre caractère est lu. À la lecture de 'B', la chaine 'ABB' est formée et recherchée dans la table. Comme elle ne s'y trouve pas, elle y est rajoutée avec le code 6 et le code de la chaine 'AB' est transmis. Le scénario de codage du message au complet est illustré dans le tableau de la figure 4.21. Notons que le code 3 pour 'AA', 5 pour 'BA', 6 pour 'ABB', 7 pour 'BC', 9 pour 'ABA' et 10 pour 'ABC' n'ont jamais été utilisés dans le codage du message de l'exemple. Il aurait fallu un message plus long pour permettre une nouvelle apparition de ces suites et par conséquent une utilisation de ces codes. L'algorithme LZW est donc plus efficace pour des messages longs. La première partie d'une chaine d'entrée sert toujours à initialiser la table des codes et à l'adapter aux statistiques de l'entrée. De petites chaines d'entrée sont pénalisées par une efficacité médiocre de LZW.

Message	Émission	Ajout à la table
		(A,0) (B,1) (C,2)
A		
A	0 (A)	(AA,3)
B	0 (A)	(AB,4)
A	1 (B)	(BA,5)
B		
B	4 (AB)	(ABB,6)
C	1 (B)	(BC,7)
A	2 (C)	(CA,8)
B		
A	4 (AB)	(ABA,9)
B		
C	4 (AB)	(ABC,10)
A	8 (CA)	

fig 4.21: Codage et construction de la table de codage

La décompression LZW est le processus inverse de la compression. Elle utilise les codes de sortie pour créer la chaine d'entrée sans pour autant avoir la table de codage; elle la définit au fur et à mesure, comme c'est le cas à la compression. L'algorithme de décompression LZW dans sa forme simple est le suivant:

```
initialiser la table de codage avec les symboles de l'alphabet
lire code
sortir chaine(code)
tant qu'il y a des codes en entrée
faire
    lire nouveau_code
    si nouveau_code dans table de codage
    alors
        chaine = chaine(nouveau_code)
    sinon
        chaine = chaine(code)
        chaine = concaténation(chaine, caractère)
    finsi
    sortir chaine
    caractère = premier caractère de chaine
    ajouter concaténation(chaine(code), caractère) dans table de codage
    code = nouveau_code
fait
```

Algorithme de décompression LZW

Ainsi, comme pour la compression, une nouvelle chaine est rajoutée à la table des chaines à chaque réception d'un nouveau code. Pour les codes connus, une simple conversion du code est faite vers la suite de symboles qu'il représente. À chaque réception d'un code, une nouvelle chaine est rajoutée à la table des chaines. Cette nouvelle chaine est constituée de l'ancienne chaine codée concaténée au premier caractère de la chaine que l'on vient de décoder. Dans le tableau de la figure 4.22, nous illustrons le décodage de 001412448.

Code	Décodage	Ajout à la table
		(A,0) (B,1) (C,2)
0	A	
0	A	(AA,3)
1	B	(AB,4)
4	AB	(BA,5)
1	B	(ABB,6)
2	C	(BC,7)
4	AB	(CA,8)
4	AB	(ABA,9)
8	CA	(ABC,10)

fig 4.22: Décodage et construction de la table de codage

La partie ¨sinon¨ dans l'algorithme ci-haut, est utilisée dans un cas particulier qui cause certains problèmes. En effet, il arrive parfois que le décodeur reçoit des codes inconnus. Ce cas se présente lorsqu'on fait face à une suite de symboles ayant une séquence de la forme *CchaineCchaineC* et que *Cchaine* existe déjà dans la table de codage où *chaine* est une chaine quelconque et *C* un caractère quelconque. En effet, en traitant *'CchaineC'*, le compresseur émet le code de *'Cchaine'* et rajoute *'CchaineC'* à la table des chaines. *'C'* initialise ensuite la nouvelle chaine pour encore former la suite *'CchaineC*'. Le compresseur émet alors le code de *'CchaineC'*. C'est le seul cas, au codage, où un code créé à l'étape i est émis à l'étape i+1. Il représente aussi le seul cas, au décodage, où un code reçu n'est pas encore dans la table des chaines. Le décompresseur ne pourra pas décoder le code de *'CchaineC'* en le recevant puisque ce code ne sera pas encore dans sa table de codage. En effet, il ne connaitra pas

encore une extension de la supposée chaine précédente. Un exemple s'impose. Le tableau suivant illustre le codage et le décodage du message ACACBCACACAB avec l'alphabet de trois lettres {A,B,C}.

CODAGE			DÉCODAGE		
Message	Code émis	Ajout à la table	Code reçu	Décodage	Ajout à la table
		(A,0) (B,1) (C,2)			(A,0) (B,1) (C,2)
A					
C	0 (A)	(AC,3)	0	A	
A	2 (C)	(CA,4)	2	C	(AC,3)
C					
B	3 (AC)	(ACB,5)	3	AC	(CA,4)
C	1 (B)	(BC,6)	1	B	(ACB,5)
A					
C	4 (CA)	(CAC,7)	4	CA	(CB,6)
A					
C					
A	7 (CAC)	(CACA,8)	7	CAC	(CAC,7) ◄
B	0 (A)	(AB,9)	0	A	(CACA,8)
	1 (B)		1	B	(AB,9)

fig 4.23: Illustration du codage et décodage LZW

On remarque qu'il y a un léger décalage entre l'ajout d'un nouveau code dans la table des chaines du codeur par rapport à celui ajouté dans la table des chaines du décodeur. Le cas particulier est celui de la réception du code 7 par le décodeur puisque ce dernier ne le connait pas encore. Ceci est du au fait que la génération du code 7 de la suite 'CAC' dans le codage survient à l'étape qui précède immédiatement son émission. En recevant le code inconnu 7, le décompresseur le génère dans sa table en utilisant les codes précédents, ceci en prenant la chaine du dernier code reçu (4='CA') et en concaténant sa chaine respective avec son premier caractère, cela donne la suite 'CAC'. Heureusement, c'est le seul cas où le décompresseur rencontre un code indéfini. Tous les préfixes d'une chaine, apparaissant dans la table des chaines, se trouvent aussi dans la table. Les préfixes d'une chaine de caractères sont toutes les suites de caractères commençant au début de la chaine et ayant une longueur inférieure à celle de la chaine. Ceci est une propriété caractéristique fondamentale de la table de codage avec LZW. Puisqu'en enlevant le dernier symbole de n'importe quelle chaine dans la table de codage on obtient une autre chaine qui existe obligatoirement dans la table, on peut représenter chaque chaine par deux codes. Le premier identifie la sous-chaine qui existe déjà dans la table et le deuxième représente le dernier caractère. La table a ainsi une entrée à taille fixe (deux codes) quelle que soit la taille de la chaine. Cette technique revient à une technique de compression (voir exploitation de l'ordre des données dans 4.2.2.2) qui permet d'avoir moins d'espace en mémoire lors de la compression. Elle permet aussi d'utiliser une plus grande table de codage donc un codage plus efficace. Notons que le code généré par LZW peut être de longueur quelconque. Mais une fois choisie, cette longueur doit rester fixe et être partagée par le codeur et le

décodeur. Le code doit néanmoins avoir plus de bits que le nombre de bits nécessaires pour coder un unique symbole de l'alphabet. Si l'alphabet est la table ASCII, comme c'est souvent le cas, le code doit avoir plus de huit bits parce que les 256 premiers codes sont utilisés pour coder les différents symboles. Les codes suivants sont alloués aux chaines traitées par l'algorithme. Ainsi, lorsqu'on utilise 12 bits pour le code de sortie, et c'est le cas par défaut de l'algorithme LZW, les codes de 0 à 255 sont réservés aux symboles individuels et les codes 256 à 4 095 sont alloués succéssivement aux chaines traitées.

A	0	A	0
B	1	B	1
C	2	C	2
AA	3	0A	3
AB	4	0B	4
BA	5	1A	5
ABB	6	4B	6
BC	7	1C	7
CA	8	2A	8
ABA	9	4A	9
ABC	10	4C	10

fig 4.24: Compression de la table de codage

Le modèle présenté avec LZW est un modèle adaptatif. Il s'adapte dynamiquement au fur et à mesure de la réception des symboles d'entrée. Ceci engendre un «en-tête de rodage» qui est une première partie dans la chaine d'entrée au cours de laquelle la table de codage est initialisée. Cet en-tête de rodage peut être plus ou moins long dépendamment de la répartition des symboles et des suites de symboles. Une répercussion importante est alors ressentie avec des messages courts. En effet, de petites chaines d'entrée sont pénalisées par une efficacité médiocre puisque la table de codage, qui est adaptative, «n'a pas le temps» d'être optimale. Plusieurs codes sont créés mais jamais ré-utilisés, comme le montre notre exemple dans la figure 4.21.

Schématiquement, LZW remplace des chaines de symboles d'une longueur sans limite établies par des codes d'une longueur fixe. La longueur de ce code varie généralement entre neuf et seize bits. On peut ainsi coder entre 512 et 65 536 chaines différentes. Les premières combinaisons du code étant en fait réservées pour les 256 symboles de la table ASCII, c'est seulement le reste des combinaisons qui servent à la codification des chaines. Ainsi en utilisant un codage sur neuf bits, on peut coder 256 chaines alors qu'avec seize bits, on code 65 280 suites de symboles. Selon le codage choisi, une taille différente de mémoire est nécessaire pour l'algorithme. En utilisant de longs codes, on peut théoriquement avoir un meilleur taux de compression en compressant une longue chaine d'entrée. Ceci est dû au fait qu'on utilise alors de plus grandes tables de codage. Dans notre implantation, nous avons testé les codifications de neuf à quatorze bits pour étudier leur faisabilité dans le cadre de la Carte Santé. Nous avons trouvé un compromis entre la taille de mémoire nécessaire, le taux de compression et la taille du code. L'algorithme LZW est relativement simple à implanter, mais nous nous sommes confrontés à deux problèmes majeurs. Le premier concerne la taille de la table de codage. Plus le taille du code est grande, plus l'espace nécessaire pour la table de codage est important. Avec des chaines de tailles variables, cet espace peut devenir trop important. Pour résoudre ce problème, nous avons mémorisé la table sous forme compressée. Chaque chaine dans la table, quelle que soit sa taille, est représentée par une paire (code, caractère). Le caractère est le dernier caractère de la chaine, alors que le code

décodeur. Le code doit néanmoins avoir plus de bits que le nombre de bits nécessaires pour coder un unique symbole de l'alphabet. Si l'alphabet est la table ASCII, comme c'est souvent le cas, le code doit avoir plus de huit bits parce que les 256 premiers codes sont utilisés pour coder les différents symboles. Les codes suivants sont alloués aux chaines traitées par l'algorithme. Ainsi, lorsqu'on utilise 12 bits pour le code de sortie, et c'est le cas par défaut de l'algorithme LZW, les codes de 0 à 255 sont réservés aux symboles individuels et les codes 256 à 4 095 sont alloués succéssivement aux chaines traitées.

A	0	A	0
B	1	B	1
C	2	C	2
AA	3	0A	3
AB	4	0B	4
BA	5	1A	5
ABB	6	4B	6
BC	7	1C	7
CA	8	2A	8
ABA	9	4A	9
ABC	10	4C	10

fig 4.24 : Compression de la table de codage

Le modèle présenté avec LZW est un modèle adaptatif. Il s'adapte dynamiquement au fur et à mesure de la réception des symboles d'entrée. Ceci engendre un «en-tête de rodage» qui est une première partie dans la chaine d'entrée au cours de laquelle la table de codage est initialisée. Cet en-tête de rodage peut être plus ou moins long dépendamment de la répartition des symboles et des suites de symboles. Une répercussion importante est alors ressentie avec des messages courts. En effet, de petites chaines d'entrée sont pénalisées par une efficacité médiocre puisque la table de codage, qui est adaptative, «n'a pas le temps» d'être optimale. Plusieurs codes sont créés mais jamais ré-utilisés, comme le montre notre exemple dans la figure 4.21.

Schématiquement, LZW remplace des chaines de symboles d'une longueur sans limite établies par des codes d'une longueur fixe. La longueur de ce code varie généralement entre neuf et seize bits. On peut ainsi coder entre 512 et 65 536 chaines différentes. Les premières combinaisons du code étant en fait réservées pour les 256 symboles de la table ASCII, c'est seulement le reste des combinaisons qui servent à la codification des chaines. Ainsi en utilisant un codage sur neuf bits, on peut coder 256 chaines alors qu'avec seize bits, on code 65 280 suites de symboles. Selon le codage choisi, une taille différente de mémoire est nécessaire pour l'algorithme. En utilisant de longs codes, on peut théoriquement avoir un meilleur taux de compression en compressant une longue chaine d'entrée. Ceci est dû au fait qu'on utilise alors de plus grandes tables de codage. Dans notre implantation, nous avons testé les codifications de neuf à quatorze bits pour étudier leur faisabilité dans le cadre de la Carte Santé. Nous avons trouvé un compromis entre la taille de mémoire nécessaire, le taux de compression et la taille du code. L'algorithme LZW est relativement simple à implanter, mais nous nous sommes confrontés à deux problèmes majeurs. Le premier concerne la taille de la table de codage. Plus le taille du code est grande, plus l'espace nécessaire pour la table de codage est important. Avec des chaines de tailles variables, cet espace peut devenir trop important. Pour résoudre ce problème, nous avons mémorisé la table sous forme compressée. Chaque chaine dans la table, quelle que soit sa taille, est représentée par une paire (code, caractère). Le caractère est le dernier caractère de la chaine, alors que le code

est le code d'une autre chaine se trouvant déjà dans la table. En effet, chaque chaine de symboles est une combinaison d'une chaine existante dans la table et d'un nouveau caractère.

Chaîne	Code
A	0
B	1
C	2
AB	3
ABC	4
CB	5
CBA	6
CBAA	7
BC	8
CBAAB	9
ABCA	10

Chaîne	Code
A	0
B	1
C	2
0B	3
3C	4
2B	5
5A	6
6A	7
1C	8
7B	9
4A	10

fig 4.25 : Compression de la table de codage

Le deuxième problème concerne la recherche de chaines. En effet, avant de coder une chaine, l'algorithme cherche cette dernière dans la table de codage. Cette recherche peut être très pénalisante lorsque la table est très grande. Une des solutions possible serait de trier la table de chaines pour pouvoir faire une recherche dichotomique et ainsi réduire le nombre de comparaisons de chaines. La recherche de chaque chaine demande alors un nombre de comparaisons de l'ordre de $\log_2$(nombre de chaines). Nous avons choisi d'implanter un algorithme de hash-code pour réduire davantage le nombre de comparaisons. L'indice d'une chaine dans la table de codage est calculé à partir de la chaine elle-même. Ainsi, une chaine recherchée peut être trouvée, avec un peu de chance, après une seule tentative. Lorsqu'une nouvelle chaine et son code doivent être rajoutés à la table, la fonction Hash-code calcule un indice dans la table à partir des symboles de la chaine. Si à l'adresse générée figure une autre chaine, la fonction gère un nouvel indice jusqu'à trouver une adresse libre dans la table. Pour permettre à la fonction hash-code de trouver plus facilement une case vide, nous avons agrandi la table de codage de 25% par rapport à la taille qu'elle devait avoir selon le code utilisé. En choisissant une bonne fonction de hash-code et une taille de table de codage initiale adéquate, on peut diminuer la moyenne du nombre de recherches d'une façon très importante.

Avec des messages relativement très petits, les données manipulées dans un dossier médical portable sur une carte à puce ne peuvent pas toujours être compressées avec efficacité en utilisant LZW.

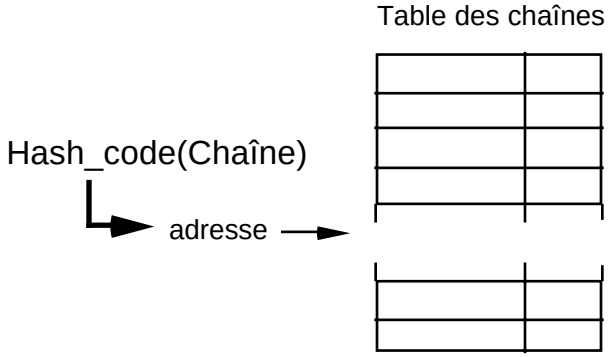


fig 4.26 : Calcul de l'indice par la fonction Hash_code

4.3.4 Codage arithmétique

Avec le codage de Huffman, on remplace chaque symbole d'entrée par un code spécifique. Le codage arithmétique contourne cette stratégie, qui fut longtemps l'état de l'art de la compression, en remplaçant des suites de symboles en entrée par des nombres réels en sortie [WITTEN 87]. Ceci rappelle le codage de Lempel, Ziv et Welch, bien que ceux-ci codent une suite de symboles par un nombre entier de taille fixe. Le code de sortie, avec le codage arithmétique, est toujours un nombre réel inférieur à 1 et supérieur ou égal à 0. Avec ce nombre réel, on peut décoder et reconstruire de façon unique la chaine d'entrée d'origine. Chaque message est donc représenté par un nombre dans l'intervalle [0,1[. Plus la taille du message est importante, plus petit est l'intervalle nécessaire pour le représenter et plus grand est le nombre de bits nécessaires pour spécifier cet intervalle. L'idée de base du codage arithmétique est que, partant de l'intervalle [0,1[, chaque symbole entrant d'un message réduit la taille de l'intervalle en accord avec les probabilités d'apparition du symbole spécifié par le modèle. Avant de coder un message, le domaine de ce dernier est [0,1[. Chaque fois qu'un symbole du message est traité, ce domaine est rétréci à sa portion allouée pour le symbole. Prenons par exemple le message COMPACTAGE, dont les symboles appartiennent à l'alphabet {A,C,E,G,M,O,P,T}. Soit le modèle figé de cet alphabet défini dans la figure 4.27, le domaine de chaque symbole est déterminé en prenant une portion du domaine global proportionellement à la probabilité d'apparition de ce symbole. L'ordre d'attribution des portions du domaine aux différents symboles n'a pas d'importance tant que le codeur et le décodeur le font de la même manière. Initialement, le domaine du message est [0,1[. À la réception du premier symbole C, le codeur rétrécit le domaine à [0.2,0.3[qui est le domaine qu'alloue le modèle à ce symbole. Ce nouveau domaine est divisé en portions redistribuées aux différents symboles au prorata de leurs probabilités. Lorsque le deuxième symbole est reçu, à savoir le O, le domaine est rétréci à son dixième puisque selon le modèle, le domaine de O est [0.7,0.8[. Ceci nous donne le nouveau domaine [0.27,0.28[. L'algorithme de codage d'un message de n'importe quelle longueur est le suivant:

Alphabet = {a,c,e,g,m,o,p,t}

Sybole	Probabilité	Domaine
A	2/10	[0.0,0.2[
C	1/10	[0.2,0.3[
E	2/10	[0.3,0.5[
G	1/10	[0.5,0.6[
M	1/10	[0.6,0.7[
O	1/10	[0.7,0.8[
P	1/10	[0.8,0.9[
T	1/10	[0.9,1.0[

fig 4.27 : Modèle figé pour l'alphabet {a,c,e,g,m,o,p,t}

```
inférieur = 0.0
supérieur = 1.0
tant qu'il y a des symboles en entrée
faire
    lire symbole
    supérieur = inférieur + (supérieur-inférieur)* supérieur(symbole)
    inférieur = inférieur + (supérieur-inférieur)* inférieur(symbole)
fait
code = inférieur
```

Algorithme de codage arithmétique

En procédant de cette manière pour coder le message COMPACTAGE, le domaine se rétrécit comme suit:

initialement	: [0	,1[
réception de C	: [0.2	,0.3[
réception de O	: [0.27	,0.28[
réception de M	: [0.276	,0.277[
réception de P	: [0.2768	,0.2769[
réception de A	: [0.2768	,0.27682[
réception de C	: [0.276804	,0.276806[
réception de T	: [0.2768058	,0.276806[
réception de A	: [0.2768058	,0.27680584[
réception de G	: [0.27680582	,0.276805824[
réception de E	: [0.2768058212	,0.2768058216[

Évolution du domaine du message COMPACTAGE

La figure 4.28 illustre clairement ce processus. La barre verticale représente le domaine avec les portions attribuées à chaque symbole de l'alphabet en tenant compte de leurs probabilités. De la gauche vers la droite, à la réception de chaque symbole, une nouvelle barre représente le nouveau domaine qui est un agrandissement de la portion allouée au dernier symbole reçu dans l'ancien domaine.

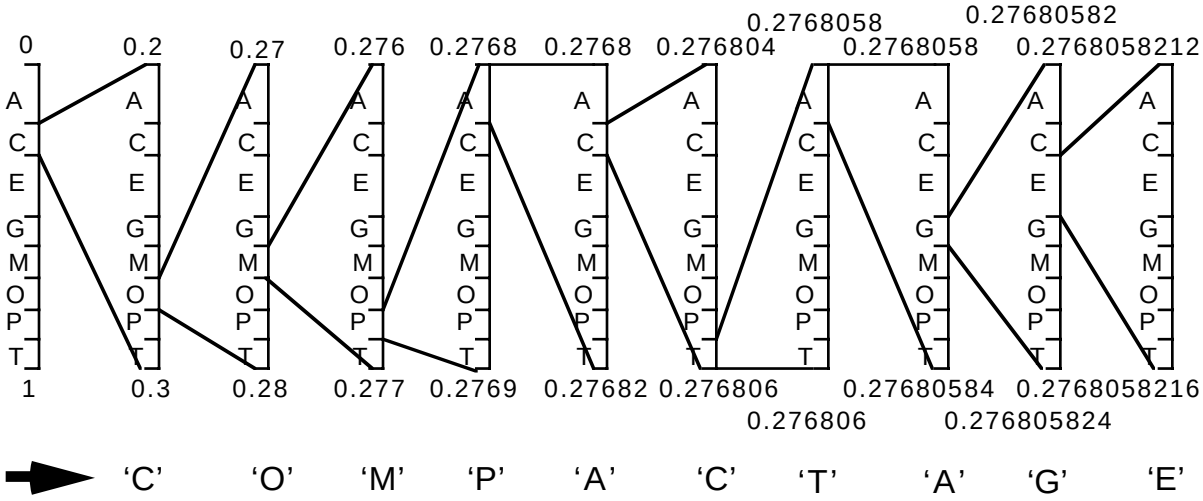


fig 4.28 : Représentation du processus du codage arithmétique

Tout nombre dans l'intervalle [0.2768058212,0.2768058216[code d'une façon unique le mesage COMPACTAGE. Par exemple, 0.2768058214 fait bien l'affaire. Dans l'algorithme décrit ci-haut, on a pris la borne inférieure du domaine comme étant le code du message. En recevant le nombre 0.2768058214, le décodeur sait tout de suite que le premier symbole est le C, puisque ce nombre se trouve dans l'intervalle attribué dans le modèle comme domaine au symbole C. En enlevant l'effet des bornes de l'intervalle de C sur ce nombre, le décodeur peut réitérer pour identifier les autres symboles de la même façon. L'algorithme de décodage est le suivant:

```
lire nombre
faire
    chercher le symbole dont l'intervalle englobe nombre
    sortir symbole
    domaine = supérieur(symbole) - inférieur(symbole)
    nombre = (nombre - inférieur(symbole))/domaine
fait  jusqu'à ce qu'il n'y ait plus de symbole
```

Algorithme de décodage arithmétique

Le processus de décompression est simple et ne fait qu'inverser les calculs effectués au codage. Il demeure néanmoins un problème lié à la détection de la fin d'un message. Le décompresseur doit savoir quand arrêter le décodage. Pour résoudre ceci, chaque message se termine par un symbole spécial de fin de message connu par le codeur et le décodeur. Ce symbole est ajouté à l'alphabet et on lui attribue une probabilité et un domaine. Le codage arithmétique tel que présenté ne traite que les symboles pris les uns indépendamment des autres, sans tenir compte de leur contexte. Il est donc efficace pour les sources d'ordre 1. Lorsque le contexte influence la distribution des symboles, l'efficacité du codage arithmétique, tel que présenté, diminue puisque le codage arithmétique exploite les fréquences d'occurrences des symboles dans la chaine d'entrée. Les différents types de redondances ne sont donc pas tous pris en considération. Nous verrons plus loin une adaptation permettant de

tenir compte du voisinage d'un symbole pour évaluer la probabilité de ce dernier.

À première vue, le codage arithmétique semble astucieux, pas très compliqué et facile à réaliser, mais tel que présenté, il apparait impraticable. En effet, rares sont les ordinateurs qui supportent des réels de plus de 80 bits. Le programme de codage arithmétique que nous avons testé accomplit le codage avec des entiers standards de 16 bits sans manipuler des réels pour représenter la borne inférieure et la borne supérieure des domaines. L'algorithme de codage décrit plus haut ne transmet rien tant que le message en entrée n'a pas été codé en entier. De même, le décodeur ne peut pas commencer la décompression tant que l'entière transmission n'a pas été terminée. Doit-on alors interrompre et recommencer le processus après chaque séquence de symboles d'une longueur détérminée? La solution est d'utiliser un schéma de transmission et de réception incrémentales dans lequel des variables entières de longueur fixe reçoivent des nouveaux bits par le poids faible, sont décalées et rejettent des bits par le poids fort. De telles variables peuvent former un nombre aussi long en bit que la mémoire le permet. Dans l'algorithme du codage arithmétique présenté, nous avons utilisé un domaine avec une borne inférieure égale à 0.0 et une borne supérieure égale à 1.0. Dans notre implantation, *supérieur* et *inférieur* sont représentés par des entiers sur 16 bits et sont respectivement égal à 0xFFFF et 0x0000 en hexadécimal. À la réception de chaque symbole de la source, *supérieur* et *inférieur* sont recalculés de la façon suivante:

```
supérieur = inférieur + (supérieur-inférieur) * supérieur(symbole)
inférieur = inférieur + (supérieur-inférieur) * inférieur(symbole)
```

Ce processus continue à chaque réception d'un symbole. On remarque alors que *supérieur* et *inférieur* convergent vers un même entier. Lorsque les bits à gauche du poids fort de *supérieur* et d'*inférieur* deviennent les mêmes, ceux-ci sont immédiatement transférés dans le code de sortie puisqu'aucun rétrécissement futur ne peut les affecter. Un décalage de bits est alors effectué dans *supérieur* comme dans *inférieur*.

Prenons l'exemple de la figure 4.27. Supposons, pour simplifier, que nous disposons de deux registres de cinq chiffres pour représenter les bornes supérieure et inférieure. Initialement, *supérieur* égale 99999 et *inférieur* égale 00000. Mais théoriquement, les deux continuent indéfiniment à droite respectivement des 9s et des 0s. Lorsqu'on reçoit le premier caractère 'C' du message COMPACTAGE, on calcule la différence des deux registres. On obtient 99999 auquel on ajoute 1 parce qu'on suppose une infinité de 9s dans *supérieur*. Avec les deux formules ci-haut, on obtient une nouvelle valeur 30000 pour *supérieur* à laquelle on retranche 1 pour obtenir 29999, pour la même raison que tout à l'heure, ainsi qu'une nouvelle valeur 20000 pour *inférieur*. On remarque alors que le premier chiffre significatif de *supérieur* et le premier chiffre significatif de *inférieur* sont égals. Comme on est sûr qu'ils ne changeront pas, on les retranche des deux variables en faisant un décalage à gauche des chiffres. Le chiffre décalé est alors mis dans la variable représentant le code de sortie.

Dans ce qui suit, nous présentons le processus de calcul pour le message COMPACTAGE:

Codage	*supérieur*	*inférieur*	code de sortie cumulatif
État initial	99999	00000	
Codage de **C** [0.2,0.3[	29999	20000	
décalage du **2**	99999	00000	.2
Codage de **O** [0.7,0.8[	79999	70000	
décalage du **7**	99999	00000	.27
Codage de **M** [0.6,0.7[	69999	60000	
décalage du **6**	99999	00000	.276
Codage de **P** [0.8,0.9[	89999	80000	
décalage du **8**	99999	00000	.2768
Codage de **A** [0.0,0.2[	19999	00000	
Codage de **C** [0.2,0.3[	03599	00000	
décalage du **0**	35999	00000	.27680
Codage de **T** [0.9,1.0[	35999	32400	
décalage du **3**	59999	24000	.276803
Codage de **A** [0.0,0.2[	31199	24000	
Codage de **G** [0.5,0.6[	28319	27600	
décalage du **2**	83199	76000	.2768032
Codage de **E** [0.3,0.5[	79599	78160	
décalage du 7	95999	81600	.27680327
décalage du 9	59999	81600	.276803279
décalage du 5	99999	81600	.2768032795

fig 4.29 : Processus de codage du message ‘COMPACTAGE’

Dans l'implantation du programme, ces mêmes opérations sont effectuées en binaire.

Le décodeur procède de la même façon, mais il utilise trois nombres au lieu de deux; *supérieur*, *inférieur* et un troisième nombre contenant les bits en entrée courants. Ce nombre est toujours entre *inférieur* et *supérieur*, qui convergent d’ailleurs continuellement vers lui. Lorsqu'on détecte un symbole, les valeurs de *supérieur* et de *inférieur* sont décalées et par le fait même, s'éloignent de la valeur du troisième entier. En effectuant les mêmes tests de comparaison sur les bits de poids fort de *supérieur* et d'*inférieur*, le décodeur identifie le moment où il faut procéder à un décalage et ainsi décoder le symbole.

Le codage arithmétique utilise des domaines qu'il attribue aux symboles en tenant compte de leurs probabilités d'apparition dans la chaine d'entrée. Comme pour l'algorithme de Huffman, un problème se pose pour le calcul des probabilités des symboles. Les méthodes de compression statistiques procèdent généralement par un premier passage sur la chaine d'entrée pour calculer les fréquences d'apparition de chaque symbole de l'alphabet. C'est seulement lors d'un deuxième passage que la compression proprement dite est faite. Les statistiques relevées sont rajoutées aux données compressées pour permettre au décodeur de faire la décompression. Comme pour l'implantation de l'algorithme de Huffman, on a adopté la solution de la compression avec un modèle adaptatif. Le codeur et le

décodeur commencent avec le même modèle et, à chaque traitement d'un symbole, les statistiques du modèle sont remises à jour de part et d'autre. Tant que que l'algorithme de mise à jour du modèle opère d'une façon identique pour le codeur et le décodeur, nous sommes assurés d'avoir un même modèle pour la compression et la décompression et nous ne sommes plus obligés de transmettre la table de codage du compresseur au décompresseur. Pour le codage arithmétique, un modèle adaptatif engendre des inconvénients importants en ce qui conserne la mise à jour du modèle. En effet, lorsqu'on traite en codage ou décodage un symbole quelconque, la mise à jour dans le modèle, de sa probabilité d'apparition engendre une mise à jour des domaines de tous les symboles. Elle implique ainsi un nombre important d'opérations élémentaires. C'est ce qui fait la «lenteur» du codage arithmétique, en comparaison à d'autres méthodes de compression.

Comme pour l'algorithme de Huffman, nous avons aussi pris en compte le contexte de chaque symbole dans le calcul des probabilités d'apparition des caractères. Tout dépendant du type des données à compresser, cette façon de procéder peut donner de meilleurs taux de compression. Le fait de pouvoir prédire un caractère avec une bonne probabilité réduit le nombre de bits nécessaires à la codification. De plus, en élargissant le contexte, on améliore les prédictions. Par contre, pour mémoriser les statistiques en tenant compte du contexte des caractères, il faut disposer de beaucoup de mémoire de stockage, ce qui n'est pas toujours possible. Avec un modèle d'ordre 0, c'est-à-dire qui ne tient pas du tout compte du voisinage des caractères, il faut seulement disposer d'un espace de 256 entiers. Par contre, pour des modèles d'ordre 2 ou 3, il faut avoir des centaines de kilobits uniquement pour les statistiques. Le modèle d'ordre 1, qui tient compte du caractère précédant le symbole traité que nous avons utilisé pour nos tests, demande 65 536 entiers pour mémoriser les fréquences des symboles. Les modèles d'ordre supérieur n'ont donc pas été testés pour des raisons d'espace mémoire.

4.3.5 Algorithme LZHUF

LZHUF est un algorithme qui utilise un codage mixte ralliant la compression basée sur les dictionnaires et la compression statistique de Huffman. Les techniques de compression basées sur les dictionnaires sont toutes les variantes dérivées de l'algorithme de Lempel et Ziv de 1977 noté LZ77 [ZIV 77] puis de leur algorithme de 1978 noté LZ78 [ZIV 78]. LZW, vu précédemment, fait partie de ces variantes et il est dérivé de LZ78. LZHUF, qui a été écrit à l’origine par Haruyasu Yoshikazi en 1989 et dont le nom vient de Lempel, Ziv et HUFfman, est essentiellement l'algorithme utilisé dans le produit commercial LHARC. LZHUF combine donc la technique de compression de Lempel et Ziv pour coder des chaines de caractères et la technique de Huffman pour trouver la meilleure représentation des codes de sortie en se basant sur la distribution des probabilités d'apparition des chaines de caractères codées.

Plusieurs variations de l'algorithme de Lempel et Ziv ont été publiées. Toutes partagent l'idée de remplacer une chaine de caractères par un pointeur qui pointe vers l'endroit où la même chaine de caractères apparait précédemment dans le texte.

Tous ces algorithmes construisent un dictionnaire de chaines de caractères, appelées phrases, et remplacent les phrases dans le texte par des pointeurs vers l'index du dictionnaire. C'est de là que vient la dénomination «technique basée sur les dictionnaires». Ces techniques sont divisées en deux grandes familles: la famille des techniques dérivées de LZ77 et la famille de celles dérivées de LZ78. La distinction majeure de ces deux familles est que la première utilise des pointeurs à deux composantes pour identifier la phrase et sa longueur, alors les pointeurs de la seconde famille ne contiennent qu'un entier pour identifier une phrase précédemment sélectionnée dans la chaine d'entrée.

LZHUF utilise dans son code compressé un mélange de pointeurs et de caractères. Les caractères sont utilisés chaque fois que le pointeur tend à prendre plus d'espace que les caractères qu'il code. Le pointeur a normalement une taille fixe puisqu'il pointe toujours vers un ensemble de taille fixe de caractères qui précèdent la position de codage courante du message d'entrée. Cet ensemble est appelé fenêtre et a une taille donnée N. La fenêtre est mobile sur l'ensemble du message. La chaine remplacée par un pointeur a une taille maximum notée F. La fenêtre mobile est constituée de N-F caractères déjà codés et de F caractères appelés 'tampon de prévision'. Au début du codage, les N-F caractères sont initialisés arbitrairement par des blancs et les F premiers caractères du message d'entrée sont chargés dans le tampon de prévision. Pour coder un caractère, on recherche dans les N-F caractères la plus longue correspondance avec la chaine de caractère se trouvant dans le tampon de prévision. Si une correspondance est trouvée avec une chaine C, un pointeur est constitué à partir d'un déplacement dans la fenêtre jusqu'à C et à partir de la longueur de la chaine correspondante C. Si la taille du pointeur est inférieure à la longueur de C, le pointeur est transmis comme code et la fenêtre est décalée dans le message d'entrée par la longueur de C. Sinon, le premier caractère du tampon de prévision est transmis et la fenêtre est décalée d'une position dans le message d'entrée. Dans le cas où il n'y a pas de correspondance dans les F-N caractères de la chaine se trouvant dans le tampon de prévision, le premier caractère du tampon de prévision est transmis et la fenêtre est décalée d'une position dans le message d'entrée.

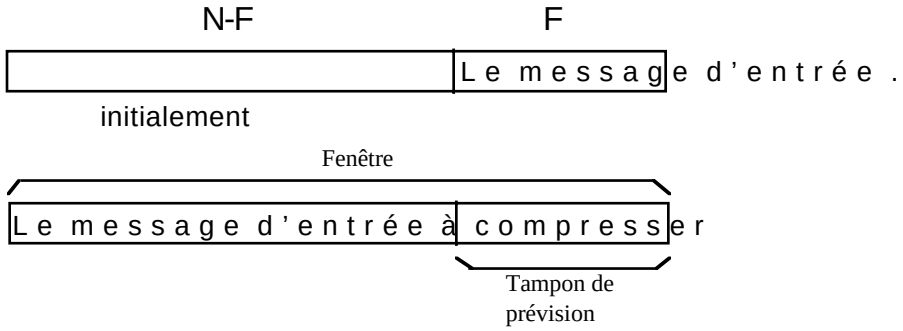


fig 4.30 : Exemple d'une fenêtre mobile avec N=32 et F=8

Pour que les pointeurs prennent le moins de place possible dans le codage, leurs représentations physiques en bit sont déterminées suivant leurs probabilités de distribution en utilisant le codage de Huffman. Un modèle adaptatif est utilisé pour éviter de transmettre la table de codage. Dans le code de sortie, un bit est rajouté à tous les pointeurs et tous les caractères pour pouvoir les distinguer lors du décodage. Le décodage est très simple. Il utilise une fenêtre au même titre que le codeur mais au lieu de rechercher une correspondance de la chaine du tampon de prévision dans les N-F caractères, il ne fait que recopier la chaine de caractères correspondante au pointeur reçu.

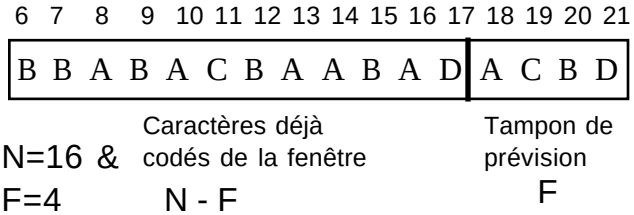


fig 4.31 : Fenêtre mobile sur le message d'entrée

Pour simplifier l'implantation de la fenêtre mobile sur le message d'entrée, il est plus intéressant de numéroter les caractères dans la fenêtre modulo N, N étant la taille de la fenêtre. Cela évite de faire des décalages consommateurs de temps d'exécution. Le décalage se fait implicitement en changeant les indices des cases après l'introduction des nouveaux symboles. Ceci se réalise au moyen d'un simple compteur (modulo N) indiquant à tout moment le tampon de prévision dans la fenêtre. De cette façon, le pointeur n'est plus constitué d'un déplacement mais simplement d'une position entre 0 et N-1 dans la fenêtre. En fait, seulement N-F positions sont utilisées. Les F positions à l'intérieur du tampon de prévision ne sont pas utilisées dans le codage des pointeurs et ceci ne détériore le taux de compression que si F n'est pas très petit devant N. Ces positions inutilisées peuvent néanmoins être employées pour le codage de messages particuliers comme la fin du message d'entrée par exemple.

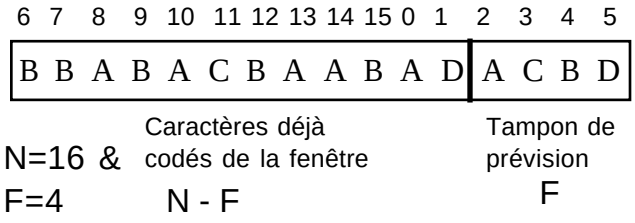


fig 4.32 : Fenêtre mobile avec une position modulo N

Dans un modèle adaptatif comme celui de LZHUF, le traitement qui est le plus consommateur de temps d'exécution est la recherche de correspondance entre le tampon de prévision et les chaines de caractères déjà codées. En effet, à chaque introduction d'un nouveau symbole dans la fenêtre, un ancien symbole la quitte, ce qui perturbe les résultats des anciennes recherches et nécessite de nouvelles prospections de correspondance. Cette exploration est d'autant plus importante que N est grand. Pour rentabiliser le temps de recherche, il est intéressant de disposer d'une liste triée pour effectuer des investigations dichotomiques.

Dans la figure 4.33, on remarque que l'insertion de la chaine ACBD du tampon de prévision nous donne directement la correspondance avec la plus longue chaine des caractères déjà codés, à savoir ACB qui est le préfixe de la chaine suivante de la liste. La plus longue correspondance avec le tampon de prévision est toujours

obtenue à partir du préfixe de la chaine qui précède ou qui suit la nouvelle chaine insérée, à savoir Cp et Cs dans la figure. Après seulement log N recherches, la correspondance est trouvée. Néanmoins, au fur et à mesure que la fenêtre se déplace sur le message d'entrée, les insertions et les suppressions de chaines dans la liste nécessitent un nombre important d'opérations. Un arbre de recherche binaire vient palier à ce problème. La figure 4.34 montre l'arbre correspondant à la liste triée précédente. Dans cet arbre, chaque noeud a deux fils; son fils gauche lui est inférieur alors que son fils droit lui est supérieur.

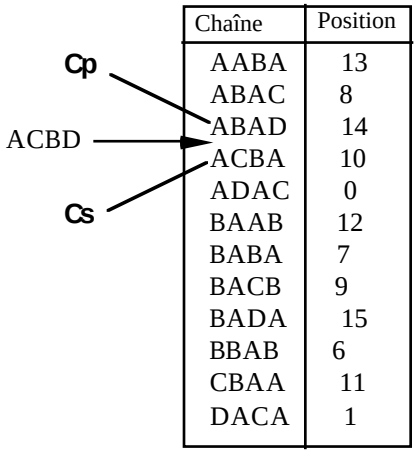


fig 4.33 : Liste triée des chaînes

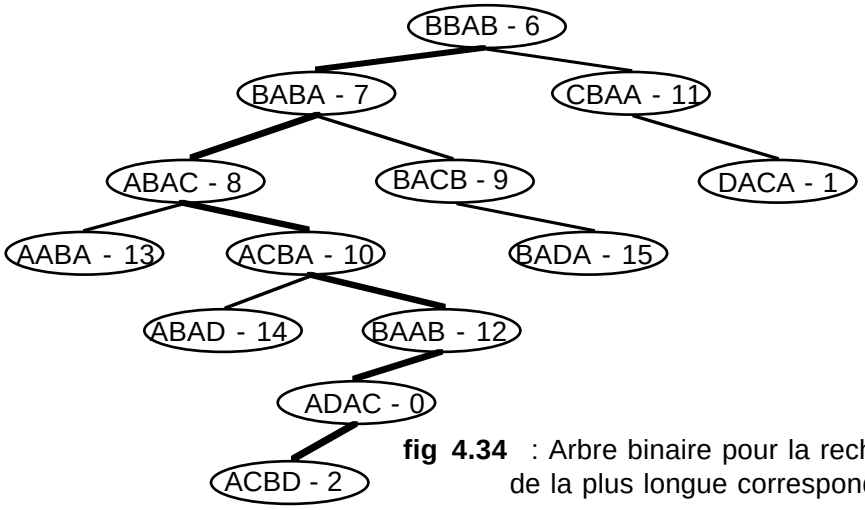


fig 4.34 : Arbre binaire pour la recherche de la plus longue correspondance

Comme on l'a vu avec une liste triée générale, la plus longue correspondance que l'on recherche se trouve être un préfixe de Cp ou Cs. Ces derniers se trouvent nécessairement dans le chemin de l'arbre qui mène au nouveau noeud inséré et correspondant à la chaine du tampon de prévision. L'un d'eux est le parent direct de ce noeud. Le second est trouvé en exécutant la règle suivante:

Si **Cp** est le parent direct alors **Cs** est le noeud où le chemin qui mène au nouveau noeud a tourné pour la dernière fois à gauche.
Si **Cs** est le parent direct alors **Cp** est le noeud où le chemin qui mène au nouveau noeud a tourné pour la dernière fois à droite.

Pour une fenêtre de taille N, l'arbre contient au maximum N noeuds. Ceci permet d'implanter l'arbre dans un tableau de N éléments. La structure de données utilisées pour l'implantation de l'algorithme comprend un premier vecteur contenant les N symboles en entrées (fenêtre) et une matrice contenant les N noeuds de l'arbre de recherche binaire. La matrice est constituée de trois vecteurs de N

éléments. Le premier contient la position des fils de gauche, le second contient la position des fils de droite et le dernier contient la position des parents. LZHUF utilise une seconde structure d'arbre pour manipuler l'arbre de Huffman, structure nécessaire à la codification des pointeurs (**voir Algorithme de Huffman**).

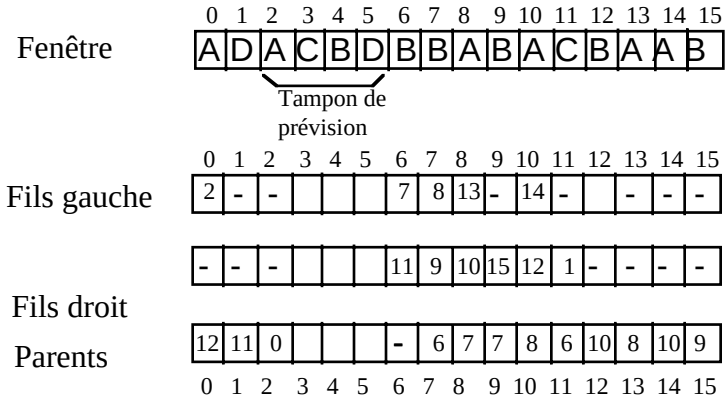


fig 4.35 : Structure de données utilisée pour représenter l'arbre binaire

Comme on peut l'observer dans la figure 4.34, l'arbre binaire peut être déséquilibré, ce qui peut réduire la performance de l'algorithme de recherche. Ce déséquilibre n'affecte pas le taux de compression mais augmente le temps d'exécution. Une amélioration de LZHUF serait d'introduire un processus pour équilibrer l'arbre binaire de recherche.

4.4 Les résultats de l'étude comparative

Les différentes techniques de compression présentées ont des efficacités de compression très différentes. L'efficacité d'une technique de compression est exprimée par un taux appelé taux de compression. Ce taux est un ratio qui mesure la diminution de la taille de la sortie par rapport à la chaine d'entrée. Dans la littérature, ce taux est généralement exprimé par:

$$\text{Taux de compression} = \frac{\text{Longueur de la chaîne d'entrée}}{\text{Longueur de la chaîne de sortie}} \qquad (1)$$

On le retrouve souvent sous différentes appellations comme degré de réduction des données, quotient de compression, ratio de compression ou efficacité de compression. Ce taux permet de comparer différentes méthodes de compression. On remarque selon l'équation (1), que plus le quotient est élevé, plus la méthode de compression employée est efficace. En prenant des tailles en bits identiques pour les entités en entrée et en sortie, ce ratio nous donne aussi le nombre moyen de symboles initiaux représentés par un code de sortie. Si par exemple ce taux vaut 2, cela signifie que le modèle de compression réduit en moyenne une séquence de deux caractères de la chaine initiale en un unique caractère dans la chaine comprimée. Dans la littérature, on peut aussi trouver d'autres ratios permettant de comparer

différentes méthodes de compression. Le chiffre de valeur en est un exemple.

$$\text{Chiffre de valeur} = \frac{\text{Longueur de la chaîne de sortie}}{\text{Longueur de la chaîne d'entrée}} \qquad (2)$$

Comme on peut le voir dans l'équation (2), le chiffre de valeur est l'inverse du taux de compression vu plus haut. Aux dires de certains auteurs, il exprime mieux le degré de réduction des données. Ce taux doit être inférieur à 1 pour que la méthode de compression soit efficace. La fraction des donnéees réduite peut aussi très facilement être déduite en retranchant le chiffre de valeur de l'unité.

$$\text{Gain} = 1 - \text{Chiffre de valeur} \qquad (3)$$

Le taux que l'on a utilisé dans nos travaux de compression ressemble beaucoup au ratio exprimé dans l'équation (2). Cependant, il est exprimé en pourcentage. Il mesure la fraction réduite des données de la chaine d'entrée. En d'autres termes, le pourcentage de compression, comme nous l'avons nommé, exprime le gain en volume effectué à la compression.

$$\text{Pourcentage de compression} = \frac{(\text{Taille du message initial} - \text{Taille du message compressé}) \times 100}{\text{Taille du message initial}} \qquad (4)$$

Le gain n'est pas toujours positif. On parle alors de perte. Le pourcentage de compression exprimé dans l'équation (4) est négatif dans ces cas. Lorsque le pourcentage est négatif, il y a augmentation et non diminution de l'espace requis pour les données. La méthode de compression est alors jugée non efficace. Ces cas sont courants lorsqu'on traite, en entrée, des chaines courtes.

4.4.1 Les données manipulées

Le pourcentage de compression, présenté dans le paragraphe précédent, permet de comparer plusieurs méthodes de compression de données et de les classer par ordre d'efficacité. Ce même taux est aussi utile pour comparer l'efficacité d'une même méthode sur des données différentes. On peut de cette manière, étudier la capacité de compression de différentes méthodes appliquées sur des données variées.

Pour comparer les différentes méthodes utilisables dans le cadre du projet des dossiers médicaux sur cartes à micro-processeur, nous avons utilisé dans notre étude des données de différents types. Dans l'ordre décroissant de l'entropie, les données utilisées sont des fichiers de bits aléatoires (E=7.989), des fichiers comportant des dossiers médicaux pré-codés sous forme binaire (E=7.680), des fichiers comportant des dossiers médicaux pré-codés avec des redondances artificielles (E=7.679), des fichiers exécutables (E=6.941) et des fichiers comportant des dossiers médicaux en texte libre (E=5.338). Les figures 4.36 à 4.39 illustrent les fréquences moyennes des 256 caractères de la table ASCII dans les fichiers de bits aléatoires, les

fichiers de dossiers médicaux pré-codés, les fichiers exécutables et les fichiers de dossiers médicaux en texte libre. Sur l'axe horizontal, nous avons les 256 caractères de la table ASCII alors que sur l'axe vertical sont représentés les pourcentages des fréquences d'apparition.

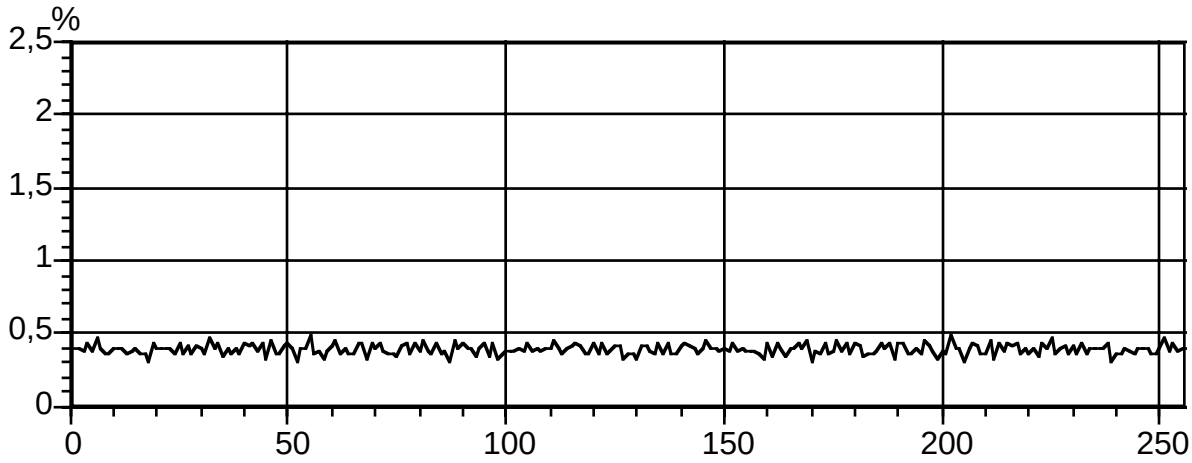


fig 4.36 : Répartition moyenne des caractères pour des données binaires de 65 536 octe

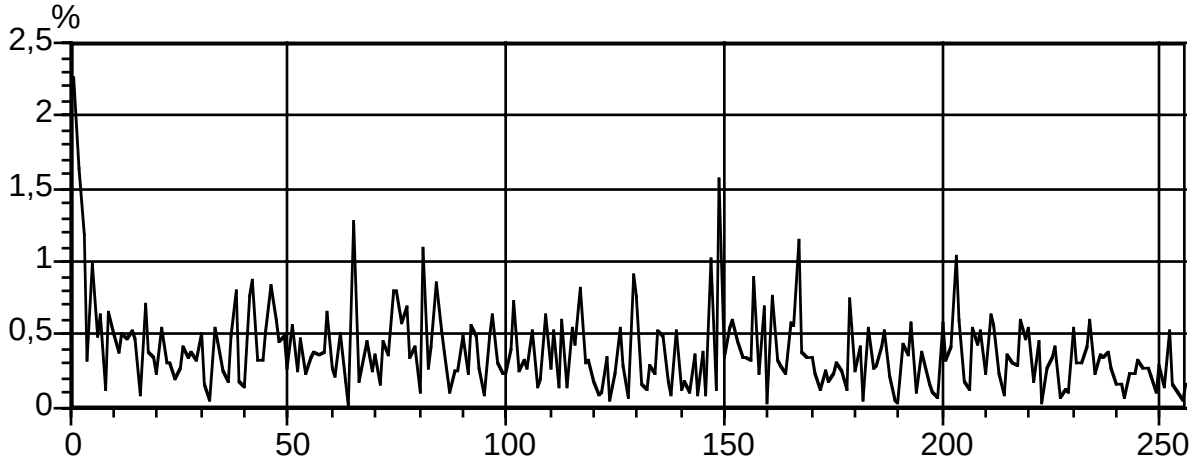


fig 4.37 : Répartition moyenne des caractères pour des dossiers médicaux pré-codés de 10 874 octets

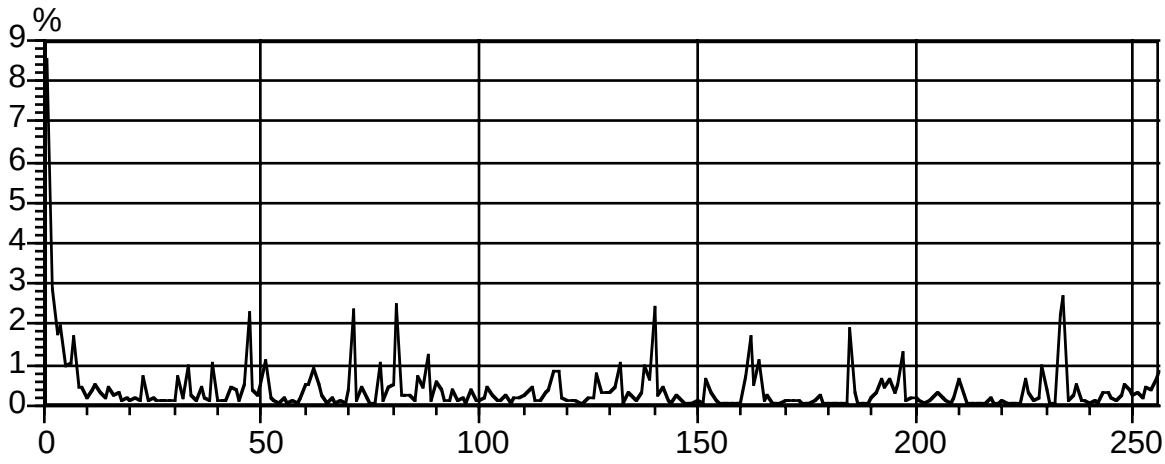


fig 4.38 : Répartition moyenne des caractères pour des exécutables de 65 536 octets

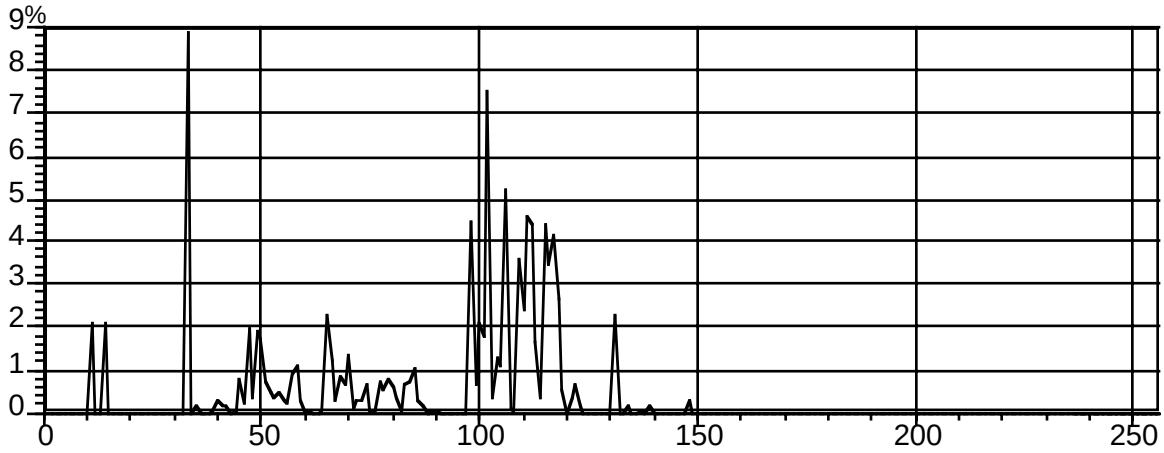


fig 4.39 : Répartition moyenne des caractères pour des dossiers médicaux en texte libre de 16 384 octets

Les bits aléatoires: Les messages composés de bits aléatoires sont les messages les plus dificilement réductibles. De part la faible redondance de ce genre de données, une compression efficace des messages constitués de bits aléatoires est très rare. Essayer d'avoir un rendement de compression positif avec des bits aléatoires représente encore un défi. Nous avons choisi ce genre de données malgré cette difficulté parce que les données réelles d'un dossier médical portable avec une structure optimale des données, s'y rapprochent sensiblement. L'étude de leur compression peut montrer la rentabilité d'incorporer un module de compression dans le projet Carte Santé.

Ces fichiers sont créés par une génération de bits aléatoires avec une distribution uniforme. Dans une équiprobabilité d'avoir le bit 1 ou 0, on obtient une distribution uniforme au niveau des caractères contenus dans les fichers. Tous les symboles de la table ASCII ont la même probabilité d'apparition dans de telles données.

Les dossiers médicaux pré-codés sous forme binaire: La structure de ces dossiers est basée sur le contenu de 12 dossiers réels. Elle ne correspond pas au DMP vu dans le chapitre II mais s'inspire des dossiers médicaux manuels utilisés chez certains praticiens. Ces fichiers se retrouvent d'ailleurs intégralement dans les fichiers comportants des dossiers médicaux en texte libre.

Nous avons introduit à l'intérieur d'un même fichier plusieurs dossiers médicaux de façon contiguë. La taille d'un dossier dans ce fichier est toujours multiple de huit bits (octets). Chaque dossier est composé de zones distinctes. Toutes les zones ne sont pas toujours présentes dans un dossier. Mais chaque dossier débute toujours par la zone identification. Il y a neuf zones possibles: identification, antécédents personnels, médication actuelle, suivi, laboratoire, radiologie, divers, autres médecins consultés et remarques. Chaque zone est composée par un identificateur, une taille en octets (maximum 4 095 octets) et de l'information. Suivant le type de la zone, la structure de l'information diffère. Chaque donnée est enregistrée d'une façon telle qu'elle prend le moins de bits possible. Les données sont donc sous forme pré-codée ou compressée à un premier niveau.

Les dossiers médicaux pré-codés avec redondance: Dans une population réelle, on remarque souvent une répétition dans les prescriptions médicales d'un patient donné. Dans un dossier médical portable, on peut donc retrouver des répétitions du nom d'un même médicament. Dans le but de se rapprocher d'avantage de la réalité des dossiers médicaux, nous avons donc introduit artificiellement des redondances de ce type dans plusieurs zones des dossiers médicaux utilisés dans les fichiers précédents.

Les fichiers exécutables: Les fichiers exécutables contiennent des patrons de bits arbitraires qui peuvent rappeler le contenu d'un dossier médical portable sur une carte à micro-processeur. Ils contiennent aussi des répétitions fréquentes de certains mots-clé ou instruction comme le saut (JMP) ou le chargement d'un registre (LGR). Cela peut s'apparenter à la répétition de certaines prescriptions médicales ou à la répétition d'un certain diagnostic ou traitement dans un vrai dossier médical.

Les dossiers médicaux en texte libre: Nous avons introduit dans ces fichiers les dossiers médicaux réels en texte libre tel que nous les avons trouvés dans les cabinets de médecins. Comme pour les dossiers médicaux pré-codés sous forme binaire, on retrouve dans un même fichier plusieurs dossiers contigus. Pour permettre une lecture plus aisée, nous avons introduit des séparateurs comme les titres des zones, le retour chariot (RC) et le caractère @. Ainsi, les différents dossiers sont séparés par la séquence RC@@RC, Les zones sont séparées par leur titre identificateur, les enregistrements d'une zone sont séparés par un RC et les champs sont séparés par le caractère @.

Mis à part la comparaison des différentes méthodes de compression, l'utilisation de texte libre comme chaine d'entrée pour nos algorithmes de compression implantés nous permet de déterminer l'opportunité d'augmenter ou de diminuer la redondance des données avant de tenter de les compresser. En effet, en traduisant

les dossiers médicaux en texte libre en notre possession sous forme binaire, nous avons remarqué un gain d'espace de 33 à 34%. Cette compression logique, et en partie physique, n'a fait que mettre l'information sous la structure présentée ci-haut, sans pour autant diminuer le contenu sémantique des dossiers. Une diminution importante des redondances a donc été effectuée. Sur de telles données, le taux de compression espéré n'est pas très élevé. On peut alors se demander s'il est vraiment nécessaire de réduire au préalable la redondance avant d'effectuer une compression 'statistique'. Il peut être plus intéressant d'effectuer directement une compression statistique et ainsi obtenir un meilleur taux de compression globale. Les tests répondent à de telles questions.

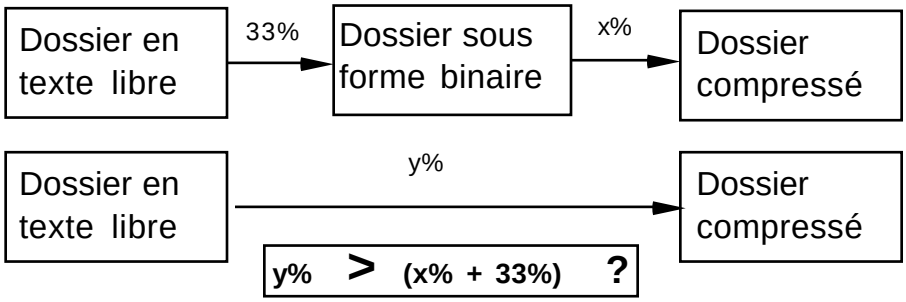


fig 4.40 : Opportunité d'augmenter ou de diminuer la redondance avant la compression

4.4.2 Les facteurs de mesure

Les quatre algorithmes implantés ont été testés sur un micro-ordinateur compatible avec un microprocesseur INTEL 80386 d'une fréquence de 25 Mhertz et une vitesse d'accès au disque de 19 ms.

En pratique, la mesure de performance d'algorithmes ou de modèles de compression se fait généralement en comparant les taux de compression obtenus. Mais il y a d'autres facteurs de mesure importants, comme le temps et la quantité de mémoire requise pour le programme de codage et de décodage [BELL 90]. Il est évident qu'il faut assurer que les besoins du programme de compression et de décompression ne dépassent pas les capacités de la machine sur laquelle les programmes sont exécutés. Pour un même modèle, les besoins en ressources du programme de codage et du programme de décodage sont souvent différents. Étant donnés les différents facteurs de mesure, il n'est pas trivial de présenter une comparaison pratique de différentes méthodes de compression. Les aspects de performance sont tributaires du type et de la taille des données à compresser. C'est la raison pour laquelle nous avons choisi d'utiliser des données de différents types pour étudier la performance des algorithmes sur ces types de données. Afin de faciliter la comparaison, nous avons aussi préféré utiliser les mêmes fichiers de données pour tous les algorithmes.

Les facteurs de mesure que nous avons retenus pour la comparaison des

algorithmes sont le taux de compression, le temps d'exécution et la quantité de ressources requises.

Taux de compression: Bien qu'il soit subjectif, le taux de compression est le facteur le plus intéressant. Il est évident que compresser un fichier contenant une suite d'un même caractère et un fichier contenant une suite de caractères aléatoires avec un même algorithme ne donne pas le même taux de compression. Un taux de compression ne peut donc pas caractériser une méthode de codage. Néanmoins, en exécutant deux méthodes de codage sur les mêmes données, le taux de compression obtenu peut être indicateur de performance. Le taux que nous avons utilisé est le suivant:

$$\text{Pourcentage de compression} = \frac{(\text{Taille du message initial} - \text{Taille du message compressé}) \times 100}{\text{Taille du message initial}}$$

Ce taux donne le pourcentage de gain d'espace par rapport à la taille initiale. S'il est positif, ce taux indique donc un gain. L'algorithme est alors dit efficace pour le type de données compressées. Si le taux est négatif, alors l'algorithme est jugé inefficace pour le type de données compressées puisqu'il y a une augmentation dans la taille des données. Pour la majorité des algorithmes et quel que soit le type de données manipulées, le taux de compression est sensible à la taille des données à compresser. La performance est généralement inversement proportionnelle à la taille de la chaine d'entrée. C'est la raison pour laquelle nous avons utilisé un échantillon de fichiers de tailles variables: 1024, 8192, 16 384 et 65 536 octets. Les données à compresser dans les zones de la carte à microprocesseur étant de taille réduite, la sensibilité des algorithmes de compression à la petitesse de la chaine d'entrée est importante à mesurer.

Temps d'exécution: Les temps d'exécution ont été mesurés avec l'horloge du DOS. Nous avons mesuré les temps de compression ainsi que les temps de décompression. Ces derniers sont présentés à l'annexe C. Les temps que nous avons porté à l'annexe comportent le temps de compression, le temps de chargement de programme (environ 0.66 seconde) ainsi que le temps de lecture et d'écriture sur le disque (environ 0.00025 seconde par caractère).

Quantité de ressources requises: Les quatre implantations que nous avons testées demandent toutes une quantité de mémoire et de disque généralement disponibles sur des micro-ordinateurs. Il faut dire que ces algorithmes ont été choisis de manière à ce qu'il puisse être exécutés sur un micro-ordinateur. Le facteur de mesure de la quantité de ressources requises a donc été considéré pour le choix des algorithmes et de leurs implantations. Nous ne faisons pas ressortir ce facteur dans nos résultats. Néanmoins, il est à noter que les besoins en mémoire augmentent avec l'ordre du modèle. Ainsi, un modèle de contexte d'ordre 1 utilise 256 fois plus de mémoire pour sauvegarder le contexte que le modèle d'ordre 0. Les besoins en mémoire se multiplient par 256 à chaque fois que l'on accroit l'ordre du modèle. LZW, pour sa part, requiert plus de mémoire à chaque fois que l'on augmente la taille en bit du

code de sortie. Ainsi, LZW avec un code de taille neuf bits requiert un espace pour sauvegarder 256 chaines de caractères. Par contre, LZW avec un code de taille 16 bits requiert un espace pour 65 280 chaines de caractères.

4.4.3 Les expériences de compression

4.4.3.1 Description

Huffman: Le codage de Huffman s'avère l'un des plus lents algorithmes testés. Les temps d'exécution restent néanmoins acceptables pour des tailles de données réduites. Le temps de la décompression est sensiblement égal à celui de la compression. L'efficacité du codage de Huffman s'accroit avec la taille des données à compresser. Ainsi, on remarque que le gain d'espace obtenu est plus important pour les grands fichiers. Le codage de Huffman n'est efficace que pour les données ayant une entropie faible comme les textes et les programmes exécutables. Il ne présente aucun intérêt pour les données binaires aléatoires ou les dossiers médicaux pré-codés. En effet, avec ces données, il y a plutôt une perte qu'un gain d'espace.

LZW: L'algorithme de Lempel, Ziv et Welch réagit différemment vis-à-vis des données selon la taille du code de sortie. Plus la taille du code de sortie est petite, plus l'algorithme est efficace avec des petites chaines d'entrée. LZW à petit code de sortie perd de son efficacité avec de grands fichiers. En effet, la table des chaines se remplit très vite et l'algorithme devient statique. Toute la compression se fait alors en se basant sur les statistiques de la première portion du fichier à compresser. Cette lacune peut être améliorée en adoptant une meilleure stratégie lorsque la table des chaines se remplit, comme l'effacement du contenu de la table et son remplissage à nouveau ou l'élimination des chaines les moins utilisées. On remarque que LZW9 est plus intéressant pour les petites tailles alors que LZW14 présente plus d'intérêt pour les grands fichiers. Ainsi, LZW avec un code de sortie sur 14 bits a été capable de compresser le fichier des bits aléatoires de 65 536 octets. LZW ne s'avère pas efficace pour des données binaires aléatoires ou pour les dossiers médicaux pré-codés, exeption faite des très grandes tailles. Son efficacité est plus notable pour les fichiers texte. Le temps requis pour la décompression est légèrement inférieur à celui requis pour la compression. Grâce à la fonction de Hash-code, le temps nécessaire pour le codage ou le décodage ne varie pas beaucoup avec la taille du code de sortie, bien qu'avec LZW14, la taille de la table des chaines est sensiblement supérieure à la taille de la table des chaines de LZW9.

Codage Arithmétique: Le codage arithmétique est l'algorithme le plus lent de tous les algorithmes testés. Pour les grandes tailles de fichiers, l'algorithme de décompression a même dépassé le seuil de temps fixé. Le temps nécessaire pour la décompression est notablement supérieur au temps requis pour la compression. Bien qu'il soit lent, le codage arithmétique reste encore utilisable pour une carte à microprocesseur puisque le temps de codage et de décodage de petites chaines est acceptable. Ce codage donne des résultats très intéressants pour la compression de dossiers médicaux pré-codés. Avec un modèle de contexte d'ordre 3, on n'obtient pas de gain d'espace sur les dossiers médicaux pré-codés. Par contre, l'efficacité du

modèle est très importante sur le texte puisqu'on obtient un taux de compression qui frôle les 78%. L'exécution du codage avec un modèle d'ordre 3 demande beaucoup trop de temps, ce qui nous a obligé à arrêter l'exécution du programme à plusieurs occasions. Il est néanmoins intéressant de noter que le codage arithmétique a été capable de compresser des fichiers contenant des bits aléatoires.

LZHUF: L'algorithme LZHUF est celui qui a présenté les meilleurs résultats dans l'ensemble. Les temps d'exécution, bien qu'ils soient lents, sont acceptables. La décompression nécessite moins de temps que la compression. Mis à part les fichiers de bits aléatoires, tous les fichiers des autres classes de types de données ont été compressés avec efficacité. Avec LZHUF, on note un taux de compression très encourageant pour les dossiers médicaux pré-codés. Le taux de compression d'un texte de 64 Ko est de 75% alors que les dossiers médicaux pré-codés sont compressés à 15% pour un fichier 8 Ko.

4.4.3.2 Comparaison

Nous avons choisi un programme de compression commercial PKzip 1.1 de PK-ware comme programme de référence pour avoir un ordre de grandeur dans la comparaison des algorithmes de compression que nous avons implantés. C'est un produit qui donne de très bons taux de compression et les temps d'exécution sont excellents. PKzip se fonde sur un algorithme ayant les propriétés des schémas basés sur les dictionnaires comme LZW et LZHUF. Nous avons donc exécuté PKzip sur les mêmes données utilisées pour les autres algorithmes. Les figures suivantes montrent un classement des algorithmes en fonction du taux de compression et du temps d'exécution selon le type de données compressées.

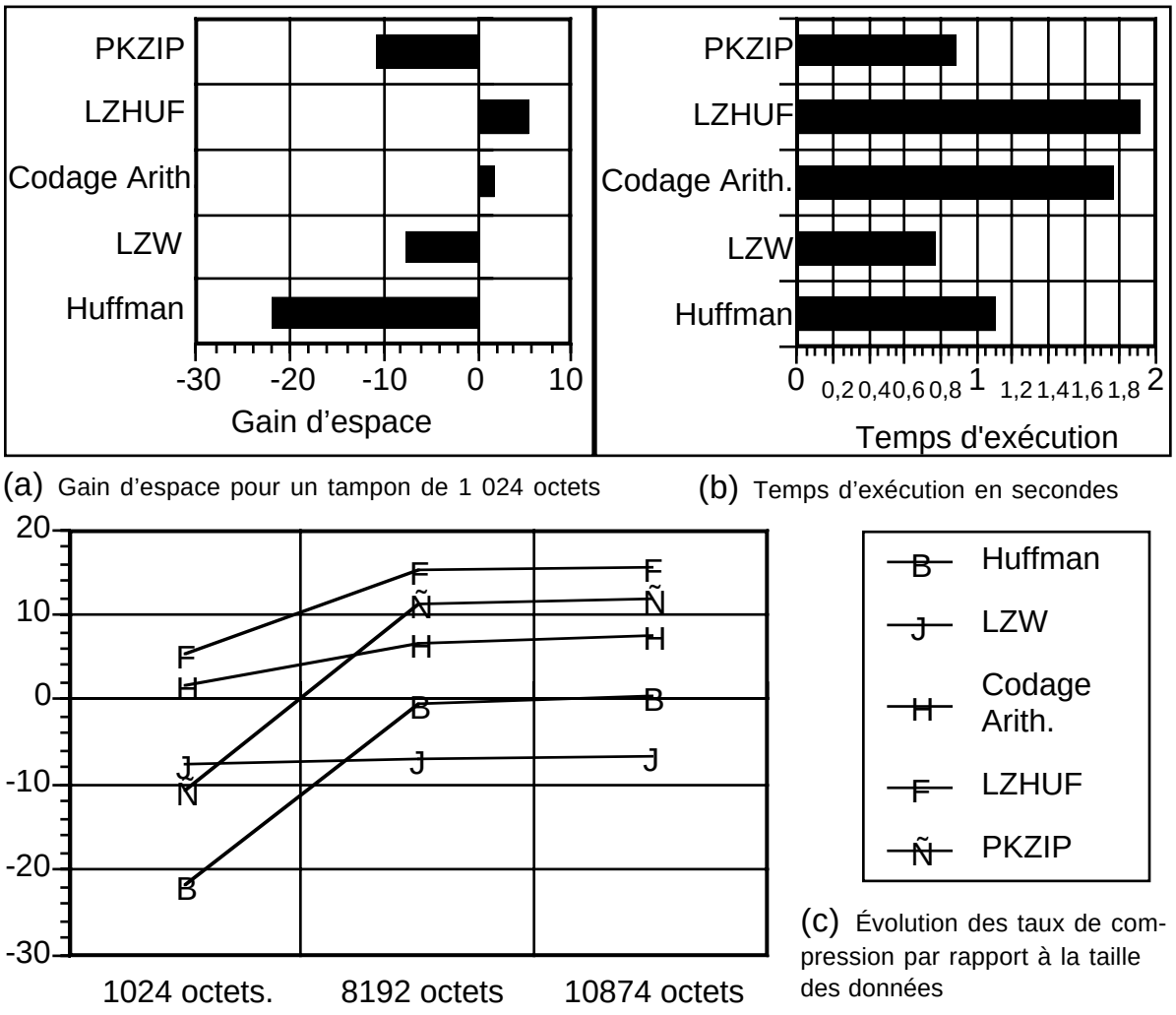
En annexe D, nous retrouvons des tableaux de comparaison permettant de classer les algorithmes par gain d'espace. Dans ce «hit-parade» ne figure à chaque fois que le meilleur de chaque famille d'algorithme. En effet, on dispose de cinq familles d'implantation. Chaque famille est constituée de variantes d'implantations du même algorithme. Ainsi, la première famille est celle de l'algorithme de Huffman où l'on retrouve l'implantation classique utilisant un modèle de contexte fixe d'ordre 0 et celle basée sur l'interdépendance entre deux caractères successifs, à savoir le modèle de contexte d'ordre 1. La deuxième famille comporte les différentes implantations de LZW avec des tailles de code de sortie variant de 9 bits à 14 bits. La troisième est celle du codage arithmétique où l'on retrouve le modèle de contexte fixe d'ordre 0, d'ordre 1 puis celui d'ordre 3. La quatrième famille comporte l'implantation unique de LZHUF alors que la dernière famille est celle du «programme témoin» PKzip. Chaque famille de programmes est toujours représentée dans les tableaux par un seul programme pour un type particulier de données. C'est le programme qui présente les meilleurs résultats pour la compression des données en question qui figure dans le tableau. Le type d'implantation est représenté entre parenthèses après le nom de l'algorithme.

Il est à noter que l'algorithme de Huffman basé sur l'interdépendance entre les caractères successifs a toujours donné de meilleurs résultats que l'algorithme de Huffman classique avec un modèle de contexte d'ordre 0. Dans la famille de LZW, le codage sur 9 bits a été plus rentable avec les dossiers médicaux pré-codés avec ou

sans redondance ainsi qu'avec les fichiers de bits aléaloires. Toutefois, les fichiers de bits aléatoires de taille importante ont donné des résultats très intéressants quand le codage était effectué sur 14 bits. En général, plus la taille de l'information est importante, plus le code de sortie de taille supérieure est davantage rentable. Pour les fichiers texte, par exemple, le code sur 9 bits est plus rentable pour compresser une information de 1 024 octets, alors que le code sur 14 bits est le plus intéressant pour un fichier de 65 536 caractères.

La figure 4.41 illustre une comparaison des algorithmes pour des données binaires aléatoires. Les graphiques 4.41.a et 4.41.b montrent le taux de compression et le temps d'exécution réalisés par chacun des algorithmes sur un fichier de 65 536 octets aléatoires. On voit que seul le codage arithmétique et LZW arrivent à compresser l'information. Les autres algorithmes augmentent légèrement la taille du fichier initial. Cependant, le codage arithmétique demande beaucoup de temps d'exécution. Le graphique 4.41.c montre une évolution du taux de compression pour tous les algorithmes. On voit que ce type de données est difficilement compactable et on remarque l’echec de compression de tous les algorithme. LZW avec un codage

de 14 bits présente les meilleurs résultats pour des bits aléatoire lorsqu'il s'agit de fichiers de grande taille. Par contre, avec des tailles réduites, c'est le codage arithmétique qui présente le plus d'avantages malgrès sa relative lenteur. Il est à noter que le «programme témoin» PKzip est lui aussi incapable de compresser l'information. Devant l'impossibilité de compresser l'information, PKzip laisse l'information telle quelle et rajoute une centaine de caractères de gestion. C'est ce qui explique la diminution du taux de perte avec l'augmentation de la taille du fichier.



(a) Gain d'espace pour un tampon de 1 024 octets

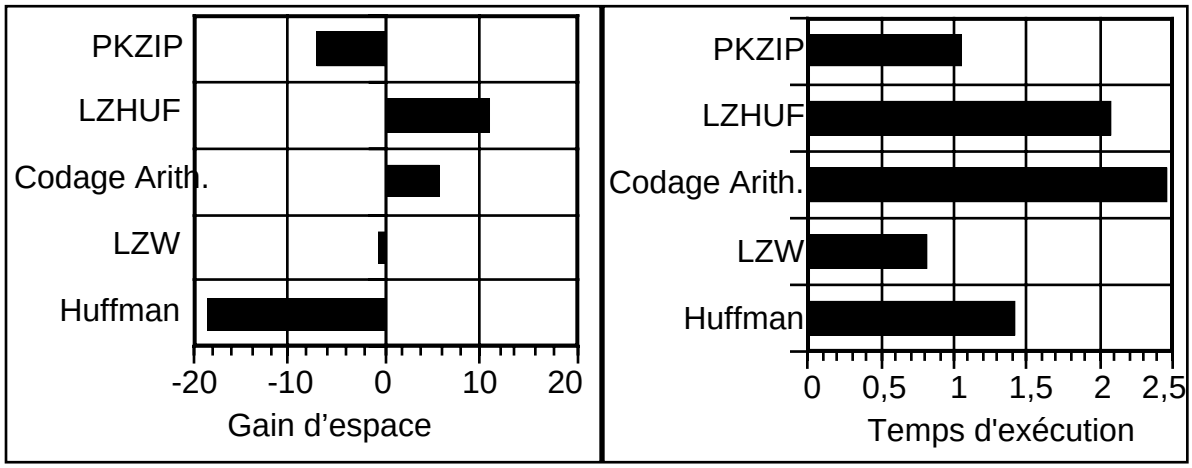
(b) Temps d'exécution en secondes

(c) Évolution des taux de compression par rapport à la taille des données

fig 4.42 : Comparaison des algorithmes pour des dossiers médicaux pré-codés

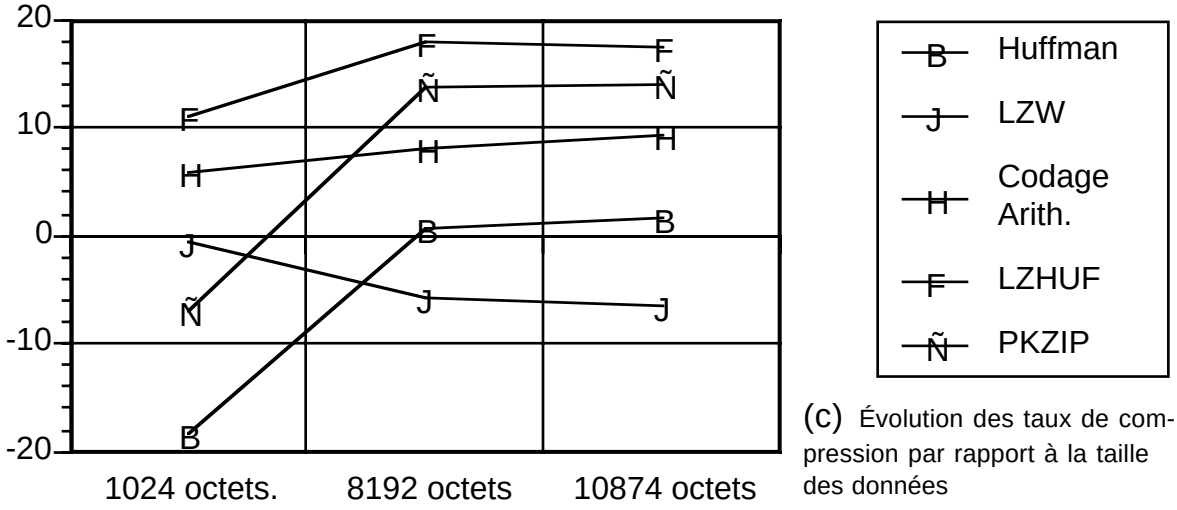
La figure 4.42 montre une comparaison des algorithmes pour des dossiers médicaux pré-codés. Les fichiers de petite taille sont difficilement compressés. En fait, seuls LZHUF et le codage arithmétique arrivent à avoir un gain d'espace lorsqu'il s'agit d'un fichier de 1 024 octets. PKzip ne codifie pas l'information et il rajoute 110 caractères de gestion. Cela explique l'augmentation de 10.74% de la taille de l'information. Au fur et à mesure que la taille de l'information à compresser augmente, les possibilités de compression augmentent aussi. C'est ainsi que tous les algorithmes testés, exception faite de LZW, arrive à gagner de l'espace avec des

dossiers médicaux pré-codés. soulignons la performance remarquable de LZHUF avec plus de 15% de gain en compressant des dossiers médicaux pré-codés totalisant 10 874 octets. PKzip ne gagne, avec les mêmes données, que 11.78%. Il est aussi à noter la quasi-stabilité de LZW quelle que soit la taille des données. Par contre, l'algorithme de Huffman, qui n'est pas rentable pour de petites tailles, parvient à faire des gains d'espace lorsqu'il s'agit de tailles importantes.



(a) Gain d'espace pour un tampon de 1 024 octets

(b) Temps d'exécution en secondes



(c) Évolution des taux de compression par rapport à la taille des données

fig 4.43: Comparaison des algorithmes pour des dossiers médicaux pré-codés avec redondances

La figure 4.43 montre une comparaison des algorithmes pour des dossiers médicaux pré-codés comportant des redondances. On remarque que les redondances artificielles rajoutées ne modifient pas beaucoup l'allure des courbes d'évolution des compressions effectuées. Seules les valeurs des taux de compression s'améliorent légèrement par rapport aux valeurs obtenues lors des compressions des dossiers médicaux pré-codés sans redondance ajoutée. Cependant, on peut remarquer une diminution de la performance de LZW avec l'évolution de la taille des données. Ceci est dû à la saturation de la table des chaines qui ne contient que 256 chaines puisque le codage est de neuf bits. Dans ces graphiques nous avons choisi de faire figurer les valeurs obtenues avec la compression des dossiers médicaux pré-codés avec les redondances artificielles 2. En effet, les résultats obtenus avec la

compression des dossiers médicaux pré-codés et avec les redondances artificielles 1 ne diffèrent pas beaucoup des résultats dans la figure 4.42.

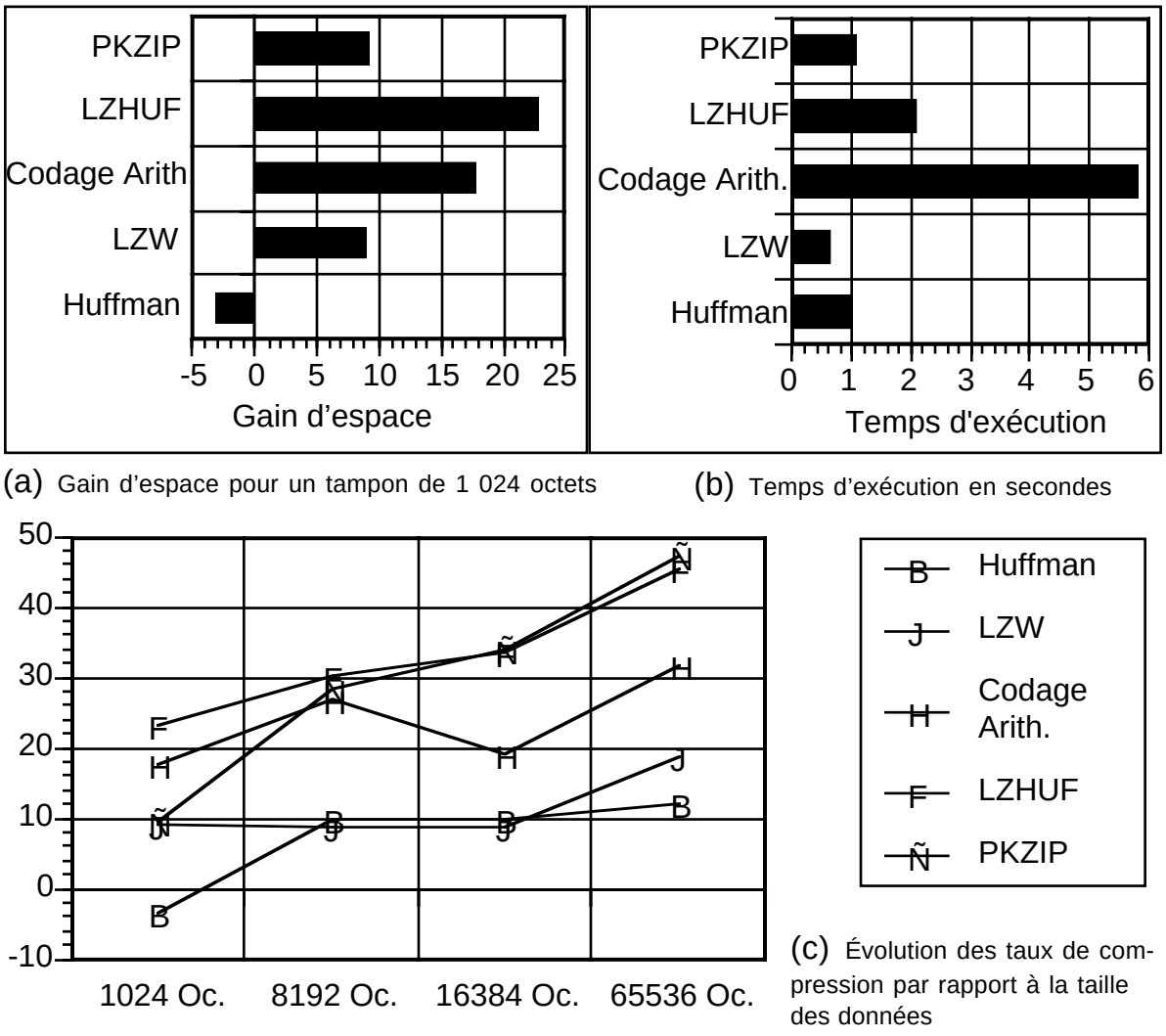
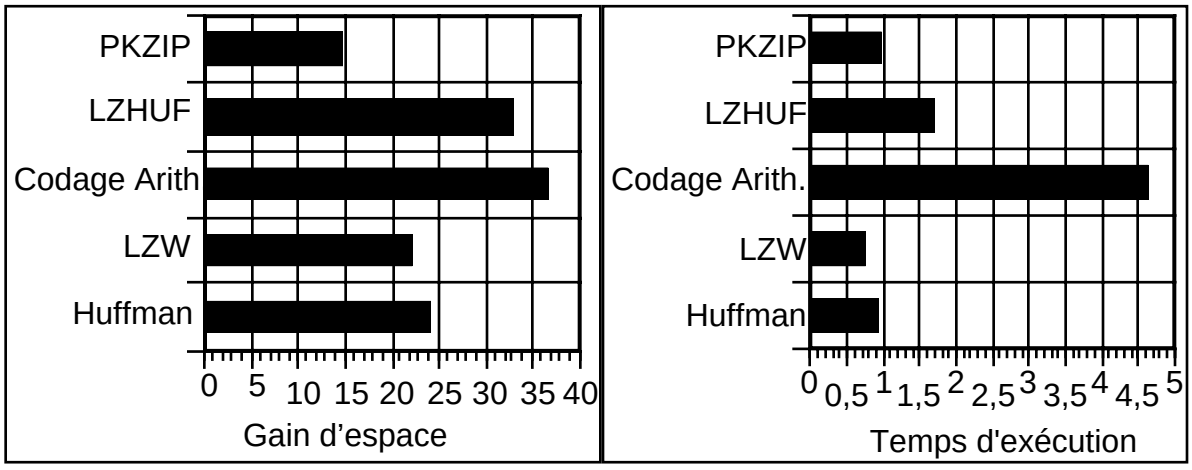


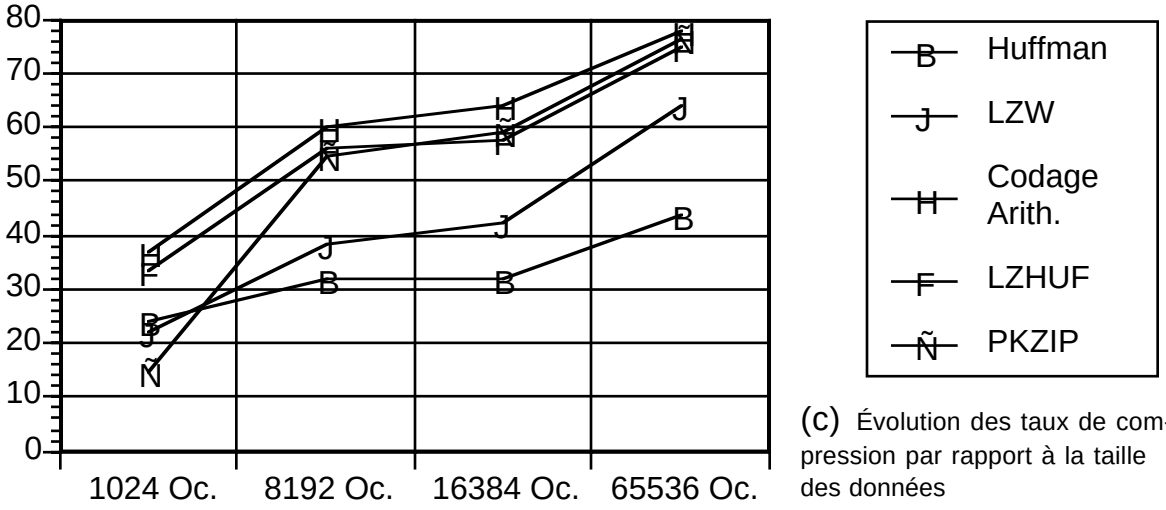
fig 4.44: Comparaison des algorithmes pour des programmes exécutables

La figure 4.44 montre une comparaison des algorithmes pour des programmes exécutables. Mis à part l'algorithme de Huffman pour un fichier de 1 024 octets, tous les algorithmes on été rentables dans la compression des programmes exécutables. Pour de petites tailles, c'est LZHUF qui présente les meilleurs résultats. À partir de 16 384 octets, c'est PKzip, le «programme témoin», qui devient plus intéressant. Les taux de compression obtenus avec tous les algorithmes sont néanmoins rentables. Le taux de compression du codage arithmétique semble diminuer à partir de 16 384 octets. En fait, c'est parce que le modèle de contexte d'ordre 3, qui est le plus rentable des modèles du codage arithmétique pour des petites tailles de fichiers exécutables, demande trop de temps d'exécution pour de grands fichiers. Nous avons donc retenu les performances du modèle d'ordre 1 pour les fichiers de taille 16 384 et 65 536 octets. C'est ce qui donne l'impression d'une diminution de performance. Il faut aussi noter les exigences toujours croissantes en temps d'exécution du codage arithmétique. Pour les grands fichiers, le temps nécessaire est parfois trop élevé.



(a) Gain d'espace pour un tampon de 1 024 octets

(b) Temps d'exécution en secondes



(c) Évolution des taux de compression par rapport à la taille des données

fig 4.45: Comparaison des algorithmes pour des dossiers médicaux en texte libre

La figure 4.45 montre une comparaison des algorithmes pour des dossiers médicaux en texte libre. Tous les algorithmes ont été capables de compresser les données fournies. On note la performance remarquable du codage arithmétique qui a donné les meilleurs taux de compression. Toutefois ses temps d'exécution ne sont pas très encourageants. Il est à remarquer que c'est le modèle de contexte d'ordre 3 qui a été retenu de la famille des implantations du codage arithmétique. LZHUF donne de très bons résultats. Ces derniers sont meilleurs que ceux de PKzip pour des petits fichiers. LZHUF demande néanmoins beaucoup de temps d'exécution. Cela est dû au besoin continuel de mise à jour de son arborescence. Le décodage de LZHUF, pour ce genre de données, est beaucoup plus rapide que le codage.

Pour tester les possibilités de compression de très petites chaines d'entrée binaires, comme c'est la cas pour les dossiers médicaux pré-codés, nous avons utilisé les trois algorithmes les plus performants avec ce type de données de petite taille. LZHUF, le codage arithmétique d'ordre 1 et le codage arithmétique d'ordre 0

présentent les meilleurs résultats pour les fichiers de bits aléatoires et de dossiers médicaux pré-codés de 1 024 octets. Nous avons donc constitué un échantillon de tampons de 512 puis de 256 octets de bits aléatoires et de dossiers médicaux pré-codés. Nous avons remarqué qu'aucun des trois programmes n'est capable de compresser des tampons de 256 ou 512 octets de bits aléatoires. Par contre, les trois programmes compressent les tampons comportant les dossiers médicaux pré-codés et le codage arithmétique arrive aussi à obtenir de bons résultats de compression avec des dossiers médicaux pré-codés de 256 octets. Soulignons aussi que le résultat obtenu pour les tampons de même catégorie varie beaucoup d'un tampon à l'autre. Pour des dossiers médicaux pré-codés, les résultats sont parfois positifs et parfois négatifs. Il y a donc intérêt à étudier la possibilité de compression avant le codage des données plutôt que d'effectuer une compression systématique de toute information. La moyenne des taux de compression obtenue est résumée dans les deux tableaux qui suivent. Nous rappelons que lorsque le taux de compression est positif, cela signifie qu'il y a en fait une augmentation de taille. Ce n'est que lorsque ce dernier est négatif que l'on peut parler de compression et de gain d'espace.

Pourcentage de compression moyen pour dix tampons de 512 et 256 octets contenant des bits aléatoires

	256 octets		512 octets	
	Taille résultat	Pourcentage de compression	Taille résultat	Pourcentage de compression
LZHUF	268.6	+ 4.92 %	536.1	+ 4.70 %
Codage Arit. ordre 0	262.3	+ 2.46 %	525.5	+ 2.63 %
Codage Arit. ordre 1	258	+ 0.78 %	514	+ 0.39 %

Pourcentage de compression moyen pour dix tampons de 51 et 256 octets contenant des dossiers médicaux pré-codés

	256 octets		512 octets	
	Taille résultat	Pourcentage de compression	Taille résultat	Pourcentage de compression
LZHUF	260.3	+ 1.67 %	501.2	- 2.11 %
Codage Arit. ordre 0	254	- 0.78 %	506.6	- 1.05 %
Codage Arit. ordre 1	256.6	+ 0.23 %	508.6	- 0.66 %

4.5 Réflexions sur les résultats

La majorité des techniques de compression ont été formulées dans le but de

réduire grand corpus de données. En effet, ce sont les grands volumes qui posent des problèmes de stockage ou de transmission. En observant les résultats présentés, nous remarquons un net avantage à coder de longues chaines d'entrée. Plus la chaine à coder est longue, plus le taux de compression est intéressant. Inversement, lorsque la chaine d'entrée est trop courte, la compression est impossible et nous notons alors une augmentation de volume. Cela est contraire au but du codage. Il existe donc un seuil dans la taille d'un bloc à coder en dessous duquel le rendement de la compression est négatif. Les techniques de compression dynamiques nécessitent, en effet, un en-tête de rodage ou d'adaptation dans la chaine d'entrée au cours de laquelle l'algorithme s'adapte dynamiquement aux statistiques caractérisant la source. Tout dépendant des distributions des symboles et des chaines, cet en-tête de rodage peut être plus ou moins long. Lors de son codage, le taux de compression est très bas sinon négatif. C'est ce qui pénalise les chaines d'entrée de courte taille. Les techniques non adaptatives, comme celle de Huffman, nécessitent par contre le stockage d'une table de codage dont la taille devient très importante. En effet, lorsqu'il s'agit de codage de petites chaines, la taille de la table de codage devient non négligeable devant la taille de l'ensemble du code, ce qui n'avantage guère les entrées courtes. Malheureusement, c'est le cas des données du DMP manipulées et gérées dans la carte à puce. En ce qui concerne le projet Carte Santé, il est plus avantageux de compresser un bloc de données entier dans la carte à microprocesseur que de tenter de coder chaque transaction médicale à part.

Les techniques de compression sont généralement très liées aux types de données. Ainsi, on retrouve des modèles sémantiquement dépendants pour la compression de données très spécifiques. C'est le cas des images, par exemple. Étant donné la compléxité des données manipulées dans un dossier médical portable pré-codé et la distribution quasi-uniforme des symboles de telles données, la technique de compression qui nous intéresse est une technique universelle, c'est-à-dire une technique qui ne tienne pas compte de la sémantique véhiculée par les données à compresser.

En tenant compte des résultats des taux de compression et le temps de traitement des différents algorithmes, la technique de compression que nous conseillons pour le projet Carte Santé est l'algorithme LZHUF. LZHUF présente les meilleurs taux de compression pour des données semblables à celles manipulées dans un dossier médical portable. De plus, son temps de traitement est acceptable pour des tailles similaires à celles des blocs dans une carte à microprocesseur. Néanmoins, pour atteindre un temps d'exécution acceptable dans une application médicale où le temps de réponse est très important, il est plus intéressant d'implanter l'algorithme de compression en logiciel exécuté dans la mémoire de l'ordinateur plutôt que dans le microprocesseur de la carte intelligente. Le microprocesseur de la carte est relativement lent, ce qui augmenterait sensiblement le temps de réponse. D'autant plus que l'implantation matérielle du programme de compression dans la carte à puce obligerait la transmission des données intégrales non compressées, donc un temps de transmission plus important.

En créant les dossiers médicaux pré-codés sous forme binaire à partir des

dossiers en texte libre pour notre échantillon de données, nous obtenons une diminution de redondance d'aproximativement 33%. Ce pré-codage est d'ores et déjà une compression sémantiquement dépendante fondée sur les différentes techniques de compression logiques. Bien qu'elle soit faible, la compression des dossiers médicaux pré-codés sous forme binaire permet d'obtenir une compression globale supérieure à celle des dossier médicaux en texte libre. C'est pourquoi nous croyons que dans le cas des dossiers médicaux portables, il est préférable de diminuer la redondance en compressant dans une première étape avec des techniques élémentaires sémantiquement dépendantes, avant de véritablement compesser avec une technique de compression universelle. Les tableaux des résultats placés en annexe prouvent qu'il n'est pas avantageux d'augmenter la redondance dans le cas des dossiers médicaux pour obtenir des taux de compression élevés, comme c'est le cas des données textuelles en général.

À la lumière des tests de compression effectués, nous pouvons remarquer qu'il n'est pas toujours opportun de compresser. En effet, il arrive que la compression provoque une augmentation de volume. C'est le cas des données dépourvues de redondance. Pour obtenir le résultat escompté, c'est-à-dire occuper toujours le minimum d'espace sur la carte, il est préférable d'implanter un mécanisme qui permette de savoir si une information dans une carte à microprocesseur est compressée ou non. Ce mécanisme permet de connaitre la nécessité de la décompression lors de la lecture des données. La compression ne devient plus systématique mais intelligente et elle ne s'accomplit que si elle est nécessaire. Pour réaliser cette compression intelligente, nous proposons d'accompagner chaque zone de données par un bit (bit de compression) indiquant si l'information est compressée. Ainsi, si l'information est compressable, elle est réduite et le bit de compression l'accompagnant est mis à 1. Dans le cas contraire, l'information est stockée à l'état brut sans compression et le bit reste à 0. Au moment de la lecture des données sur la carte, l'information est décodée ou ne l'est pas suivant l'état du bit de compression. Le bit de compression est implanté dans l'en-tête de chaque bloc physique de la carte à microprocesseur si la technologie le permet, ou est représenté par le premier bit de chaque bloc de données. Ce protocole permet de ne jamais avoir une augmentation de volume des données au moment où elles sont stockées sur la carte. Dans le pire des cas, aucun gain d'espace n'est réalisé. Cependant, ce protocole permet de diminuer le temps de réponse lors de la lecture des données dans les cas où la décompression n'est pas nécessaire.

Malgré la fiabilité de la mémoire d'une carte à puce qui garantit que les données inscrites ne risquent pas d'être altérées, nous avons jugé intéressant d'étudier la sensibilité des algorithmes de décompression à de légères altérations du code compressé. Théoriquement, si l'on veut obtenir le message d'origine après la décompression, il faut que le code compressé reste intact pour être interprété correctement par le décodeur. Nous avons donc compressé un texte de taille raisonnable (1 024 octets) avec les différents algorithmes étudiés. Nous avons ensuite rajouté du «bruit», c'est-à-dire que nous avons effectué de légères modifications de bits sur deux caractères, au début, au millieu ou à la fin du code compressé. Le code ainsi «parasité» a ensuite été traité par les décompresseurs respectifs. Nous avons re-

marqué que la majorité des algorithmes sont extrêmement sensibles au bruit dans le code et ce, quel que soit le lieu (au début, au milieu ou à la fin du fichier). Les résultats sont résumés dans le tableau qui suit:

Algorithmes	Bruit au début du fichier	Bruit au milieu du fichier	Bruit à la fin du fichier
LZHUF	Perte totale Parasites et augmentation de taille Détection erreur:NON	Perte de 95% de l'info Parasites et augmentation de taille Détection erreur:NON	Perte de 35% de l'info Parasites multiples et dispersés Détection erreur:NON
Codage Arithmétique	Perte de 95% de l'info + quelques parasites Arrêt du programme	Perte de 50% de l'info + quelques parasites Arrêt du programme	Perte de 5% de l'info + quelques parasites Arrêt du programme
LZW	Perte de 80% de l'info Parasites et augmentation de taille Détection erreur:NON	Perte de 75% de l'info + quelques parasites Détection erreur:OUI	Perte de 5% de l'info + quelques parasites Détection erreur:OUI
Huffman	Perte totale Parasites multiples et dispersés Détection erreur:NON	Parasites minimes et localisés Détection erreur:NON	Parasites minimes et localisés Détection erreur:NON
PKzip	Perte de 90 % de l'info Parasites Détection erreur:OUI	Perte totale Parasites Détection erreur:OUI	Perte de 20 % de l'info + quelques parasites Détection erreur:OUI

Ces données illustrent que le résultat est imprévisible. Les bruits peuvent être détectés ou interprétés comme des codes. Les plus grands dommages sont obtenus avec des bruits au début du code. Avec du bruit à la fin du code, les dégâts sont moindres sinon minimes. Les dégâts sont considérés minimes si les parasites n'infectent pas le texte d'une façon disparate mais que le texte demeure intact jusqu'à un certain point, comme c'est le cas avec le codage arithmétique. Lorsque les parasites sont dispersés, on ne peut plus distinguer entre le texte correct et le texte infecté. En général, les algorithmes sont très sensibles aux bruits et il est préférable d'implanter un mécanisme permettant de vérifier la présence de bruit dans le code compressé avant le décodage, ce qui n'est pas sans difficulté.

Cette étude sur les différents algorithmes montre clairement les avantages que peut apporter la compression des données à la Carte Santé. Avec l'implantation relativement simple d'un module transparent et intelligent de compression de données, on peut stocker plus de transactions médicales dans un dossier médical portable, allonger la vie d'une Carte Santé et rendre l'application médicale plus conviviale et acceptable aux yeux des utilisateurs et des bénéficiaires.

CHAPITRE V

Implantation d'une Carte Santé

Nous présentons dans ce chapitre le projet Carte Santé et des propositions satisfaisant certaines contraintes liées à l'implantation d'un système utilisant une carte à microprocesseur comme média d'information. Bien qu'hétérogène et acceptant une diversité de logiciels et de matériels informatiques, le système que nous présentons est un système ouvert, conçu pour permettre à de nouvelles applications d'être intégrées sans changement important et permettre la modification du dossier médical portable durant le cycle de vie normal d'une carte, par ajout de champs ou restructuration, sans pour autant détériorer l'ancien contenu du point de vue sémantique ou opérationnel.

5.1 Présentation du projet Carte Santé

Les renseignements concernant la santé des patients et bénéficiaires des services de soins de santé au Québec sont éparpillés entre cliniques privées, hôpitaux et C.L.S.C. Lorsqu'un patient se présente chez un médecin, faute d'information, un ensemble d'examens et d'investigations est systématiquement fait et refait, parfois même à tort. Centraliser l'information pour permettre l'accès aux données médicales aux professionnels de la santé impliquerait un investissement énorme en sécurité qui, à terme, peut être utopique et refusé par l'opinion publique pour des raisons de respect à la vie privée. Une carte à mémoire intelligente contenant des informations pertinentes sur la santé d'un usager pourrait aider les professionnels du domaine de la santé à dresser rapidement et d'une façon fiable un portrait global de l'état de santé de son titulaire. En permettant une meilleure circulation, entre des professionnels consultés, d'une information fiable relative à l'état de santé de son titulaire, la carte à microprocesseur est un outil idéal pour contrer la dispersion et la perte de l'information médicale tout en garantissant une sécurité des données confi-

dentielles. La carte ne doit pas alors être perçue comme un instrument de contrôle mais plutôt comme un moyen pour améliorer la qualité des soins. Dans une première étape, la carte, qui n'est point un dossier médical complet mais un «aide-mémoire», contient des informations parcellaires destinées à attirer l'attention d'un praticien sur des éléments significatifs du passé médical de son patient. La carte pourrait, dans une seconde étape, contenir des informations de gestion pour accélérer et améliorer l'administration des dossiers.

À compter de mars 1992, la RAMQ compte mettre à l'essai la Carte Santé dans la région de Rimouski, qui est une région assez représentative de l'ensemble de la population du Québec.

5.1.1 Le projet pilote

Le projet Carte Santé de Rimouski est une expérience pilote servant à déterminer l'utilité de la carte à mémoire médicale qui ne sera implantée dans d'autres régions du Québec que si les résultats sont concluants. Pour une évaluation profonde et détaillée, l'expérience de Rimouski durera trois ans dont seulement 18 mois pour l'expérimentation sur le terrain. Elle portera sur plus de 9 000 personnes qui, sur une base volontaire, recevront une carte. Trois groupes sont particulièrement visés à savoir: les nourrissons de moins de 24 mois, les personnes âgées de plus de 60 ans et les femmes enceintes. Plus de 1 600 citoyens de la municipalité de St-Fabien près de Rimouski seront aussi invités à se munir d'une carte, ce qui permettrait d'étudier l'impact de l'implantation du même système sur une population entière. Les professionnels de la santé sont eux aussi impliqués sur une base volontaire. Ils sont composés de médecins, de pharmaciens, d'infirmières et d'ambulanciers. Plusieurs applications informatiques, que nous détaillons dans les paragraphes qui suivent, sont impliquées dans ce projet d'envergure. Notre travail consiste surtout à concevoir et à développer une de ces applications qui permettra de manipuler l'information d'un dossier médical portable au sein d'une carte à microprocesseur.

5.1.2 Les applications impliquées

Plusieurs applications sont impliquées dans le projet pilote: des applications pharmaceutiques, une application médicale, une application de services d'urgence et une application de service de vaccination. L'application médicale MÉDICARTE et l'application de services d'urgence sont spécialement et entièrement conçues et développées pour le projet pilote, par le Département de médecine sociale et préventive de l'Université Laval sous l'environnement Windows de Microsoft. Elles utilisent et manipulent les données médicales telles que conçues dans le DMP par les utilisateurs eux-mêmes et décrites dans le chapitre II de cet essai. Le développement d'une nouvelle application vient du fait que sur le site pilote, aucune application médicale informatique importante n'ait encore été implantée. Ce qui n'est pas le cas pour les applications pharmaceutiques. Plusieurs pharmacies disposent déjà d'une application informatique. Bien que MÉDICARTE ait été conçue et développée sur mesure pour le projet, les applications pharmaceutiques existent déjà et les praticiens y sont familiarisés. Elles ne sont donc pas remplacées, mais simplement

adaptées par leurs fournisseurs respectifs afin de permettre la manipulation des cartes à microprocesseur. C'est ce qui advient aussi à VAXIN, une application développée par le Département de santé communautaire du centre hospitalier de l'Université Laval pour la gestion des vaccinations utilisée dans les Centres locaux de santé communautaire (CLSC). Il existe un autre ensemble d'applications impliquées dans le projet comme l'application responsable de la personnalisation et l'émission des cartes, l'application de maintenance et l'application d'analyse des statistiques sur les Cartes Santé. Nous appelons cet ensemble «applications de gestion et de maintenance».

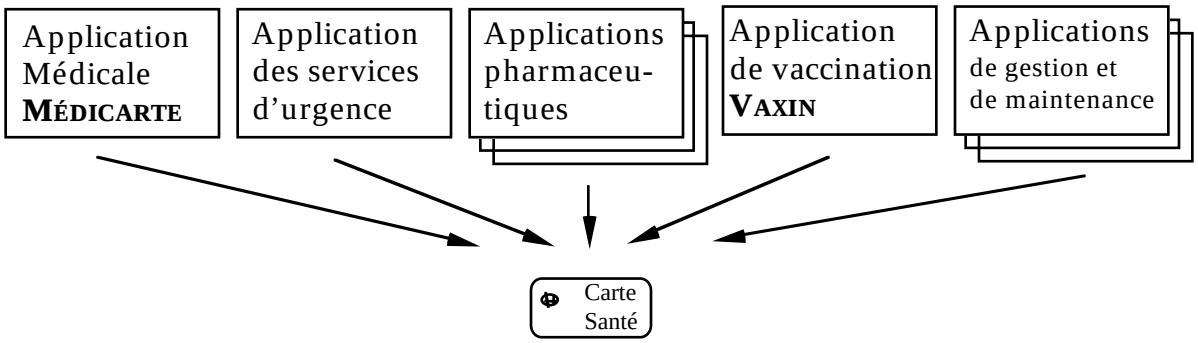


fig 5.1 : Les applications impliquées

5.1.3 Indépendance des applications vis-à-vis des technologies

Les applications qui manipulent les données sur la carte intelligente ne s'adressent pas directement au microprocesseur de la carte à puce. Elles le font à travers un noyeau qui contrôle tous les accès à la carte. Ce noyeau est appelé Coquille. Il permet un accès plus facile aux données au moyen d'un ensemble restreint d'instructions simples et d'un protocole de communication spécifique. L'ensemble de ces instructions est rassemblé dans une librairie de fonction succinte appelée librairie protocolaire. L'avantage de la librairie protocolaire est qu'elle garantit aux applications d'avoir toujours à utiliser les mêmes instructions pour manipuler les données sur la Carte Santé même si la technologie de la carte venait à changer. Les applications n'ont donc pas à se soucier des fonctionnalités et des particularités des technologies de cartes à microprocesseurs. La librairie protocolaire est compilée avec les applications qui souhaitent traiter avec le système d'exploitation de la Carte Santé et devient un simple moyen de communication. Comme nous le voyons dans la figure 5.2, le système est stratifié autour du masque de la carte. Ceci permet des modifications sans peine et une portabilité aisée. Ainsi, toute application peut s'adapter au moyen

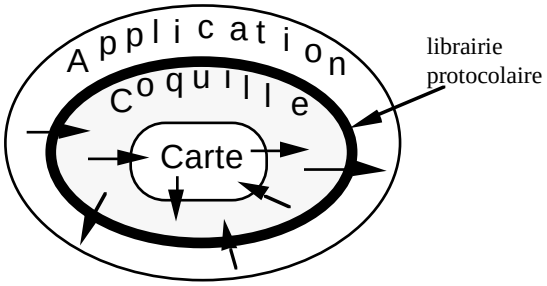


fig 5.2 : Stratification des logiciels de manipulation de la carte

de la librairie protocolaire pour accéder à la Carte Santé. La stratification permet aussi de modifier certaines couches sans toucher à d'autres. Ainsi, si la structure physique des données du DMP venaient à changer, seule la Coquille serait modifiée et on ne serait pas obligé de porter des modifications à toutes les applications de la couche supérieure. En effet, ce n'est que la Coquille qui traite les informations du DMP au niveau physique alors que les applications ont des vues logiques sur les données.

5.1.4 Les technologies possibles

Plusieurs technologies de cartes intelligentes existent déjà sur le marché. Chacune a ses propres caractéristiques et elles ne fonctionnent pas toutes de la même façon puisque chacune à un système d'exploitation et, une librairie de fonctions et de paramètres différents. Même les protocoles de communication varient malgré les tentatives de standardisation (voir chapitre I). Théoriquement, toute carte ayant un maximum d'espace utilisable et un bon masque permettant de gérer des blocs autonomes, peut être utilisée dans le cadre du projet pilote. Néanmoins, vu les prix élevés des technologies performantes de cartes intelligentes et vu les différents besoins du projet, on peut très bien permettre à différentes technologies de se côtoyer au sein d'un même projet. En effet, comme on dispose de plusieurs catégories de bénéficiaires (nouveaux nés, personnes âgées, femmes enceintes...), et que chaque catégorie a ses propres besoins d'espace et sa propre structure de DMP (voir chapitre II), on peut associer à chaque catégorie de bénéficiaires une technologie de carte particulière. Ainsi, plusieurs technologies, ayant des fonctionalités et des performances différentes peuvent être utilisées en même temps et par le même logiciel. La gestion d'un tel ensemble hétérogène demeure cependant complexe et doit être complètement transparente aux applications utilisant les données du DMP, qui ne doivent pas se soucier des problèmes techniques liés à la technologie de la carte utilisée. Le mécanisme permettant de reconnaître différentes technologies et de réagir en conséquence est décrit au paragraphe 5.4.2.4.

5.1.5 La Coquille

La Coquille est une couche logicielle entre les applications et les technologies de cartes intelligentes. C'est en fait un serveur logiciel permettant aux différentes applications d'utiliser des cartes à microprocesseur sans se soucier des aspects techniques associés à chaque technologie. Ainsi, les applications peuvent manipuler des informations sur des cartes intelligentes, moyennant un ensemble standard restreint d'instructions relativement simples rappelant les instructions de manipulation de fichiers, comme l'ouverture et la fermeture d'une session, l'écriture et la lecture ainsi que quelques fonctions de gestion. Toutes les applications écrivent et lisent des informations sur les cartes via la Coquille qui canalise toute communication vers la carte. Elles sont ainsi totalement indépendantes des cartes intelligentes et de leurs structures de données. Une application ne voit pas la structure physique des données telles qu'elles sont inscrites sur la carte et n'est en connaissance que de la vue logique de ces dernières (voir chapitre II). La Coquille est composée de deux modules ayant des fonctionalités différentes et bien précises. Le premier module est appelé

Interface Applicative. Il interagit avec les applications et les utilisateurs et communique leurs requêtes au deuxième module qui est le Pilote. Le Pilote interagit avec les cartes à microprocesseur et sert d'intermédiaire entre les cartes et l'Interface Applicative. En fait, la Coquille est constituée logiquement de trois niveaux hiérarchiques: une couche d'échange et de liaison avec le monde des applications (librairie protocolaire), une couche de gestion interne (Interface Applicative) et une couche de correspondance avec le monde des cartes (Pilote). La communication entre ces trois niveaux ne se fait que deux à deux, toujours via la couche de gestion interne. C'est-à-dire que l'Interface Applicative communique avec la librairie protocolaire et avec le Pilote mais le Pilote n'a aucun lien direct avec la librairie protocolaire. Par la suite, nous confondons délibérément la première et la seconde couche dans le même module de l'Interface Applicative. En effet, la librairie protocolaire n'est qu'un moyen de communication pour l'Interface Applicative.

Librairie protocolaire échange et liaison avec les applications	Niveau 1
Interface Applicative gestion interne	Niveau 2
Pilote correspondance avec les cartes	Niveau 3

fig 5.3 : les couches logiques de la Coquille

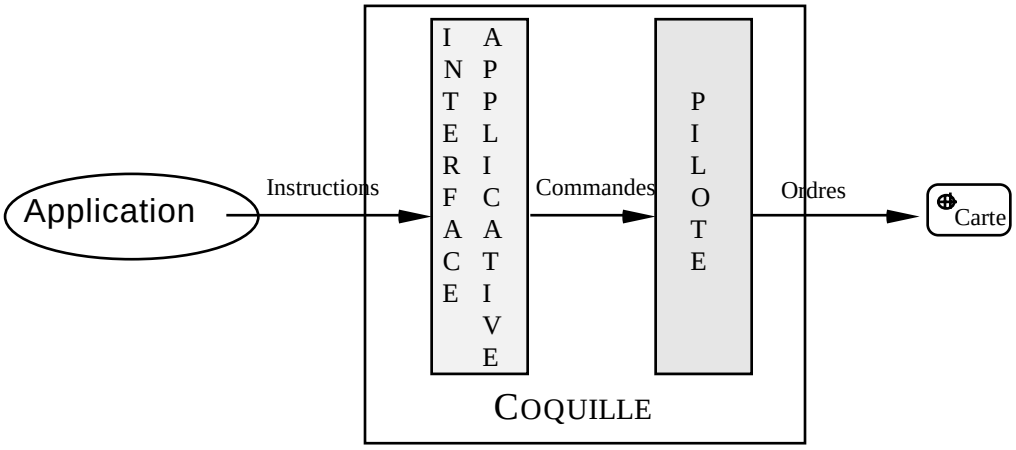


fig 5.4 : Canalisation dans la Coquille des transactions des applications destinées à la carte

Pour pouvoir distinguer les différentes communications qui existent entre les couches du système, nous avons appelé «instructions» les requêtes des applications reçues par l'Interface Applicative. Les requêtes de l'Interface Applicative destinées au Pilote sont quant à elles appelées «commandes». Enfin, les requêtes du Pilote aux cartes intelligente sont appelées «ordres». Une instruction peut amener l'Interface Applicative à générer plusieurs commandes suivant la correspondance des données logiques et physiques, puisqu'il peut arriver, dans le cas général, qu'une zone logique soit éclatée en plusieurs zones physiques sur la carte. De même, une commande peut amener le Pilote à générer plusieurs ordres. En effet, le Pilote génère des ordres d'écritures pour les données de gestion et peut éclater une transaction en plusieurs lectures et écritures. C'est le cas des signatures, par exemple. Chaque transaction est accompagnée d'une signature. Or, cette dernière est stockée à part dans une table de signatures (voir chapitre II) et seulement son index accompagne les données de la transaction. Le Pilote génère donc pour chaque

transaction des ordres de recherche, donc de lecture dans la table des signatures pour vérifier l'existence de la signature courante. Si celle-ci existe, son index dans la table est utilisé dans la transaction, sinon elle est rajoutée comme une nouvelle entrée à la table des signatures.

Il y a deux versions de la Coquille. Une est développée pour l'environnement Windows de Microsoft. C'est un logiciel à part entière pour communiquer avec les applications sous Windows. La seconde est développée sous DOS. C'est une application résidente en mémoire et communique avec les applications DOS comme les applications pharmaceutiques, par exemple. Les applications impliquées sont des entités distinctes de la Coquille quel que soit l'environnement choisi.

5.1.5.1 L'Interface Applicative

L'Interface Applicative est un module permettant aux applications d'interagir avec les cartes intelligentes au moyen d'un ensemble de 12 instructions. Elle reçoit des requêtes des applications et prend en main la correspondance entre les données logiques et les données physiques. Nous rappelons que les applications ont une vue logique des données alors que l'information peut avoir physiquement toute une autre structure sur la carte. Après avoir contrôlé les droits d'accès aux données des acteurs auxquels prétendent les applications, l'Interface Applicative génère des commandes au Pilote. Chaque transaction logique est succeptible d'être sectionnée en plusieurs transactions physiques. Ce qui fait qu'une instruction peut engendrer plusieurs commandes. L'Interface Applicative garantit alors l'intégrité de cet ensemble de commandes (voir 5.3.2.1). Le contrôle d'accès qu'effectue l'Interface Applicative est un niveau de sécurité qui se rajoute à celui de la carte intelligente et permet d'éviter les transactions vaines qui vont d'office être refusées par la carte. Ce niveau de sécurité permet aussi de contrôler les droits d'accès en annulation, en suppression et même en impression, des droits qui ne sont pas pris en compte par la carte ou du moins par toutes les technologies de cartes. Elle joue donc un rôle de sécurité important. C'est d'ailleurs elle qui procède à l'identification et l'authentification, s'il y a lieu, des cartes d'habilitation et de bénéficiaires. Ceci évite la connaissance et la manipulation des mots de passe et des clés par les applications, une manipulation qui est très souvent jugée dangeureuse. Certains dialogues avec l'utilisateur comme la demande d'insertion de carte, la demande et la lecture des clés ainsi que l'affichage de messages d'avertissement sont effectués par l'Interface Applicative au moyen d'une interface usager utilisant un ensemble de fenêtres et d'icones standardisé. Une autre fonction de l'Interface Applicative permet à celle-ci de gérer plusieurs sessions d'acteurs différents en même temps. Ainsi, plusieurs médecins ou pharmaciens peuvent utiliser le même poste de travail en concurrence sans être obligé de ré-initialiser leur session de travail en introduisant à chaque fois leur carte d'habilitation et leur numéro d'identification personnel. En somme, l'Interface Applicative assure une indépendance des applications vis-à-vis des problèmes liés aux technologies de cartes et des représentations physiques des données. Elle facilite la programmation de la communication avec les cartes et valide les accès à l'information. Elle constitue le lien entre les applications et le Pilote qui lui communique directement avec les cartes.

5.1.5.2 Le Pilote

Le Pilote est la couche logicielle servant de lien direct avec la carte à microprocesseur. Il gère toutes les manipulations physiques des données. Sa tâche principale est de reconnaître la technologie de la carte qu'il traite et de générer des ordres appropriés en conséquence pour écrire et lire de l'information dans la mémoire de la puce. C'est à lui qu'incombe la tâche de gérer la mémoire en créant et en allouant des blocs ainsi que de détecter les saturations et de contrôler leur résolution suivant les politiques choisies. Il s'occupe de l'analyse et de la codification des informations destinées à la carte suivant leurs types de données avant leur stockage, ainsi que de leur décodification et leur remise en forme avant leur transfert à l'Interface Applicative. Le Pilote se charge aussi de la compression et de la décompression des données pour optimiser l'utilisation de l'espace sur la carte. Chaque transaction physique qui lui parvient de l'Interface Applicative génère plusieurs ordres bien spécifiques et le Pilote assure l'intégrité de cet ensemble d'ordres à la carte. Il assure donc l'atomicité de la transaction physique.

5.2 Manipulation des données

Les données ne sont pas vues et manipulées de la même façon par l'Interface Applicative et le Pilote. Le Pilote ne voit que les données physiques telles qu'elles sont inscrites sur la carte. Quant à l'Interface Applicative, elle est en mesure de traiter les structures logiques et physiques des données. En général, les données passent par trois étapes de transmission. La première concerne la communication de l'application avec l'Interface Applicative. Au cours de cette étape, les données sont toutes en caractères, même les chaînes de bits et les chiffres. L'information est alors structurée en transactions logiques subdivisées en champs et regroupées pour chaque transaction dans une unique chaîne de caractères. La seconde étape concerne la communication entre l'Interface Applicative et le Pilote. Au cours de cette étape, les données sont aussi toutes en caractères mais structurées en transactions physiques subdivisées en champs. Les données de chaque transaction physique sont regroupées dans une unique chaîne de caractères. La dernière étape concerne la communication entre le Pilote et la carte. Au cours de cette étape, les données sont codées en binaire de manière à occuper le moins d'espace possible et transmises en chaînes de bits dont la structure n'est connue que par le Pilote (voir Annexe B). Au cours de la première et la seconde étape, la manipulation des données suit des règles de format bien précises que nous présentons ci-dessous. Nous présentons aussi les précautions prises pour assurer l'intégrité des données véhiculées sur ces trois étapes ainsi qu'un mécanisme permettant l'annulation d'une information déjà inscrite sur la carte.

5.2.1 Le format des données manipulées

Le protocole de communication entre les applications et l'Interface Applicative exige que les données soient transmises en caractères et que chaque champ soit séparé par un séparateur en l'occurrence le caractère '**@**'.

Exemple : '**@**champ_1**@**champ_2**@**champ_3**@**NULL'.

La chaîne de caractères est toujours suivie par un caractère nul (NULL). Lorsqu'un champ est absent, il suffit de mettre deux '**@**' l'un à la suite de l'autre.

Exemple : '**@**champ_1**@@**champ_3**@**NULL'. Le champ 2 n'existe pas.

Le protocole exige aussi que le nombre de champs dans la partie des données de la requête soit limité et fixe. Ainsi, on a une limite de 255 champs au maximum et pour chaque requête concernant une zone logique donnée, on doit fixer le nombre de champs entre 1 et 255. Cette restriction nous impose donc d'avoir un nombre fixe de séparateurs '**@**' dans la chaîne comportant les données de la requête. Cette codification est très bonne lorsqu'il s'agit de champs indépendants et dont on connaît le nombre. Cependant, la codification des suites de champs interreliés se pose. En effet, il n'est pas possible de codifier une suite de champs dont on ne connaît pas le nombre. Une suite de champs peut avoir un nombre d'éléments variables. C'est le cas des vaccins ou des médicaments par exemple, où l'on peut avoir différents cas de figures: un, deux, trois ou plus de vaccins ou médicaments à la fois. Pour résoudre ce problème, nous proposons de fixer le nombre de séparateurs utilisés pour transmettre une suite de champs. De ce fait toute suite aura une longueur fixe et connue. On suggère de prendre le nombre maximum d'éléments dans une suite égale à 15. Le nombre 15 est choisi parce qu'il reflète la représentation physique qui est utilisée pour emmagasiner les données sur la carte. En effet, toute suite d'information est représentée par un vecteur sur la carte. Ce vecteur commence toujours par un indicateur du nombre d'éléments. Cet indicateur est codé sur quatre bits. Il peut donc représenter un chiffre entre 0 et 15. Dans ce vecteur, le nombre d'élements est variable et toujours inférieur à 16. Chaque élément du vecteur a exactement la taille requise pour emmagasiner un champ, quelle que soit sa structure. En fixant ainsi le nombre d'éléments d'une suite de champs par transaction logique, on facilite le travail qui consiste à transcrire physiquement les données. Si le nombre de champs dépasse le maximum fixé (15), l'application transmet une première requête avec les 15 premiers champs puis transmet les champs restants dans une seconde transaction. Ceci évite au processus qui fait la transcription physique de faire lui-même cette division en sous-suites. Si le nombre de champs est inférieur au maximum, la transaction logique comprendra quand même 15 champs. Les champs supplémentaires seront vides. Comme pour la représentation physique, le premier champ d'une suite est toujour l'indicateur du nombre d'éléments. Ce choix répond à des soucis d'uniformité que nous verrons plus bas.

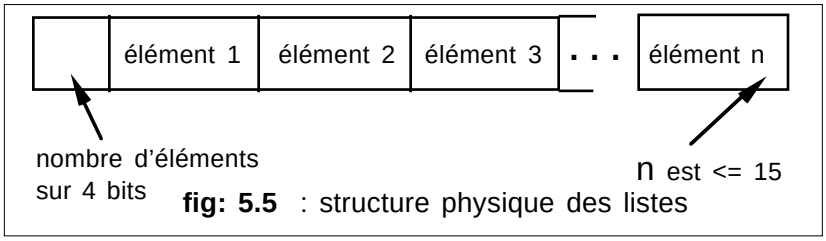


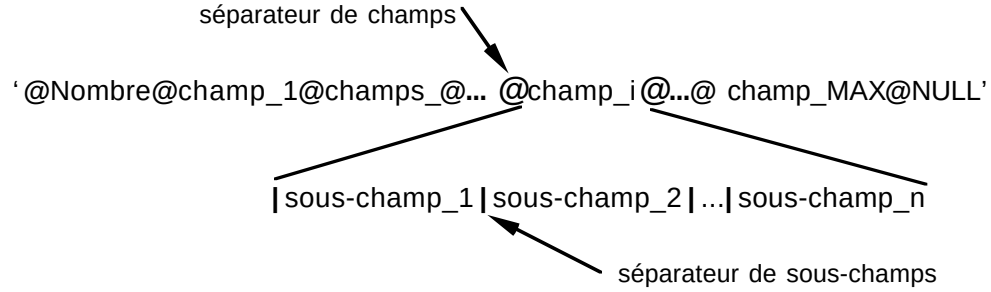
fig: 5.5 : structure physique des listes

Exemple 1:
Nombre d'élements = 5
données de la transaction 1=

'**@**5**@**champ_1**@**champ_2**@**champ_3**@**champ_4**@**champ_5**@@@@@@@@@@@@**NULL'

Exemple 2:
Nombre d'élements =18
données de la transaction 1=
'@15@champ_1@champ_2@champ_3@champ_4@champ_5@champ_6@champ_7@champ_8@champ_9
@champ_10@champ_11@champ_12@champ_13@champ_14@champ_15@NULL'
données de la transaction 2=
'@3@champ_16@champ_17@champ_18@@@@@@@@@@@@@@NULL'

Ceci est relativement simple lorsqu'on traite des éléments d'une suite qui ont une structure simple. C'est le cas des vaccins. Chaque vaccin est un simple code entre 0 et 255. Le problème est encore plus délicat lorsque les éléments de la suite sont de type complexe. C'est notamment le cas des médicaments. En effet, chaque médicament est composé de plusieurs champs; le DIN, la quantité, le PRN, la répétition... Appelons-les «sous-champs». Nous proposons d'introduire dans ces cas de figures complexes un nouveau séparateur, en l'occurrence le caractère '|'. Ce séparateur sert uniquement à distinguer les sous-champs. Il précède chaque sous-champ de manière à ce qu'il y ait autant de séparateurs que de sous-champs.



Sachant que le nombre de sous-champs d'un champ de médicament est fixe, en l'occurrence cinq, il est facile de coder une suite de médicaments de cette façon.

Exemple :
Nombre de médicament = 2
données de la transaction =
'@2@|DIN_1|Quantité_1|PRN_1|Durée_1|renouvellement_1@
|DIN_2|Quantité_2|PRN_2|Durée_2|renouvellement_2@@@@@@@@@@@@@@NULL'

Le mécanisme qui reconnaît et traite les champs (Interface Applicative) n'a pas besoins d'analyser les sous-champs. Il les considère comme parties intégrantes du champ. Tout ce qui est entre 2 caractères '@' est considéré comme étant un champ et est traité comme un tout. C'est un second mécanisme qui connaît la structure de chaque champ qui va reconnaître et analyser les sous-champs délimités par des '|'.

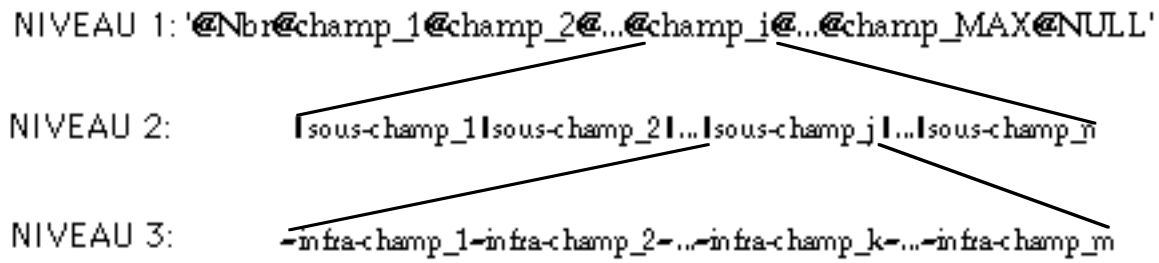
La codification des sous-champs se complique lorsque le nombre de ceux-ci est variable et inconnu. On parle alors de suite de sous-champs. C'est le cas des diagnostics. Les diagnostics sont une suite de champs diagnostiques. Chaque champ est composé d'un nombre de sous-champs fixe et d'une suite de sous-champs de conduites. Nous avons affaire à une *suite de suites.* Le mécanisme qui reconnaît les sous-champs ne limite pas le nombre de séparateurs '|' utilisés pour structurer un champ. Ceci nous permet de considérer les éléments d'une suite contenue dans un

champ comme étant des sous-champs. La suite sous-champs est donc «aplatie» pour composer un nombre indéterminé de sous-champs. Ces derniers sont des éléments de la suite précédés par un champ indiquant leur nombre. Cet indicateur est important puisqu'il permet de connaître le nombre de sous-champs, ce dernier étant variable.

Exemple
Nombre de diagnostics=2
Nombre de conduites du diagnostic 1 = 2
Nombre de conduites du diagnostic 2 = 3
données de la transaction=
'@2@|Table_1|Code_1|2|conduite_1.1|conduite_1.2
@|Table_2|Code_2|3|conduite_2.1|conduite_2.2|conduite_2.3@@@@@@@@@@@@@@@NULL'

Dans notre exemple, les conduites sont des simples codes ayant tous la même taille fixe. Au même titre que l'exemple des vaccins et des médicaments, si les conduites étaient de type complexe, c'est à dire composées de sous-éléments, on serait confronté à un autre problème non résolu avec seulement nos deux séparateurs (**@** et **|**). En effet, un sous-champ peut être structuré et composé de sous-éléments. Appelons-les «infra-champs». Pour distinguer les infra-champs au sein d'un sous-champ, nous introduisons un nouveau séparateur, en l'occurrence le caractère '**~**'. Chaque infra-champ est alors précédé par ce séparateur. Dans le Dossier Médical Portable, ce cas de figure se présente avec les variables de suivi. En effet, les variables de suivi sont constituées par une suite d'ensembles de variables complexes. Chaque ensemble est composé d'une sous-suite de variables d'un même type. Les variables de ces sous-suites sont à leur tour structurées en sous-éléments; type secondaire, code et valeurs. Ces sous-éléments sont des infra-champs. Dans notre cas, leur nombre est fixe et connu. Cependant on pourrait avoir un nombre variable d'infra-champs et traiter ainsi des *suites de suites de suites* d'éléments simples. Dans ce cas, l'ensemble des infra-champs d'une suite est précédé par un indicateur du nombre d'éléments permettant de traiter des suites de longueur variable.

La codification utilisant les trois séparateurs nous permettent de représenter des données ayant trois niveaux de complexité. On a, dans un même champ de données, une explicitation récursive à deux niveaux (sous-champs et infra-champs).



En réalité, on peut avoir quatre niveaux de complexité. En effet, la vraie limite emposée par l'Interface Applicative concerne le nombre de séparateurs '@' dans un enregistrement. Ce nombre doit être fixe et ne dépassant pas 255. Dans une transac-

tion de lecture ou d'écriture, on peut retrouver autant d'enregistrements que l'on veut dans une même chaîne terminée par le caractère nul.

5.2.2 Intégrité des données sur la carte

Les données du DMP peuvent être altérées à plusieurs étapes de l'utilisation de la Carte Santé. Un mauvais traitement affligé à la carte, par exemple, peut engendrer des altérations minimes. Ceci n'est pas fatal et peu probable, vu la grande fiabilité de la mémoire de la puce. L'intégrité des données est surtout compromise lors de l'écriture de l'information sur la carte. Un retrait manuel forcé de la carte ou une coupure électrique au cours d'une écriture physique suspend l'écriture et compromet la l'intégrité des données. L'information est alors partielle sur la carte. Une écriture partielle est parfois plus grave et dangeureuse que la non-écriture, surtout lorsqu'il s'agit d'information compressée; comme on l'a vu dans le chapitre IV, pour être décompressée, une information compressée doit rester intacte. Pour garantir l'intégrité des données sur la carte pour les applications qui les traitent, la Coquille assure les applications que les écritures qu'elles effectuent sont correctement et intégralement inscrites sur la carte. Dans le cas d'une interruption de l'écriture, celle-ci est considérée par la Coquille comme non effectuée et l'information partielle se trouvant sur la carte n'est pas transmise à l'application lors de la lecture. Ce mécanisme assurant l'intégrité des données est appelé mécanisme d'atomicité puisqu'il garantit l'intégralité d'une transaction d'écriture. Comme nous allons le voir, le mécanisme d'atomicité peut apporter des corrections à l'altération et recouvrer dans certains cas l'intégrité des données lorsqu'il s'agit de transactions physiques.

5.2.2.1 Atomicité logique

Une instruction d'écriture concernant une zone logique et transmise par une application à la Coquille peut générer plusieurs transactions physiques si les champs qui la composent sont sectionnés et stockés dans différentes zones physiques de la carte. C'est le cas général où les acteurs n'ont pas les mêmes droits d'accès sur toutes les variables d'une même transaction logique. Même si dans le cas du DMP une transaction logique génère une seule transaction physique parce que les champs d'un enregistrement sont stockés dans une même zone physique, le cas général pose des problèmes d'intégrité des données. Ces problèmes doivent être pris en considération. En supposant qu'une transaction logique TL soit sectionnée en plusieurs transactions physiques $\{TP_1, TP_2, ..., TP_n\}$, la transaction logique d'écriture TL n'est considérée comme terminée que lorsque toutes ses transactions physiques correspondantes TP1 à TPn sont transmises intégralement et avec succès à la carte. Pour garantir à l'application qui a envoyé l'instruction d'écriture, que les données de la transaction logique TL sont intègres dans la carte, il faut s'assurer que l'ensemble des transactions physiques $\{TP_1, TP_2, ..., TP_n\}$ soient toutes effectuées avec succès. Dans le cas contraire, il faut assurer à l'application que l'information partielle ne sera pas transmise et considérée comme inexistante. Pour ce faire, le mécanisme d'atomicité logique utilise un drapeau de signalisation pour indiquer le début puis la fin de la suite des transactions physiques TP_1 à TP_n. Dans le cas où le drapeau de

signalisation de la fin n'est pas positionné, cela indique que la transaction logique n'a pas été terminée et que ses données ne sont pas intégralement transmises à la carte. Positionner le drapeau de signalisation du début du processus consiste à inscrire, sur la carte dans la zone de contrôle, le numéro séquentiel et l'identification de la transaction logique. Positionner le drapeau de signalisation de la fin du processus consiste à effacer dans la zone de contrôle le numéro séquentiel et l'identificateur de la transaction logique qui vient d'être terminée. Ceci évite de cumuler des informations à la fois sur les bonnes et mauvaises transactions et de ne garder que les informations sur les transactions logiques qui n'ont pas été effectuées correctement pour pouvoir prendre les mesures nécessaires lors des lectures. Le numéro séquentiel d'une transaction logique peut être déterminé de deux façons différentes: soit en lisant une des zones physiques rattachées à la transaction logique en question et en dénombrant les enregistrements, soit en gérant sur la carte, dans la zone de contrôle, des compteurs pour toutes les différentes transactions logiques. Lors d'une lecture, si un drapeau de signalisation existe toujours dans la zone de contrôle, les données de la transaction logique correspondante au drapeau ne seront pas transmises à l'application.

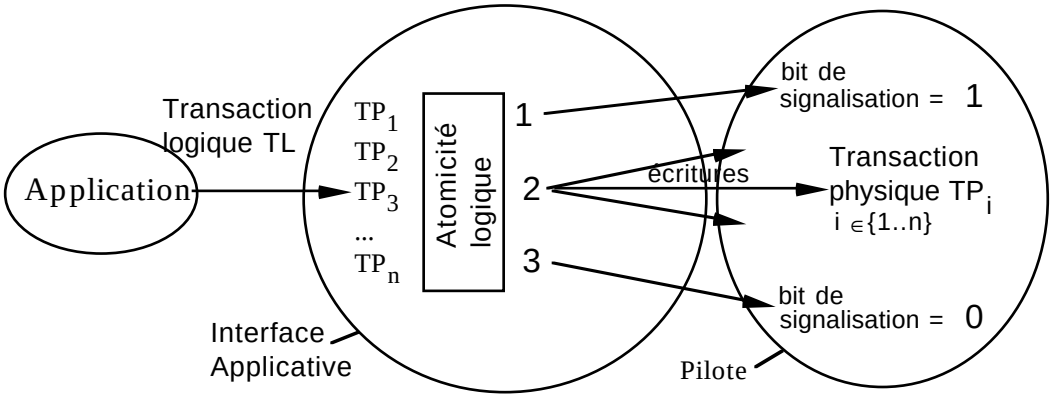


fig 5.6 : Mécanisme d'atomicité logique

5.2.2.2 Atomicité physique

L'atomicité physique garantit qu'un ordre d'écriture a été effectué avec succès. Il est assuré par le Pilote. Un ordre d'écriture peut être refusé par le microprocesseur de la carte dans le cas où l'accès à la zone de données considérée est interdit ou que la mémoire de la carte est saturée. La saturation peut parfois être résolue par le Pilote, mais dans le cas contraire, elle est définitive. Dans ces deux cas de refus d'écriture, l'échec est signalé à l'Interface Applicative qui prend alors les mesures nécessaires. L'impossibilité d'écriture peut aussi être due à des phénomènes externes comme une coupure d'électricité ou un retrait manuel prématuré de la carte. Dans ces deux cas, le Pilote tente de recouvrer l'erreur à la prochaine occasion. Le mécanisme d'atomicité ne fait que garder des traces pour savoir si une transaction d'écriture a été interrompue. Le procédé est simple: avant de transmettre l'ordre d'écriture avec les données à la carte, le Pilote inscrit, dans un fichier sur le disque dur du site, les données et l'identification de la zone physique, l'identificateur de l'acteur ayant souhaité l'écriture ainsi que la date de la transaction. Un drapeau de

signalisation est ensuite positionné dans la zone de contrôle de la carte réservée au Pilote, pour indiquer que l'écriture a commencé. Après la transmission de l'ordre d'écriture avec les données et la vérification du code retour de la carte, le drapeau de signalisation est supprimé de la zone de contrôle. Enfin, l'information inscrite dans le disque dur du site est effacée. Le drapeau de signalisation inscrit dans la zone de contrôle comprend l'identificateur de la zone, l'identificateur de l'acteur responsable de l'écriture et la date de la transaction. Si une séance d'écriture est interrompue, ce drapeau n'est pas supprimé. Lors d'une nouvelle session, la zone de contrôle est vérifiée. Si ce drapeau existe, le Pilote peut savoir qu'une écriture a été interrompue et peut identifier la zone concernée, l'acteur ainsi que la date de l'opération. Avec ces informations, le Pilote tente de rechercher l'information sur le disque dur du site et rétablit l'intégrité physique de la transaction. Si l'information ne se trouve pas dans le fichier parce que l'accident a eu lieu sur un autre site par exemple, l'erreur est signalée à l'Interface Applicative qui se charge de prendre les mesures nécessaires.

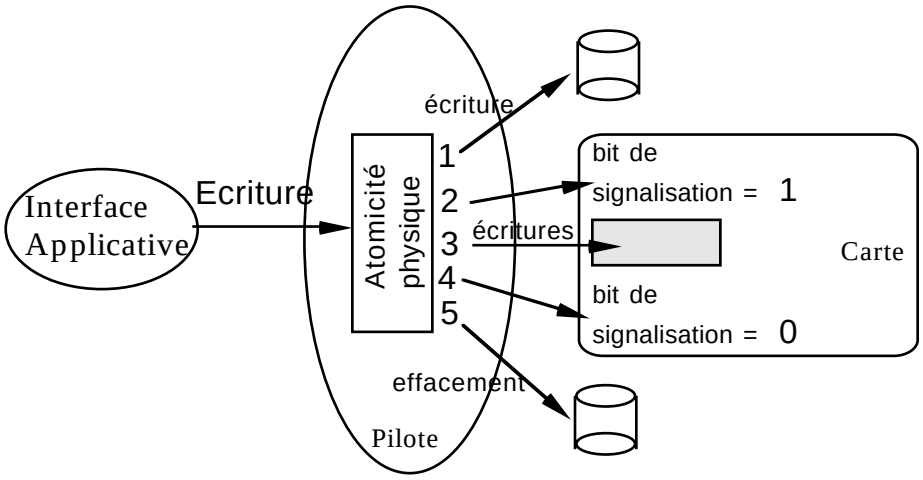


fig 5.7 : Mécanisme d'atomicité physique

Il est à signaler que lorsqu'une transaction logique n'est pas sectionnée en plusieurs transactions physiques, c'est-à-dire que toutes les variables de la transaction logique sont stockées dans la même zone physique, comme c'est le cas pour le DMP, le mécanisme assurant l'atomicité logique n'est pas nécessaire. C'est le mécanisme d'atomicité physique qui assure alors l'intégrité des données de ces zones logiques.

5.2.3 Annulation d'une information sur la carte

Une information du DMP inscrite sur la carte n'est jamais supprimée. Même si cette dernière est erronée, la suppression est interdite. Il existe cependant la possibilité d'annuler une information fausse ou obsolète. L'annulation peut se faire de deux façons différentes. La première est une annulation logique par ajout d'une nouvelle information contre-disant une ancienne donnée. Si par erreur le médecin un introduit 'O+' pour le groupe sanguin à la place de 'AB -', par exemple, il peut introduire une nouvelle fois le groupe sanguin. L'application qui reçoit les deux informations ne considère alors que la plus récente et n'affiche pas l'information

erronée. Il est à noter que dans ce cas, la Coquille transmet à la lecture les deux informations sans distinction. La deuxième possibilité d'annulation est plutôt physique et n'est permise qu'à l'émetteur. Ce droit d'accès en annulation n'est contrôlée que par la Coquille qui identifie l'acteur et non par le microprocesseur de la carte. Chaque transaction physique accompagnée d'une date et d'une signature comporte un bit de validation. Ce bit est normalement mis à 0 pour indiquer que l'information est valide. Lors d'une opération d'annulation, ce bit est mis à 1 pour signaler au Pilote que toute la transaction est annulée. Lorsque la Coquille reçoit une commande de lecture, les données annulées et les données non annulées sont indifféremment lues. La Coquille a alors deux possibilités selon sa paramétrisation, soit transmettre à l'application seulement les données non annulées, soit transmettre toutes les données et signaler celles qui sont annulées et celles qui ne le sont pas. Cette deuxième possibilité sert par exemple au programme de maintenance pour permettre à l'émetteur de vérifier le contenu et l'historique de la carte. Le processus d'annulation que nous venons de décrire s'opère au niveau physique des données. On peut facilement concevoir un processus d'annulation au niveau logique des données. Cela consisterait à annuler une transaction logique et non une transaction physique. L'annulation se fait par l'inscription, dans une zone de gestion sur la carte, du numéro séquentiel de la transaction à annuler. Après la lecture des données par la Coquille, une consultation de la zone de gestion permet de connaître les données annulées.

5.3 Données logiques vs données physiques

Les données logiques sont organisées en tables comme des relations d'une base de données relationnelle. Chaque relation correspond à une zone logique et est composée d'enregistrements. Ces enregistrements sont formés de champs ayant des types de données bien définies (voir chapitre II). Dans une zone logique, les enregistrements sont homogènes, c'est-à-dire que tous les enregistrements sont constitués des mêmes champs. Soit les enregistrements E_i et E_j de la zone logique z, E_i et E_j ont la même structure de champs. Cette structure de champs est respectée par tous les enregistrements même si un champ identique dans deux enregistrements différents peut avoir des tailles différentes et que certains champs peuvent même être absents (ou vides) dans certains enregistrements. Les enregistrements logiques, quelle que soit la zone logique à laquelle ils appartiennent, sont toujours terminés par un champs pour indiquer la date de la transaction et un champ indiquant la signature de l'acteur auteur de la transaction.

Les données physiques sont elles aussi organisées en tables. Chaque table correspond à une zone physique et est composée d'enregistrements, sauf que les enregistrements sont hétérogènes. En effet, comme pour les enregistrements logiques, les enregistrements physiques sont formés de champs, sauf que ces derniers ont une structure de champs qui peut être différente d'un enregistrement à l'autre. Soit les enregistrements E_i et E_j de la zone physique z, E_i et E_j n'ont pas nécessairement la même structure de champs. On parle alors de type d'un enregistrement au sein d'une zone physique. À titre d'exemple, on peut citer le type d'enregistrement du groupe sanguin dans la zone d'urgence du DMP qui contient aussi

le type d'enregistrement des pathologies et prothèses et le type d'enregistrement des autres urgences. On peut avoir jusqu'à sept types d'enregistrement différents dans une même zone physique (voir annexe B). Les types d'enregistrement sont distingués par des identificateurs. Les enregistrements d'un même type dans une même zone physique sont quant à eux homogènes et gardent une même structure de champ propre au type d'enregistrement.

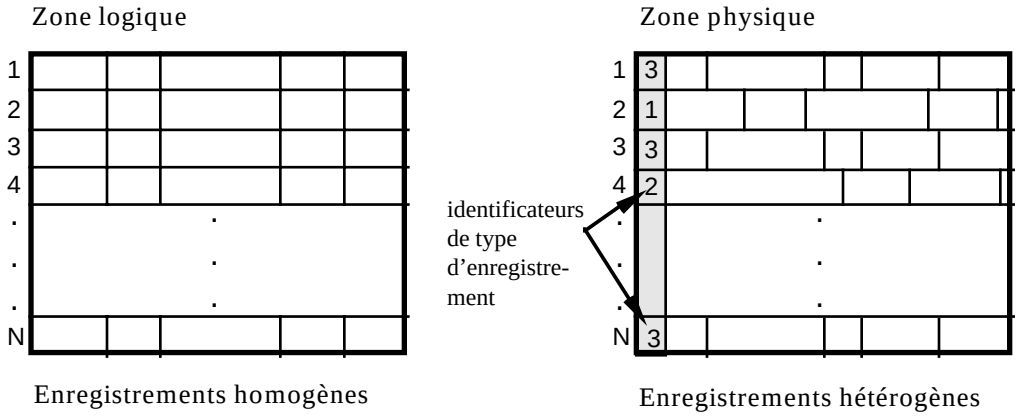


fig 5.8 : Structure des champs des zones logiques et des zones physiques

5.3.1 La table des correspondances

La Coquille se charge d'effectuer une correspondance entre les données logiques et les données physiques. Cette correspondance est établie grâce à une table de correspondance qui est externe au programme. La table est donnée en paramètre à la Coquille dans un fichier externe chiffré. Cette paramétrisation permet de modifier la correspondance des données sans porter des modifications aux programmes de la Coquille. Cette table est composée de cinq colonnes. La première et la seconde colonne indiquent la zone logique et le champ logique concernés alors que les colonnes 3, 4 et 5 indiquent respectivement le champ physique correspondant, l'identificateur du type d'enregistrement physique et la zone physique correspondante. Avec cette table, on peut connaître la structure logique d'un enregistrement logique grâce aux deux premières colonnes. Les trois dernières colonnes nous permettent de connaître les différents types d'enregistrement d'une zone physique et la structure de champ de chaque enregistrement physique. Une ligne de cette table nous donne la correspondance entre un champ d'une zone logique avec un champ d'une zone physique et indique le type d'enregistrement auquel appartient le champ physique correspondant.

Zone logique	Champ logique	Champ physique	Identificateur du type d'enregistrement	Zone physique

5.3.2 Le mécanisme de correspondance

Le mécanisme de correspondance, appelé aussi «mapping», est assuré par la Coquille à la réception des transactions d'une application. On parle alors de correspondance logique / physique ou lors du transfert d'information vers une application, on parle alors de correspondance physique / logique. Dans les deux sens, le mécanisme repose sur la table décrite dans le paragraphe précédent. Pour la correspondance logique / physique, après avoir vérifié le nombre de champs dans la transaction logique, le mécanisme fait coïncider pour chaque champ logique son champ physique correspondant. Une fois cette corrélation établie, le mécanisme rassemble les champs physiques par zones physiques concernées. Chaque sous-ensemble ainsi obtenu concerne une zone physique donnée. Les champs de chaque sous-ensemble sont alors regroupés par type d'enregistrement suivant l'identificateur de type d'enregistrement correspondant dans la table. Chaque groupe ainsi obtenu équivaut à une transaction physique. Dans le cas particulier du DMP, une transaction logique n'est pas subdivisée en plusieurs groupes mais coïncide avec une unique transaction physique qui correspond plus particulièrement à un type d'enregistrement donné dans une zone physique. La correspondance physique / logique rassemble les données de plusieurs transactions physiques dans une ou plusieurs transactions logiques en recherchant le correspondant logique de chaque champ physique dans la table de correspondance. Les champs logiques qui ne trouvent pas de correspondants restent vides. En général, une transaction physique peut contenir des champs physiques provenant de différentes transactions logiques. C'est la raison pour laquelle la correspondance physique / logique peut générer plusieurs transactions logiques pour une application. Dans le cas particulier du DMP, une transaction physique correspond, comme nous l'avons déjà précisé, à un type d'enregistrement donné d'une zone physique. Cet enregistrement coïncide directement avec une transaction logique d'une zone logique. Le mécanisme de correspondance ne fait alors que rechercher les homologues logiques des champs physiques et remplir en conséquence la chaîne de données à transmettre. Une fois cette opération terminée, les champs logiques restants sont signalés comme étant vides et la transaction logique est transmise à l'application.

5.3.3 La structure des champs et les données physiques

Nous avons présenté dans le second chapitre les différents types de données utilisés dans le DMP. On distingue les booléens, les entiers, les réels, les caractères et les dates. Chaque champ physique a un type précis et la Coquille utilise cette information pour coder toutes les données. Suivant leur type de données, les champs physiques sont transcrits pour occuper le moins d'espace possible dans la mémoire de la carte. Cependant, les champs physiques n'ont pas tous un type de données élémentaire et peuvent être structurés ou regrouper des listes comme nous l'avons présenté dans le paragraphe 5.2.1. On parle alors de sous-champs et d'infra-champs. Un champ peut donc être de type élémentaire (booléen, entier, réel, caractère ou date) ou de type structuré. Un champ structuré est composé de sous-champs de type élémentaire ou structuré en infra-champs. Les infra-champs sont forcément de type élémentaire. La structure n’a que trois niveaux possibles. Un

champ peut aussi être de type liste. On distingue alors les listes d'éléments de même type élémentaire et les listes d'éléments de même type structuré.

Pour reconnaître les champs, ceux-ci sont numérotés. Un champ logique a un numéro entre 0 et 254 et est identifié par son numéro ainsi que par le numéro de la zone logique à laquelle il appartient. Au niveau physique, on identifie aussi un champ par rapport à sa zone physique et non par rapport au type d'enregistrement auquel il appartient. Son numéro dans la zone est toujours entre 0 et 254. Un type d'enregistrement a un numéro au sein d'une zone physique et est identifié par ce numéro et le numéro de la zone à laquelle il appartient. Sachant que le numéro 255 est réservé pour la gestion interne, cette technique de numérotation et d'identification limite le nombre de champs logiques à 254 par zone logiques ainsi que le nombre de champs physiques à 254 par zone physique, tout type d'enregistrement confondu. Par contre, chaque zone physique a sa propre limite quant au nombre de types d'enregistrement qui tient à la taille de l'identificateur (voir annexe B).

5.3.3.1 La table des structures des zones

La table de correspondance que nous utilisons avec le «mapping» nous permet d'avoir la structure en champs d'une transaction mais pas la structure d'un champ en sous-parties. Deux nouvelles tables nous permettent de reconstituer la structure à deux niveaux des champs. La première table contient la composition des champs structurés en sous-champs. Elle comporte deux colonnes: la première pour identifier le champ et la seconde pour identifier le sous-champ. Elle est utilisée chaque fois qu'un champ a un type structuré ou un type liste d'éléments structurés. La deuxième table contient la composition des sous-champs structurés en infra-champs. Elle comporte aussi deux colonnes: la première pour identifier le sous-champ et la seconde pour identifier l'infra-champ. Elle est utilisée chaque fois qu'un sous-champ a un type structuré ou un type liste d'éléments structurés. Ces deux tables peuvent être fusionnées dans une même table. En effet, comme nous allons le voir au paragraphe 5.3.4, ces deux tables sont regroupées dans la même structure de données.

5.3.3.2 La table des codifications

Après avoir identifié les structures atomiques des éléments informationnels constituant les transactions de données, le Pilote procède à la codification des informations élémentaires suivant leurs types de données respectifs. La table des codifications sert à reconnaître le type de données de chaque champ, sous-champ ou infra-champ et à codifier l'information en conséquence. Lors de la lecture, cette table sert à la codification en permettant d'identifier les bits à appréhender par composant de la structure. Un composant d'une structure est soit un champ, un sous-champ ou un infra-champ. La table associe à chaque composant un type de données. Ce type de données est structuré sur un entier de 16 bits. Les trois premiers bits du premier octet de cet entier indiquent le type de données primaire du composant. Les types de données sont codés de la façon suivante:

0	000	booléan
1	001	entier
2	010	réel

3	011	caractère
4	100	date
5	101	liste d'éléments simples
6	110	liste d'éléments structurés
7	111	structure

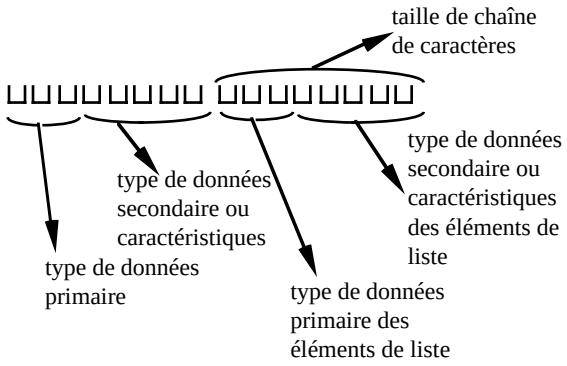


fig 5.9 : Structure d'un indicateur de type

Dans le premier octet, les cinq bits qui suivent indiquent, dans le cas du type booléen, le nombre de positions binaires utilisées, ce qui nous limite à 32 positions binaires possibles (bitstream de 32). Dans le cas des entiers, ils codifient le type utilisé. Rappelons que nous avons 17 types d'entiers différents (voir 2.5: les types de données manipulés). De la même façon, ces cinq bits codifient le type utilisé pour le cas des réels. Nous rappelons aussi que nous avons prévu dix types de réels différents. Dans le cas des caractères, nous disposons de quatre codifications possibles (BCD, ISO5, ISO6 et ASCII7). Les cinq bits servent alors à indiquer la codification utilisée alors que le deuxième octet indique la taille de la chaîne, qui devrait normalement se limiter à 64 mais nous permettant dans un usage futur d'augmenter la taille des chaînes à 255. Si la taille de la chaîne est variable, alors le deuxième octet contient zéro (0) et la taille est indiquée par les premiers bits de la chaîne de caractères elle même (voir figure 2.16, page 58). Dans le cas des dates, les cinq bits codifient les quatre types de dates que nous avons présentés au chapitre II. Si le composant est une liste d'éléments de type simple alors le deuxième octet indique le type des éléments de cette liste de la même façon que le premier octet: les trois premiers bits indiquent le type primaire et les cinq derniers indiquent le type secondaire. Ceci nous limite à n'avoir que des listes de booléen, d'entiers, de réels ou de dates puisque ceux-ci n'utilisent qu'un octet pour leur identification. On ne peut donc pas avoir des listes de chaînes de caractères, ce qui est le cas pour le DMP. Les listes de structures et les types structurés indiquent au Pilote qu'il faut procéder à une analyse du composant. Nous rappelons que les infra-champs ne peuvent pas avoir un type structuré ou être des listes.

5.3.4 Les structures de données utilisées pour le «mapping» et la codification

Pour des raisons de performance et d'économie d'espace-mémoire, les différentes tables sont regroupées dans deux structures de données principales. La première structure concerne le mécanisme de correspondance et la description du type de chaque champ physique. Elle est globale à l'ensemble des modules de la Coquille puisqu'elle est exploitée par l'Interface Applicative pour le mécanisme de correspondance et utilisée par le Pilote pour la codification des champs. Cette structure de données est locale au Pilote et ne se rapporte qu'aux sous-champs et infra-champs. La première est définie en langage C de la façon suivante:

```
struct strChampLogique
    {
        BYTE champ_physique;
        BYTE ident_type_enregistrement;
        BYTE zone_physique;
        int type_champ_physique;
    }

struct strEnregistrementLogique
    {
        struct strChampLogique enregistrement[MAXCHAMPLOGIQUE];
    }
struct strEnregistrementLogique TableCorrespondanceLP[ MAXZONELOGIQUE];
```

Dans la figure 5.10, on retrouve la représentation graphique de cette structure. Logiquement, elle reconstitue la table de correspondance explicitée dans le paragraphe 5.3.1 et représentée dans la figure 5.11.

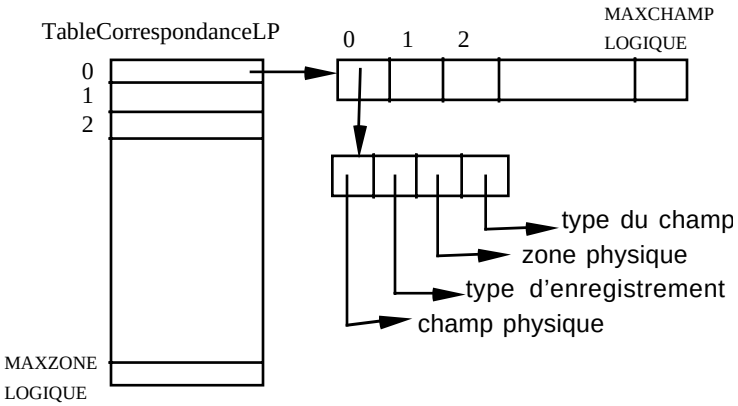


fig 5.10 : Représentation graphique de la première structure de données.

Zone Logique	Champ Logique	Champ Physique	Type	Type d'enregis-trement	Zone Physique

fig 5.11 : Représentation logique de la première structure de données

La seconde structure de données concerne la reconstitution des compositions à deux niveaux des champs structurés et des champs de type liste. Elle contient aussi des informations sur les types de données élémentaires de chaque sous-champ ou infra-champ. Le Pilote fait appel à celle-ci chaque fois qu'un champ est de type structuré. Lorsqu'un champ est de type élémentaire, le Pilote utilise alors uniquement la première structure de données. La seconde structure de données est sous forme de table à quatre colonnes: trois colonnes pour spécifier la composition des champs et sous-champs en composants élémentaires et une quatrième colonne pour

indiquer le type du composant. Lorsqu'un champ est élémentaire, la colonne «sous-champ» et la colonne «infra-champ» contiennent le numéro 255 indiquant que le type de la ligne courante appartient au champ. De même lorsqu'un sous-champ est de type élémentaire, la colonne «infra-champ» contient le numéro 255 pour indiquer que le type est celui du sous-champ. Dans un autre cas, le type est celui de l'infra-champ. Normalement, les champs élémentaires ne figurent pas dans cette table puisqu'ils sont déjà spécifiés dans la première structure. Dans la table, le numéro du champ englobe le numéro de la zone physique à laquelle il appartient. De ce fait, le numéro est un entier dont le premier octet indique la zone physique et le second octet indique le champ même. La deuxième structure de données est définie en langage C de la façon suivante:

Champ	Sous-Champ	Infra-Champ	Type

fig 5.12 : table pour la composition et la codification

```
struct strCodificationComposante
    {
        BYTE zone_physique;
        BYTE champ_physique;
        BYTE sous_champ_physique;
        BYTE infra-champ_physique;
        int type_composant;
    }

struct strCodificationComposante TableCodification[MAXCOMPOSANTES];
```

5.4 Le Pilote

Le Pilote est le module qui communique avec l'Interface Applicative et les cartes. Il reçoit des commandes de l'Interface Applicative, transmet des ordres aux cartes puis, à la lumière des réponses des cartes, il répond aux commandes de l'Interface Applicative. L'Interface Applicative et le Pilote communiquent au moyen d'une zone mémoire commune. Ils partagent des variables globales et lors de l'appel du Pilote, l'Interface Applicative lui transmet une commande et des pointeurs de tampons à utiliser pour l'échange de données. Le Pilote se trouve donc être une fonction (en langage C) avec deux paramètres: le premier identifie la commande à réaliser et le second contient les pointeurs des tampons d'échange de données. Le Pilote est réalisé avec une architecture ouverte permettant une modification simple. En effet, de nouvelles fonctionnalités peuvent être ajoutées à tout moment. De la même façon, des anciennes fonctionnalités peuvent être supprimées ou modifiées. Cette architecture ouverte confère donc au Pilote un caractère versatile.

5.4.1 Le «Super Driver» ou Pilote Universel

Le Pilote dispose d'un ensemble d'ordres universels qui s'appliquent à tous les cas de figure et qui lui permettent de traiter diverses cartes. Cette universalité du

Pilote permet aux applications d'être relativement indépendantes de l'évolution technologique des cartes intelligentes. Le Pilote est lui aussi évolutif et s'adapte à son environnement informatique.

Dans certaines circonstances, le «Super Driver» joue le rôle d'accélérateur [GAMACHE 90]. En effet, il garde un Tampon miroir où il emmagasine pour la durée d'une session une image fidèle du contenu des zones de la carte. Après chaque lecture physique, les données sont stockées en mémoire dans ce tampon pour éviter d'accéder de nouveau à la carte à la suite d'un ordre de lecture ultérieure pour les mêmes données. Les accès à la carte sont relativement lents. Ainsi, en évitant de lire une zone déjà lue, le Pilote accélère l'exécution en fournissant directement les données emmagasinées dans son tampon. Quant à elles, les écritures sont exécutées à chaque fois sur la carte pour assurer une intégrité des données. Néanmoins, les informations dans les tampons miroirs sont automatiquement mises à jour en conséquence pour accélérer les lectures suivantes. Il est à noter que ce n'est pas la totalité de la carte qui est lue et emmagasinée en mémoire mais seulement le contenu des zones lues à la suite d'une commande de l'Interface Applicative. Les zones non consultées ne sont pas inutilement lues. Cette sauvegarde temporaire en mémoire est aussi une garantie lors des écritures puisque le Pilote dispose ainsi de l'information en mémoire tant que l'écriture physique n'a pas été achevée.

5.4.2 Les fonctionnalités

Globalement, le Pilote a quatre fonctions principales et des fonctions annexes. Ces fonctions principales sont de formater et d’analyser les données, de compresser et décompresser les données, de reconnaître les différentes technologies et de générer des ordres à la carte. Les fonctions annexes concernent la génération des statistiques, l’assurance de l’intégrité physique des données et le lien avec le micro-serveur du Centre d’automatisation et de paiement des services de santé (CAPSS). Dans ce paragraphe, nous ne faisons référence qu’aux fonctions dites principales.

Nous présentons à l’annexe C une analyse fonctionnelle du système Carte Santé avec une focalisation sur le Pilote de l’application Coquille réalisée avec la méthode E.P.A.S. (Entrée, Processus, Accumulation, Sortie) [MOULIN 88]. On retrouve dans cette annexe le diagramme global du système Carte Santé ainsi que les plans des nouveaux systèmes-objets liés à la focalisation du processus Pilote. Tous les diagrammes sont accompagnés d’un guide de présentation expliquant la fonctionnalité de chaque processus.

5.4.2.1 Formatage et analyse des données

Le formatage et l’analyse des données est une fonction délicate qui consiste à analyser la chaîne de caractères contenant les données pour reconnaître les champs, les différents sous-champs et les infra-champs. Avant de retransmettre des données à l’Interface Applicative, ce module reconstruit la chaîne de caractères telle que reçue. C’est ce qui constitue le formatage des données. Le formatage procède par

une recodification des données binaires reçues de la carte en utilisant la table de codification et la table des structures des zones pour reconnaître les champs dans les suites de bits, puis il reconstitue la chaîne de caractères avec des séparateurs entre les champs, les sous-champs et les infra-champs tels que décrits au paragraphe 5.2.1. L'analyse effectue le travail inverse à la réception de l'information. En recherchant les séparateurs **@**, **|** et **~**, le module reconnaît les champs, les sous-champs et les infra-champs puis procède à une véritable analyse syntaxique en comparant la structure obtenue avec la structure donnée par la table des structures des zones. Si le résultat de la comparaison est positif, l'analyse continue par coder chaque champ par son code respectif suivant son type de données identifiés dans la table de codification. Une élimination des redondances est aussi effectuée en factorisant les champs par date et signature, si ceux-ci appartiennent à la même date et à la même signature. Le formatage remet alors ces redondances, si nécessaires, pour que l'information soit renvoyée telle que reçue.

5.4.2.2 Compression des données

Le module de compression des données, comme son nom l'indique, compresse l'information destinée au DMP pour tenter de stocker plus de données sur la carte et de rallonger la vie de celle-ci. Lors d'une écriture d'une information I dans une zone physique de la carte, le contenu complet de la zone Iz est lu, auquel le Pilote rajoute la nouvelle information. Le module tente ensuite de compresser l'information Iz+I. S'il y a gain d'espace, l'information Iz+I est stockée compressée. Dans le cas contraire, l'information est stockée non compressée. Un bit de signalisation indique à tout moment si le contenu d'une zone est compressée ou non. Lors d'une lecture, ce bit est d'abord consulté. Dans le cas où l'information est reconnue compressée, le module la décompresse. La vérification de la possibilité de compression vient du fait qu'une information de petite taille n'est pas toujours compressable et une augmentation de taille est alors notée. Ce qui est contraire au but escompté. Le Pilote fait appel à la compression et décompression avant chaque ordre d'écriture et de lecture. L'algorithme utilisé est LZHUF (voir chapitre IV).

5.4.2.3 Génération d'ordres pour la carte

Générer des ordres pour la carte intelligente est la fonction principale du Pilote. Toute la communication entre la carte et le Pilote se base sur ces derniers. Globalement, il y a des ordres entrants et des ordres sortants relativement à la carte. Les ordres entrants sont ceux qui porte l'information du Pilote vers la carte alors que les ordres sortants sont ceux qui ramènent de l'information de la carte vers le Pilote. Tous les ordres, qu'ils soient entrants ou sortants, gèrent un délai (time-out). Cette temporisation est un des paramètres du Pilote et est la même pour tous les ordres. Après le dépassement du temps imparti, l'exécution d'un ordre est jugée comme un échec. Les ordres sont implantés par des fonctions en langage C. Ces fonctions sont toutes normalisées et ont les mêmes paramètres à savoir: un tampon de données (CQ_TamponDonnées) utilisé pour les données en entrée pour les fonctions entrantes et utilisé pour les données en sortie pour les fonctions sortantes, un indicateur de l'état de la carte après exécution (CQ_StatutCarte) et un indicateur de l'état de l'opération après exécution (CQ_StatutOrdre).

CQ_StatutCarte peut avoir les valeurs suivantes:

0 si l'ordre est accepté et exécuté par la carte
1 si la carte est absente
2 si la carte est muette ou inconnue
3 si la carte est bloquée
4 si la carte est non personnalisée
5 si la protection n'est pas levée
6 si la carte est saturée

CQ_StatutOrdre peut avoir les valeurs suivantes:

0 si tout est correct
1 si le délai accordé est dépassé (time-out)
2 si les paramètres reçus sont incorrects
3 si une interruption externe a eu lieu

La déclaration d'une de ces fonctions est alors:

```
char fonction(BYTE * CQ_TamponDonnées, char CQ_StatutCarte, char CQ_StatutOrdre);
```

Les fonctions renvoient toutes un code retour (CQ_CodeRetour) qui a deux valeurs possibles: 1 lorsque l'ordre a été exécuté sans problème et 0 lorsqu'il y a eu un quelconque problème lors de l'exécution.

Nous présentons dans ce qui suit deux librairies: une pour les ordres communs de gestion et une deuxième réservée aux ordres de maintenance.

5.4.2.3.1 Ordres communs de gestion

Les ordres communs de gestion sont des ordres indispensables pour permettre à une application quelconque de manipuler des données sur une carte à microprocesseur. Nous avons relevé onze ordres en tout, constituant un ensemble nécessaire et suffisant de fonctions pour effectuer les opérations élémentaires sur une carte comme la lecture, l'écriture ou l'ouverture et la fermeture du microcircuit électrique. Nous présentons dans ce qui suit le descriptif détaillé des fonctions de cet ensemble.

Initialiser le lecteur (ordre entrant): Certains lecteurs de cartes intelligentes nécessitent une initialisation pour connaître le protocole et les paramètres de communication. Cette initialisation informe le lecteur du port utilisé, de la vitesse de transfert, etc. Certains lecteurs, dits «intelligents», attendent aussi le transfert d'un profil d'accès qui leur est propre. Cette fonction permet aussi d'initialiser les variables globales nécessaires au driver de la technologie.

Mettre la tension (ordre entrant): Le microcircuit d'une carte à microprocesseur nécessite une alimentation électrique externe pour fonctionner. Cet ordre permet

d'appliquer un voltage (Vcc et Vpp) aux contacts de la puce en invoquant une remise à zéro de tous les registres du microprocesseur. En générant un signal d'horloge, la carte est mise à l'état initial. Après cet ordre, la carte renvoie une réponse au RESET. Sans mise en tension, la carte reste muette à tout ordre. Il est donc nécessaire de transmettre cet ordre à la carte au début d'une session. S'il n'y a aucune carte dans le lecteur, un code de retour est transmis en conséquence, ce qui permet de réaliser au moyen de cet ordre une fonction d'attente d'insertion.

Enlever la tension (ordre entrant): Il est recommandé de ne pas retirer une carte alors qu'il existe une tension sur les connecteurs. Une telle manoeuvre peut endommager la puce. C'est pourquoi on suggère de toujours enlever la tension aux bornes de la puce avant de la retirer. Les lecteurs qui permettent l'insertion totale des cartes enlèvent automatiquement le voltage des connecteurs de la carte avant de l'éjecter. En générant un signal d'horloge, cet ordre permet de mettre le voltage des lignes à 0 volts. Après l'exécution de cet ordre, la carte redevient muette.

Éjecter la carte (ordre entrant): Certains lecteurs «avalent» la carte au complet en permettant l'insertion totale de la carte. De tels lecteurs évitent les accidents dûs au retrait prématuré des cartes pendant une opération d'écriture. L'ordre d'éjection s'adresse au lecteur pour lui demander de couper l'alimentation électrique au niveau des connecteurs avec la puce et d'éjecter la carte à l'extérieur. Si le lecteur utilisé ne retient pas la carte au complet et qu'il est possible de la retirer manuellement, cet ordre attend le retrait de la carte en effectuant une boucle jusqu'à ce que celle-ci soit sortie du lecteur ou jusqu'au dépassement du temps imparti (time-out).

Ouvrir une zone physique (ordre entrant): L'ouverture d'une zone physique consiste à se positionner au début d'une zone pour préparer une lecture ou une écriture éminente. Cet ordre permet donc de se positionner sur une zone donnée. Il suppose que les contrôles d'accès ont préalablement été faits. En cas de refus de la carte, le Code de retour le signale. Si la zone désirée n'existe pas, l'ordre d'ouverture se charge de la créer en faisant appel à l'ordre de création de zone physique. Avec une carte disposant d'une hiérarchie de répertoires, cet ordre se charge de parcourir le chemin de l'arborescence menant à la zone désirée. Lors de l'appel de cet ordre, le tampon CQ_TamponDonnées contient l'identificateur de la zone sur le premier caractère.

Créer une zone physique (ordre entrant): L'ordre de création de zones physiques permet d'allouer un bloc de mémoire à une zone donnée. Il respecte la politique de gestion de mémoire utilisée. Cet ordre se charge de faire les liens logiques entre différents blocs de mémoire d'une même zone physique. Lors de l'appel de cet ordre, le tampon CQ_TamponDonnées contient l'identificateur de la zone sur le premier caractère.

Lire dans une zone physique (ordre sortant): L'ordre de lecture effectue la lecture d'une zone physique au complet. Si la zone a déjà été lue au court de la même session, l'ordre de lecture ne s'adresse pas à la carte mais consulte le tampon de la zone en mémoire réservée au Pilote. Dans le cas contraire, la zone est lue dans sa totalité et est en même temps emmagasinée dans un tampon. L'utilisation en par-

allèle d'un tampon pour stocker les données permet de minimiser les accès en lecture à la carte. Ces accès sont consommateurs de temps d'exécution. En effet, il est inutile de lire une information qui a déjà été lue. Si une nouvelle lecture de la carte s'impose, cet ordre vérifie si l'information lue est compressée et le signale dans ses paramètres de retour. Lors de l'appel de cet ordre, le tampon CQ_TamponDonnées contient l'identificateur de la zone sur le premier caractère. En sortie, ce tampon contient sur les deux premiers caractères la taille en octets des données transmises, suivie des données proprement dites. Les données peuvent contenir le caractère nul. C'est pourquoi le caractère nul n'est pas utilisé comme indicateur de fin de chaîne.

Écrire dans une zone physique (ordre entrant): Suivant le statut de la zone physique, l'ordre d'écriture effectue une écriture ou une réécriture. L'écriture consiste en un ajout d'informations alors que la réécriture consiste en une modification d'informations comme c'est le cas pour les zones de gestion. Dans tous les cas, le contenu complet de la zone est réécrit dans la carte. L'ordre d'écriture procède par le transfert du contenu de la zone dans le tampon en mémoire si l'information n'y est pas encore emmagasinée. Pour une écriture, l'opération consiste alors à ajouter la nouvelle information dans le tampon et de réécrire ce dernier. Une réécriture consiste à modifier le contenu du Tampon avant de le réécrire sur la carte. L'utilisation en parallèle d'un tampon pour stocker les données permet de minimiser les accès en lecture à la carte. Lors de l'appel de cet ordre, le tampon CQ_TamponDonnées contient l'identificateur de la zone sur le premier caractère, suivi de deux caractères contenant la taille en octets des données transmises. À partir du quatrième caractère, on retrouve les données proprement dites. Les données peuvent contenir le caractère nul. C'est pourquoi le caractère nul n'est pas utilisé comme indicateur de fin de chaîne.

Soumettre une clé (ordre entrant): L'ordre de soumission de clé transmet une clé à la carte en lui demandant de la comparer avec celle en sa possession. Il doit recevoir l'identificateur de la clé pour transmettre les codes d'exécution nécessaires à la carte. Cet ordre se charge d'assembler la clé dans le format physique dans lequel elle est stockée dans la puce. Dans le cas d'un refus de la part de la carte, le code retour y fait référence. Lors de l'appel de cet ordre, le tampon CQ_TamponDonnées contient l'identificateur de la clé à soumettre sur le premier caractère, suivi de la taille de la clé sur un caractère et de la clé proprement dite.

Analyser la réponse au RESET (ordre sortant): Cet ordre permet d'analyser la réponse au RESET transmise par la carte après sa mise en tension. La réponse au reset est standardisée et son analyse permet de reconnaître, par exemple, la technologie de la carte utilisée ou le protocole de communication utilisé. L'analyse de la réponse au RESET fait appel à l'ordre de mise en tension de la puce.

Analyser le statut de la carte (ordre sortant): Cet ordre permet de connaître l'espace disponible dans la mémoire de la carte. Il fait appel à l'ordre de lecture des zones physiques et calcule la proportion de la mémoire qui reste encore disponible pour l'écriture de nouvelles données en terme de pourcentage. Il permet aussi de connaître l'acteur titulaire de la carte. Après exécution, le tampon CQ_TamponDonnées

contient sur le premier caractère la proportion en pourcentage de l'espace disponible. Sur les deux caractères suivant. On retrouve la capacité disponible de la mémoire en octets. Enfin sur le quatrième octet est indiqué l'acteur titulaire de la carte.

5.4.2.3.2 Ordres de maintenance

Les ordres de maintenance ne sont pas utilisés par les applications pharmaceutiques ou médicales mais sont exclusivement réservés aux applications de gestion et de maintenance de l'émetteur. Ils permettent à l'émetteur de gérer les différentes clés d'accès et d'effectuer certaines opérations sur la carte lors de sa phase de personnalisation ou sa phase de blocage (hibernation). Les quatre ordres de maintenance principaux sont les suivants:

Créer une clé (ordre entrant): Cet ordre permet de créer toutes les clés de la carte lors de la phase de personnalisation de celle-ci. Si le masque de la carte le permet, cette fonction permet aussi de rajouter de nouvelles clés au cours de la phase d'exploitation. Suivant la technologie, cette fonction vérifie la phase du cycle de vie de la carte et fait une demande de création de clé en fournissant la nouvelle clé au masque. Elle renvoie un code retour indiquant l'état d'exécution de l'ordre. Lors de l'appel de cet ordre, le tampon CQ_TamponDonnées contient l'identificateur de la clé à créer sur le premier caractère, suivi de la taille de la clé sur un caractère et de la clé proprement dite.

Modifier une clé (ordre entrant): Tout dépendant des possibilités offertes par la technologie de la carte, cet ordre permet de modifier une ancienne clé. Il peut être effectué sur une carte d'habilitation comme sur une carte de bénéficiaire. Il permet par exemple de modifier le numéro d'identification personnel (NIP) d'un médecin si celui-ci le désire. Pour être modifiée, une ancienne clé doit normalement avoir déjà été soumise ainsi que la clé de l'émetteur. La soumission de ces clés se fait avec l'ordre de soumission de clé de la librairie des ordres communs de gestion. L'ordre de modification de clé renvoie un code retour indiquant l'état de son exécution. Lors de l'appel de cet ordre, le tampon CQ_TamponDonnées contient l'identificateur de la clé à modifier sur le premier caractère, suivi de la taille de la nouvelle clé sur un caractère et de la nouvelle clé proprement dite.

Personnaliser une carte (ordre entrant): Cet ordre permet de signaler sur la carte la fin de la phase de personnalisation et le début de la phase d'exploitation. Il consiste à 'brûler un fusible' sur la carte pour indiquer définitivement ce changement d'état. Suivant les technologies, certaines opérations ne sont plus permises après la fin de la phase de personnalisation. Cet ordre est irréversible. Il renvoie un code retour indiquant si la carte l'a bien accepté.

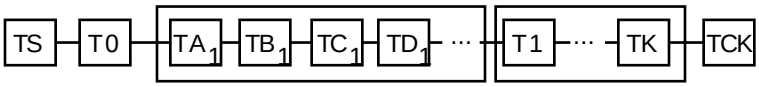
Déverrouiller une carte (ordre entrant): Lorsqu'une carte détecte que l'on essaye d'accéder frauduleusement à ses données en décelant des soumissions de clés erronées consécutives (ratification), elle s'auto-verrouille et passe en phase de blocage. Elle devient totalement inactive. Cet ordre de maintenance permet de

déverrouiller la carte et de la ramener à la phase d'exploitation. Il exige la soumission de la clé de l'émetteur qui doit être faite avec l'ordre de soumission de clé de la librairie d'ordres communs de gestion. L'ordre de déverrouillage peut échouer et la carte peut alors être annulée. Un code retour indique l'état d'exécution de la fonction.

5.4.2.1 Reconnaissance des différentes technologies

La particularité originale du Pilote, le «Super Driver» est qu'il peut reconnaître et dialoguer avec différents types et technologies de cartes intelligentes. Le mécanisne de reconnaissance des technologies se base sur la standardisation de la réponse au RESET des microprocesseurs des cartes intelligentes [ISO 89]. On retrouve dans la réponse au RESET des informations sur le type de communication, le protocole, le masque, le manufacturier de la puce, etc. Avec ces informations, le Pilote peut se configurer pour pouvoir dialoguer avec la carte.

Les normes ISO définissent la structure de la réponse au RESET comme étant constituée d'un caractère de synchronisation (**TS**) suivi d'un maximum de 32 caractères structurées comme suit: un caractère de format (**T0**), une suite de caractères d'interface (**TA_i**, **TB_i**, **TC_i**, et **TD_i**), une suite de caractères d'historique (**T1** à **TK**) et un caractère de vérification (**TCK**).



Les caractères d'interface globaux TA_1, TB_1, TC_1, et TD_1 indiquent les paramètres du protocole de communication et doivent être interprétés avant de procéder à toute transmission. Le caractère initial TS donne une séquence de bits de synchronisation permettant de définir une convention de codage des octets de toute l'information qui suit. T0 est le seul caractère obligatoire après Ts. Il comporte deux parties. Les quatre bits de poids fort, appelés **Y1**, indiquent respectivement, en position 1, la présence des caractères d'interface TA_1, TB_1, TC_1, et TD_1. Les bits de Y1 sont donc des bits de présence. Les quatres bits de poids faible, appelés **K**, indiquent le nombre de caractères d'historique T1 à TK. TA_1 donne des informations sur la fréquence et le rapport de transmission. TB_1 renseigne sur le voltage Vpp et l'ampérage Ipp. TC_1 définit le nombre d'unités de temps à attendre entre deux octets alors que TD_1 indique, sur ses quatre bits de poids faible, le protocole de communication **T** et signale sur ses quatre bits de poids fort la présence d'une autre séquence de caractères d'interface. Lorsque T=0, le protocole est un protocole de transmission de caractères asynchrone bidirectionnel. Lorsque T=1, le protocole est un protocole de transmission de paquet asynchrone bidirectionnel. Lorsque le protocole n'est pas standardisé par ISO, T=14. Les caractères d'historique ne sont pas définis par le standard ISO mais par le manufacturier de la puce. Ce sont eux qui intéressent au premier niveau le mécanisme de reconnaissance des technologies. Ils contiennent des informations sur le microprocesseur, le masque, l'application, etc. T1 est utilisé pour identifier la référence du composant. T2 contient la référence du masque. Lorsque T2=0x09, par

exemple, il s'agit alors d'une carte Bull CP8 mask9. Lorsque T2=0x20, la carte est une Bull MP. Pour une carte Gemplus MCOS, T2=0xAB. Les caractères d'historique qui suivent contiennent des informations qui diffèrent d'un constructeur à l'autre. T3 contient par exemple, pour les cartes Gemplus, la version du code du masque. On peut aussi trouver des informations sur l'application contenue dans la carte ou sur l'état de la carte par rapport à son cycle de vie. Avec toutes ces informations et leurs combinaisons, le Pilote est capable de reconnaître la technologie de la carte insérée. Une fois la technologie identifiée, le Pilote se charge de faire un aiguillage vers les fonctions propres à cette technologie pour pouvoir dialoguer adéquatement avec la carte. L'aiguillage s'effectue grâce à une matrice de fonctions. Cette matrice contient les pointeurs de toutes les fonctions nécessaires de toutes les technologies reconnues. Chaque ligne de la matrice correspond à une technologie donnée et chaque colonne correspond à une fonction de base donnée.

Technologie	Mettre la tension	Enlever la tension	Écrire	Lire	…
IBM	@fonction_MT_IBM	@fonction_ET_IBM	@fE_IBM	@fL_IBM	…
Bull MP	@fonction_MT_BullMP	@fonction_ET_BullMP	@fE_Bull-MP	@fL_Bull-MP	…
Bull mask 9	@fonction_MT_BullM9	@fonction_ET_BullM9	@fE_Bull M9	@fL_Bull M9	…
GemPlus MCOS	@fonction_MT_MCOS	@fonction_ET_MCOS	@fE_MCOS	@fL_MCOS	…
…	…	…	…	…	

Les pointeurs de fonctions sont des adresses des points d'entrée des fonctions dans le segment de code exécutable. Elles sont une des particularités du langage C que nous avons choisi pour l'implantation de notre système. L'intérêt de ces pointeurs est de pouvoir transmettre une adresse de fonction comme paramètre à une autre fonction ou de pouvoir déclarer un tableau de fonctions. C'est ce second usage que nous exploitons ici. Une matrice de fonction permet de gérer facilement les appels de fonctions en identifiant les indices des lignes aux différentes technologies et les indices des colonnes aux différentes fonctions de base. La matrice est déclarée en langage C de la façon suivante:

```
char (* CQ_MatriceFonctions[NB_TECHNOLOGIES][NB_FONCTIONS])();
```

NB_TECHNOLOGIES et NB_FONCTIONS sont des constantes indiquant respectivement le nombre maximum de technologies reconnues et le nombre maximum de fonctions de base définies. La contrainte d'une telle matrice est que toutes les fonctions de celle-ci doivent être de même type et avoir les mêmes arguments. Nous avons vu précédemment que toutes nos fonctions de base respectent cette contrainte puisqu'elles sont normalisées (**voir 5.4.2.3: Génération d'ordres pour la carte**).

Le Pilote dispose de deux variables: CQ_Technologie et CQ_Fonction_de_base.

CQ_Technologie est initialisée par l'identificateur de la technologie concernée après sa reconnaissance. CQ_Fonction_de_base contient l'identificateur de la fonction désirée. Les identificateurs sont définis par des constantes entières. L'exemple suivant montre la définition de quelques-uns de ces identificateurs avec le langage C.

```
#define TECHNO_IBM                0
#define TECHNO_BULL_MP            1
#define TECHNO_BULL_M9            2
#define TECHNO_GEMPLUS_MCOS       3
...
#define F_METTRE_TENSION          0
#define F_ENLEVER_TENSION         1
#define F_ECRIRE_ZONE             2
#define F_LIRE_ZONE               3
#define F_SOUMETTRE_CLE           4
...
```

L'initialisation de la matrice se fait de la façon suivante:

```
CQ_MatriceFonctions[TECHNO_IBM][METTRE_TENSION]=CQ_MettreTension_IBM;
CQ_MatriceFonctions[TECHNO_BULL_MP][METTRE_TENSION]=CQ_MettreTension_BMP;
...
```

L'appel à une de ces fonctions se fait alors comme suit:

```
Code_Retour =
 (* CQ_MatriceFonctions[CQ_Technologie][CQ_Fonction_de_base]
                                    (buffer, CQ_SatutOrdre, CQ_StatutCarte);
```

5.5 Intégration d'un micro-serveur

On se propose d'introduire le micro-serveur du CAPSS dans les pharmacies en même temps que le projet pilote Carte Santé. Ceci permettrait, d'après l'association du Québec des pharmaciens propriétaires (AQPP), de minimiser le matériel informatique sur le comptoir du pharmacien puisque le micro-serveur pourrait avoir un lecteur de carte intelligente déjà incorporé.

5.5.1 Présentation du mico-serveur du Centre d'automatisation et de paiement des services de santé (CAPSS)

Le micro-serveur est un «pont» entre le système autonome du pharmacien et une

centrale d'assureurs. Il permet de faire des paiements directs après autorisation. C'est un boîtier comportant un petit écran de deux lignes, un clavier de 32 touches, une unité de traitement avec un processeur de la famille 80286 avec une fréquence de l'horloge de 10 MHz, un modem, un lecteur de cartes magnétiques et un lecteur de carte à puces. Il fait véhiculer des demandes d'information émises par le système autonome du pharmacien ainsi que les réponses de la centrale. Les demandes d'informations prennent la forme de requêtes de réclamation de médicaments, d'annulation de réclamation et d'interrogation de dépôt. À chaque demande lui est associée une réponse.

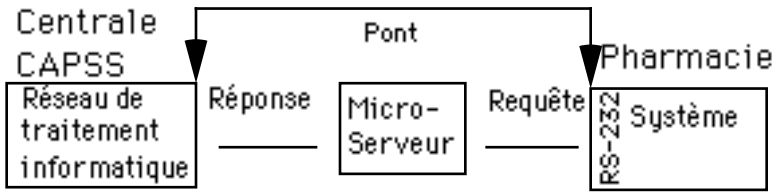
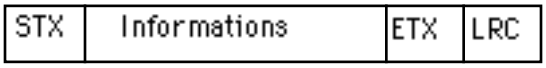


fig 5.13 : Un pont entre la pharmacie et la centrale

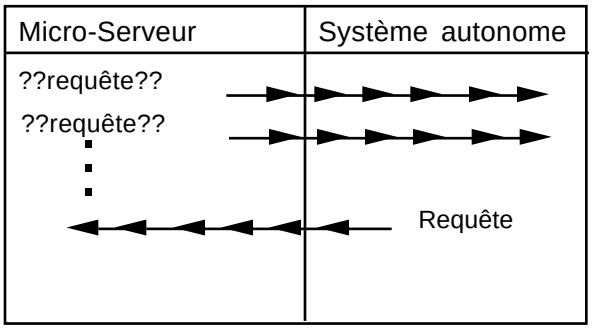
Le micro-serveur identifie le type de la requête reçue, la valide, décortique son information et la transforme dans un format reconnu par la centrale. Il communique avec le système autonome via un port série de type RS 232. Il gère les requêtes en attente et s'occupe de la sécurité de l'information du réseau. Un maximum de 30 requêtes en attente est possible au micro-serveur. Leur gestion se fait en FIFO. L'échange d'informations entre le micro-serveur et le système autonome se fait avec un protocole de communication bien établi. Tous les messages échangés entre le micro-serveur et le système autonome sont encapsulés par deux caractères **(STX)** au début du message et **(EXT)** à la fin de celui-ci. Les messages ainsi encadrés s'appellent «paquets». Afin de valider l'information, un autre caractère est calculé en fonction du message (**LRC** pour Longitudinal Redundancy checks) et rajouté à la suite du paquet.



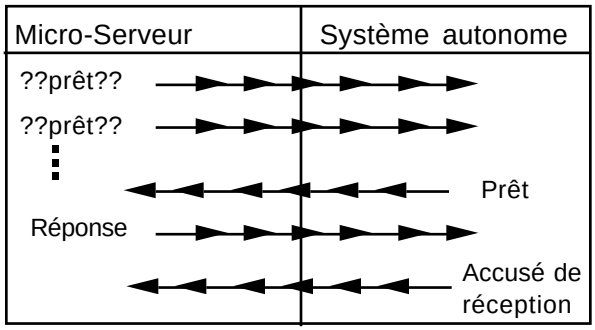
Le récepteur qui reçoit un paquet d'informations accuse réception soit par **(ACK),** si le **(LRC)** est correct, soit par **(NAK)** dans le cas contraire. La partie qui émet un paquet doit donc attendre un acquittement du récepteur. À la suite de trois échecs **(NAK)** ou si aucun acquittement n'est reçu par l'émetteur, ce dernier suspend la communication.

C'est le micro-serveur qui sollicite le système autonome de la pharmacie (pooling). Il y a deux états possible: l'**interrogation** où le micro-serveur est prêt à recevoir de l'information du système autonome et la **transmission** où le micro-serveur a de l'information à transmettre au système autonome. La transmission a priorité sur l'interrogation. Il est possible d'accumuler plusieurs requêtes en attente de traitement. De ce fait, les interrogations faites au système autonome sont quasi-continuelles.

INTERROGATION



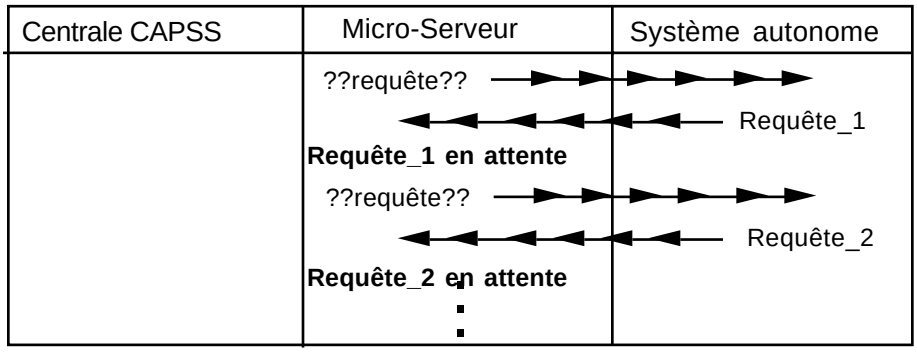
TRANSMISSION



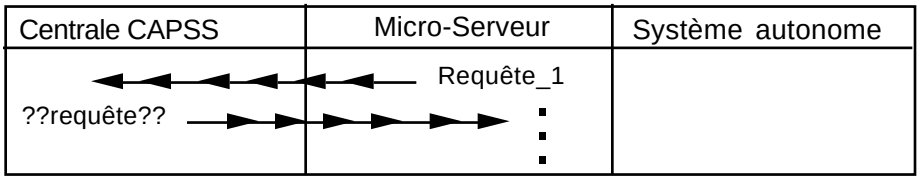
Le micro-serveur est configuré et initialisé avant le démarrage de l'application CAPSS. Ces opérations peuvent être déclenchées à distance par la centrale. Le lien entre le micro-serveur et la centrale est établi soit sur une ligne d'affaires (réseau commuté), soit sur une ligne 3101 ou 3201. Les transactions entre le micro-serveur et la centrale se font séquentiellement. Le micro-serveur attend la réponse à une requête avant d'en envoyer une autre. La première étape, lors d'une communication avec la centrale, est la composition du numéro de téléphone. Pour une bonne répartition des ressources du réseau, le micro-serveur ne conserve la ligne que pour un maximum de trois (3) transactions. C'est pourquoi le micro-serveur doit pouvoir accumuler les requêtes avant de les envoyer à la centrale.

Il y a quatre étapes de traitement: la réception d'une requête provenant du système autonome, l'envoi de la requête du micro-serveur à la centrale, la réception de la réponse de la centrale par la micro-serveur et l'envoi de la réponse du micro-serveur au système autonome.

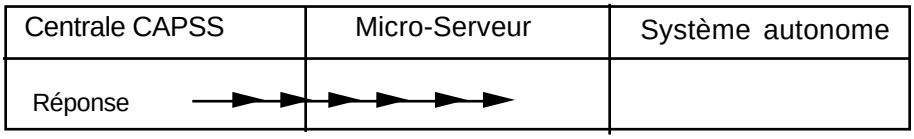
Récéption d'une requête provenant du système autonome



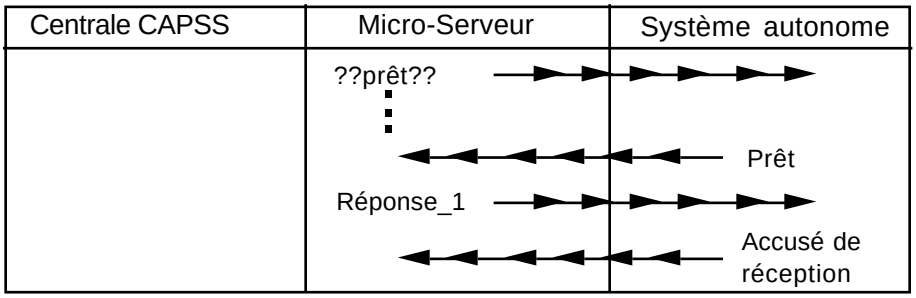
Envoi de le requête du micro-serveur à la centrale



Réception de la réponse de la centrale par la micro-serveur



Envoi de la réponse du micro-serveur au système autonome



Il y a quatre types de transactions:

Autorisation de réclamation: C'est une réclamation à la centrale pour effectuer un transfert après autorisation. Dans le cas où la centrale détecte une erreur d'authentification et si l'erreur persiste trois fois consécutives, le micro-serveur est désactivé. En cas de problème de transmission avec le système autonome ou un délai d'inactivité de ce dernier, le micro-serveur envoie un accusé de réception négatif à la centrale qui effectue alors un renversement.

Annulation de l'autorisation: C'est une requête de rectification après une autorisation de reclamation erronée (demande en écriture). En cas de problème de transmission avec le système autonome ou un délai d'inactivité de ce dernier, le micro-serveur envoie un accusé de réception négatif à la centrale qui effectue alors un renversement.

Interrogation de dépôt: C'est une demande à la centrale sur l'état des différentes requêtes faites (demandes en lecture). En cas de problème de transmission avec le système autonome ou un délai d'inactivité de ce dernier, le micro-serveur n'effectue aucun renversement avec la centrale.

Message informatif: C'est un message d'erreur contenant un texte qui vise à informer l'utilisateur du système autonome d'une anomalie ou d'une action à prendre. Ce message n'est associé à aucune requête particulière. Un renversement est un message envoyé par le micro-serveur à la centrale indiquant à cette dernière de renverser une transaction, c'est-à-dire ramener à leurs valeurs initiales les informations touchées par la transaction car celle-ci n'a pas était terminée adéquatement.

5.5.2 Proposition d'un protocole de communication

Sans le projet CAPSS, seul un lecteur de carte à puce lié à l'ordinateur contenant

l'application pharmaceutique et la coquille est nécessaire. Mais avec ce projet, le lecteur n'est plus lié physiquement à l'ordinateur mais au micro-serveur qui, lui, est connecté à l'ordinateur. Le micro-serveur est alors un simple «passthrough» pour le projet Carte Santé. Le micro-serveur est lié à l'ordinateur par le port série (RS-232) comme un lecteur de carte à puces. L'application doit pourtant continuer à adresser ses commandes au lecteur de carte à travers ce même port. C'est le micro-serveur qui identifie les ordres du Pilote destinés au lecteur de carte et les distingue des requêtes destinées à la centrale CAPSS. Pour permettre cette distinction entre les requêtes CAPSS et les ordres Carte Santé, on identifie chaque information transmise sur le port-série par une entête distincte.

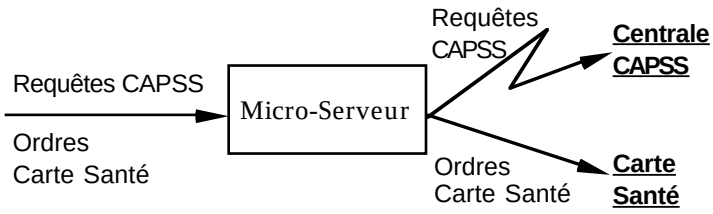


fig 5.14 : distinction des requêtes et des ordres

Les transactions CAPSS ont déjà une entête de 31 octets qui se définit comme suit:

numéro de station (NS) [5 alphanumériques]: Permet d'identifier la station d'un système autonome multi-stations.
numéro version du message (NVM) [2 alphanumériques]: Permet d'identifier la version du format de message. Pour débuter NVM est égale à "10".
numéro point de service (NPS) [10 alphanumériques]: Permet d'identifier le point de service et un compte de dépôt donné par CAPSS.
code de transaction (CT) [2 alphanumériques]: Permet d'identifier le type de la transaction.

Transactions CAPSS du système autonome vers la micro-serveur
- "10" - Autorisation de réclamation.
- "20" - Annulation d'autorisation.
- "30" - Interrogation de dépôt.

Transactions CAPSS du micro-serveur vers le système autonome
- "11" - Réclamation approuvée
- "12" - Réclamation refusée
- "13" - Réclamation refusée due à une erreur du système
- "21" - Annulation approuvée
- "22" - Annulation refusée
- "23" - Annulation refusée due à une erreur du système
- "31" - Interrogation approuvée
- "32" - Interrogation refusée
- "33" - Interrogation refusée due à une erreur du système
- "99" - Message informatif.

Identificateur de requête (IR) [12 alphanumériques]: Permet de faire la correspondance entre les réponses et les requêtes.

NS	NVM	NPS	CT	IR
5αn	2αn	10αn	2αn	12αn

Les transactions Carte Santé ne changent pas l'en-tête et gardent ce même format

fixe. Les champs NS, NVM, et NPS ne sont pas utilisés par le Pilote et sont toujours mis à zéro. CT permet d'identifier le type des ordres à la carte:

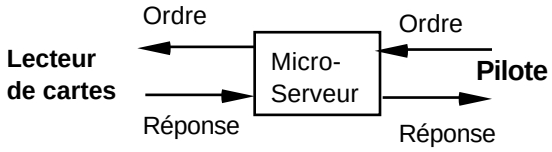
"50" - Ordre de lecture.
"51" - Ordre d'écriture.
"52" - Ordre de traitement.

Ainsi que les réponses du micro-serveur en cas d'erreur:

"59" - Ordre incompris

Enfin IR, comme ci-haut, permet de faire la correspondance entre les réponses de la carte et les ordres transmis, même si ces derniers doivent s'attendre mutuellement puisque le micro-serveur ne mémorise pas les ordres à la carte. C'est donc grâce au code de transaction (CT) que le micro-serveur peut distinguer entre les requêtes CAPSS à laisser en attente puis à transmettre à la centrale CAPSS et les ordres du Pilote à transmettre sans attente au lecteur de cartes à puce.

Le micro-serveur fonctionne en «pooling». C'est lui qui sollicite le système autonome avant chaque transmission ou réception. Le Pilote doit donc attendre la sollicitation du micro-serveur pour transmettre un ordre à la carte ou s'attendre à recevoir une réponse de celle-ci. Le micro-serveur reçoit plusieurs requêtes du système autonome avant de les transmettre une à une à la centrale CAPSS. Il peut ainsi en mémoriser jusqu'à 30. Pour les ordres du Pilote, par contre, il doit les transmettre dès leur réception, au lecteur de cartes sans les mémoriser.



Le micro-serveur a huit (8) états de traitement. Concernant le projet CAPSS, il y en a quatre (4); la réception d'une requête du système autonome (mise en attente de la requête), l'envoi d'une requête à la centrale CAPSS, la réception d'une réponse de la centrale CAPSS et l'envoi d'une réponse au système autonome. Concernant la Carte Santé, il y en a aussi quatre (4): la réception d'un ordre pour la carte (traitement sans mise en attente), la transmission d'un ordre au lecteur, la réception d'une réponse de la carte et la transmission d'une réponse de la carte.

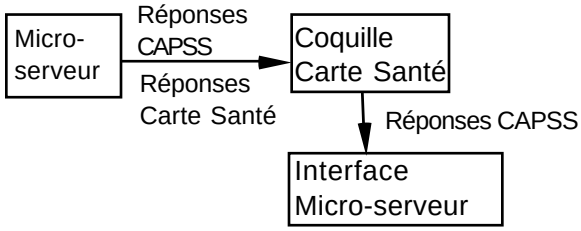


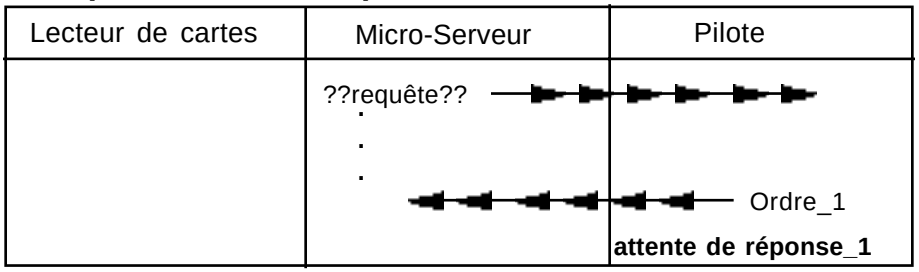
fig 5.15 : distinction des réponses CAPSS et Carte Santé

En recevant une requête ou un ordre, le micro-serveur peut les distinguer grâce à l'entête du message reçu. De son côté, le système autonome, qui contient à la fois la coquille Carte Santé et l'interface micro-serveur, reçoit des réponses du micro-serveur et doit faire la distinction entre les réponses de la carte à microprocesseur destinées à la coquille et les réponses de la centrale CAPSS destinées à l'interface micro-serveur. Pour ce faire, le Pilote se charge de recevoir les messages du Micro-serveur et de faire véhiculer ceux du CAPSS à l'interface micro-serveur. Le Pilote intercepte donc tous les messages entrant sur le port RS-232. Il traite les messages le concernant (ayant un CT commençant par '5') et laisse passer les autres

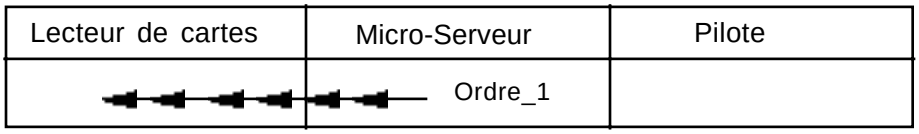
messages.

Il y a deux états possibles vis-à-vis de l'interface micro-serveur ou du Pilote: l'**interrogation** où le micro-serveur est prêt à recevoir de l'information de l'interface micro-serveur ou du Pilote et la **transmission** où le micro-serveur a de l'information à transmettre à l'interface micro-serveur ou au Pilote. Chaque état est subdivisé en deux, l'une pour CAPSS et l'autre pour le Pilote de Carte Santé. La transmission a priorité sur l'interrogation. De même, la transaction Carte Santé a priorité sur la transaction CAPSS car la première n'est pas mémorisée et nécessite une réponse rapide.

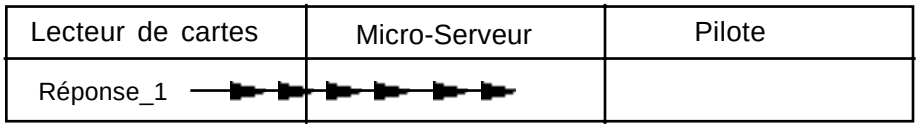
Récéption d'un ordre pour la carte



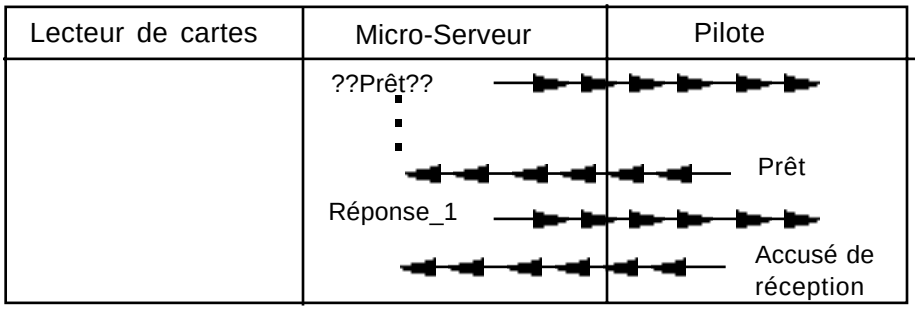
Transmission d'un ordre au lecteur



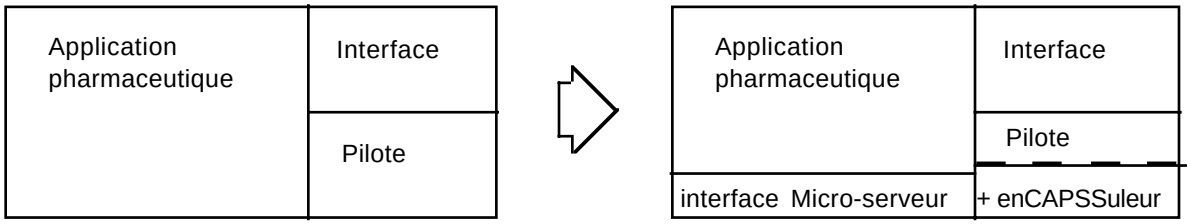
Réception d'un e réponse de la carte



Transmission d'un e réponse de la carte



L'adjonction du micro-serveur au système Carte Santé engendre de nouvelles spécifications pour le Pilote. L'analyse conceptuelle de ce processus est donnée à l'annexe C. Pour permettre la communication entre la coquille et le micro-serveur, une autre couche logicielle doit être rajoutée. Appelons cette nouvelle couche EnCAPSSuleur.



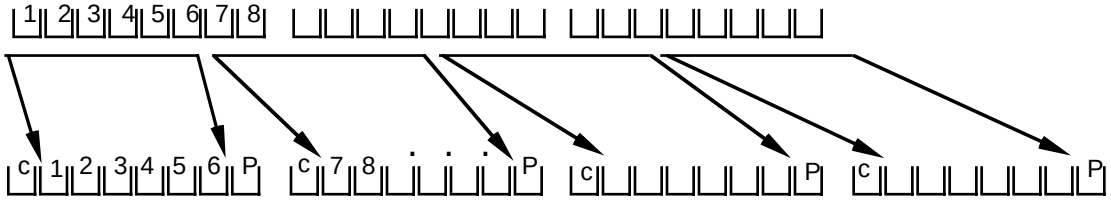
Logiciels SANS Micro-Serveur

Logiciels AVEC Micro_Serveur

L'EnCAPSSuleur est un module qui fait la communication avec le micro-serveur. Ses tâches sont les suivantes:

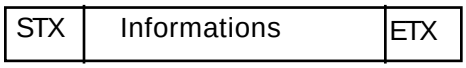
À l'émission:

1- Restructure l'information par ensemble de six (6) bits avec deux (2) bits de contrôle. Les octets du message sont traités par le micro-serveur par ensemble de six bits. Ceci est fait pour éviter d'avoir des caractères de contrôle (inférieurs à 32) dans le message. Ainsi le premier bit d'un octet est un bit de contrôle toujours à 1. Ce dernier est suivi des six bits qui à leur tour sont suivi d'un bit de parité géré par le port série. L'EnCAPSSuleur doit donc diviser le message à transmettre en ensemble de six bits, ajouter un bit de contrôle à 1 et réserver un bit pour la parité.

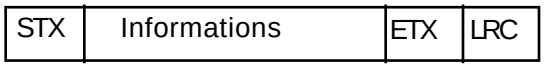


2- Rajoute un en-tête d'identification à l'information. L'EnCAPSSuleur rajoute un en-tête au message à transmettre pour permettre au micro-serveur de l'identifier comme étant à Carte Santé et de l'acheminer vers le lecteur de carte. L'en-tête est de 31 caractères. Les 17 premiers sont toujours à zéro et le 18 ème égal à '5' (voir plus haut le code de transaction CT).

3- Encapsule cette information par un STX au début et un ETX à la fin. Les messages sont encapsulés par deux caractères indiquant le début du texte (STX) et la fin du texte (ETX).



4- Calcule le LRC du paquet d'informations. Avant d'envoyer le message sur le port-série, l'EnCAPSSuleur calcule le LRC (Longitidinal Redundancy Cheks) et le rajoute au message pour permettre au micro-serveur de valider le message à la réception.



5- Envoie l'information sur le port série. L'EnCAPSSuleur vérifie la disponibilité du micro-serveur en attendant la sollicitation de ce dernier avant d'envoyer le message

sur le port-série. Le pooling exige une attente d'un message de sollicitation avant la transmission d'un ordre sur la ligne.

6- Retransmet l'information dans le cas d'une mauvaise réception de la part du micro-serveur. Dans le cas où le micro-serveur répond par un message ayant une entête avec CT='59' pour des raisons d'incompréhension de l'ordre transmis (mauvais LRC, bit de parité incorrecte...), l'EnCAPSSuleur retransmet le message.

À la réception:
1- Calcule le LRC des réponses transmises par le micro-serveur. À la réception d'un message, l'EnCAPSSuleur calcule son LRC et le vérifie avec le LRC reçu.

2- Envoie un ACK ou un NAK après vérification du LRC. Dans le cas où la vérification des deux LRC est positive l'EnCAPSSuleur émet un acquittement positif (ACK) et un acquittement négatif (NAK) dans le cas contraire.

3- Désencapsule le paquet d'informations. L'EnCAPSSuleur enlève le caractère de début et de fin de texte (STX et ETX).

4- Vérifie l'en-tête d'identification. L'en-tête d'identification est vérifiée pour identifier les messages destinés à la Coquille de Carte Santé. Les messages CAPSS sont alors acheminés directement à l'interface micro-serveur.

5- Enlève l'en-tête d'identification. Les messages identifiés comme étant destinés à Carte Santé sont alors dépourvus des en-têtes d'identification.

6- Restructure l'information par octets. À la réception d'un message, l'EnCAPSSuleur doit restructurer l'information par ensemble de huit bits en éliminant le bit de contrôle et celui de parité après les avoir vérifiés.

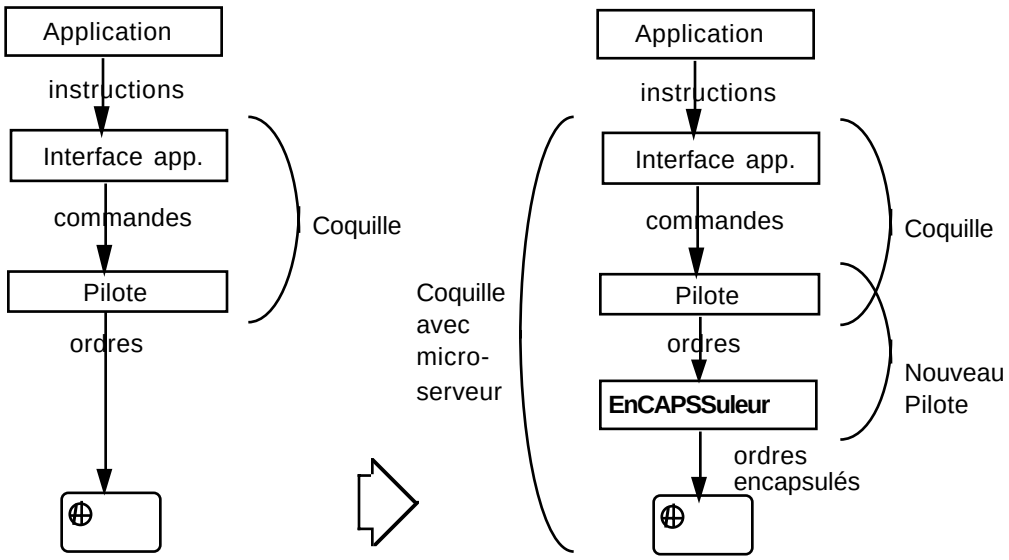


fig 5.16 : Ajout de l'EnCAPSSuleur dans la structure de la Coquille

5.6 Les statistiques pour l'évaluation du projet

Le projet-pilote Carte Santé à Rimouski doit fournir l'information nécessaire pour permettre d'étudier et d'évaluer la faisabilité d'une généralisation de l'utilisation d'une Carte Santé dans toute la province du Québec. C'est pourquoi il est très important de colliger quelques statistiques au cours du projet pilote. Ces informations doivent absolument être anonymes interdisant les liens avec les professionnels de la santé et les bénéficiaires des cartes. Chaque transaction reuceillie à des fins de statistiques est identifiée par un numéro qui n'a aucun lien avec le porteur de la carte mais plutôt avec la carte elle même pour ne pas permettre de retracer des propriétaires des données médicales. De cette façon, la seule identité emmagasinée est fictive et ne permet que de faire une compilation des données. Les données sont relevées et stockées par la Coquille d'une façon entièrement transparente et automatique. Elles sont emmagasinées sur le disque dur de chaque appareil et sont périodiquement compilées et transmises à un site central pour être analysées et interprétées. Le comité d'analyse (monitoring) utilisera ces données au fur et à mesure pour étudier la rentabilité du projet. Ces données serviront aussi à faire une étude plus réaliste sur la saturation des cartes à mémoire (voir chapitre III). La collecte automatique des données est faite à trois niveaux. Le Pilote, qui manipule l'information au niveau de la carte, se charge des données physique directement liées au DMP. Il calcule par exemple le nombre de transactions d'écriture par consultation, le nombre de zones touchées par chaque transaction d'écriture et le nombre de bits écrits par chaque transaction dans chaque zone touchée. Ceci permettrait d'étudier le cycle de vie de la carte avant sa saturation définitive. L'Interface Applicative capture quant à elle des informations sur l'utilisation des zones logiques et l'activité des acteurs. Ceci permettrait d'évaluer la pertinence de certaines zones logiques du DMP et d'étudier l'importance de certaines données vis-à-vis des différents type d'acteurs dans le système à travers les fréquences d'accès aux informations. L'application médicale MÉDICARTE, qui a été spécialement développée pour le projet, capture aussi des informations pour des analyses ultérieures. Ainsi, des informations sur l'utilisation de certaines fenêtres et certaines fonctions sont retenues pour pouvoir évaluer le produit et porter des modifications d'amélioration en tenant compte de l'utilisation de l'application par l'utilisateur final.

Une zone pour les statistiques est prévue dans la carte d'habilitation et la carte de bénéficiaire pour pouvoir y inscrire certaines données de gestion pour la capture d'informations.

5.7 Perspectives pour le Pilote

Dans le projet Carte Santé, le microprocesseur de la carte est surtout utilisé pour la gestion de la sécurité. La Carte Santé est par conséquent dépendante de son environnement. Pour limiter cette dépendance, on peut envisager de faire effectuer d'autres fonctions à la carte en utilisant les ressources de son calculateur. Ceci exige un masque spécifique qui engloberait de nouvelles fonctionnalités. Une migration de certaines fonctions du Pilote peut alors se faire, comme les fonctions de codification par exemple. Toutefois la vitesse d'exécution du microcalculateur actuel met des en-

traves à ce transfert. Dans la même optique, on peut envisager la migration d'une partie du Pilote vers le lecteur de cartes pour en faire un lecteur intelligent. La partie visée du Pilote concerne la codification ainsi que la compression et décompression des données. Avec des fonctions câblées dans un lecteur intelligent, l'exécution des transferts d'informations entre une carte et une application peut être sensiblement accélérée.

* * *

CONCLUSION

Le problème majeur auquel se heurte le dossier médical portable est le problème de la saturation. La saturation des zones de la carte est un problème inévitable vu le caractère cumulatif des données du dossier médical portable. La taille actuelle des mémoires de cartes intelligentes est de l'ordre de 64 Kbits, ce qui est nettement insuffisant. Une augmentation de la taille de la mémoire disponible sur une carte est une condition *sine qua non* pour alléger la saturation de la mémoire et permettre la manipulation sur une carte intelligente d'un dossier médical plus complet ayant un cycle de vie plus long. Les pronostics quant à l'évolution des tailles de mémoires EEPROM ne convergent pas tous mais on ne prévoit pas une taille importante de plus de 128 Kbits sur le marché avant 18 mois. Toutefois, de nouvelles technologies à grande capacité de stockage s'offrent comme solution. C'est le cas des cartes optiques à laser. Sur ces dernières, on peut même envisager d'emmagasiner des électrocardiogrammes ou des radiographies, ce qui n'est techniquement pas possible actuellement avec les cartes intelligentes. Leur capacité importante permet de gérer un dossier plus complet et complexe. Cependant, ces cartes n'offrent actuellement aucune sécurité intégrée quant à l'accès aux données qu'elles emmagasinent. Les recherches commencent à étudier la possibilité de jumeler de telles cartes avec un microprocesseur ou de leur associer un mécanisme de sécurité. Une étude est aussi en cours au Laboratoire d'Informatique de GEstion (LIGE) de l'Université Laval pour sécuriser une carte optique par l'adjonction d'une couche de logiciel chargée du chiffrement des informations.

La plupart des cartes intelligentes qui existent actuellement sur le marché ont été conçues et réalisées pour le domaine bancaire. Leur masque est bien spécifique et toutes ses fonctionnalités sont directement rattachées à une très haute sécurité exigée par les applications financières. Il faut se demander quelle carte parmi celles qui existent sur le marché est la plus rentable dans un domaine médical. Le choix n'est pas facile d'autant plus qu'une adaptation s'impose. Il nous paraît important de concevoir une carte spécialement pour le dossier médical portable au lieu d'adapter une carte bancaire au domaine de la santé. Un masque spécifique pour une carte

santé ne comporterait que les strictes fonctionnalités nécessaires pour la gestion sécuritaire et adéquate d'informations médicales partagées. Il pourrait aussi inclure une fonction d'authentification mutuelle entre la carte santé et la carte d'habilitation. Étant toujours en évolution, le marché de la santé justifierait un tel choix.

Une étude des impacts de l'utilisation des cartes à microprocesseur sur les organisations et sur les méthodologies de développement des systèmes qui en font usage est actuellement réalisée au département d'informatique de l'université Laval. À prime abord, il en ressort que les applications qui utilisent des cartes à mémoires ont une conception totalement différente des projets informatiques classiques. En effet, la mémorisation des données est répartie sur plusieurs sites et le transfert des données se fait par un média sécuritaire, autonome et très particulier: la carte intelligente. La carte apporte une grande diversité et une importante richesse aux nouvelles applications informatiques. Il est évident que l'évolution constante des cartes et la diminution inévitable de leur coût permettra la construction et la prolifération d'applications conviviales. La mise en place d'une telle application profitera des capacités d'une carte intelligente pour améliorer d'une façon significative les prestations de soins de santé au Québec. En ce sens, cette mise en place contribuera de façon évidente à l'évolution de la qualité de vie de demain.

* * *

BIBLIOGRAPHIE
ET
RÉFÉRENCES BIBLIOGRAPHIQUES

ABRAHAMSON 89 ABRAHAMSON D.M., **An adaptive dependency source model for data compression**, Communication of ACM, v 32 n 1, january 1989, pp. 77-83.

ALSBERG 75 ALSBERG, P. **Space and time savings through large data base compression and dynamic restructuring**, Proc. IEEE, v 63, n 8, 1975, pp.1114-1122.

ANON 88 Anonyme, **Smart card storage technologies**, Electronic library, v 6 n 6, december 1988, pp.432-438.

AXIS 89 **Les cartes à mémoires (les clés indispensables pour les systèmes de communication)**. Agence Axis, 1989, Eco-Media, Edition Milan-Midia. ISBN 2-86726-412-10.

BEAUD 85 BEAUD Michel, **L'art de la thèse**, Édition la Découverte, Paris 1985.

BELL 90 BELL Timothy, CLEARY John et WITTEN Ian, **Text Compression**, Prentice-Hall, 1990, ISBN 0-13-911991-4.

BRICH 88 BRICH R., **Into a 3rd Genaration- The 'Active' Card**, SMART-CARD'88: International Conference and Workshop on Smart Card Applications and Technologies, 1988, p. 6 pp. 3 vol.

BRIGHT 88 Bright Roy, **Smart Cards: Principles, Practice, Applications**, Elis Hard, England, 1988, 173p. ISBN 0-7458-0374-1

BULL 87 **Micro card single-service mask family: technical overview**, Micro card technologies, Inc. 1987.

BULL 89 **L'environnement MP proposé par Bull CP8**, Bull CP8, septembre 1989.

CANTIN 90 CANTIN Réjean, Conférence au séminaire international sur les applications de la carte à mémoire, Stéria Canada, Montréal, mai 1990.

CANTIN 91 CANTIN Réjean, **Projet Carte-Santé du Québec**, Conférence à l'Expo-Congrès International des Technologies d'Information, Montréal, septembre 1991.

CHABANE 90 CHABANE Rezzik, **Compression des données**, Mémoire de maîtrise, Département d'informatique, Université Laval, janvier 1990.

DESOKY 88 DESOKY A. et GREGORY M., **Compression of text and binary files using adaptive Huffman coding techniques**, Conference Proceedings, IEEE Southeastcon, Knoxville, TN, april 11-13 1988, pp.660-663.

GALLAGER 78 GALLAGER, R. G. **Variation on a theme by Huffman**, IEEE Trans. Infor. Th. v 24, n 6, 1978, pp.668-674.

GAMACHE 90 GAMACHE A., ARDOUIN P. et al, **Caractéristiques systémiques et techniques d'une carte santé réalisée avec une carte intelligente**, Colloque sur l'integration de la technologie des cartes intelligentes dans le développement des systèmes, Université Laval, Québec, 13 et 14 septembre 1990.

GAMACHE 91-a GAMACHE A. et ARDOUIN P., **A Formal Model for Analysis of Memory Allocation and Saturation in the Design of Smart Card Based System**, Proceedings of SCAT-91, Washington, may 1991, pp.134-145.

GAMACHE 91-b GAMACHE A., ARDOUIN P. & al, **Effects of memory saturation in the design of a portable medical record**, Conference proceedings, Third global conference on patient cards, Barcelona, march 12-15 1991, pp.132-136.

GCPC 91 Actes de la troisième conférence mondiale sur les «Cartes-Patients»,, Barcelone, Espagne, 12-15 mars 1991.

GUEZ 88 GUEZ F., ROBERT C. et LAURET A., **Les cartes à microcircuit; techniques et applications**, Masson, 1988, ISBn 2-225-81346-9

GUILLOU 88 GUILLOU L., DAVIO M. et QUISQUATER JM., **L'état de l'art en matière de techniques à clé publique: hasard et redondance**, Annales des télécommunications, v 43 n 9-10, septembre-octobre 1988, pp.489-505.

HAMMING 80 HAMMING R.W., **Coding an Information Theory**, Prentice-Hall, 1980.

HOPKINS 91 HOPKINS Robin. **The Exeter care card**, Conference proceedings, Third global conference on patient cards, Barcelona, march 12-15 1991, pp.327-332.

HUFFMAN 52 HUFFMAN, D. A. **A method for the construction of minimum redundancy codes**, Proceedings of the IRE 40, 1952, pp.1098-1101.

IBM 90 **Transaction security system: programming guide and reference**, IBM corporation, may 1990.

ISO 89 ISO / IEC 7816-3, International Standard, Identification cards - Integrated circuit(s) cards with contacts (Electronic signals and transmission protocols), 1989.

KNUTH 81 KNUTH Donald E. **The art of computer programming**, v 1-2-3, Addison-wesley, 1981.

KNUTH 85 KNUTH, D. E. **Dynamic Huffman coding**, Journal of Algorithms v6, 1985, pp.163-180.

LAW 91 LAW Averill M. and KELTON W. David, **Simulation modeling and analysis**, second edition, McGraw-Hill, 1991, ISBN 0-07-036698-5

MARTIN 88 MARTIN S. L., **Smart card development expands as standard near final approval**, Computer design, v 27 n 16, septembre 1988, pp.30-31.

MONOD 91 MONOD Elsbeth. **Smart cards in the French social and health sector**, Conference Proceedings, Third global conference on patient cards, Barcelona, march 12-15 1991, pp. 22-30.

MOULIN 88 MOULIN Bernard, **La méthode E.P.A.S. pour la modélisation et la conception de systèmes**, support de cours, Département d'informatique, Université Laval, Québec, 1988.

NELSON 91 NELSON, M.R. **Arithmetic coding and statistical modeling**, Dr Dobbs journal, february 1991.

OMURA 89 OMURA jim K., **Smart card to create electronic signatures**, IEEE International Conference on Communications- ICC'89, Boston, 1989, pp. 1160-1164.

PARADINAS 88 PARADINAS Pierre, **La Biocarte, intégration d'une carte à microprocesseur dans un réseau professionnel santé**. Thèse de doctorat en informatique présentée à l'Université de Lille (France), novembre 1988.

PIKE 81 PIKE, J. **Text compression using a 4 bit coding scheme**, The Computer Journal, v 24, n 4, 1981, pp.324-330.

RAMABADRAN 89 RAMABADRAN T.V., et COHN, D. L., **An adaptive algorithm for the compression of computer data**, IEEE Transactions on Communications v 37, n 4 april 1989, pp.317-324.

REGHBATI 81 REGHBATI, H., K. **An overview of data compression techniques**, Computer april 1981, p.71-75.

RODEH 81 RODEH, M., PRATT, V.R. et EVEN, S. **Linear Algorithm for data compression via string matching**. Jounal of the ACM, v 28, n 1, 1981, pp.16-24.

RUBIN 76 RUBIN, F. **Experiments in text file compression**, Commun. ACM v 19, n 11, 1976, pp.617-623.

RUCKWOOD 88 RUCKWOOD R., **The Active Smartcard- The future travelers' companion**, SMARTCARD'88: International Conference and Workshop on Smart Card Applications and Technologies, 1988, p. 12 pp. 3 vol.

RUTH 72 RUTH, S. S., et KREUTZER, P. J. **Data compression for large business files**, Datamation, v 18, n 9, 1972, pp.62-66.

SCAT 91 Conference Proceedings, Smart card systems, Applications et Technol-

ogies, Advanced Security & Identification Technology, Washington DC, may 28-31 1991, Edited by Paul Oyer and Magie DeHoust.

SEATON 91 SEATON Brendan. **Analysis of patient card technologies**, Conference proceedings, Third global conference on patient cards, Barcelona, march 12-15 1991, pp.7-10.

SECURTECH 91 Conference Proceedings, The innovative security technology, Crystal city, Va, USA, april 15-17 1991.

SHANNON 48 SHANNON , C.E., **A Mathematical theory of communication**, Bell Syst. Tech. J. v 27, 1948, pp.379-423, 623-656.

STONER 79 STONER J.A., **Data compression: Methods and complexity issues**, Ph.D Thesis, Departement of electrical engeneering and computer science, Princeton University, Princeton, 1979.

TOSHIBA 90 **Toshiba IC card developer's kit: technical reference manual**, Toshiba corporation, 1990.

URROWS 89 URROWS Henry et URROWS Elizabeth, **Future of transactional card technologies**, Optical Information Systems, v 9, n 4, jully-august 1989, pp. 190-208.

VITERBI 79 VITERBI, J. A, et OMURA, J,K., **Principles of Digital Communications and Coding**, McGraw HillJ, 1979.

WATERBURY 89 WATERBURY R.C., **Smart card speed transactions**, InTech technologies, v 36, n 11, november 1989, p.80.

WELCH 84 WELCH, T. A. **A technique for high-performance data compression**. IEEE Computer, v 17, n 6, 1984, pp.8-19.

WITTEN 87 WITTEN, I. H., NEAL, R. M., et CLEARY, J. G., **Arithmetic coding for data compression,** Commun. ACM, v 30, n 6, 1987, pp.520-540.

ZAIANE 91 ZAIANE R., GAMACHE A., ARDOUIN P. et DURANT, P., **Design constraints and implementation of a PMR smart card based software**, Conference proceedings of ACT, Toronto, november 1991.

ZIV 72 ZIV,J., **Coding of Sources with unknown statistics-Part I: Probability of encoding error, Part II: Distortion relative to a fidelity criterion**, IEEE Trans. Inf. Th. v 18, n 3, 1972, pp.384, 394.

ZIV 77 ZIV, J., et LEMPEL A. **A universal algorithm for sequential data compression**, IEEE Trans. Inf. Th. v 23, n 3, 1977, pp.337-343.

ZIV 78 ZIV, J., et LEMPEL A. **Compression of individual sequences via variable-rate coding**, IEEE Trans. Inf. Th. v 24, n 5, 1978, pp.530-536.

ANNEXE A

Les zones physiques et leur contenu

Dans ce qui suit, nous présentons les 15 zones physiques du DMP avec leur contenu respectif divisé par groupe de données. Les données d'un même groupe se trouvent toujours ensemble sur la carte et sont toujours inscrites par la même transaction. À chaque groupe lui correspond un type d'enregistrement dans la zone physique. Un groupe de données est inscrit dans la carte avec une date et une signature indiquant la date de l'inscription et l'identification de l'acteur ayant inscrit ces informations. Pour des soucis de clarté et de simplification, nous ne représentons pas la date et la signature avec le contenu des zones. La signature est en fait un indice d'une signature dans la table contenue dans la zone S2 (zone des signatures des prestataires).

Nous avons ajouté la zone logique d'origine de chaque champ à la suite de celui-ci pour permettre au lecteur de faire un lien avec les données des zones logiques. Nous présentons aussi, dans des tableaux, le profil d'accès des acteurs pour chaque zone physique. Dans ces tableaux, où on ne retrouve que les acteurs «professionnels de la santé», la lettre 'L' indique un droit de lecture alors que la lettre 'E' indique un droit d'écriture.

Identification (zone physique 1):

Médecin	Pharmacien	Ambulancier	Infirmière	Infirmière en chef
LE	LE	L	LE	LE

Cette zone comprend huit groupes d'informations distincts. Les sept premiers groupes sont composés par des champs spécifiques. Le huitième est composé de tous les champs de l'identification à la fois. Il permet de coder en une seule fois la première inscription dans cette zone qui comporte généralement toutes les données à la fois. Les sept autres groupes permettent les modifications ou les ajouts.

groupe 1 Nom (L1)

groupe 2 Prénom (L1)

groupe 3 sexe (L1)
Date de naissance (L1)

groupe 4 Numéro d'assurance maladie (L1)
Expiration de la carte d'assurance maladie (L1)

groupe 5 Numéro du dossier hospitalier (L1)

groupe 6 Personne à prévenir (L1)
groupe 7 Numéro de téléphone (L1)
groupe 8 Ensemble de toutes les données des autres groupes

Urgence (zone physique 2):

Médecin	Pharmacien	Ambulancier	Infirmière	Infirmière en chef
LE	L	L	L	L

Cette zone comprend trois groupes d'informations distincts. Le groupe sanguin est codé avec le rhésus. Il y a 16 pathologies et prothèses possibles. Elles sont introduites par ensemble, ce qui nécessite une codification de suites. Le nombre maximum de pathologies introduites à la fois est 16. Les informations d'urgence sont un texte d'un maximum de 64 caractères.

groupe 1 Groupe Sanguin (L2)

groupe 2 Pathologies et prothèses (L3)

groupe 3 Autres informations d'urgence (L4)

Médication (zone physique 3):

Médecin	Pharmacien	Ambulancier	Infirmière	Infirmière en chef
LE	LE			L

Cette zone ne comprend qu'un seul groupe d'informations contenant les données pour une médication. Cependant, les médicaments, qui comprennent donc plusieurs variables, sont introduits par ensemble d'un maximum de 16 médications.

groupe 1 DIN (L5)
Quantité délivrée (L5)
PRN (Pro-Ré-Nata) (L5)
Durée du traitement (L5)
Nombre de renouvellement (L5)

Intolérances médicamenteuses et allergies (zone physique 4):

Médecin	Pharmacien	Ambulancier	Infirmière	Infirmière en chef
LE	LE	L	L	L

Cette zone comprend trois groupes d'informations distincts; les intolérances et allergies aux médicaments ainsi que les autres allergies. Les intolérances et les allergies médicamenteuses sont des DINs introduits par ensemble d'un maximum de 16 éléments à la fois, ce qui nécessite une codification de suites. Les autres allergies sont en texte libre d'un maximum de 64 caractères.

groupe 1 Liste des codes d'intolérances médicamenteuses (L6)

groupe 2 Liste des codes d'allergies médicamenteuses (L7)
groupe 3 Autres allergies (L8)

Vaccination (zone physique 5):

Médecin	Pharmacien	Ambulancier	Infirmière	Infirmière en chef
LE	L	L	LE	LE

Cette zone ne comprend qu'un seul groupe d'informations. Ce groupe contient uniquement des codes de vaccins, qui sont introduits par ensemble de maximum 16 éléments. Il y a 128 codes de vaccins possibles. Chaque vaccin est accompagné d'une

date de vaccination.

groupe 1 Liste des vaccins avec leur date (L9)

Suivi du poids (zone physique 6):

Médecin	Pharmacien	Ambulancier	Infirmière	Infirmière en chef
LE	L		LE	LE

Cette zone ne comprend qu'un seul groupe d'informations contenant une seule donnée: le poids, qui est introduit individuellement.

groupe 1 Poids (L10)

Suivi pédiatrique (zone physique 7):

Médecin	Pharmacien	Ambulancier	Infirmière	Infirmière en chef
LE			LE	LE

Cette zone comprend quatre groupes d'informations: le premier contient la taille, le second le périmètre crânien et le troisième contient les deux données ensembles, à savoir la taille et le périmètre crânien. Le troisième ensemble permet d'éviter la répétition de la date et la signature lorsque les deux données sont inscrites en même temps. Le quatrième groupe contient un ensemble d'informations pédiatriques toujours introduites regroupées.

groupe 1 Taille (L24)

groupe 2 Périmètre crânien (L24)

groupe 3 Taille (L24)
Périmètre crânien (L24)

groupe 4 Gestation (L25)
Complications (L25)
Travail spontané ou déclenché (L25)
Durée totale du travail (L25)
Type d'accouchement (L25)
Apgar 1 minute (L25)
Apgar 5 minutes (L25)
Apgar 10 minutes (L25)
Réanimation (L25)
Alimentation (L25)

Suivi général et prévention (zone physique 8):

Médecin	Pharmacien	Ambulancier	Infirmière	Infirmière en chef
LE				

Cette zone comprend quatre groupes d'informations: les diagnostics, les examens, l'occupation professionnelle et des données concernant les pressions artérielles et la fréquence cardiaque. Les diagnostics sont introduits par groupe de maximum 16. Chaque diagnostic comporte un groupe de conduite contenant un maximum de 16 éléments. Les examens sont aussi introduits par groupe de 16 au maximum.

groupe 1 Liste des diagnostics et conduites (L15)

groupe 2 Liste des examens de prévention (L16)

groupe 3 Occupation (L11)

groupe 4 Fréquence cardiaque (L23)
Pression artérielle systolique (L23)
Pression artérielle diastolique (L23)

Autres suivis (zone physique 9):

Médecin	Pharmacien	Ambulancier	Infirmière	Infirmière en chef
LE				

Cette zone comprend quatre groupes d'informations. Chaque groupe contient un certains nombre de variables toujours introduites ensemble.

groupe 1 Cholestérol total (L18)
HDL-cholestérol (L18)
LDL-cholestérol (L18)
Triglycéride (L18)
date de mesure (L18)

groupe 2 Fructosamine (L19)
Glycémie à jeûn (L19)
Glycémie 2 heures p.c. (L19)
Hémoglobine glycosylée (L19)
Protéinurie (24 heures) (L19)
Date de mesure (L19)

groupe 3 Calcium (L21)
HCO3 (L21)
Hématocrite (L21)
Hémoglobine (L21)
PH (L21)
Phosphore (L21)
Potassium (L21)
Prétéinurie (L21)
Sodium (L21)
Date de mesure (L21)

groupe 4 Débit de pointe (L22)
Indice de saturation (L22)
PCO2 (artérielle) (L22)
PO2 (artérielle) (L22)
Date de mesure (L22)

Variables de suivi (zone physique 10):

Médecin	Pharmacien	Ambulancier	Infirmière	Infirmière en chef
LE				

Cette zone comprend des variables de différents types. Elle permet de faire le suivi de plusieurs variables de la zone physique 9 en choisissant soi-même leurs combi-

naisons ou d'autres variables qui seront définies ultérieurement et ayant un type de données quelconque. Ces variables peuvent même être rajoutées au cours du projet. Les variables de cette zone sont regroupées par éléments de même type (voir 2.5). Pour chaque type de données, on peut représenter 32 variables différentes.

groupe 1 Liste de variables

Antécédents et créatinine (zone physique 11):

Médecin	Pharmacien	Ambulancier	Infirmière	Infirmière en chef
LE				

Cette zone comprend quatre groupes d'informations distincts. Les antécédents familiaux en code et en texte libre, les antécédents personnels et des informations servant pour différents suivis. Les antécédents familiaux sont introduits par ensemble d'un maximum de 16 codes. Les antécédents personnels sont aussi introduits par ensemble de 16 codes au maximum. Les codes peuvent provenir de plusieurs tables différentes.

groupe 1 Liste des antécédents familiaux (L12)

groupe 2 Autres antécédents familiaux (L13)

groupe 3 Table (L14)
Code (L14)
Année (L14)

groupe 4 BUN (urée) (L20)
Créatinine (L20)
Date de mesure (L20)

Ophtalmologie (zone physique 12):

Médecin	Pharmacien	Ambulancier	Infirmière	Infirmière en chef
LE				

Cette zone comprend quatre groupes d'informations, tous regroupant des données ophtalmologiques. Ces données ont été séparées de cette façon en se basant sur la manière de saisie de celles-ci par l'application. Les données de chaque groupe sont normalement inscrites en même temps.

groupe 1 Acuité visuelle: 2 yeux sans lunette (L17)
Acuité visuelle: 2 yeux avec lunette (L17)
Acuité visuelle: l'oeil droit sans lunette (L17)
Acuité visuelle: l'oeil droit avec lunette (L17)
Acuité visuelle: l'oeil gauche sans lunette (L17)
Acuité visuelle: l'oeil gauche avec lunette (L17)
Date de mesure de l'acuité (L17)

groupe 2 Papille droite (L17)
Papille gauche (L17)
Date de mesure de la papille (L17)

groupe 3 Réfraction droite-sphère (L17)

Réfraction gauche-sphère (L17)
Réfraction droite-force du cylindre (L17)
Réfraction gauche-force du cylindre (L17)
Réfraction droite-axe (L17)
Réfraction gauche-axe (L17)
Date de mesure de la réfraction (L17)

groupe 4 Tension occulaire droite (L17)
Tension occulaire gauche (L17)
Date de mesure de la tension (L17)

Résumés des grossesses antérieures (zone physique 13):

Médecin	Pharmacien	Ambulancier	Infirmière	Infirmière en chef
LE				L

Cette zone comprend quatre groupes d'informations. Toutes les données de cette zone concernent les grossesses antérieures. L'information de chaque groupe provient d'une zone logique différente et est donc inscrite séparément. Toutes les variables d'un même groupe sont saisies et inscrites sur la carte en même temps, d'où cette subdivision en quatre parties.

groupe 1 Gravida (L26)
Para (L26)
Aborta (L26)
Nombre d'enfants vivants (L26)

groupe 2 Date d'accouchement (L28)
Durée du travail (L28)
Durée de la grossesse (L28)
Mode d'accouchement (L28)
Dilatation curetage (L28)
Type d'anesthésie (L28)
Garçon ou fille (L28)
Poids (L28)
Liste des particularités (L28)

groupe 3 Liste des facteurs de risque (L32)

Suivi de grossesse et visites prénatales (zone physique 14):

Médecin	Pharmacien	Ambulancier	Infirmière	Infirmière en chef
LE				L

Cette zone est subdivisée en deux groupes d'informations. Chaque groupe d'informations provient d'une zone logique à part et est saisi et inscrit sur la carte en même temps.

groupe 1 Date d'accouchement (L27)
Suites de couches normales (L27)
Hémorragie puerpérale (L27)
Fièvre (L27)

groupe2 Date de visite (L33)
Semaines de grossesse (L33)

Hauteurs fond utérin (L33)
Albuminerie (L33)
Glycosurie (L33)
Coeur foetal (L33)
Mouvement foetal (L33)
Oedème (L33)
Présentation du foetus (L33)
Remarque (L33)

Résultats de tests pour grossesse (zone logique 15):

Médecin	Pharmacien	Ambulancier	Infirmière	Infirmière en chef
LE				L

Cette zone comprend sept groupes d'informations. Chaque groupe contient des données venant d'une autre zone logique, sauf pour les groupes 4, 5 et 6 qui contiennent tous les trois des données de la même zone logique 39. Cette division en trois groupes est dictée, pour des soucis d'économie d'espace, par le fait que ces données ne sont pas nécessairement toujours inscrites ensemble.

groupe 1
Résultat amniocentèse (L29)
Date amniocentèse (L29)

groupe 2
Age gestationnel (DDM) (L30)
Age gestationnel (échographie) (L30)
Percentile (L30)
Morphologie (L30)
Profil Bio-Physique (L30)

groupe 3
Immunoglobine anti D (L31)
Date immunoglobine (L31)

groupe 4
Anticorps Toxoplasmose (L34)
Anticorps Rubéole (L34)
Anticorps HBs (L34)
Anticorps VDRL (L34)

groupe 5
Cytologie (L34)

groupe 6
D.P.A. selon clinique (L34)
D.P.A. selon échographie (L34)
D.P.A. selon DDM (L34)

groupe 7
Hb (1,2 & 3) (L35)
Hte (1,2 & 3) (L35)
Coombs directs (1,2 & 3) (L35)
Coombs indirects (1,2 & 3) (L35)
Glycémie (L35)
Urine (L35)

ANNEXE B

La représentation physique des données dans les zones physiques

Nous allons présenter dans ce qui suit la codification de chaque champ du contenu des 15 zones physiques du DMP. Les champs ne sont pas regroupés comme dans l'annexe A, mais donnés séquentiellement. Dans cette codification, on ne retrouve pas la date et la signature qui ont été omises pour des raisons de simplification. La date est codée en nombre de jours après une date particulière, en l'occurrence le 1er mars 1992. Sur dix (10) bits, 1 024 jours peuvent représenter des dates jusqu'au 19 décembre 1994. La codification de la signature et de la date de transaction est la suivante:

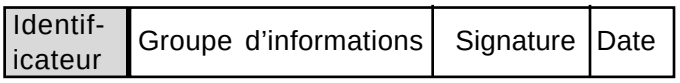
Signature	(adresse vers table)	8	bits
Date	(Date_4)	10	bits

Nous n'avons pas représenté la codification des identificateurs de groupes d'informations. Celle-ci dépend du nombre de groupes dans une zone et varie entre zéro, un, deux, trois et quatre bits. Dans la codification d'un champ, la lettre 'n' signifie une répétition, donc une codification d'une suite. 'n' représente le nombre d'éléments dans la suite.

Pour chaque zone physique, nous présentons une représentation graphique de son contenu.

Identification (zone physique 1):

Taille du nom	(de 0 à 30)	5	bits
Nom	(Car [ISO5])	n*5	bits
Taille du prénom	(de 0 à 20)	5	bits
Prénom	(Car [ISO5])	n*5	bits
Sexe		1	bits
Date de Naissance	(Date_1)	17	bits
Numéro d'assurance maladie	(4 Car [ISO5] + 2 Chif [BCD])	28	bits
Date d'expiration	(Date_3)	11	bits
Numéro de dossier (à l'hôpital de Rimouski)	(6 Chif [BCD])	24	bits
Taille de personne-à-prévenir	(de 0 à 50)	6	bits
Personne-à-prévenir	(Car [IS05])	n*5	bits
Numéro de téléphone	(10 Chif [BCD])	40	bits

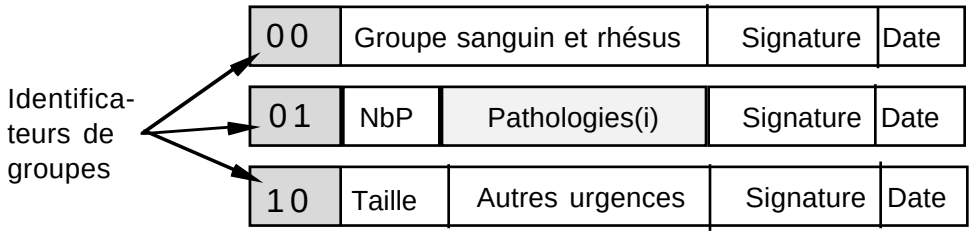


L'identificateur de groupe d'informations est sur trois bits pour coder les huit groupes d'informations possibles. La structure de l'ensemble d'informations diffère d'un groupe à l'autre (voir 2.6.1).

Urgence (zone physique 2):

Groupe sanguin et rhésus	(code pour 8)	3	bits
Nombre de pathologies	(de 0 à 15)	4	bits

Liste des numéros de pathologie	(n codes pour 16)	n*4	bits
Taille de autres-urgences	(de 0 à 63)	6	bits
Autres-urgences	(Car [ASCII7])	n*7	bits

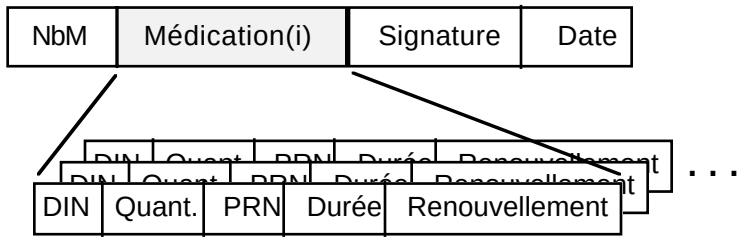


Médication (zone physique 3):

Nombre de médicaments	(de 0 à 15)	3	bits
Liste de *Médicaments(i)*	(structuré)	n*57	bits

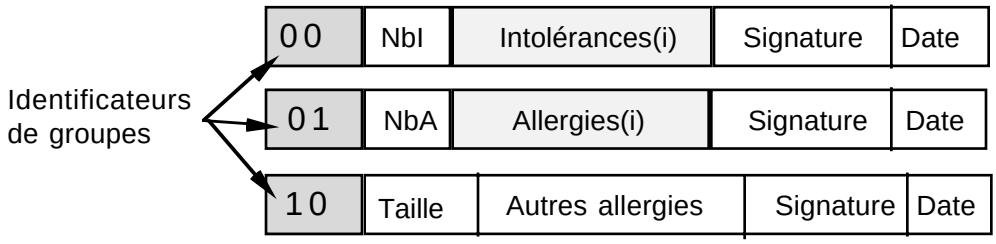
La structure de médicaments(i):

Code du médicament (DIN)	(8 Chif [BCD])	32	bits
Quantité délivrée	(de 0 à 1023)	10	bits
Pro_Ré_Nata (PRN)		1	bit
Duré du traitement	(de 0 à 255 jours)	8	bits
Nombre de renouvellements	(de 0 à 63)	6	bits



Intolérances médicamenteuses et allergies (zone physique 4):

Nombre des intolérances médicamenteuses	(de 0 à 15)	4	bits
Liste des intolérances médicamenteuses (DIN)	(n * 8 Chif [BCD])	n*32	bits
Nombre des allergies médicamenteuses	(de 0 à 15)	4	bits
Liste des allergies médicamenteuses (DIN)	(n * 8 Chif [BCD])	n*32	bits
Taille de autres-allergies	(de 1 à 64)	6	bits
Autres-allergies	(Car [ASCII7])	n*7	bits



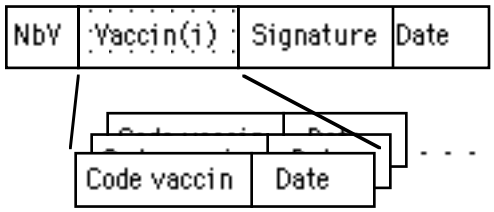
Vaccination (zone physique 5):

Nombre de vaccins	(de 0 à 15)	4	bits

Liste de *vaccins*	(structuré)	n*24	bits

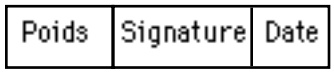
La structure de vaccins(i):

Code du vaccin	(code pour 256)	8	bits
Date de vaccination	(Date_2)	16	bits



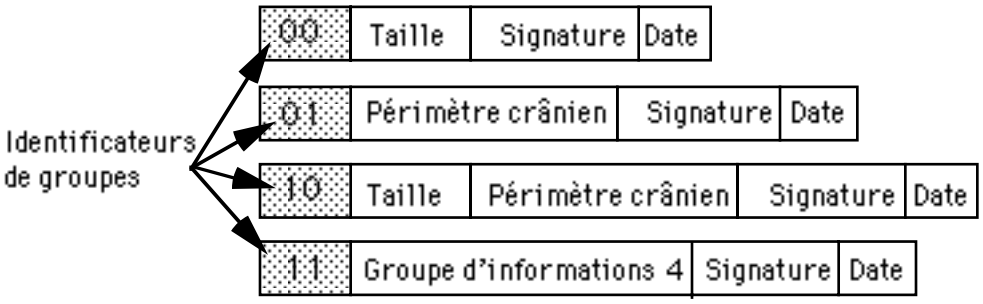
Suivi du poids (zone physique 6):

Poids	(de 1 à 262 144 grammes)	18	bits



Suivi pédiatrique (zone physique 7):

Taille	(de 1 à 128 cm)	7	bits
Périmètre crânien	(de 1 à 64 cm)	6	bits
Groupe information 4:			
Gestation	(entre 20 et 51)	5	bits
Complications		1	bit
Travail spontané ou déclenché		1	bit
Durée totale du travail en heure	(entre 0 et 63)	6	bits
Minutes supplémentaires	(0 pour 00 et 1 pour 30)	1	bit
Type d'accouchement	(code pour 32)	4	bits
Apgar 1 minute	(entre 0 et 10)	4	bits
Apgar 5 minutes	(entre 0 et 10)	4	bits
Apgar 10 minutes	(entre 0 et 10)	4	bits
Réanimation		1	bit
Alimentation		1	bit



Suivi général et prévention (zone physique 8):

Nombre de diagnostics	(de 0 à 15)	4	bits
Liste de*Diagnostics(i)*	(structuré)		

La structure de diagnostic(i):

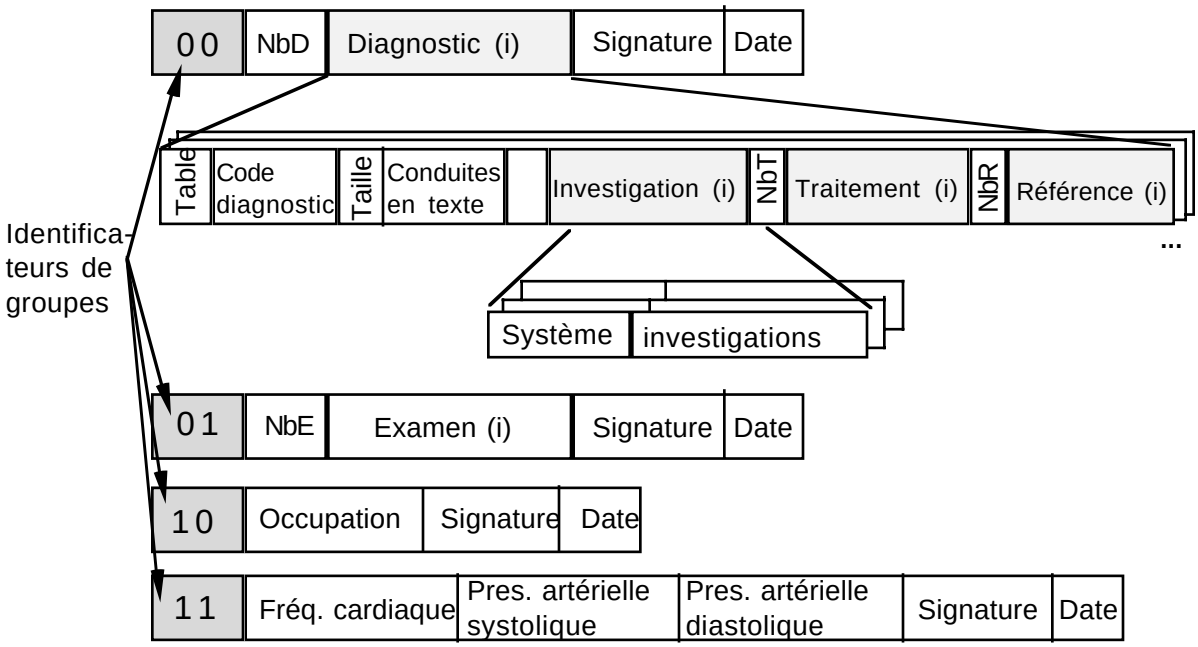
Numéro de table	(Code pour 4)	2	bits
Code du diagnostic	(5 Car (ISO6))	30	bits
Taille de conduites en texte libre	(de 0 à 63)	6	bits
Conduites en texte libre	(Car[ASCII7]	n*7	bits
Nombre d'investigations	(de 0 à 15)	4	bits
Liste d'*investigations*	(structuré)	n*18	bits

La structure d'investigation(i):

Système	(code pour 32)	4	bits
Investigations	(bits stream)	13	bits

Nombre de traitements	(de 0 à 15)	4	bits
Liste de traitements	(n codes pour 8)	n*3	bits
Nombre de références	(de 0 à 15)	4	bits
Liste de références	(n codes pour 64)	n*6	bits

Nombre d'examens	(jusqu'à 16)	4	bits
Liste des numéros des examens de prévention	(n codes pour 64)	n*6	bits
Occupation professionnelle	(code pour 128)	7	bits
Fréquence cardiaque	(de 1 à 400)	9	bits
Pression artérielle systolique	(de 0 à 400)	9	bits
Pression artérielle diastolique	(de 0 à 200)	8	bits



Autres suivis (zone physique 9):

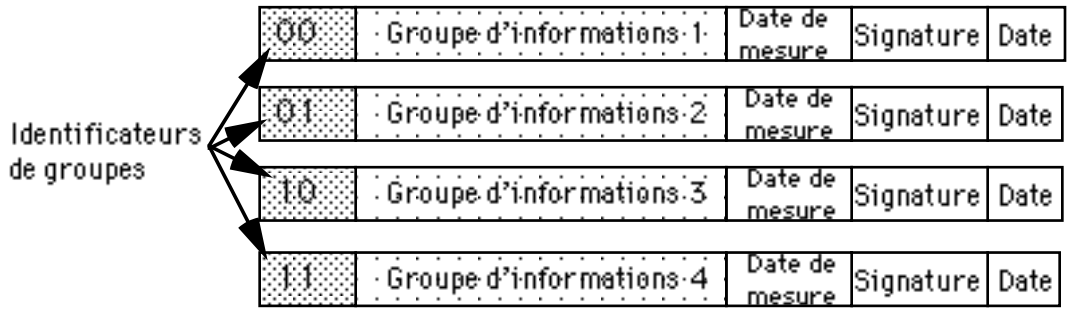
Groupe information 1:

Cholestérol total	(réel_5 de 00.00 à 99.99)	14	bits
HDL-cholestérol	(réel_5 de 00.00 à 99.99)	14	bits
LDL-cholestérol	(réel_5 de 00.00 à 99.99)	14	bits
Triglycéride	(réel_5 de 00.00 à 99.99)	14	bits
Date de mesure	(Date_3)	11	bits

Groupe information 2:

Fructosamine	(entier_9 de 100 à 999)	10	bits

Glycémie à jeûn	(réel_4 de 1.0 à 99.0)	11	bits
Glycémie 2 heures p.c.	(réel_4 de 1.0 à 99.0)	11	bits
Hémoglobine glycosylée	(entier_6 de 1 à 99)	7	bits
Protéinurie (24 heures)	(entier_6 de 0 à 99)	7	bits
Date de mesure	(Date_2)	16	bits
Groupe information 3:			
Calcium	(réel_2 de 1.00 à 4.00)	11	bits
HCO3	(entier_6 de 0 à 99)	7	bits
Hématocrite	(entier_6 de 0 à 99)	7	bits
Hémoglobine	(entier_8 de 30 à 300)	9	bits
PH	(entier_6 de 0 à 100)	7	bits
Phosphore	(réel_2 de 0.00 à 9.99)	11	bits
Potassium	(réel_1 de 2.0 à 9.9)	8	bits
Prétéinurie		1	bit
Sodium	(entier_6 de 100 à 170)	7	bits
Date de mesure	(Date_2)	16	bits
Groupe information 4:			
Débit de pointe	(entier_9 de 20 à 700)	10	bits
Indice de saturation	(réel_2 de 0.00 à 1.02)	11	bits
PCO2 (artérielle)	(réel_4 de 00.0 à 99.9)	11	bits
PO2 (artérielle)	(réel_7 de 000.0 à 999,9)	14	bits
Date de mesure	(Date_2)	16	bits



Variables de suivi (zone physique 10):

Nombre de type	(jusqu'à 4)	2	bits
Liste de ***Types Complexes*** *(i)*	(structuré)		

La structure de Types complexes(i):

Type primaire des variables	(Code pour 4)	2	bits
Nombre de variables	(jusqu'à 16)	4	bits
Liste de***Variables (i)***	(structuré)		

La structure de Variables(i):

Sous-type	(diffère d'un type primaire à l'autre)		
Code de la variable	(Code pour 32)	5	bits
Valeur de la variables	(diffère d'un type à l'autre et sous-type à l'autre)		

Il y a quatre types primaires: booléen, entier, réel et chaîne de caractères. Pour les booléen, il n'existe pas de sous-type. Par contre pour les autres types, nous avons des sous-types différents. Pour chaque type primaire, il y a les types secondaires suivants:

Type primaire entier:

sous-type entier_3	» identificateur codé sur 3 bits	000
sous-type entier_5	» identificateur codé sur 3 bits	001
sous-type entier_6	» identificateur codé sur 3 bits	010
sous-type entier_8	» identificateur codé sur 3 bits	011
sous-type entier_9	» identificateur codé sur 3 bits	100
sous-type entier_10	» identificateur codé sur 3 bits	101
sous-type entier_13	» identificateur codé sur 3 bits	110
sous-type entier_15	» identificateur codé sur 3 bits	111

Une variable de type primaire *entier* est codée:

Sous-type	(code pour 8)	3 bits
Valeur	(diffère d'un sous-type à l'autre)	(Voir 2.5)

Type primaire réel:

sous-type réel_2	» identificateur codé sur 2 bits	00
sous-type réel_4	» identificateur codé sur 2 bits	01
sous-type réel_5	» identificateur codé sur 2 bits	10
sous-type réel_9	» identificateur codé sur 2 bits	11

Une variable de type primaire *réel* est codée:

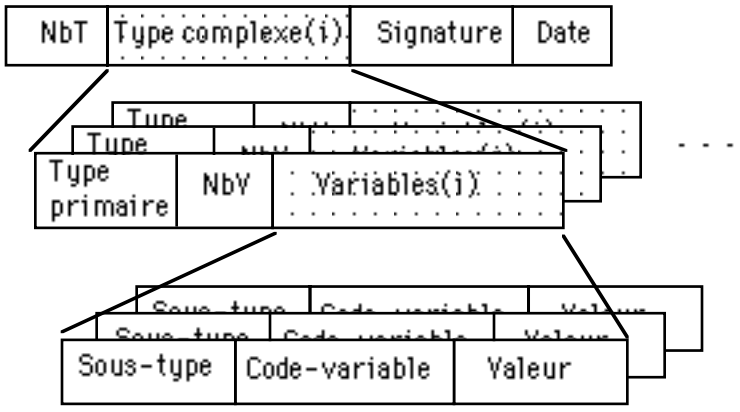
Sous-type	(code pour 4)	2 bits
Valeur	(diffère d'un sous-type à l'autre)	(voir 2.5)

Type primaire chaîne de caractères:

sous type BCD	» identificateur codé sur 2 bits	00
sous type ISO5	» identificateur codé sur 2 bits	01
sous type ISO6	» identificateur codé sur 2 bits	10
sous type ASCII7	» identificateur codé sur 2 bits	11

Une variable de type primaire *chaîne de caractère* est codée:

sous-type	(code pour 4)	2 bits
Taille de la chaîne	(jusqu'à 64)	6 bits
Valeur	(diffère d'un sous-type à l'autre)	(voir 2.5)



Antécédents et créatinine (zone physique 11):

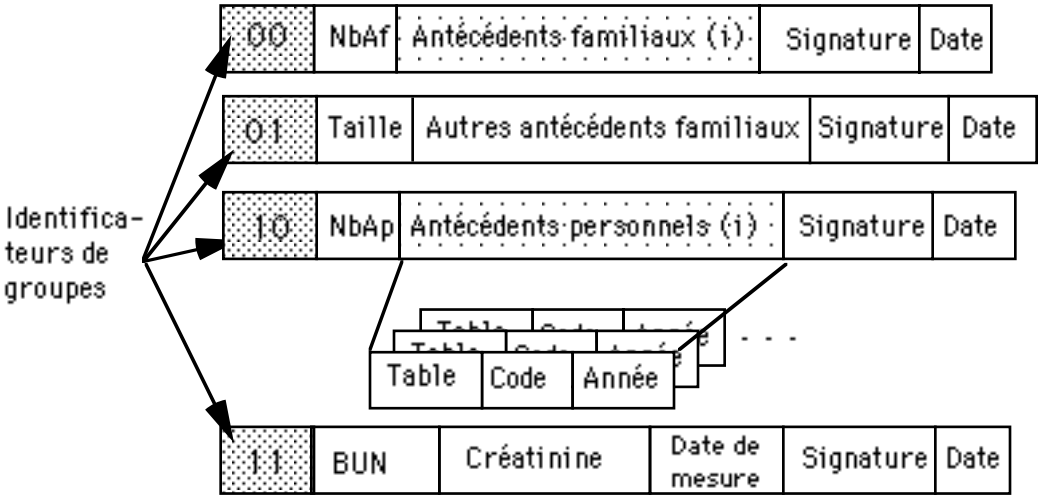
Nombre d'antécédents familiaux	(de 0 à 15)	4 bits
Liste des antécédents familiaux	(n codes pour 14)	n*4 bits
Taille de autres-antécédents-familiaux	(de 1 à 64)	6 bits
Autres-antécédents-familiaux	(car [ASCII7])	n*7 bits

Nombre antécédents personnels	(de 0 à 15)	4	bits
Liste d' *antécédents personels (i)*	(structuré)	n*45	bits

La structure d'antécédents personnels(i):

Table	(Code pour 4)	2	bits
Code	(6 car [ISO6])	36	bits
Année	(entier_6 de 0 à 99)	7	bits

BUN (urée)	(réel_4 de 00.0 à 99.9)	11	bits
Créatinine	(entier_9 de 10 à 999)	10	bits
Date de mesure	(Date_2)	16	bits



Ophtalmologie (zone physique 12):

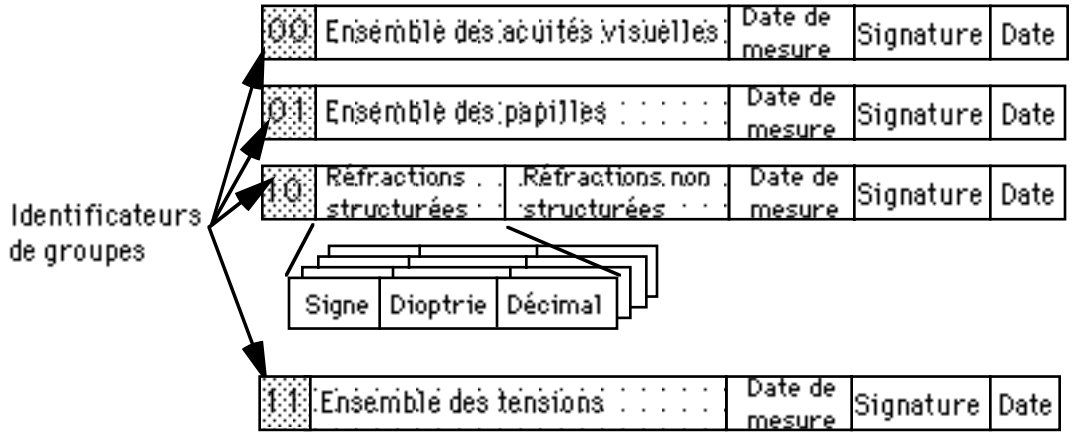
Acuité visuelle: 2 yeux sans lunette	(code pour 13)	4	bits
Acuité visuelle: 2 yeux avec lunette	(code pour 13)	4	bits
Acuité visuelle: l'oeil droit sans lunette	(code pour 13)	4	bits
Acuité visuelle: l'oeil droit avec lunette	(code pour 13)	4	bits
Acuité visuelle: l'oeil gauche sans lunette	(code pour 13)	4	bits
Acuité visuelle: l'oeil gauche avec lunette	(code pour 13)	4	bits
Date de mesure de l'acuité	(Date_3)	11	bits
Papille droite	(entier_3 de 0 à 10)	4	bits
Papille gauche	(entier_3 de 0 à 10)	4	bits
Date de mesure de la papille	(Date_3)	11	bits
Réfraction droite-sphère	(structuré 1)	8	bits
Réfraction gauche-sphère	(structuré 2)	8	bits
Réfraction droite-force du cylindre	(structuré 3)	8	bits
Réfraction gauche-force du cylindre	(structuré 4)	8	bits

Les structures 1, 2, 3 & 4 des réfractions:

Signe	(+ ou -)	1	bit
Dioptrie	(entrier_5 de 0 à 25)	5	bits
Décimal	(code pour 4)	2	bits

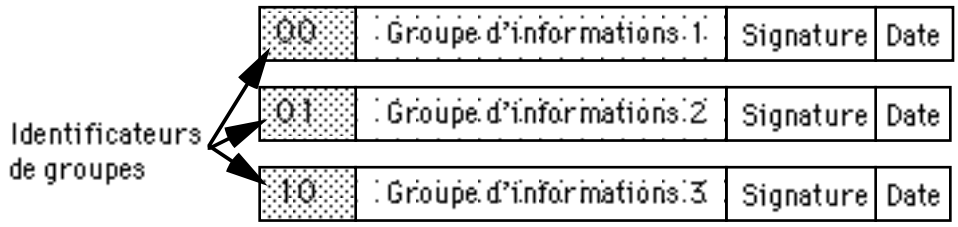
Réfraction droite-axe	(entier_7 de 0 à 180)	8	bits
Réfraction gauche-axe	(entier_7 de 0 à 180)	8	bits
Date de mesure de la réfraction	(Date_3)	11	bits
Tension occulaire droite	(entier_6 de 0 à 99)	7	bits
Tension occulaire gauche	(entier_6 de 0 à 99)	7	bits

Date de mesure de la tension	(Date_3)	11	bits



Résumés des grossesses antérieures (zone physique 13):

Groupe information 1:			
Gravida	(entier_3 de 0 à 15)	4	bits
Para	(entier_3 de 0 à 15)	4	bits
Aborta	(entier_3 de 0 à 15)	4	bits
Nombre d'enfants vivants	(entier_3 de 0 à 15)	4	bits
Groupe information 2:			
Date d'accouchement	(Date_2)	16	bits
Durée du travail	(entier_5 de 0 à 63)	6	bits
Durée du travail en minute	(00 min ou 30 min)	1	bit
Durée de la grossesse	(entier_5 de 0 à 63)	6	bits
Mode d'accouchement	(code pour 32)	5	bits
Dilatation curetage		1	bit
Type d'anesthésie	(code pour 15)	4	bits
Garçon ou fille		1	bit
Poids	(entier_12 de 0 à 8192 grammes)	13	bits
Nombre des particularités	(de 0 à 15)	4	bits
Liste des particularités	(n codes pour 64)	n*6	bits
Groupe information 3:			
Nombre de facteurs de risque	(de 0 à 15)	4	bits
Liste des facteurs de risque	(n codes pour 64)	n*6	bits

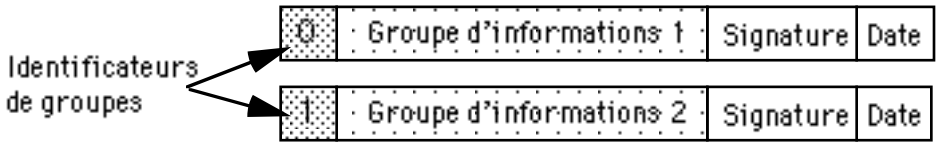


Suivi de grossesse et visites prénatales (zone physique 14):

Groupe information 1:			
Date d'accouchement	(Date_2)	16	bits
Suites de couches normales		1	bit
Hémorragie puerpérales		1	bit
Fièvre	(bits stream)	5	bits

Groupe information 2:

Date de visite	(Date_2)	16	bits
Semaines de grossesse	(entier_5 de 0 à 63)	6	bits
Hauteurs fond utérin	(entier_5 de 0 à 63)	6	bits
Albuminerie		1	bit
Glycosurie		1	bit
Coeur foetal		1	bit
Mouvement foetal		1	bit
Oedème		1	bit
Présentation du foetus	(code pour 4)	2	bits
Taille de remarque	(de 0 à 63)	6	bits
Remarque	(car [ASCII7])	n*7	bits



Résultats de tests pour grossesse (zone logique 15):

Groupe information 1:

Résultat amniocentèse		1	bit
Date amniocentèse	(Date_2)	16	bits

Groupe information 2:

Age gestationnel (DDM)	(entier_5 de 0 à 63)	6	bits
Age gestationnel (échographie)	(entier_5 de 0 à 63)	6	bits
Percentile	(entier_6 de 0 à 100)	7	bits
Morphologie		1	bit
Profil Bio-Physique	(code pour 6)	3	bits

Groupe information 3:

Immunoglobine anti D		1	bit
Date Immunoglobine anti D	(Date_2)	16	bits

Groupe information 4:

Anticorps Toxoplasmose	(entier_4 de 0 à 31)	4	bits
Anticorps Rubéole	(entier_4 de 0 à 31)	4	bits
Anticorps HBs	(entier_4 de 0 à 31)	4	bits
Anticorps VDRL	(entier_4 de 0 à 31)	4	bits

Groupe information 5:

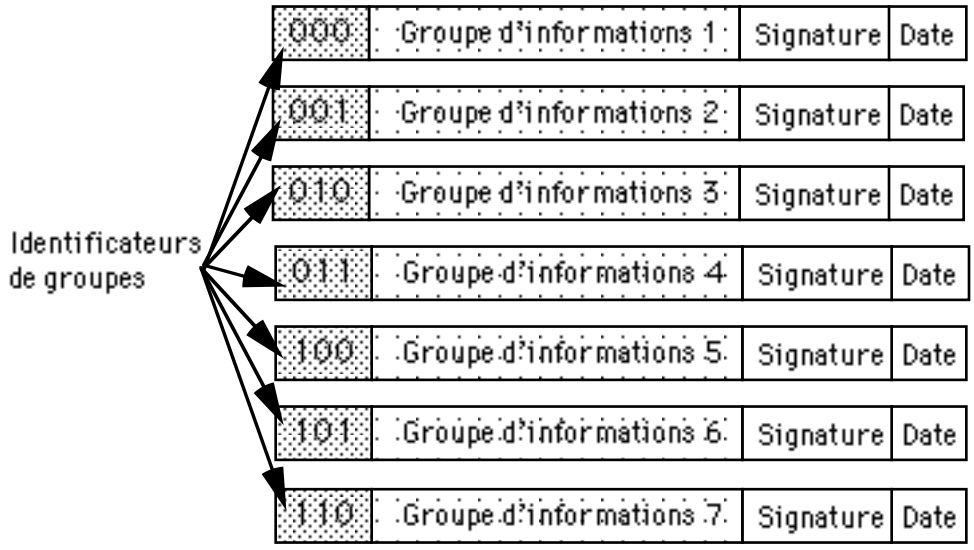
Cytologie	(code pour 3)	2	bits

Groupe information 6:

D.P.A. selon clinique	(Date_2)	16	bits
D.P.A. selon échographie	(Date_2)	16	bits
D.P.A. selon DDM	(Date_2)	16	bits

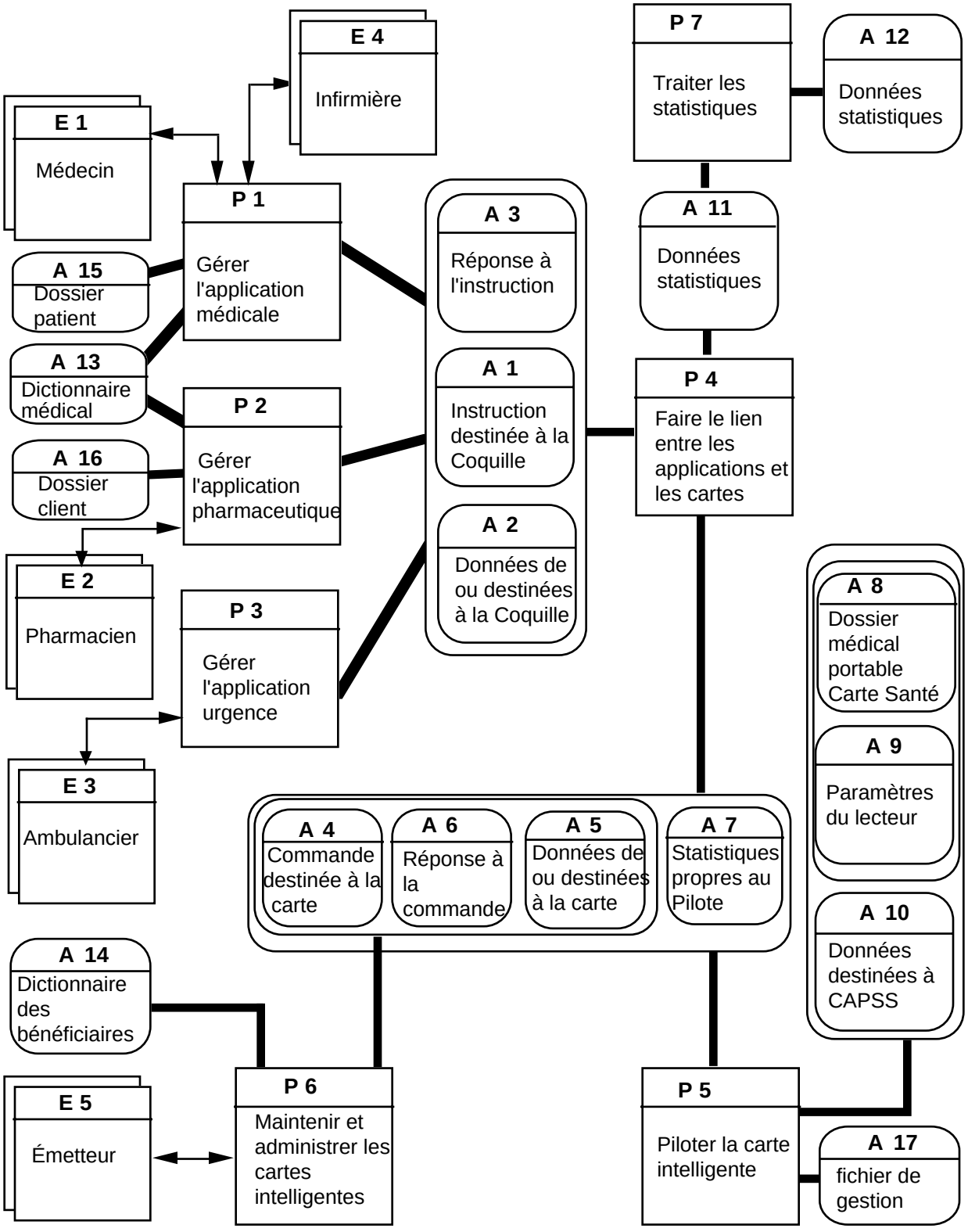
Groupe information 7:

Hb (1,2 & 3)	(entier_7 de 0 à 200)	8	bits
Hte (1,2 & 3)	(entier_7 de 0 à 200)	8	bits
Coombs directs (1,2 & 3)		1	bit
Coombs indirects (1,2 & 3)		1	bit
Glycémie	(entier_4 de 0 à 31)	5	bits
Urine		1	bit


Identificateurs
de groupes
000
Groupe d'informations 1
Signature
Date
001
Groupe d'informations 2
Signature
Date
010
Groupe d'informations 3
Signature
Date
011
Groupe d'informations 4
Signature
Date
100
Groupe d'informations 5
Signature
Date
101
Groupe d'informations 6
Signature
Date
110
Groupe d'informations 7
Signature
Date

ANNEXE C

Diagramme global du système Carte Santé



Guide de présentation
Diagramme global du système Carte Santé

Perspective de modélisation:
Objectif: Décrire le système Carte Santé
Limites: L'émetteur de la carte, le médecin, le pharmacien, l'infirmière et l'ambulancier.

P1 Gérer l'application médicale

Le processus "Gérer l'application médicale" (P1) reçoit des transactions médicales du médecin (**E1**) et de l'infirmier(ère) (**E4**) sous forme de requêtes d'enregistrement et de consultation destinées au dossier patient (**A15**) et au DMP (**A8**). Il utilise un dictionnaire médical (**A13**) pour transcrire ses codes et ses données puis, transmet des instructions (**A1**) et des données (**A2**) au processus **P4** pour les véhiculer à la carte intelligente. En retour, il reçoit une réponse à sa requête (**A3**), met à jour le dossier patient (**A15**) et répond aux requêtes du médecin (**E1**) et de l'infirmier(ère) (**E4**) à partir de la réponse **A5** et du dossier patient (**A15**).

P2 Gérer l'application pharmaceutique

Le processus "Gérer l'application pharmaceutique" (P2) reçoit des transactions pharmaceutiques du pharmacien(ne) (**E2**) sous forme de requêtes d'enregistrement et de consultation destinées au dossier client (**A16**) et au DMP (**A8**). Il utilise un dictionnaire médical (**A13**) pour transcrire ses codes et ses données puis, transmet des instructions (**A1**) et des données (**A2**) au processus **P4** pour les véhiculer à la carte intelligente. En retour, il reçoit une réponse à sa requête (**A3**), met à jour le dossier client (**A16**) et répond aux requêtes du (de la) pharmacien(ne) (**E2**) à partir de la réponse **A5** et du dossier client (**A16**).

P3 Gérer l'application urgence

Le processus "Gérer l'application urgence" (P3) reçoit des transactions médicales de l'ambulancier (E3) destinées à mettre à jour ou consulter le DMP (**A8**) puis transmet des instructions (**A1**) et des données (**A2**) au processus **P4** pour les véhiculer à la carte intelligente. En retour, il reçoit une réponse à sa requête (A3) et répond à **E3** à partir de **A5**.

P4 Faire le lien entre les application et les cartes

Le processus "Faire le lien entre les application et les cartes" (P4) reçoit des processus **P1**, **P2** et **P3** des instructions et des données (**A1** et **A 2**) destinées à la Coquille. Il génère des commandes (**A4**) ainsi que des données destinées à la carte (**A5**) au processus **P5** à partir des données **A2**. En retour, il reçoit du processus **P5** une réponse à chaque commande (**A6**), des données de la carte (**A5**) et des statistiques propres au Pilote (**A7**). Il transmet des données statistiques (**A11**) au processus **P7** et

répond aux requêtes des processus **P1**, **P2** et **P3** par un message **A3** et des données **A2**.

P5 Piloter la carte intelligente

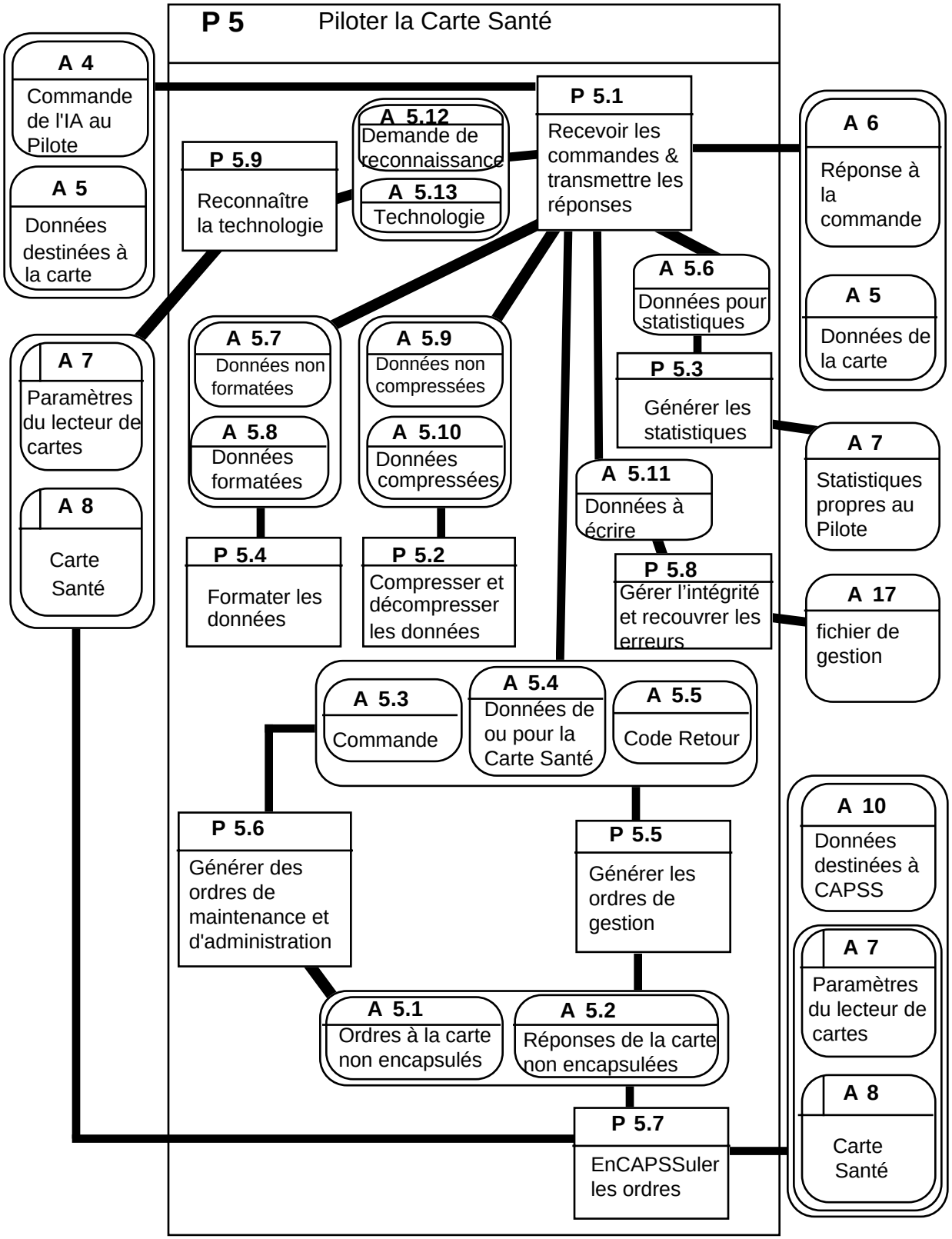
Le processus "Piloter la carte intelligente" (P5) reçoit des commandes (A4) et des données destinées à la carte (**A5**) du processus **P4** ou **P6**. Il transmet alors des informations (**A8**, **A9** et **A10**) au micro-serveur qui les transmettra au lecteur et à la carte intelligente ou directement aux destinataires s'il n'y a pas de micro-serveur. Le processus **P5** renvoie ensuite une réponse aux commandes (**A6**) et des données de la cartes (**A5**) au processus appelant après avoir analysé les données du DMP (**A8**). Il calcule des statistiques (**A7**) sur les informations qu'il manipule et les transmet au processus **P4**. Le processus **P5** utilise un fichier de gestion (**A17**) pour sauvegarder des informations nécessaires pour la gestion et le recouvrement de certaines erreurs.

P6 Maintenir et administrer les cartes intelligentes

Le processus "Maintenir et administrer les cartes intelligentes" (P6) reçoit des instructions de gestion de l'émetteur de carte (**E5**) et utilise le fichier des bénéficiaires (**A14**). Il transmet des commandes de maintenance et de gestion (**A4**) et des données (**A5**) au processus **P5** qui les transmet à son tour à la carte. En retour, le processus **P6** reçoit une réponse à la commande (**A6**) et des données de la carte (**A5**).

P7 Traiter les statistiques

Le processus "Traiter les statistiques" (P7) reçoit des données statistiques (**A11**) du processus **P5** et les emmagasine dans un fichier de statistiques (**A12**).


P 5
Piloter la Carte Santé
A 4
Commande de l'IA au Pilote
A 5
Données destinées à la carte
A 7
Paramètres du lecteur de cartes
A 8
Carte Santé
P 5.1
Recevoir les commandes & transmettre les réponses
A 5.12
Demande de reconnaissance
A 5.13
Technologie
P 5.9
Reconnaître la technologie
A 6
Réponse à la commande
A 5
Données de la carte
A 5.6
Données pour statistiques
P 5.3
Générer les statistiques
A 7
Statistiques propres au Pilote
A 5.7
Données non formatées
A 5.8
Données formatées
A 5.9
Données non compressées
A 5.10
Données compressées
A 5.11
Données à écrire
P 5.4
Formater les données
P 5.2
Compresser et décompresser les données
P 5.8
Gérer l'intégrité et recouvrer les erreurs
A 17
fichier de gestion
A 5.3
Commande
A 5.4
Données de ou pour la Carte Santé
A 5.5
Code Retour
P 5.6
Générer des ordres de maintenance et d'administration
P 5.5
Générer les ordres de gestion
A 10
Données destinées à CAPSS
A 7
Paramètres du lecteur de cartes
A 8
Carte Santé
A 5.1
Ordres à la carte non encapsulés
A 5.2
Réponses de la carte non encapsulées
P 5.7
EnCAPSSuler les ordres

Guide de présentation
Processus P5: Piloter la Carte Santé

Perspective de modélisation:
Objectif: Décrire le Pilote de la Carte Santé
Limites:

P 5.1 Recevoir les commandes et transmettre les réponses

Le processus "Recevoir les commandes et transmettre les réponses" appelle le processus **P5.9** pour reconnaître les technoilogies de cartes en lui transmettant une demande de reconnaissance (**A5.12**) et reçoit en contrepartie l'identification de la technologie (**A5.13**). Il reçoit des commandes (**A4**) et des données destinées à la carte (**A5**) de l'Interface Applicative, appelle le processus **P5.4** pour formater et analyser les données (**A5.8**) lors d'écriture ou pour obtenir des données non formatées (**A5.7**) lors des lectures. Il fait appel au processus **P5.2** pour compresser ou décompresser des données (**A5.9** et **A5.10**) et au processus **P5.3** pour générer des statistiques (**A7**) en lui transmettant des données (**A5.6**). Le processus **P5.1** transmet des commandes (**A5.3**) et des données (**A5.4**) au processus **P5.5** ou au processus **P5.6** pour générer des ordres à la carte. Les processus **P5.5** et **P5.6** lui renvoient les données de la carte (**A5.4**) et un code retour (**A5.5**). En retour ,le processus **P5.1** transmet la réponse à la commande (**A6**) et les données de la carte (**A 5**) à l'Interface Applicative. Pour assurer l'intégrité des données écrites sur la carte, le processus **P5.1** fait appel au processus **P5.8** en lui transmettant des données à écrire sur disque (**A5.11**). Au début d'une session et dans le cas où on détecte une interruption d'une ancienne transaction d'écriture sur la carte, le processus **P5.1** reçoit des données (**A5.11**) du processus **P5.8** pour tenter de recouvrer l'erreur.

P5.2 Compresser et décompresser les données

Le processus "Compresser et décompresser les données" (P5.6) effectue la compression des données ou leur expansion (décompression). À la réception d'un ordre de compression et des données d'origine non traitées (**A5.9**), le processus vérifie l'occasion de comprimer les données et effectue la compression si celle-ci est possible. Lorsqu'il reçoit un ordre de décompression et des données de la carte (**A5.10**), le processus vérifie si ces données sont compressées et effectue leur expansion dans le cas positif.

P 5.3 Générer les statistiques

Le processus "Générer les statistiques" (P5.3) reçoit des données (**A5.6**) et effectue des calculs à des fins de statistiques (**A 7**).

P5.4 Formater les données

Le processus "Formater les données" (P5.4) est activé par le processus "Recevoir les commandes" (**P5.1**) à la réception de données non formatées (**A5.7**). En retour, le

processus remet des données formatées et analysées (**A5.8**). À la réception de données formatées (**A5.8**), le processus transforme ceux-ci et remet des données non formatées (**A5.7**). Le formatage consiste à analyser les données, à enlever certaines redondances d'informations et de coder les données destinées à la Carte Santé.

P5.5 Générer les ordres

Le processus "Générer les ordres" (P5.5) reçoit une commandes pour la Carte Santé (**A5.3**) et des données pour la Carte Santé (**A5.4**) du processus "Recevoir les commandes" (**P5.1**), génère des ordre à la Carte Santé (**A8**) via le processus **P5.7** en transmettant des ordres non encapsulés à la carte (**A5.1**) et retourne les données de la Carte Santé (**A5.4**) ainsi qu'un code retour (**A5.5**) indiquant l'état d'exécution de la commande (**A5.3**) après avoir reçu les réponses non encapsulées de la carte (**A5.2**) du processus **P5.7**.

P5.6 Générer les ordres de maintenance et d'administration

Le processus "Générer les ordres de maintenance et d'administration" (P5.6) reçoit une commandes pour la Carte Santé (**A5.3**) et des données pour la Carte Santé (**A5.4**) du processus "Recevoir les commandes" (**P5.1**), génère des ordres à la Carte Santé (**A8**) via le processus **P5.7** en transmettant des ordres non encapsulés à la carte (**A5.1**) et retourne les données de la Carte Santé (**A5.4**) ainsi qu'un code retour (**A5.5**) indiquant l'état d'exécution de la commande (**A5.3**) après avoir reçu les réponses non encapsulées de la carte (**A5.2**) du processus **P5.7**.

P 5.7 EnCAPSSuler les ordres

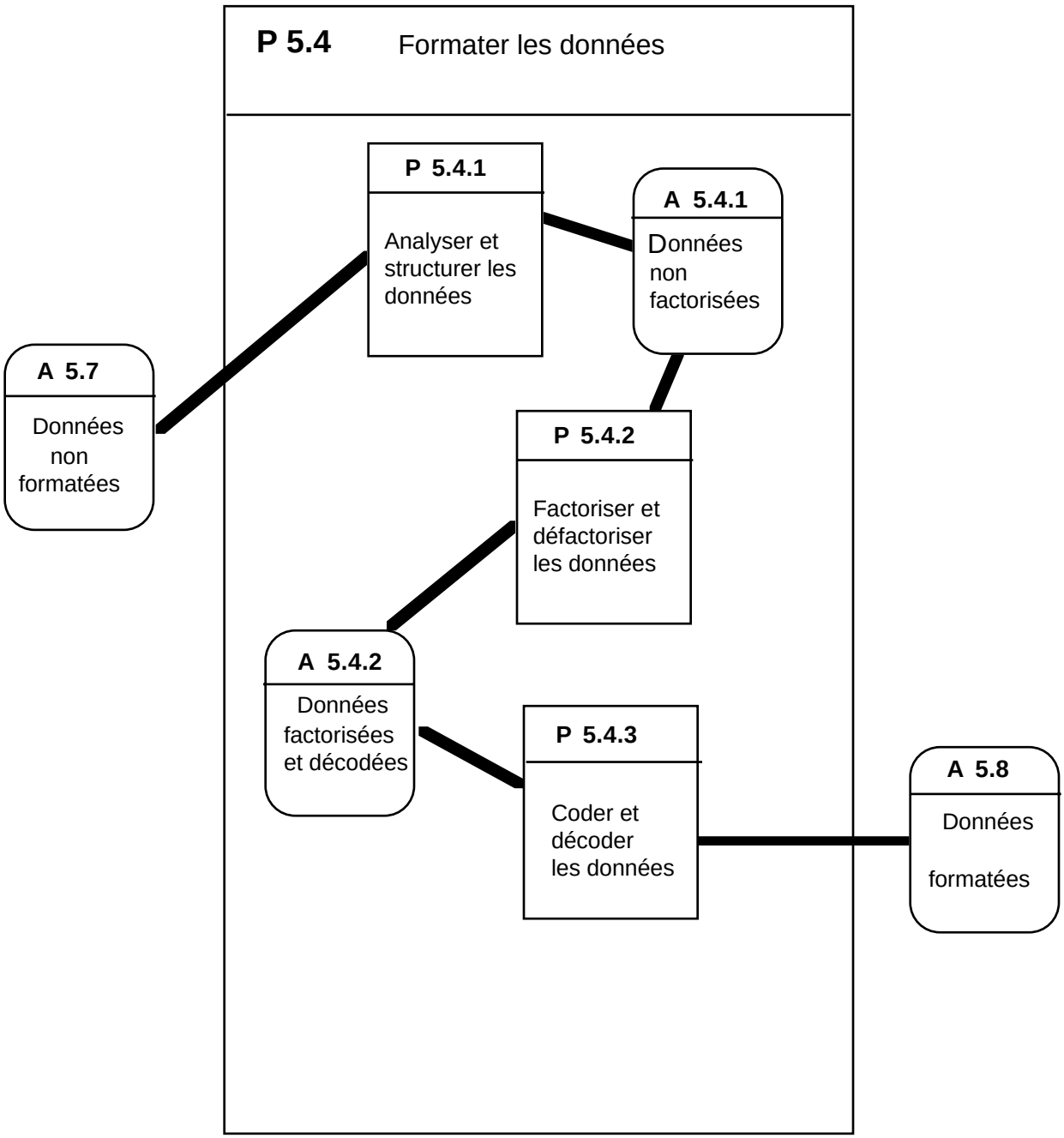
Le processus "EnCAPSSuler les ordres" (P5.7) reçoit un ordre non encapsulé destiné à la carte (**A5.1**) et lui rajoute des caractère de contrôle exigés par le protocole de communication avec le micro-serveur de CAPSS puis le transmet à la Carte Santé (**A8**) et le lecteur des cartes (**A9**). En recevant des données du micro-serveur, le processus vérifie si le message lui est destiné. Dans le cas positif, il lui élimine les caractères de contrôle pour remettre au processus appelant la réponse de la carte non encapsulée (**A5.2**). Dans le cas négatif, les données destinées à CAPSS (**A10**) sont redirigées sur le port-série.

P 5.8 Gérer l'intégrité et recouvrer l'erreur

Le processus "Gérer l'intégrité et recouvrer l'erreur" (P5.8) assure l'intégrité des données des transactions physiques. Il reçoit des données (**A 5.11**) et les stocke dans le fichier de gestion (**A17**). À la fin de la transaction, il efface les mêmes données du fichier. Au début d'une session, dans le cas d'une détection d'une écriture non intègre, le processus **P5.8** lit les données du fichier de gestion (**A17**) et les transmet au processus **P5.1** pour recouvrer l'intégrité de l'information sur la carte.

P 5.9 Reconnaître la technologie

Le processus "Reconnaître la technologie" (P5.9) analyse la réponse au RESET de la carte (**A8**) après avoir reçu une demande de reconnaissance de technologie (**A5.12**). Une fois la technologie reconnue, il transmet son identificateur (**A5.13**) au processus P**5.1**.


P 5.4
Formater les données
P 5.4.1
Analyser et structurer les données
A 5.4.1
Données non factorisées
A 5.7
Données non formatées
P 5.4.2
Factoriser et défactoriser les données
A 5.4.2
Données factorisées et décodées
P 5.4.3
Coder et décoder les données
A 5.8
Données formatées

Guide de présentation
Processus P5.4:Formater les données

Perspective de modélisation:
Objectif: Décrire le formatage des données
Limites:

P5.4.1 Analyser et restructurer les données

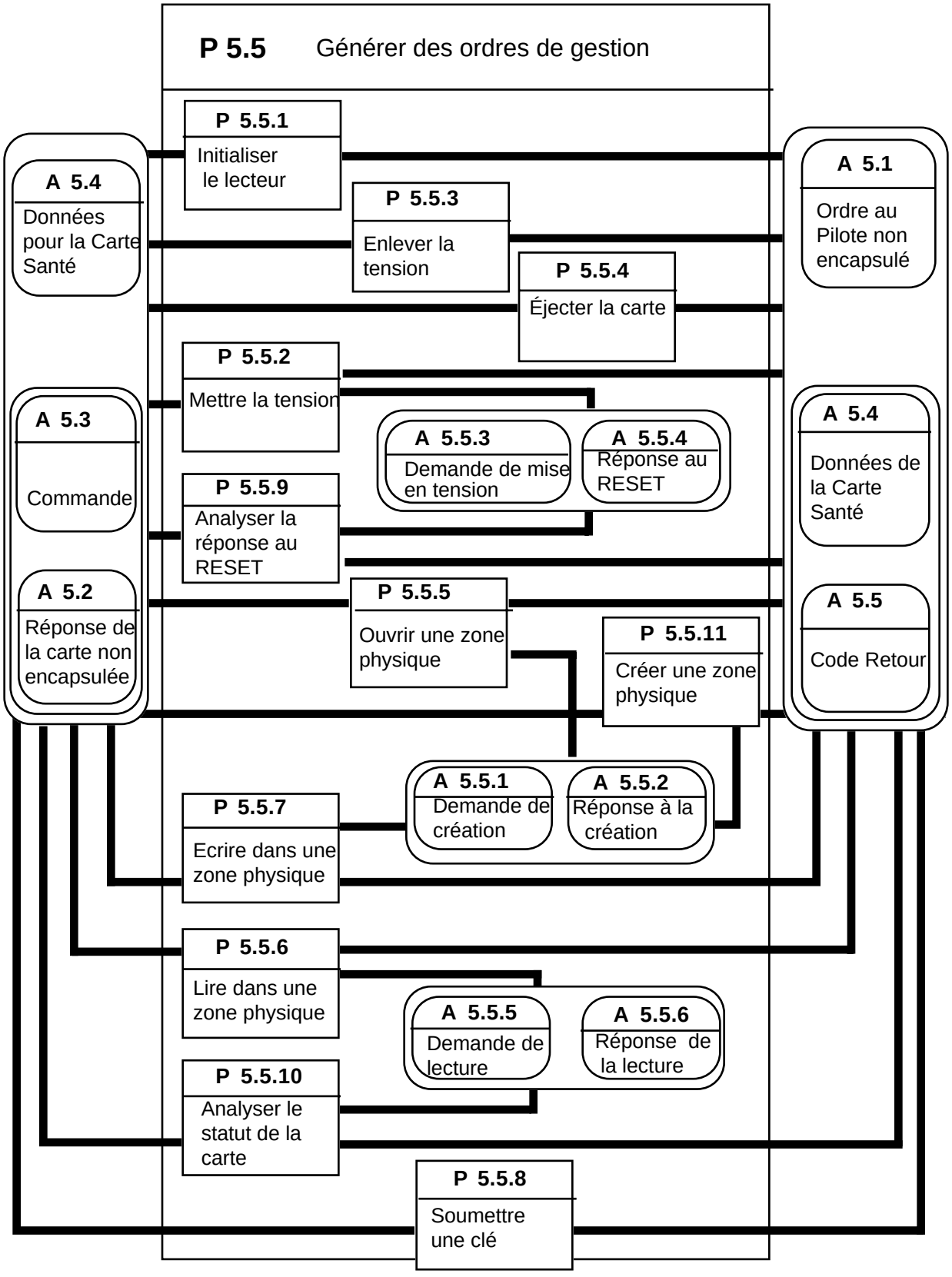
Le processus “Analyser et restructurer les données” (P5.4.1) reçoit des données non formatées (**A5.7**), analyse les champs, sous-champs et infra-champs et restructure les données dans un format pour la carte. Il transmet des données non encore factorisées (**A5.4.1**) au processus **P5.4.2**. De même, il reçoit des données non factorisées (**A5.4.1**), restructure les champs et sous-champs et retourne des données non formatées (**A5.7**) telles que comprises par le processus appelant.

P5.4.2 Factoriser et défactoriser les données

Le processus “Factoriser et défactoriser les données” (P5.4.2) reçoit des données non factorisées (**A5.4.1**), enlève certaines redondances et retourne des données factorisées (**A5.4.2**). De même, il reçoit des données factorisées (**A5.4.2**), rajoute des redondances et retourne des données non factorisées (**A5.4.1**).

P5.4.3 Coder et décoder les données

Le processus “Coder et décoder les données” (P5.4.3) reçoit des données factorisées et non codées (**A5.4.2**), code l’information avec un codage plus restreint (BCD, ISO5, ISO6 ...) selon le contenu de celle-ci et leur type de données et retourne des données formatées (**A5.8**). De même, le processus reçoit des données formatées (**A5.8**), décode l’information et retourne des données décodées (**A5.4.2**).


P 5.5 Générer des ordres de gestion
P 5.5.1
Initialiser le lecteur
P 5.5.3
Enlever la tension
P 5.5.4
Éjecter la carte
P 5.5.2
Mettre la tension
P 5.5.9
Analyser la réponse au RESET
P 5.5.5
Ouvrir une zone physique
P 5.5.11
Créer une zone physique
P 5.5.7
Ecrire dans une zone physique
P 5.5.6
Lire dans une zone physique
P 5.5.10
Analyser le statut de la carte
P 5.5.8
Soumettre une clé
A 5.4
Données pour la Carte Santé
A 5.3
Commande
A 5.2
Réponse de la carte non encapsulée
A 5.1
Ordre au Pilote non encapsulé
A 5.4
Données de la Carte Santé
A 5.5
Code Retour
A 5.5.3
Demande de mise en tension
A 5.5.4
Réponse au RESET
A 5.5.1
Demande de création
A 5.5.2
Réponse à la création
A 5.5.5
Demande de lecture
A 5.5.6
Réponse de la lecture

Guide de présentation
Processus P5.5: Générer les ordres

Perspective de modélisation:
Objectif: Décrire les fonctions de manipulation et d'accès à la Carte Santé
Limites:

P5.5.1 Initialiser le lecteur

Le processus "Initialiser le lecteur" (P5.5.1) reçoit un ordre d'initialisation de lecteur (**A5.6**) et le transmet au lecteur via le processus **P5.6**. Le processus initialise selon la technologie utilisée une zone de travail global.

P5.5.2 Mettre la tension

Le processus "Mettre la tension" (P5.5.2) reçoit un ordre de mise en tension (**A5.6**) ou une demande de mise en tension (**A5.5.3**) et les transmet à la carte (**A5.1**) via l'enCAPSSuleur (**P5.7**). Une réponse au RESET (**A5.5.4**) est retournée au processus **P5.5.9** si celui-ci en a fait la demande.

P5.5.3 Enlever la tension

Le processus "Enlever la tension" (P5.5.3) reçoit un ordre de mise hors-tension (**A5.6**) et le transmet à la carte. L'alimentation électrique de la carte est alors arrêtée.

P5.5.4 Éjecter la carte

Le processus "Éjecter la carte" (P5.5.3) fait éjecter la carte du lecteur après avoir reçu l'ordre d'éjection (**A5.6**). Si la technologie de la carte ne le permet pas, le processus attend le retrait manuel de la carte.

P5.5.5 Ouvrir une zone physique

Le processus "Ouvrir une zone physique" (P5.5.5) est chargé de l'ouverture d'une zone après la réception d'une commande (**A5.6**). Si l'ouverture est en écriture et que la zone n'existe pas, le processus envoie une demande de création de zone (**A5.5.1**) au processus **P5.13** qui lui remet une réponse sur la création (**A5.5.2**).

P5.5.6 Lire dans une zone physique

Le processus "Lire dans une zone physique" (P5.5.6) est chargé de la lecture de l'information de la carte. Il reçoit un ordre de lecture (**A5.6**) ou une demande de lecture (**A5.5.5**) et les transmet à la carte via le processus **P5.7** après avoir vérifié que la lecture soit permise. Les données de la Carte Santé (**A5.7**) sont alors données en retour à l'appelant. Le processus lit toute une zone physique à la fois.

P5.5.7 Écrire dans une zone physique

Le processus “Écrire dans une zone physique” (P5.5.7) est chargé de l’écriture de l’information de la carte. Il reçoit un ordre d’écriture (**A5.6**) et le transmet à la carte avec les données (**A5.7**) via le processus **P5.7**. Si la zone à écrire n’existe pas, le processus transmet uns demande de création de zone (**A5.5.1**) au processus **P5.13**.

P5.5.8 Soumettre une clé

Le processus “Soumettre une clé” (P5.5.8) reçoit un ordre de soumission de clé **(A5.6)** et le transmet à la carte via le processus **P5.7**. Un code retour indiquant la validité de la clé **(A5.8)** est envoyé à l’appelant.

P5.5.9 Analyser la réponse au RESET

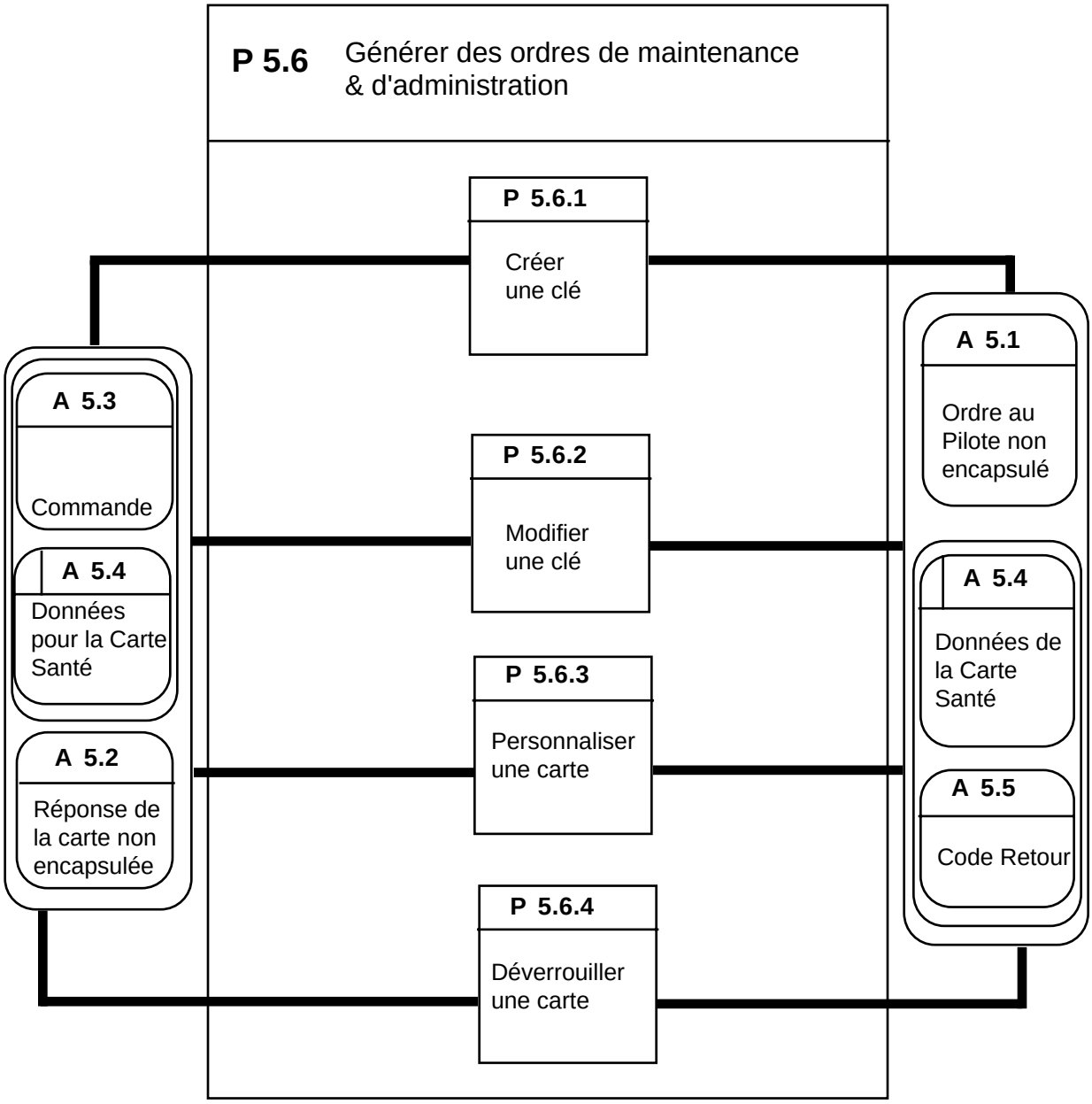
Le processus “Analyser la réponse au RESET” (P5.5.9) reçoit un ordre de remise à zéro de la carte **(A5.6)** et le transmet à la carte via le processus **P5.7**. La réponse au RESET **(A5.2)** est alors analysée pour voir la validité de la carte et identifier la technologie. Les résulats **(A5.7)** sont ensuite transmis au processus **P5.1**. **P5.5.9** peut faire appel au processus **P5.5.2** pour mettre en tension la carte et ceci, en lui transmettant une demande de mise en tension (**A.5.5.3**). Le processus de mise en tension lui renvoie alors une réponse au RESET (**A.5.5.4**).

P5.5.10 Analyser statut de Carte Santé

Le processus “Analyser statut de Carte Santé” (P5.5.9) reçoit un ordre d’identification de la carte (**A5.6**). Après avoir vérifié que la lecture soit permise sur la carte, il retourne le statut du porteur de la carte (bénéficiaire, médecin, pharmacien,...), la capacité totale de la mémoire de la carte ainsi que l’espace encore disponible sur celle-ci après l’avoir calculé (**A5.7**). **P5.5.10** peut faire appel au processus **P5.5.6** en lui transmettant une demande de lecture de zone (**A.5.5.5**).

P5.5.11 Créer une zone physique

Le processus “Créer une zone physique” (P5.5.11) reçoit un ordre de création de zone (**A5.6**) ou une demande de création de zone (**A5.5.1**) et les transmet à la carte via le processus **P5.7**. Un code retour indiquant l’état d’exécution (**A5.8**) est transmis au processus **P5.1** ou une réponse à la demande de création (**A5.5.5**) est transmise à l’appelant.


P 5.6
Générer des ordres de maintenance & d'administration
P 5.6.1
Créer une clé
P 5.6.2
Modifier une clé
P 5.6.3
Personnaliser une carte
P 5.6.4
Déverrouiller une carte
A 5.3
Commande
A 5.4
Données pour la Carte Santé
A 5.2
Réponse de la carte non encapsulée
A 5.1
Ordre au Pilote non encapsulé
A 5.4
Données de la Carte Santé
A 5.5
Code Retour

Guide de présentation
Processus P5.6: Générer les ordres de maintenance et d'administration

Perspective de modélisation:
Objectif: Décrire les fonctions de manipulation et d'accès à la Carte Santé
Limites:

P5.6.1 Créer une clé

Le processus "Créer une clé" (P5.6.1) reçoit un ordre de création de clé (A5.3) et le transmet à la carte via le processus **P5.7** avec la nouvelle clé (**A5.4**). Un code retour indiquant l'état d'exécution (**A5.5**) est transmis à l'appelant.

P5.6.2 Modifier une clé

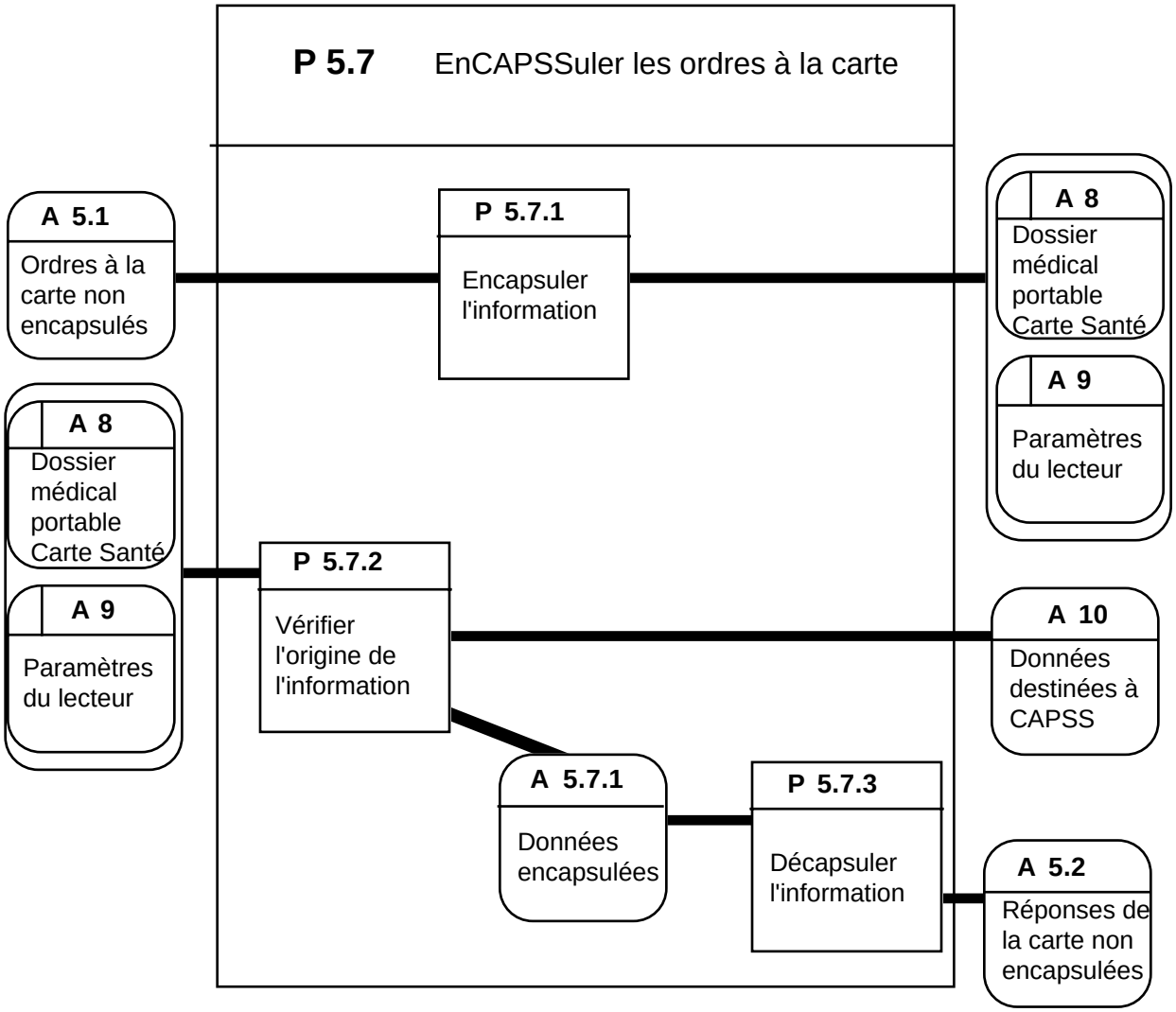
Le processus "Modifier une clé" (P5.6.2) reçoit un ordre de modification de clé (**A5.3**) et le transmet à la carte via le processus **P5.7** avec la nouvelle clé (**A5.4**). Un code retour indiquant l'état d'exécution (**A5.3**) est transmis au processus **P5.1**.

P5.6.3 Personnaliser une carte

Le processus "Personnaliser une carte" (P5.6.3) reçoit un ordre de personnalisation (**A5.3**) et le transmet à la carte. Un code retour indiquant l'état d'exécution (**A5.3**) est transmis au processus appelant.

P5.6.4 Déverrouiller une carte

Le processus "Déverrouiller une carte" (P5.6.4) reçoit un ordre de déverrouillage de carte (**A5.3**) et le transmet à la carte. Un code retour indiquant l'état d'exécution (**A5.3**) est transmis au processus appelant.


P 5.7 EnCAPSSuler les ordres à la carte
A 5.1
Ordres à la carte non encapsulés
P 5.7.1
Encapsuler l'information
A 8
Dossier médical portable Carte Santé
A 9
Paramètres du lecteur
A 8
Dossier médical portable Carte Santé
A 9
Paramètres du lecteur
P 5.7.2
Vérifier l'origine de l'information
A 10
Données destinées à CAPSS
A 5.7.1
Données encapsulées
P 5.7.3
Décapsuler l'information
A 5.2
Réponses de la carte non encapsulées

Guide de présentation
Processus P5.7: EnCAPSSuler les ordres à la carte

Perspective de modélisation:
Objectif: Décrire les fonctions d'encapsulation de l'information avant transmission vers le micro-serveur et après réception du micro-serveur.
Limites:

P5.7.1 Encapsuler l'information

Le processus “Encapsuler l'information” (P5.7.1) reçoit un message destiné à la carte (**A5.1**). Il l'enveloppe avec des caractères de contrôle et le transmet à la carte (**A8** et **A9**) via le micro-serveur.

P5.7.2 Vérifier l'origine de l'information

Le processus “Vérifier l'origine de l'information” (P5.7.2) reçoit un message du micro-serveur et vérifie si ce dernier provient de la carte (**A8** et **A9**). Dans le cas positif, les données (**A5.7.1**) sont transmises au processus **P5.7.3**. Sinon, elles sont acheminées à l'interface de CAPSS (**A10**).

P5.7.3 Décapsuler l'information

Le processus “Décapsuler l'information” (P5.7.3) reçoit des données encapsulées (**A5.7.1**). Il élimine les caractères de contrôle et transmet des données non encapsulées (**A5.2**).

ANNEXE D

Classement des algorithmes par taux de compression

FICHIERS CONTENANT DES BITS ALÉATOIRES

Taille initiale =1 024 octets (Entropie = 7.835)

Algorithmes	Taille résultat	Pourcentage
Codage Arithmétique (1)	1027 octets	+ 0.29 %
LZHUF	1069 octets	+ 4.39 %
PKzip	1134 octets	+ 10.74 %
LZW (9 bits)	1152 octets	+ 12.50 %
Huffman (1)	1326 octets	+ 29.49 %

Taille initiale = 8 192 octets (Entropie = 7.989)

Algorithmes	Taille résultat	Pourcentage
Codage Arithmétique (1)	8223 octets	+ 0.37 %
LZHUF	8297 octets	+ 1.28 %
PKzip	8304 octets	+ 1.36 %
Huffman (1)	8514 octets	+ 3.93 %
LZW (9 bits)	9186 octets	+ 12.13 %

Taille initiale =16 384 octets (Entropie = 7.989)

Algorithmes	Taille résultat	Pourcentage
Codage Arithmétique (1)	15550 octets	- 5.09 %
LZHUF	16497 octets	+ 0.68 %
PKzip	16498 octets	+ 0.69 %
Huffman (1)	16706 octets	+ 1.96 %
LZW (9 bits)	18227 octets	+ 11.24 %

Taille initiale = 65 536 octets (Entropie = 7.989)

Algorithmes	Taille résultat	Pourcentage
LZW (14 bits)	51954 octets	- 20.72 %
Codage Arithmétique (1)	53629 octets	- 18.16 %
PKzip	65650 octets	+ 0.17 %
LZHUF	65674 octets	+ 0.21 %
Huffman (1)	65860 octets	+ 0.49 %

FICHIERS CONTENANT DES DOSSIERS MÉDICAUX PRÉ-CODÉS

Taille initiale =1 024 octets (Entropie = 7.438)

Algorithmes	Taille résultat	Pourcentage
LZHUF	968 octets	- 5.46 %
Codage Arithmétique (0)	1006 octets	- 1.75 %
LZW (9 bits)	1103 octets	+ 7.71 %
PKzip	1134 octets	+ 10.74 %
Huffman (1)	1247 octets	+ 21.77 %

Taille initiale = 8 192 octets (Entropie = 7.675)

Algorithmes	Taille résultat	Pourcentage
LZHUF	6934 octets	- 15.35 %
PKzip	7273 octets	- 11.21 %
Codage Arithmétique (1)	7655 octets	- 6.55 %
Huffman (1)	8222 octets	+ 0.36 %
LZW (9 bits)	8763 octets	+ 6.97 %

Taille initiale =10 874 octets (Entropie = 7.680)

Algorithmes	Taille résultat	Pourcentage
LZHUF	9125 octets	- 15.44 %
PKzip	9592 octets	- 11.78 %
Codage Arithmétique (1)	10048 octets	- 7.59 %
Huffman (1)	10813 octets	- 0.56 %
LZW (9 bits)	11611 octets	+ 6.77 %

FICHIERS CONTENANT DES DOSSIERS MÉDICAUX PRÉ-CODÉS AVEC REDONDANCE

Taille initiale =1 024 octets (Entropie = 7.174)

Algorithmes	Taille résultat	Pourcentage
LZHUF	910 octets	- 11.13 %
Codage Arithmétique (1)	965 octets	- 5.76 %
LZW (9 bits)	1031 octets	+ 0.68 %
PKzip	1095 octets	+ 6.93 %
Huffman (1)	1213 octets	+ 18.45 %

Taille initiale = 8 192 octets (Entropie = 7.599)

Algorithmes	Taille résultat	Pourcentage
LZHUF	6718 octets	- 17.99 %
PKzip	7060 octets	- 13.81 %
Codage Arithmétique (1)	7540 octets	- 7.95 %
Huffman (1)	8143 octets	- 0.59 %
LZW (9 bits)	8669 octets	+ 5.82 %

Taille initiale =12 629 octets (Entropie = 7.638)

Algorithmes	Taille résultat	Pourcentage
LZHUF	10429 octets	- 17.42 %
PKzip	10856 octets	- 14.04 %
Codage Arithmétique (1)	11469 octets	- 9.18 %
Huffman (1)	12437 octets	- 1.52 %
LZW (9 bits)	13448 octets	+ 6.49 %

FICHIERS CONTENANT DES PROGRAMMES EXÉCUTABLES

Taille initiale =1 024 octets (Entropie = 6.542)

Algorithmes	Taille résultat	Pourcentage
LZHUF	787 octets	- 23.14 %
Codage Arithmétique (3)	842 octets	- 17.77 %
PKzip	928 octets	- 9.37 %
LZW (10 bits)	930 octets	- 9.17 %
Huffman (1)	1059 octets	+ 3.41 %

Taille initiale = 8 192 octets (Entropie = 6.870)

Algorithmes	Taille résultat	Pourcentage
LZHUF	5729 octets	- 30.06 %
PKzip	5860 octets	- 28.46 %
Codage Arithmétique (3)	5986 octets	- 26.92 %
Huffman (1)	7378 octets	- 9.93 %
LZW (12 bits)	7471 octets	- 8.80 %

Taille initiale =16 384 octets (Entropie = 7.028)

Algorithmes	Taille résultat	Pourcentage
PKzip	10842 octets	- 33.82 %
LZHUF	10892 octets	- 33.52 %
Codage Arithmétique (1)	13232 octets	- 19.23 %
Huffman (1)	14782 octets	- 9.77 %
LZW (13 bits)	14972 octets	- 8.61 %

Taille initiale = 65 536 octets (Entropie = 6.941)

Algorithmes	Taille résultat	Pourcentage
PKzip	34593 octets	- 47.21 %
LZHUF	35707 octets	- 45.51 %
Codage Arithmétique (1)	44740 octets	- 31.73 %
LZW (14 bits)	53350 octets	- 18.59 %
Huffman (1)	57508 octets	- 12.24 %

FICHIERS CONTENANT DES DOSSIERS MÉDICAUX EN TEXTE LIBRE

Taille initiale =1 024 octets (Entropie = 5.346)

Algorithmes	Taille résultat	Pourcentage
Codage Arithmétique (3)	649 octets	- 36.62 %
LZHUF	684 octets	- 33.20 %
Huffman (1)	776 octets	- 24.21 %
LZW (9 bits)	798 octets	- 22.07 %
PKzip	873 octets	- 14.74 %

Taille initiale = 8 192 octets (Entropie = 5.308)

Algorithmes	Taille résultat	Pourcentage
Codage Arithmétique (3)	3275 octets	- 60.02 %
LZHUF	3579 octets	- 56.31 %
PKzip	3700 octets	- 54.83 %
LZW (11 bits)	5069 octets	- 38.12 %
Huffman (1)	5587 octets	- 31.79 %

Taille initiale =16 384 octets (Entropie = 5.338)

Algorithmes	Taille résultat	Pourcentage
Codage Arithmétique (3)	5932 octets	- 63.77 %
PKzip	6715 octets	- 59.01 %
LZHUF	6963 octets	- 57.50 %
LZW (12 bits)	9460 octets	- 42.26 %
Huffman (1)	11144 octets	- 31.98 %

Taille initiale = 65 536 octets (Entropie = 4.463)

Algorithmes	Taille résultat	Pourcentage
Codage Arithmétique (3)	14458 octets	- 77.93 %
PKzip	15359 octets	- 76.56 %
LZHUF	16366 octets	- 75.02 %
LZW (14 bits)	23672 octets	- 63.87 %
Huffman (1)	36915 octets	- 43.67 %