n°d'ordre 1630

Laboratoire d'Informatique Fondamentale de Lille

50396
1995
529

Thèse

présentée à

L'Université des Sciences et Technologies de Lille

pour l'obtention du titre de

Docteur en Informatique

par

Thierry Peltier

La Carte Blanche : Un nouveau système d'exploitation pour objets nomades

Soutenue le 7 décembre 1995 devant le jury :

Georges Grimonprez, Président.
Régis Beuscart,
William J Caelli, Rapporteurs.
Eric Alzay,
Vincent Cordonnier,
Gilles Goncalves,
Pierre Paradinas, Examinateurs.

UNIVERSITÉ DES SC. ET TECHN. LILLE I
BIBLIOTHÈQUE UNIVERSITAIRE

UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE
U.F.R. d'I.E.E.A. Bât M3. 59655 Villeneuve d'Ascq CEDEX
Tél. 20.43.47.24 Fax. 20.43.65.66

Je remercie Georges Grimonprez qui me fait l'honneur de présider le jury de cette thèse. Je lui suis également très reconnaissant d'avoir ponctué ce travail par ses conseils et de l'avoir examiné avec intérêt.

J'exprime toute ma reconnaissance envers Messieurs William Caelli et Régis Beuscart d'avoir accepter d'examiner cette thèse et de l'attention qu'ils y ont portée.

Je remercie Gilles Goncalves qui me fait l'honneur de participer au jury.

Mes remerciements vont particulièrement à Vincent Cordonnier et à Pierre Paradinas qui m'ont suivi tout au long de ce travail.

J'exprime ma gratitude au C.N.R.S. et à la société Gemplus qui m'ont permis d'effectuer cette thèse par leur co-financement.

Je tiens également à remercier Guillaume Brouard, Olivier Caron, Patrick George, Jean-Marie Place, Patrick Trane, et Pierre-Alain Vast, qui ont, à un moment, participer au projet de Carte Blanche.

Enfin, merci à toute l'équipe de RD2P.

Table des Matières

Chapitre 5 Le Système d'Exploitation pour la Carte Blanche 87

Introduction

Aujourd'hui, l'homme devient de plus en plus nomade, aussi bien pour son travail que pour ses loisirs. La mobilité le prive d'un environnement où il dispose du confort et de tout le matériel nécessaire à ses activités. Il faut donc faciliter sa tâche lors de ses déplacements et lui permettre d'emporter les informations qu'il doit exploiter en tout lieu et à tout moment. Il existe pour cela divers objets nomades, qui suivent un individu et lui rendent différents services aux moments où il en a besoin.

La sécurité des objets nomades est très importante car toutes les informations qu'ils contiennent et tous les services qu'ils rendent, ne concernent pas toutes les activités du porteur. Certaines données sont confidentielles ou présentent un caractère critique. De plus, ces objets peuvent être facilement perdus ou volés et leur contenu doit être protégé. Les cartes à micro-processeur sont des objets nomades qui intègrent pleinement ces considérations de sécurité.

L'engouement que ces objets nomades suscitent, pose un problème d'encombrement. Un individu qui veut disposer de tous les objets nomades ne peut pas les porter tous sur lui. De plus, l'utilisation individuelle de chacun d'eux est simple mais la présence de plusieurs rend moins aisée leur manipulation.

Le regroupement des fonctions de plusieurs objets nomades, la sécurité et d'autres nouveaux besoins, nécessitent l'étude de nouvelles caractéristiques fonctionnelles. C'est principalement l'objectif de l'axe de recherche sur les systèmes d'exploitation pour cartes à micro-processeur, dans lequel s'inscrit le projet de la Carte Blanche. Ce projet nous amène donc à définir un nouvel objet nomade et son système d'exploitation.

Ce mémoire se décompose en six chapitres. Le premier chapitre tente de présenter un panorama des objets nomades actuels et d'en extraire les caractéristiques qu'il serait intéressant de retrouver dans la Carte Blanche.

Le chapitre deux décrit plus particulièrement les cartes à micro-processeur dont la facilité d'utilisation et la sécurité sont les principaux intérêts. Nous abordons d'abord leurs caractéristiques physiques puis nous établissons les grandes classes d'applications pour cartes à micro-processeur. Pour les plus représentatives d'entre elles, leur masque, c'est à dire leur système d'exploitation rudimentaire, est ensuite détaillé. Cette description permet de montrer les spécificités des cartes à micro-processeur.

Ces spécificités mettent en évidence les mécanismes qui confèrent aux cartes leur sécurité. Nous terminons ce chapitre par un bilan des atouts de la carte à micro-processeur à retenir pour la Carte Blanche, et de ses limites qu'il faudra dépasser.

Nous pouvons alors définir dans le troisième chapitre les concepts de la Carte Blanche. Le premier concept est un nouveau modèle de fonctionnement pour objet nomade. Il intègre toutes les réflexions survenues tout au long des chapitres précédents. Il permet à la Carte Blanche de laisser son porteur choisir la ou les applications qu'il veut trouver sur son objet nomade et de les faire évoluer en quantité et en qualité selon ses besoins. Le second concept introduit la notion de partenaire qui permet d'obtenir pour la Carte Blanche une sécurité au moins équivalente à celle de la carte à micro-processeur. Cette notion est le résultat de l'analyse des besoins des différents acteurs autour de la Carte Blanche (le porteur, les émetteurs d'application, ...). Le troisième concept concerne les applications pour la Carte Blanche. Ces applications possèdent leurs propres traitements et peuvent les exécuter dans la Carte Blanche. Cette caractéristique permet également de profiter de la présence de plusieurs applications sur le même support pour que ces applications puissent coopérer entre elles. Enfin, le dernier concept est relatif au porteur. La Carte Blanche est un objet nomade personnel. Elle est anonyme à l'achat, et il peut être nécessaire de la titulariser au nom du porteur à tout moment. D'autre part, le porteur doit être le seul à décider des applications qui fonctionnent sur sa Carte Blanche. Des moyens doivent lui être donnés, en particulier pour contrôler le jeu d'applications dont il a besoin.

Ces concepts de la Carte Blanche requièrent le développement d'un véritable système d'exploitation. Les spécifications de ce système sont données au chapitre quatre. Ce chapitre décrit tous les éléments de base manipulés par le système, autour desquels sont regroupées les notions abordées dans les concepts. Les règles nécessaires à la sécurité d'accès de la Carte Blanche sont également énoncées.

Le cinquième chapitre montre l'implémentation des éléments dans le système d'exploitation et décrit les primitives de manipulation de ces éléments. Les mécanismes d'exécution, de protection logicielle ou matérielle, et de coopération des applications y sont largement détaillés.

Enfin, le chapitre six traite, dans une première partie, de l'élaboration d'un prototype du système d'exploitation qui permet l'évaluation et la démonstration des concepts de la Carte Blanche. Ce prototype est basé sur le cahier des charges que forment les chapitres quatre et cinq. Pour que le prototype soit complètement opérationnel, nous avons développé une chaîne de compilation d'application pour la Carte Blanche, l'application étant écrite en langage évolué. Ce compilateur fait l'objet de la seconde partie du chapitre.

Chapitre 1
Les Objets Nomades

1.1 Introduction

Ce chapitre présente les objets nomades dans leur ensemble pour en extraire les caractéristiques fonctionnelles. D'abord nous allons essayer de donner une définition de l'objet nomade, appellation plus récente que les matériels qu'elle désigne. Ensuite nous abordons les aspects sociaux et juridiques des objets nomades qui jouent un rôle très important quant à leur acceptation par une importante population de porteurs, et quant à l'existence de nombreuses utilisations de ces objets nomades. Enfin nous faisons une tentative d'énumération des objets nomades d'aujourd'hui, et pour chacun d'eux nous dégageons les fonctions que nous devons retrouver dans les objets nomades de demain.

1.2 Définition

Il est très difficile de donner une définition générale d'un objet nomade, tant ce terme peut être employé dans différents domaines. Effectivement, on peut le voir comme un simple petit objet que l'on trouve dans nos poches comme une clef ou une carte d'identité, ou qui accompagne toujours une chose mobile comme la carte grise d'une voiture.

Dans ce document, nous étudions des objets nomades informatiques dans lesquels les objets cités précédemment peuvent apparaître sous une forme informatisée.

Dans le domaine qui nous intéresse, le terme d'objet nomade est très controversé. Certains pensent qu'il est bien adapté car il montre l'aspect itinérant et la disponibilité en tout lieu. Ce terme convient donc bien à un objet que l'on veut emporter partout avec soi et que l'on veut utiliser dès que l'on en a besoin. L'objet nomade doit donc disposer de tous les outils nécessaires pour cela, et les détracteurs du terme le qualifient alors de gadget comprenant tous les outils que l'on prend avec soi «au cas où». Ils pensent également que le terme nomade lui donnent un caractère hasardeux préjudiciable à un appareil de très haute technologie.

L'objet nomade est donc un appareil électronique qui possède les caractéristiques suivantes :

- Sa taille et son poids sont réduits

  Effectivement, pour être nomade, cet objet doit être facilement transporté et manipulé. Il doit également être robuste.

  D'ailleurs, les formats retenus sont du plus grand au plus petit : des formats papiers classiques A4 et A5, jusqu'au format carte de crédit
- Il est utilisable en tout lieu, à tout moment

  Il suit une personne ou un mobile et peut être utilisé sur place. Il s'utilise seul ou en connexion à un système d'information.
- Il dispose de moyens de communication

  L'objet nomade doit parfois se connecter à d'autres systèmes d'information pour communiquer avec le monde extérieur, lui fournir, échanger ou obtenir des services ou de l'information. Du point de vue des systèmes hôtes sur lesquels l'objet nomade va successivement se connecter, les déplacements du porteur et de son objet nomade peuvent être considérés comme un réseau de communication virtuel.
- Il contient de l'information

  L'intérêt de porter sur soi l'objet nomade est qu'il contient des informations dont le porteur peut avoir besoin constamment. Ces informations sont soit personnelles, soit relatives à une fonction applicative de l'objet nomade, voire même confidentielles. La majeure partie de ces informations provient de systèmes extérieurs par les voies de communication de l'objet nomade. D'ailleurs, chaque objet nomade appartient à des systèmes d'informations globaux qui l'environnent.
- Il est doté de capacité de traitement des informations

  L'objet nomade n'a pas pour seules fonctions le chargement, la mémorisation et la restitution des données, il est capable de les traiter au cours des utilisations et d'en créer de nouvelles.

  Il peut faire preuve d'une certaine intelligence dans la communication, le contrôle d'accès aux données et dans l'interface homme-machine.

1.3 Aspects sociaux et juridiques

Les objets nomades n'ont un intérêt que dans une utilisation spatiale et temporelle. En fait, des besoins de nomadisme ont toujours existé mais par manque de moyens appropriées, ils trouvaient des solutions dans les systèmes fixes et les réseaux de communication. Aujourd'hui, les objets nomades contenant leurs propres informations et disposant de leur propre capacité de traitement, peuvent obtenir les mêmes services en se connectant en différents endroits. La banalisation des bornes d'accès pose à l'utilisateur des problèmes de confiance dûs au manque de transparence des actions réalisées. Cette crainte est d'autant plus justifiée que l'objet nomade peut comporter des applications totalement indépendantes et surtout des informations personnelles, voire confidentielles. De plus, comment prouver l'utilisation frauduleuse de l'objet nomade quand tout est électronique ?

Cet aperçu des problèmes montre l'importance des considérations sociales et juridiques des objets nomades que nous décrivons ci-après.

1.3.1 Aspects sociaux

Les objets nomades seront vraisemblablement les outils de l'homme moderne qui vit dans un monde de plus en plus informatisé. Mais cette fois-ci, ils ne sont plus destinés qu'aux seuls informaticiens mais à une très large proportion de la population. Pour cela, l'informatique doit être la plus transparente possible, et les applications doivent être d'un abord facile, et surtout naturel. Maintenant, l'informatique doit s'adapter à Monsieur Tout-Le-Monde et non plus le contraire. De même, la machine ira chercher l'information là où elle se trouve par des réseaux de communication, et cela à la place de l'utilisateur où qu'il soit.

La composante communication de l'objet nomade avec d'autres systèmes d'information est à l'origine de nombreux problèmes. En effet, une personne peut être réticente à utiliser un objet nomade car elle ne sait pas ce qui se passe lors d'une connexion. L'utilisateur ne voit pas les données échangées : les traces de chaque connexion sur l'objet nomade et sur les sites hôtes ne constituent-elles pas un moyen de surveillance de tous ses faits et gestes ? Les données confidentielles ne sont-elles pas divulguées frauduleusement ?

De même, de nouvelles applications apparaîtront uniquement si l'objet nomade peut prouver que la confidentialité, l'intégrité des données et les traitements des applications sont assurés. En effet, la sécurité générale des applications est très importante lors de la connexion sur des sites qui ne correspondent pas à des applications de l'objet nomade.

La disponibilité des objets nomades doués de moyens avancés de communication, permet de toujours contacter le porteur, quel que soit le lieu où il se trouve et ce qu'il fait. Cette caractéristique peut mettre un frein à l'utilisation des objets nomades car elle va à l'encontre du respect de la vie privée. Il faut alors donner à l'utilisateur, la possibilité de «neutraliser» les contacts venant de l'extérieur.

1.3.2 Aspects juridiques : les nouvelles technologies doivent faire leurs preuves

Pour les objets nomades et leur environnement, toutes les informations sont numériques. Quel que soit leur support, magnétique ou mémoire silicium, et leur mode de transmission, comment prouver l'origine ou l'arrivée d'une information «électronique» dans un système juridique où tout est basé sur la preuve reposant sur le papier ?

En effet, la connaissance déjà ancienne du papier, de l'écriture manuscrite et des encres utilisées a permis l'échafaudage de techniques d'évaluation de la véracité d'une preuve. Ainsi, l'authenticité d'une signature manuscrite, la falsification de texte ou l'identité de l'auteur du texte , peuvent être juridiquement prouvées par analyse graphologique .

En informatique, le transfert de données laisse des traces seulement si celle-ci sont explicitement mémorisées durablement sur un support magnétique ou électronique. De plus, la modification d'une

mémoire n'est pas technologiquement détectable. Les informations écrasent simplement les précédentes. Nous devons user de méthodes complémentaires pour déceler toute anomalie. Ainsi, les procédés cryptographiques permettent de protéger l'intégrité, la confidentialité, et même de signer électroniquement des documents numériques. Mais ceux-ci font appel à des raisonnements mathématiques que les experts juridiques doivent bien assimiler. En effet, en cas de litige, ces experts doivent comprendre comment la preuve numérique a été construite pour qu'ils puissent juger de leur degré de fiabilité. De façon plus générale, les nouvelles technologies doivent établir leur système de preuves pour que les experts puissent discerner le vrai du faux.

Maintenant, si ces méthodes ne sont pas utilisées et que l'objet nomade est ouvert sur un vaste réseau de communication, comment prouver l'origine et la véracité d'informations ? En réalité, dans des systèmes très fermés, comme les services de carte bancaire ou de télépéage, les informations collectées peuvent servir et servent de preuves.

Enfin, les objets nomades contiennent des informations personnelles et constituent à eux seuls un fichier nominatif. Ils sont alors réglementés par la C.N.I.L. (Commission Nationale Informatique et Libertés) qui protège également toutes les données informatiques contre les usages abusifs.

1.4 Tentative d'énumération

Nous allons essayer de donner une liste non exhaustive des matériels qui répondent à la définition d'objets nomades ou qui s'en approchent. Le choix n'est pas dû au hasard : nous présentons les objets qui possèdent des fonctions qu'il serait intéressant de retrouver dans l'objet nomade générique de demain.

Nous avons classé les objets nomades par utilisation de la plus générale à la plus particulière. Cependant ce classement n'est pas exclusif puisque différents objets nomades utilisent les mêmes technologies ou peuvent s'interfacer entre eux.

1.4.1 Les téléphones mobiles

Nous trouvons dans cette catégorie les téléphones cellulaires de voiture ou portables, les téléphones à la nouvelle norme GSM (Group System Mobil) [Mou91] ainsi que les bi-bops. La principale fonction du téléphone mobile est de contacter ou d'être contacté par une personne ou un système d'information quel que soit l'endroit où se trouvent le porteur et son correspondant. L'abonnement est attaché à un équipement personnel, dont l'utilisation est souvent limitée à un territoire. Mais le GSM attache l'abonnement à un individu grâce à sa carte SIM (Subscriber Identification Module) [SIM94] personnelle, indépendante de l'équipement téléphonique.

Une autre caractéristique intéressante est la préservation de la composante privée de l'individu. D'une part, le téléphone ne donne pas la localisation du poste appelé. D'autre part, si l'abonné ne veut pas être dérangé, le téléphone peut être désactivé sans qu'il soit considéré en dérangement par le réseau.

Enfin, certains téléphones de voiture possèdent un fonctionnement «mains libres» qui utilise la commande vocale. Cette fonction fait appel au filtrage de bruit ambiant et bien sûr à la reconnaissance vocale. Cette dernière apporte une sécurité au téléphone car une personne non habilitée à l'utiliser ne sera pas reconnue par le système.

1.4.2 Les ordinateurs portables

Cette catégorie comprend les ordinateurs portables, et dans une moindre mesure les ordinateurs de poche. Ils ont à peu près les mêmes fonctions que les ordinateurs de bureau avec l'avantage de pouvoir les transporter avec soi. Cependant, leurs capacités de stockage et de traitement sont plus faibles, quoique cela soit de moins en moins vrai.

Certains portables possèdent leur propre alimentation qui les rend autonome pendant un temps qui reste limité. Ils possèdent des dispositifs de tolérance à l'arrêt brutal d'un traitement (comme par exemple, des systèmes de reprise après panne et des alarmes), et des systèmes d'économie de l'énergie.

L'aspect nomade de ces ordinateurs implique :

- le besoin de relier l'ordinateur nomade à un système d'information. D'ailleurs, l'origine de ces ordinateurs était le besoin de prendre avec soi de l'information se trouvant dans le site central et de pouvoir l'utiliser sur le terrain. Au retour, les informations stockées devaient être restituées dans le site. Mais une même donnée a pu être dupliquée sur plusieurs ordinateurs portables et donc évoluer différemment. La restitution de cette donnée doit rendre compte de toutes les modifications tout en assurant la cohérence des informations du site
- une exposition accrue à la perte, à la destruction et à leur utilisation frauduleuse. Ils ne disposent pourtant d'aucune sécurité particulière pour pallier ces problèmes

Les ordinateurs portables sont évolutifs. Nous entendons par évolutif, le fait qu'ils savent s'adapter aux besoins de leur utilisateur principal : le porteur. Ce dernier peut ajouter des nouvelles applications qui lui sont nécessaires, et en détruire d'autres dont il ne se sert plus. Donc, les programmes applicatifs que les ordinateurs portables contiennent, peuvent être modifiés et supprimés, d'autres peuvent être ajoutés. De plus, ces applications peuvent même être développées sur l'ordinateur.

Les ordinateurs portables fonctionnent sous des systèmes d'exploitation analogues à ceux des ordinateurs de bureau, type MS-DOS.

1.4.3 Les ordinateurs sans clavier

Les ordinateurs sans clavier sont destinés à une utilisation permettant le déplacement aisé. Ils sont équipés d'un stylet permettant d'écrire ou de dessiner sous une forme plus ou moins libre, sur l'écran sensitif de l'ordinateur. Dans certains cas, le doigt peut remplacer le stylet. L'usage des ordinateurs sans clavier est aussi général que celui des ordinateurs portables puisque, à priori, la saisie manuscrite peut remplacer celle au clavier. En effet, un système de reconnaissance d'écriture permet de transfor-

mer les mots manuscrits en leur équivalent ASCII compréhensible par la machine. Mais il s'avère encore peu efficace pour de longues saisies et l'utilisation des ordinateurs sans clavier est restreinte à des applications interactives où de faibles quantités d'informations sont à fournir, le plus souvent sous forme simplifiée (par exemple : l'ordinateur donne un choix avec cases à cocher). Le mode de fonctionnement impose l'autonomie de l'appareil.

Les ordinateurs sans clavier ne disposent pas plus de système de sécurité que les précédents. Toutefois, sur certains d'entre eux, la reconnaissance d'écriture sert d'authentification de l'opérateur. En effet, l'ordinateur peut être conçu pour reconnaître l'écriture de tout le monde moyennant le respect d'une certaine typographie. Sinon, l'utilisation proprement dite de l'ordinateur, commence par une phase d'apprentissage de l'écriture d'un opérateur donné, et dans ce cas, l'authenticité de ce dernier peut être vérifiée.

Ces ordinateurs sont également évolutifs. La différence réside dans le fait que les applications sont obligatoirement chargées dans la machine, et doivent être développées sur des ordinateurs classiques avec des compilateurs croisés. En effet, l'utilisation d'un clavier reste aujourd'hui le meilleur moyen de saisir beaucoup de données de type texte, comme le programme source d'une application.

Des nouveaux systèmes d'exploitation spécifiques aux équipements portables de cette catégorie apparaissent : ce sont PenPoint [CS91] de Go Corporation et le Pen-Windows de Microsoft. Ils offrent aux applications tous les outils nécessaires à la reconnaissance d'écriture, au fonctionnnement multi-processus et à la gestion de l'interface homme/machine graphique (multi-fenêtrage).

1.4.4 Les assistants numériques

Nous classons dans cette catégorie les agendas électroniques, les organiseurs, les P.D.A. (Personal Digital Assistants), les Newtons d'Apple et les Personal Communicators d'EO. Ces objets nomades sont de petits ordinateurs ciblés sur une ou plusieurs applications immuables. Aucune de ces machines ne permet l'ajout de nouvelle application. Ils ne sont donc pas évolutifs.

Les assistants possèdent des fonctions qui conviennent bien à l'utilisation nomade. D'abord, ils sont dotés d'une interface graphique et se commandent au doigt et au stylet. Ensuite, ils ont souvent d'origine des fonctions de téléphonie : ils peuvent recevoir des appels téléphoniques et des télécopies, ainsi que les envoyer grâce à la numérotation automatique; ils peuvent également filtrer les appels. Enfin, le plus innovant se trouve dans les métaphores qui facilitent beaucoup l'utilisation de l'assistant. L'usage d'icônes, d'une nouvelle gestuelle et de messages compréhensibles par la machine sont des métaphores. Les métaphores de type message, en étroite collaboration avec les applications, simulent une forme d'intelligence. En effet, les messages sont interprétés de façon à effectuer l'action associée utilisant une ou plusieurs applications. Par exemple, le message «appeler Bob» va déclencher la recherche dans l'application Répertoire du dénommé Bob, puis la transmission de son numéro de téléphone à l'application Téléphone qui effectuera l'appel. Cet exemple donne une idée de la coopération entre applications.

Les applications sont des outils personnels conçus pour coopérer ensemble. Le porteur va placer dans son assistant des informations qui le concernent ou qui n'intéressent que lui, telles que ses rendez-vous de la semaine dans l'Agenda et des adresses et des numéros de téléphone dans le répertoire. Bien que ces données peuvent avoir un caractère confidentiel, les assistants ne sont toujours pas sécurisés. Ils ne possèdent ni personnalisation nominative du porteur, ni identification et authentification de celui-ci, à l'exception, pour certains, d'une authentification par la reconnaissance d'écriture.

Les assistants numériques fonctionnent pour la plupart sur des systèmes d'exploitation spécifiques souvent dédiés aux applications associées. Mais deux grands éditeurs de logiciels concurrents pensent normaliser les systèmes d'exploitation pour assistants numériques. Ce sont General Magic avec le Magic Cap et Microsoft avec PadWindows. Ces systèmes intègrent le multi-processus, l'interface graphique, la reconnaissance d'écriture, et les métaphores.

1.4.5 Les cartes à micro-processeur

Les cartes à micro-processeur sont des objets nomades qui sont en pleine émergence. Les cartes à micro-processeur sont également dédiées à une ou plusieurs applications. Mais la différence avec les assistants est qu'elles possèdent un cycle de vie. Celui-ci commence par une phase de personnalisation pendant laquelle ces applications sont installées et un porteur (humain ou objet mobile) est associé irrévocablement à la carte par un identifiant, en général le nom, qui est d'ailleurs également écrit sur le support plastique de la carte. Cette phase permet donc un certain choix d'applications. Ensuite, les cartes à micro-processeur basculent définitivement dans la phase d'utilisation pendant laquelle aucune évolution n'est plus possible.

Les cartes à micro-processeur fonctionnent toujours en communication avec un système hôte. La fonction communication des cartes à micro-processeur est donc primordiale. Elles ne sont d'ailleurs pas autonomes puisqu'elles sont alimentées par l'hôte. Leur fonctionnement de type «Plug and Play» est bien adapté aux objets nomades : le système hôte et la carte se reconnaissent puis réalisent automatiquement la ou les fonctions qu'ils ont en commun.

La caractéristique principale des cartes à micro-processeur est sa haute sécurité. En effet, des contrôles d'accès très stricts aux données sont gérés par le logiciel de base de la carte, et il est très difficile, voire impossible de les contourner, même par des procédés physiques. De plus, la sécurité des échanges est renforcée par des encryptages des messages transférés. Enfin, les cartes à micro-processeur ne sont ni duplicables, ni falsifiables et il est très difficile d'en concevoir des fausses essentiellement pour des raisons de coût de fabrication.

Il n'y a pas a proprement parlé de traitement applicatif dans la carte. Les manipulations des données se résument à une simple gestion de fichiers et à quelques fonctions de sécurité, par exemple de type cryptographique.

Toutes ces caractéristiques sont issues du logiciel de base des cartes à micro-processeur, appelé masque, sorte de système d'exploitation rudimentaire et spécialisé, très différent des précédents. Les

cartes à micro-processeur et leur masque sont détaillés dans le chapitre 2 : Les Cartes à Micro-Processeur.

1.4.6 Les cartes PCMCIA

Les cartes au format PCMCIA sont des objets facilement transportables qui deviennent depuis 2 ans des équipements standards pour les ordinateurs portables.

Les premières générations de cartes PCMCIA apportaient à tout ordinateur équipé pour les recevoir , de nouvelles fonctions comme la fourniture d'applications et de mémoires de stockage programmables, ou l'équipement de matériels périphériques comme un fax modem. L'ordinateur accède à leurs fonctions par une interface de communication évoluée. Ces premières cartes ne peuvent prétendre au titre d'objet nomade tel que nous l'avons défini, car il ne possède aucune capacité de traitement. De plus, les accès aux fonctions par l'ordinateur ne sont pas du tout protégés.

Cependant, une nouvelle norme récente prend en considération une plus large variété de cartes, et notamment des cartes PCMCIA comprenant un micro-processeur et des processeurs spécialisés (co-processeurs arithmétique, cryptographique, ...) comme par exemple la **carte PCMCIA Fortessa**. Ces cartes sont alors de véritables objets nomades dont la communication avec l'ordinateur peut être entièrement contrôlée.

Leur format et leur capacité à contenir des équipements performants (micro-processeur, modem), ainsi que leur interface informatique évoluée, font qu'elles seraient sûrement idéales pour héberger les fonctions d'un objet nomade nouvelle génération se situant à la jonction de l'évolution des assistants numériques et de celle des cartes à micro-processeur. En effet, elles pourront, par rapport aux cartes à micro-processeur, contenir des micro-processeurs plus puissants avec des co-processeurs adaptés aux besoins du nomadisme, des capacités de mémoire ROM, EEPROM et RAM plus importantes, permettant d'y héberger un véritable système d'exploitation et des applications aussi nombreuses que variées.

1.4.7 Les jeux électroniques de poche

Les jeux électroniques de poche peuvent être considérés comme des objets nomades. En effet, le jeu est également une activité lors des déplacements. Il est intéressant qu'un objet nomade qui nous suit partout puisse servir de temps à autres à la détente de son porteur. D'ailleurs, bon nombre d'ordinateurs ou d'assistants numériques possèdent plusieurs jeux.

De plus, la fonction jeu, en incitant le porteur à utiliser plus souvent son objet nomade, a un second rôle : l'augmentation de l'activité et de la réceptivité de celui-ci. En effet, lorsqu'il est en période d'activité, l'objet nomade peut être réceptif à d'autres applications et réaliser d'autres fonctions en arrière plan de la fonction jeu, comme par exemple, se situer dans le réseau environnant et consulter sa boîte aux lettres externe.

1.5 Conclusions

Les objets nomades ne sont qu'au début de leur existence. L'étude que nous venons de faire doit servir de guide dans l'élaboration de la prochaine génération. Nous donnons ci-après le résumé des caractéristiques fonctionnelles des objets nomades. Le constructeur d'un nouvel objet nomade pourra y puiser les fonctions nécessaires pour obtenir la finalité recherchée.

L'interface Homme-Machine joue un rôle important dans l'acceptation des objets nomades par une large population de porteurs. Les interfaces innovantes dans ce domaine sont :

- La commande et la synthèse vocales pour un fonctionnement «mains libres»
- Le «Plug and Play» pour une utilisation rapide et efficace
- La manipulation de l'objet nomade comme une feuille de papier : on peut écrire et dessiner sur l'écran avec un stylet, et, pour plus de facilité, le doigt peut remplacer le stylet. Ce type d'interface requiert un module de reconnaissance d'écriture
- Le pilotage aisé de l'objet nomade grâce à l'usage d'icônes et de métaphores qui sont parfois dites «intelligentes»

La connectique est également essentielle. Il est intéressant que l'objet nomade puissent se connecter de différentes manières. Il doit pouvoir communiquer avec des systèmes d'information, notamment avec des micro-ordinateurs et avec d'autres objets nomades. La communication peut être directe, par infrarouge ou par câble, ou encore passer par des réseaux spécialisés ou par celui du téléphone. D'ailleurs, la téléphonie est un moyen habituel de communiquer, ne changeant pas, ainsi, les habitudes de porteur. L'objet nomade peut composer automatiquement un numéro, recevoir des appels, et même émettre et obtenir des télécopies (et cela sans papier !).

Il est souhaitable que les applications d'un objet nomade soient à la fois multiples et évolutives. D'une part, les activités d'un porteur peuvent être nombreuses, et d'autre part, elles évoluent au cours du temps. Un même objet nomade comportant plusieurs applications permettrait au porteur de n'en posséder qu'un. Il faudrait aussi que chaque application puisse être mise à jour indépendamment des autres. De plus, les informations personnelles ou confidentielles étant regroupées, poseraient moins de problèmes de redondance, de mise à jour et de sécurité.

Le développement et la maintenance des applications pour objets nomades, ne s'effectuent pas sur l'objet nomade mais sur des machines mieux adaptées et disposant d'un environnement de programmation.

L'autonomie des objets nomades dépend de leur mode de fonctionnement. Le mode connecté permet d'alimenter l'objet nomade pendant son fonctionnement. L'utilisation indépendante oblige à une autonomie importante (équivalente à plusieurs jours de travail). Dans ce cas, il faut opter pour des systèmes économiseurs d'énergie, comme la mise en veille des composants non utilisés, et pour un détecteur d'usure de batterie.

Enfin, la sécurité des objets nomades actuels est bien trop faible, compte tenu des risques auxquels ils sont exposés. Par exemple, il semble judicieux qu'un objet nomade qui est personnel, comporte les coordonnées de son porteur, ne serait-ce que pour qu'il puisse lui être restitué en cas de perte. Mais le véritable besoin de sécurité se pose au niveau des informations confidentielles et des applications. En cas de perte ou de vol, des personnes non habilitées ne doivent pas pouvoir y accéder. De plus, une application ne doit pas forcément pouvoir accéder aux autres applications, ou alors suivant certaines contraintes. Enfin, l'ouverture des objets nomades sur d'autres systèmes d'information via la communication doit inciter à renforcer cette sécurité. Ceci est d'autant plus vrai avec les communications sans fil, où des transmissions peuvent s'effectuer à l'insu du porteur.

La sécurité des objets nomades nous semble primordiale. C'est pourquoi, nous avons choisi d'étudier plus en détail les cartes à micro-processeur qui offrent un haut degré de sécurité de par leur conception matérielle et la nature des fonctions du logiciel de base encarté.

Chapitre 2
Les Cartes à Micro-Processeur

2.1 Introduction

Les cartes à puce sont de plus en plus répandues dans le monde. Elles sont des objets personnels, utilisées pour les loisirs ou le travail, qui commencent à se trouver dans toutes les mains, aussi bien celles des adultes que celles des enfants. Mais d'un point de vue technique, il faut distinguer les cartes à mémoire, telles les télécartes, des cartes à micro-processeur comme les cartes bancaires.

Les cartes à mémoire ont été inventées et brevetées par le Français Roland MORENO entre 1974 et 1979 sous le nom de «procédé et dispositif de commande électronique» [Guez88]. L'origine de cette idée est la sécurisation des cartes de crédit par une mémorisation plus sûre des données, des fonctions d'identification du demandeur et la commande d'action mécanique du terminal bancaire (distribution de billets par exemple). Ce dispositif pallie les inconvénients des cartes à bande magnétique, à savoir leur faible capacité de mémorisation, leur sensibilité aux champs magnétiques, l'usure due à la lecture (très faible distance entre la tête de lecture et la piste magnétique) mais surtout la facilité de lire ou de modifier les données et de dupliquer ces cartes. Roland MORENO a donc imaginé un circuit intégré contrôlant la sécurité de la mémoire de type EPROM qu'il contient. Ces circuits adaptés sur un support plastique s'appellent des cartes à logique câblée. Ces dernières se sont généralisées avec la télécarte française et non pas pour les cartes de crédit auxquelles elles étaient initialement destinées. Des brevets japonais sur les cartes avec et sans contact ont été également déposés en 1975 et 1976.

Les progrès en matière d'intégration et le réel besoin de sécuriser les cartes bancaires ont permis l'avènement d'une nouvelle génération de cartes à puce en 1981 : les cartes à micro-processeur. Celles-ci offrent une plus grande richesse d'utilisation et une sécurité plus fine.

La deuxième catégorie de cartes à puce dotées d'un micro-processeur, et donc de capacité de traitements, répond en grande partie à la définition de l'objet nomade du chapitre 1. Nous allons présenter dans ce chapitre ses caractéristiques et surtout ses différents systèmes d'exploitation, car la sécurité qu'elles apportent nous semble très importante pour les objets nomades. Nous en extrairons les points forts et les points faibles qui serviront de base à l'étude d'un nouveau système d'exploitation pour objet nomade.

2.2 Description générale

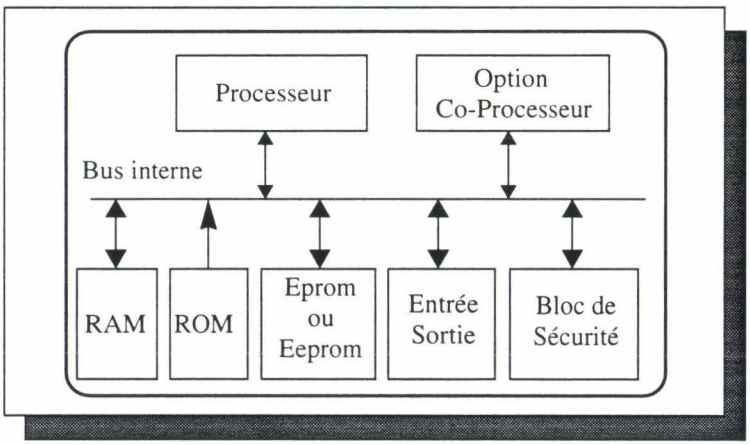
Les cartes à logique câblée et les cartes à micro-processeur ont certaines caractéristiques communes : les caractéristiques physiques, de contact et de communication, qui sont d'ailleurs normalisées par l'OSI (Organisation de Standardisation Internationale). Nous les détaillons dans l'annexe A.

Les autres caractéristiques que nous présentons dans ce paragraphe, sont propres aux cartes à micro-processeur et dépendent essentiellement des constructeurs.

2.2.1 Caractéristiques du micromodule

Le micromodule des cartes à micro-processeur n'est pas seulement un simple micro-processeur mais un véritable et minuscule micro-ordinateur. Autour d'un micro-processeur 8 bits, celui-ci est composé de différents types de mémoire, d'un port d'entrées/sorties et d'un bloc de sécurité. Récemment, il peut également comporter optionnellement un co-processeur.

FIGURE 1 Architecture du micromodule



Les mémoires sont :

- la mémoire morte ou ROM (Read Only Memory) qui contient le logiciel de base de la carte, celui que le micro-processeur exécute dès sa mise sous tension. Son contenu est fixé une fois pour toutes à la fabrication du micromodule
- la mémoire de stockage programmable une seule fois ou (E)PROM (Programmable ROM) dans laquelle sont inscrites les références de la carte et des informations applicatives. La place occupée par une information, même devenue obsolète, ne peut pas être récupérée
- la mémoire de stockage programmable et effaçable ou EEPROM (Electrically Erasable PROM) remplace ou complète parfois l'EPROM. Son avantage est de pouvoir être effacée puis réécrite pour effectuer une mise à jour d'information
- la mémoire vive ou RAM (Random Access Memory) est la mémoire de travail nécessaire au micro-processeur pour exécuter le logiciel de base. Elle sert également de tampon pour les entrées/sorties

Toutes ces mémoires sont de faible capacité du fait de la taille réduite du composant. Par exemple, pour un micromodule récent, le ST16F48 de SGS-Thomson, nous disposons de 16 koctets de ROM, 288 octets de RAM et 8 koctets d'EEPROM.

Le port d'entrées/sorties est très rudimentaire. Il s'agit d'une bascule dans l'espace d'adressage de la mémoire RAM qui est relié à la pastille de contact de la carte. Le protocole de communication est alors entièrement pris en charge par le logiciel de base de la carte, pour la sérialisation et la désérialisation des octets, ainsi que pour le respect du temps de passage d'un bit et le temps d'attente maximum entre deux octets.

Le bloc de sécurité est l'ensemble des dispositifs physiques contrôlant le mode de fonctionnement du micromodule. Nous le détaillerons dans le paragraphe suivant.

De récent micromodule développé conjointement par l'Université Catholique de Louvain La Neuve (Belgique) avec le projet «Corsair» et Philips (carte TB-200 RSA commercialisée), intègre un co-processeur MPU (Multi-Precision Unit), permettant d'exécuter des algorithmes cryptographiques comme le RSA.

2.2.2 Sécurité statique

La carte à micro-processeur contient certainement le micro-ordinateur le plus protégé physiquement. Déjà, de par sa nature, la carte ne possède pas d'interface homme/machine évoluée, la seule interface est sa pastille de contact. D'ailleurs, le contact d'entrées/sorties donne sur le port série du micromodule, par lequel passeront toutes les transactions avec la carte sous le contrôle très strict du système d'exploitation.

Ensuite, au niveau du matériel lui-même, la sécurité s'articule en deux points :

- le micromodule est monolithique et il est difficile de distinguer ses différents composants
- le système d'exploitation est inscrit définitivement en ROM et ne peut donc être ni remplacé, ni altéré. De plus, l'adressage logique de cette mémoire est entrelacé de telle sorte qu'il soit très difficile, voire impossible de la lire avec un microscope électronique

Enfin, le système d'exploitation dispose d'un bloc de sécurité, ensemble de dispositifs physiques renforçant la sécurité lors du fonctionnement de la carte parce que le système d'exploitation les consulte régulièrement. Ce sont :

- un détecteur de tension qui vérifie que la tension est suffisante, notamment pour écrire dans la mémoire (des unités consommées par exemple)
- un détecteur de fréquence qui vérifie qu'on n'abaisse pas la fréquence pour augmenter le délai de garde
- une couche de passivation et un détecteur de lumière qui préservent le micromodule d'une attaque matérielle. Cette couche occulte le composant et si elle est grattée pour faire apparaître la puce de silicium, le détecteur peut alerter le système

Il peut s'ajouter à cela une matrice de sécurité qui régie l'accès aux différents types de mémoire. Par exemple, la mémoire ROM est accessible uniquement en exécution, ce qui interdit

son listage. Par opposition, la mémoire RAM ne peut pas être exécutée. Cela interdit ainsi le chargement d'un programme pirate en RAM puis son exécution.

2.2.3 Les applications des cartes

Les applications des cartes à micro-processeur sont de plus en plus nombreuses mais elles apparaissent le plus souvent dans le classement suivant [Guez88][Bright88] :

- Les cartes de pré-paiement

  C'est la première utilisation des cartes à puce, aussi bien historiquement avec la télécarte que quantitativement pour l'enjeu économique qu'elles représentent. Leurs intérêts sont le paiement facile sans circulation d'argent liquide mais par unités d'achat spécifiques au service que l'on peut obtenir, et la certitude du paiement du service rendu. Bien que ces cartes soient souvent, pour des raisons économiques évidentes (faible coût de production), des cartes à logique câblée et mémoire à fusibles, donc non rechargeables, nous retrouvons ces applications dans les cartes à micro-processeur avec la possibilité supplémentaire de recharger des unités de paiement.
- Les cartes de paiement

  Ce sont les cartes bancaires et les porte-monnaie électroniques. Leur vente arrive en deuxième position après les télécartes. Les sommes d'argent échangées sont plus importantes et leur usage est applicable à toutes gammes de produits. La carte à micro-processeur apporte les éléments nécessaires à l'identification et l'authentification du porteur et ajoute des contrôles plus fins comme le seuil de retrait d'argent liquide. Elle mémorise également toutes les transactions effectuées, ce qui est très important pour un fonctionnement déconnecté du système central.
- Les cartes de contrôle d'accès

  Elles permettent de sécuriser soit un accès physique comme l'entrée dans un bâtiment ou dans une salle, soit un accès logique tel que la connexion à un terminal d'ordinateur.
- Les dossiers portables

  Les applications de type dossier portable commencent à se répandre. Elles consistent à mémoriser des informations concernant à la fois un individu et un domaine particulier. Nous pouvons citer les cartes Etudiant, les cartes santé [Par88] [BP91] et les cartes Ville de plus en plus nombreuses. Les avantages de ces cartes sont l'accès, seulement par des personnes habilitées, à des informations qui restent donc confidentielles, et la facilité d'utilisation pour le porteur.

Cependant, de nouvelles générations d'applications apparaissent. Il s'agit des cartes pour la télévision cryptée et des cartes d'identification SIM (Suscriber Identifier Module) [SIM94] pour les téléphones mobiles GSM (Group System Mobil) [Mou91].

D'autres caractéristiques des applications vont influencer le choix de la carte en fonction de ses aptitudes à protéger l'application. Il s'agit de son espace d'utilisation, de son mode de fonctionnement, de la relation entre l'application et le porteur, et de la nature des données de cette application.

L'espace d'utilisation d'une carte est de deux ordres. La carte est utilisée soit dans un circuit fermé, par exemple dans le cadre d'une entreprise, soit dans un système ouvert tel le système ban-

caire. Dans le premier cas, le système est déjà sécurisé et cela est facile car il est parfaitement déterminé et autonome. La carte n'est valide que dans ce système qui cherchera à la reconnaître mais la carte n'a pas besoin de reconnaître le système. Dans le cas des systèmes ouverts, des cartes compatibles d'origines différentes fonctionnent sur des sites variés. Ce n'est plus seulement au système de reconnaître la carte mais aussi à la carte de vérifier l'authenticité du site. De plus, ces cartes et applications demandent un effort de standardisation.

Lors de la connexion à une borne, la carte est soit en liaison directe avec un site serveur, soit en liaison simple avec la borne. Dans le premier cas et quelle que soit la borne utilisée, le site serveur qui est lui même sécurisé, peut toujours effectuer un certain nombre de vérifications et comptabiliser les opérations effectuées par la carte. Par contre, dans le cas d'une borne isolée et donc plus sensible à la fraude, la sécurité repose essentiellement sur la carte qui pourra prouver sa légitimité à obtenir un service de la borne et qui mémorisera de façon sûre l'opération effectuée. Quel que soit le mode de fonctionnement employé par la carte, nous devons remarquer qu'une application carte est composée de deux parties :

- la carte qui possède et sécurise des données de l'application
- le système hôte qui commande la carte et sécurise les traitements

Ensuite, une carte peut être anonyme ou nominative selon les besoins de l'application. En effet, une application de pré-paiement, comme le téléphone, est utilisée sans faire référence à un titulaire donné. La carte peut changer de porteur au cours de sa vie sans poser de problème au niveau de l'application. Par opposition, une carte bancaire n'est délivrée qu'au titulaire du compte et lui seul peut l'utiliser. Son nom est d'ailleurs inscrit dans la carte. Il faut bien distinguer le titulaire du porteur, car si souvent il s'agit de la même personne, le titulaire peut être une personne morale, comme une société, ou une famille entière, et alors la carte peut être utilisée par différents porteurs légitimes.

Enfin, les données d'une application peuvent être de natures différentes. Nous trouvons des informations immuables, des informations à caractère confidentiel et des informations de type historique. Il est intéressant d'associer des mémoires qui conviennent bien au type de données. Par exemple, de la mémoire EEPROM est utile pour des données historiques réinitialisables après traitement. De même, des données immuables sont inscrites en (E)PROM.

2.3 Les masques des cartes

Le masque d'une carte à micro-processeur est le logiciel de base, sorte de petit système d'exploitation, dont les fonctions sont :

- la gestion des entrées/sorties
- la gestion de la mémoire de stockage de données
- l'exécution des commandes
- la sécurité du matériel et des données

Le masque est inscrit dans la mémoire ROM de la carte. Ses fonctions sont donc définitivement fixées à la fabrication. A la mise sous tension, le micro-processeur exécute le code ROM qui commence par envoyer la réponse à remise à zéro conformément à la norme ISO 7816-3 [ISO3].

L'origine du nom masque provient de la fabrication des composants en silicium, et dans notre cas plus particulièrement de la ROM. Le fabricant de carte fournit à la fonderie un masque qui permet d'obtenir une mémoire ROM contenant un programme et des informations spécifiques.

2.3.1 Masque et système d'exploitation

Les masques des cartes sont très éloignés des véritables systèmes d'exploitation actuels. En effet, les caractéristiques des systèmes classiques que n'ont pas les logiciels de base des cartes sont :

- la gestion de la mémoire vive avec par exemple l'utilisation de mémoire virtuelle, des mécanismes de pagination et de mémoire cache

  La carte ne possède qu'une faible quantité de RAM qui est uniquement utilisée par le masque. Ce dernier utilise des adresses pré-allouées pour son exécution.
- la gestion de nombreux périphériques, notamment les unités de mémoire de masse et les unités d'entrées/sorties

  La carte ne dispose d'aucun périphérique. Les entrées/sorties sont directement gérées par le masque.
- la gestion d'interruptions matérielles et logicielles. La carte n'a qu'une interruption : la remise à zéro
- la gestion des processus avec chargement en mémoire, exécution du programme et libération de la mémoire

  La carte contient un seul programme - le logiciel de base - qui se trouve en permanence en mémoire ROM.

2.3.2 Caractéristiques générales

Dans un groupe de travail commun à la société Gemplus et RD2P, visant à la modélisation des masques des cartes, nous avons défini une grille d'évaluation des masques actuels. Il en ressort quatre grands points qui montrent les caractéristiques générales communes à la plupart des cartes actuelles.

2.3.2.1 Cycle de vie

Le cycle de vie d'une carte à micro-processeur est un ensemble d'étapes que la carte doit franchir et qui contribue à sa sécurité.

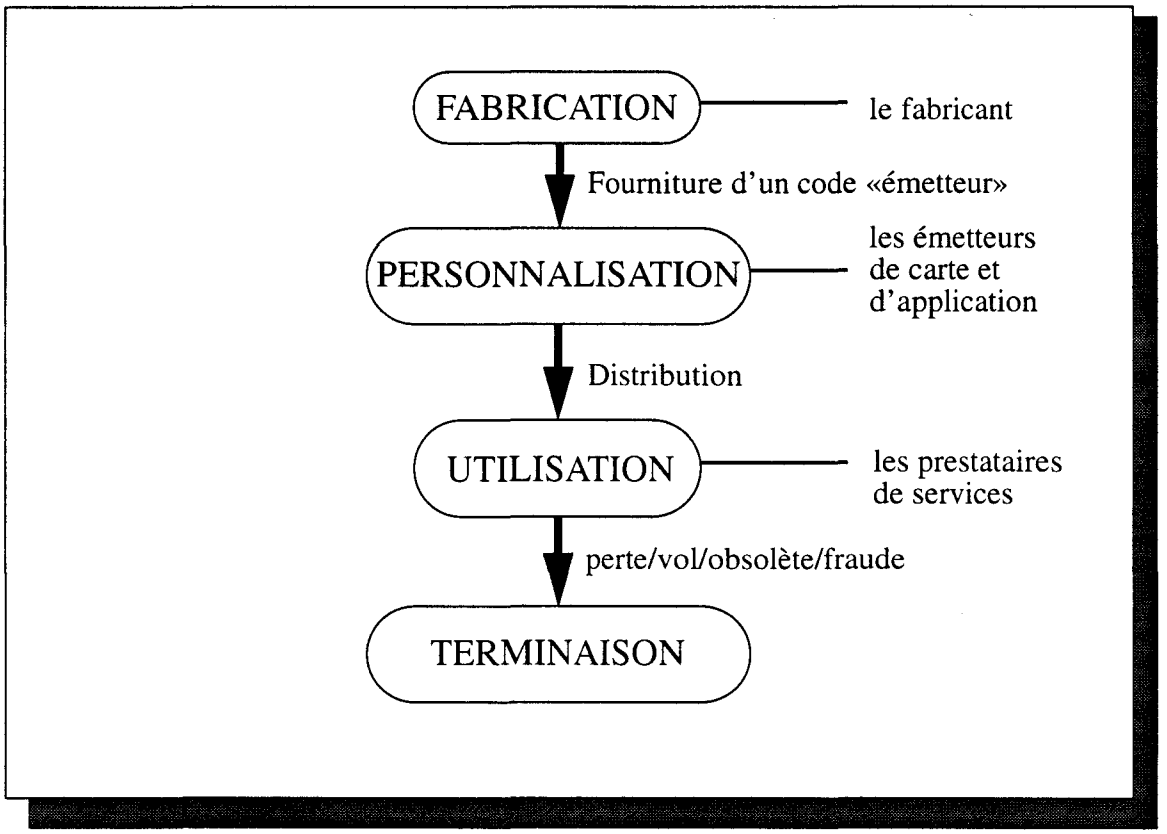
Le schéma classique du cycle de vie d'une carte est donné à la figure 2 de la page 19.

1. **La phase de fabrication** est la première du cycle de vie. Le fabricant de la carte est le concepteur de l'architecture du micromodule et de son masque. La fabrication comprend l'encartage du composant sur le morceau de plastique ainsi qu'une série de tests de conformité aux normes (notamment la résistance aux torsions) et des vérifications fonctionnelles de la puce de silicium. Une fois les tests terminés, la carte est pré-personnalisée. Il s'agit d'inscrire les références du fabricant et du masque en mémoire mais aussi un numéro de série qui distinguera cette carte d'entre toutes les autres.

2. **La phase de personnalisation** : après la fabrication, le fabricant fournit la carte à un émetteur de carte. L'émetteur de carte est l'entité qui a commandé au fabricant un lot de

cartes pour installer une ou plusieurs mêmes applications sur chacune des cartes pour les distribuer ensuite à un ensemble de porteurs. Le fabricant communique un code d'accès à l'émetteur de carte. Ce code lui donne les droits de personnalisation des cartes.

FIGURE 2 Cycle d'une carte à micro-processeur



Lorsque la carte est personnalisée, l'émetteur de carte la remet au porteur spécifié.

3. **La phase d'utilisation** est celle dont le fonctionnement est le plus sécurisé. Pendant cette phase, le porteur va insérer sa carte dans un lecteur ou une borne pour obtenir un service auprès d'un prestataire de services. Un prestataire de services est l'entité qui fournit le service au porteur lorsqu'il utilise sa carte. Il est souvent confondu avec l'émetteur de carte, par exemple, France Telecom est émetteur des télécartes et est prestataire de services via ses cabines publiques. Mais un prestataire de services peut être en contrat avec l'émetteur pour utiliser la carte de ce dernier et vendre ses services, par exemple un commerçant est un prestataire qui utilise le porte-monnaie électronique d'une banque.

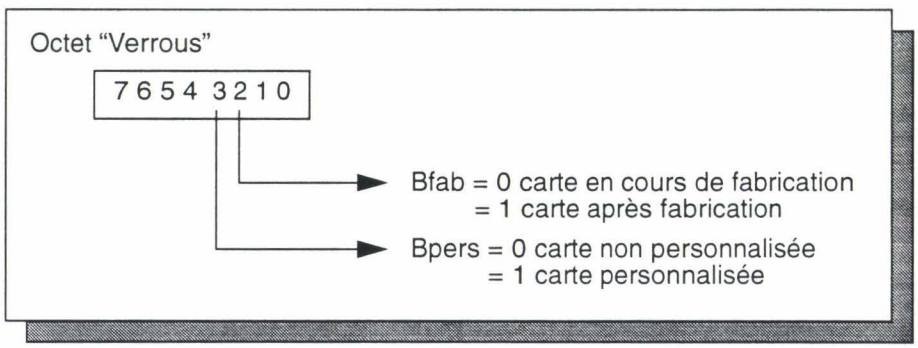
4. **La terminaison** est une phase spécifique au monde des cartes qui est due à son aspect nomade et à sa petite taille. La carte passe dans cette phase définitive lorsque :

- elle est perdue par le porteur ou volée. La terminaison est implicite et immédiate si la carte n'est plus utilisée. Si elle est utilisée par le voleur ou la personne qui l'a retrouvée, la terminaison arrivera par l'un des trois autres cas
- elle est détruite par une utilisation hors norme (flexion, température, tension, U.V., feu, champ magnétique, etc...)

- elle est bloquée par une utilisation frauduleuse (voir ratification)
- elle est saturée, c'est à dire que l'application ne peut plus écrire de nouvelles informations car il n'y a plus de mémoire et l'application ne peut pas en récupérer.

Le passage d'une phase à une autre s'effectue en basculant un bit de la mémoire EPROM de l'état bas à l'état haut. Le retour de ce bit à l'état bas est interdit soit matériellement par l'utilisation de mémoire programmable une seule fois (EPROM), soit logiquement par le masque de la carte. La bascule de ces bits est effectuée explicitement par le fabricant et l'émetteur de carte pour les passages respectifs aux phases de personnalisation et d'utilisation. Par exemple, dans la carte MCOS, il faut écrire dans l'octet "Verrous" pour basculer les bits BFab et BPers qui indique la fin respective des phases de fabrication et de personnalisation (cf. figure 3 de la page 20). Le passage à la phase de terminaison se fait, quant à lui, implicitement par le système d'exploitation et ses mécanismes de sécurité.

FIGURE 3 Indicateurs de phase d'une carte MCOS



2.3.2.2 Stockage de données

La mémoire de stockage ((E)EPROM) de la carte possède une partie réservée au **système** et une partie pour les **applications**.

1. **La partie système** contient :

- les informations inscrites lors de la pré-personnalisation
- des structures de contrôle de l'état de la carte, notamment le code de l'émetteur de carte
- des structures de contrôle du cycle de vie.

A partir de la phase d'utilisation, elle n'est plus accessible de l'extérieur.

2. **La partie application**

Cette partie est constituée du reste de la mémoire (E)EPROM et est disponible pour les applications. Dans les cartes récentes, cet espace est géré à la façon dont MS-DOS gère ses disques, c'est à dire en une arborescence de répertoires et en fichiers. Dans le monde de la carte, les répertoires sont définis pour recevoir des applications indépendantes les unes des autres. Un émetteur d'application peut ainsi stocker dans un même répertoire tous les fichiers concernant son application, et attribuer des droits d'accès à ces fichiers à des utilisateurs propres à ce répertoire. Mais les répertoires servent également à classer des fichiers d'une même application. Dans les répertoire, il existe des **fichiers généraux** dans lesquels un émetteur d'application est libre de mettre toutes sortes d'informations, et des **fichiers spéciaux spécifiques** à chaque type de carte et dont la struc-

ture est significative pour les mécanismes de sécurité (fichiers contenant une clef cryptographique par exemple) ou pour l'application prédéfinie de la carte (fichier de porte-monnaie électronique par exemple). Par exemple, dans le répertoire d'une application bancaire, nous trouvons trois fichiers. Le premier est un fichier général contenant les informations d'un relevé d'identité bancaire (RIB). Le second est un fichier spécial contenant la clef d'identification qui doit être présentée pour modifier le fichier du RIB. Enfin le troisième est un fichier spécifique à l'application qui contient les informations d'un porte-monnaie électronique et qui ne peut être utilisé que par des commandes particulières (cf paragraphe 2.3.2.3 : Langage de commandes).

L'accès aux données s'effectue en plusieurs étapes. Il faut d'abord sélectionner un répertoire qui devient le répertoire courant une ou plusieurs fois la commande de sélection de répertoire. Ensuite un fichier doit être sélectionné dans ce répertoire courant à l'aide de la commande de sélection de fichier, et devient le fichier courant. Enfin, il faut fournir l'adresse dans le fichier (le déplacement par rapport au début du fichier) et de la taille de la donnée avec une fonction d'accès au fichier. En général, les données sont binaires et sont organisées en mots de 4 octets. Les fonctions d'accès sont la lecture, l'écriture de type EPROM (passage des bits de 0 à 1 seulement) et la mise à jour pour les mémoires EEPROM qui est une écriture en deux phases : effacement par mot puis réécriture. Pour écrire un octet dans un fichier, il faut donc lire le mot contenant cet octet et le placer dans un tampon en mémoire de travail, modifier l'octet dans le tampon puis écrire le mot entier dans le fichier.

2.3.2.3 Langage de commandes

Le langage de commandes du masque se décompose en trois parties : les commandes administratives, les commandes opérationnelles et optionnellement les commandes applicatives.

- **Les commandes administratives** sont les commandes servant à la personnalisation de la carte, la création des structures de données (répertoire et fichier) et des structures de sécurité.
- **Les commandes opérationnelles** sont les commandes courantes. Il s'agit de la commande de présentation de code secret, des commandes de sélection d'un répertoire ou d'un fichier et des commandes de lecture, écriture et mise à jour de fichiers.
- **Les commandes applicatives** sont des ordres propres à une application prédéfinie à la fabrication de la carte. Elles pallient l'impossibilité pour la plupart des cartes de mettre du code applicatif lors de la personnalisation. Les cartes qui en possèdent sont donc dédiées à une application.
  Les commandes applicatives sont en général des ordres plus complexes que les commandes opérationnelles et mettent en oeuvre plusieurs opérations élémentaires. Par exemple, l'ordre «débiter» d'une carte dédiée «porte-monnaie électronique» vérifie d'abord que la somme demandée est inférieure à un plafond de retrait autorisé et au solde du compte, puis modifie ce solde en conséquence. Cette opération peut s'accompagner d'une mémorisation de la transaction dans un fichier «historique».

Certaines actions doivent se décomposer en séquences de commandes soit parce qu'elles sont complexes ou à cause de la dualité ordre «entrant» et ordre «sortant» du protocole T=0 (cf Annexe A). Par exemple, une action qui demande des données en entrée et en renvoie d'autres en sortie doit être décomposée en deux commandes consécutives. La première est un ordre entrant qui

fournit des paramètres à la carte. Elle doit être immédiatement suivie d'un ordre sortant qui récupère la réponse.

2.3.2.4 Sécurité dynamique

Le principal avantage des cartes à micro-processeur est la puissance de calcul que le masque met au service de la sécurité. Dans cette section, nous allons montrer comment la sécurité dynamique de la carte est apportée par ce système d'exploitation.

En premier lieu, la sécurité de la carte est obtenue à la mise sous tension par l'exécution incontournable du masque qui ne peut lui-même être altéré (cf. paragraphe 2.2.2 : Sécurité statique). De plus, cette exécution débute toujours par la réponse à remise à zéro qui permet d'identifier précisément et sûrement la carte à travers les octets historiques immuables (cf. Annexe A). En effet, ces octets peuvent donner des indications sur le type physique de la carte, les références du fabricant et de l'émetteur de carte et donc de l'application de la carte. Le contenu des octets historiques fait l'objet de la norme 7816-4 de l'ISO intitulée «Commandes entre industries pour l'inter-opérabilité» [ISO4].

Ensuite, la sécurité est consolidée par l'interface de la carte avec l'extérieur. En effet, toutes les commandes arrivent par le contact d'entrées/sorties entièrement sous le contrôle du système. Celui-ci analyse les ordres et ne les exécute que s'ils sont valides. L'accès direct à la mémoire n'est jamais possible. Il faut utiliser une commande du système qui assure ainsi le respect des droits d'accès aux données.

Le cycle de vie de la carte est un élément important dans la sécurité. Le jeu de commandes utilisable par la carte à un moment donné, varie en fonction de la phase en cours. Les commandes permettant la pré-personnalisation ne sont accessibles qu'à la fabrication. L'accès à la partie système de la mémoire est autorisé pour le paramètrage du système jusqu'à la phase de personnalisation comprise. Enfin, en phase d'utilisation, les commandes restant disponibles sont les commandes opérationnelles et les éventuelles commandes applicatives. Suivant le paramètrage du système, certaines commandes administratives peuvent rester valides, par exemple pour la création de nouveaux fichiers.

La protection des données est assurée en calquant une structure de contrôle sur la structure de données. Chaque fichier possède ses propres droits d'accès qui sont relatifs soit à la carte, soit au répertoire auquel le fichier appartient. Ce principe de protection oblige à éclater une structure de données cohérente en un ensemble de fichiers distincts pour pouvoir attribuer des droits distincts sur des groupes de champs différents de la structure. La manipulation d'un élément complet d'une telle structure oblige à travailler sur plusieurs fichiers.

Pour acquérir des droits, un utilisateur doit présenter un code secret de l'application. La présentation de code secret est soumise à ratification qui protège d'une recherche exhaustive d'un code secret. Le mécanisme de ratification bloque l'usage d'un code secret après un nombre limité (en général 3) de présentations fausses consécutives de ce code. Ce code secret peut être réhabilité sous contrôle de la clef de l'émetteur de carte.

Pour renforcer encore la sécurité des échanges, certaines cartes disposent d'algorithmes cryptographiques. Tout d'abord, elles utilisaient des algorithmes à clef secrète comme le DES (Data Encryption Standard) [NBS80], mais la distribution des clefs parmi toutes les cartes était problé-

matique. Les algorithmes à clef publique [GDQ89] comme le RSA (Rivest Shamir Adelman) [RSA80] sont certainement plus intéressants mais pour l'instant, ils ne peuvent pas remplacer les algorithmes à clef secrète, car ils nécessitent des puissances de calcul non compatibles avec les possibilités des micro-processeurs 8-bits actuellement utilisés. Tous les échanges de données peuvent être chiffrés rendant difficile l'espionnage de la ligne de communication. Les algorithmes cryptographiques permettent également de mettre en oeuvre des signatures électroniques et des authentifications de carte de terminal ou d'utilisateur.

2.3.3 Des masques particuliers

Après avoir dégagé les caractéristiques générales des cartes à micro-processeur, nous allons présenter les particularités de quelques cartes représentatives de l'évolution dans ce domaine.

Nous distinguons deux types de masques, les masques à usage général et les masques spécifiques à une application.

2.3.3.1 Masques à usage général

Les masques à usage général possèdent uniquement des commandes de personnalisation et des commandes opérationnelles. Ces dernières sont des ordres élémentaires comme la présentation de code secret, la sélection et la lecture de fichiers par exemple, qui sont utilisables dans le cadre de toute application.

Ces masques ne comportent aucune fonctionnalité propre à une application. Il n'existe pas de commande applicative, ni de type pré-structuré de donnée mis à part celles qui contribuent à la sécurité du système (comme les fichiers contenant les clefs de chiffrement). Les applications sont entièrement spécifiées lors de la phase de personnalisation.

1. M4 CP8

   La carte CP8 est la première carte à micro-processeur. Son masque initial nommé M1 a évolué jusqu'au masque M4 actuel. Cette carte a été conçue initialement pour répondre aux besoins des cartes bancaires. Cependant, le masque M4 que nous allons décrire maintenant reste d'un usage général.

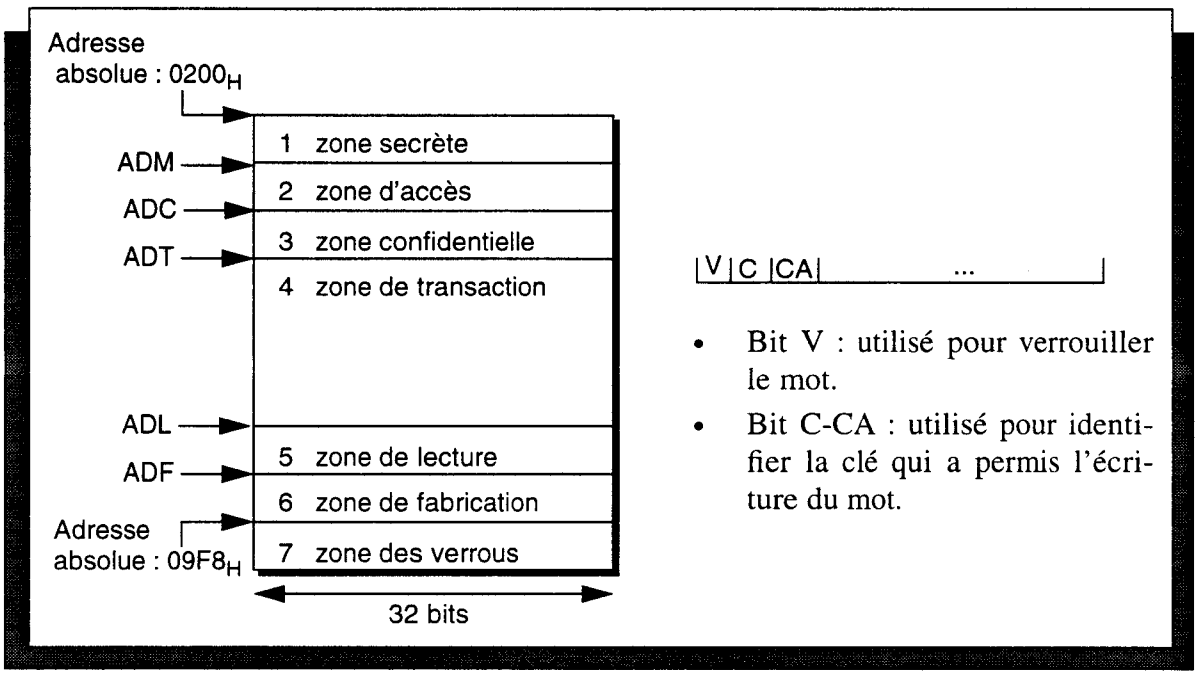
   Le masque M4 gère sa mémoire de données en 7 zones (voir figure 4 de la page 24) possédant chacune des droits d'accès particuliers pour chaque clef d'accès. Ces droits sont également fonction de la phase du cycle de vie de la carte.

   Ces zones sont toutes structurées en mots de 32 bits. La zone de données utile à l'application est la zone de transactions. Dans les mots de cette zone, 29 bits servent réellement aux données, les 3 autres servent au maintien de l'intégrité et de la confidentialité des données (2 bits indiquent la clef sous laquelle a été écrit le mot, et un bit verrouille le mot interdisant ainsi sa réécriture). L'accès aux données s'opère par adressage physique de la mémoire.

   Les clefs utilisateurs sont au nombre de cinq : la clef fabricant, utilisable uniquement en phase de fabrication, deux clefs émetteurs dont une est facultative, et deux clefs porteurs dont la seconde n'est active qu'après invalidation de la première sur demande du porteur.

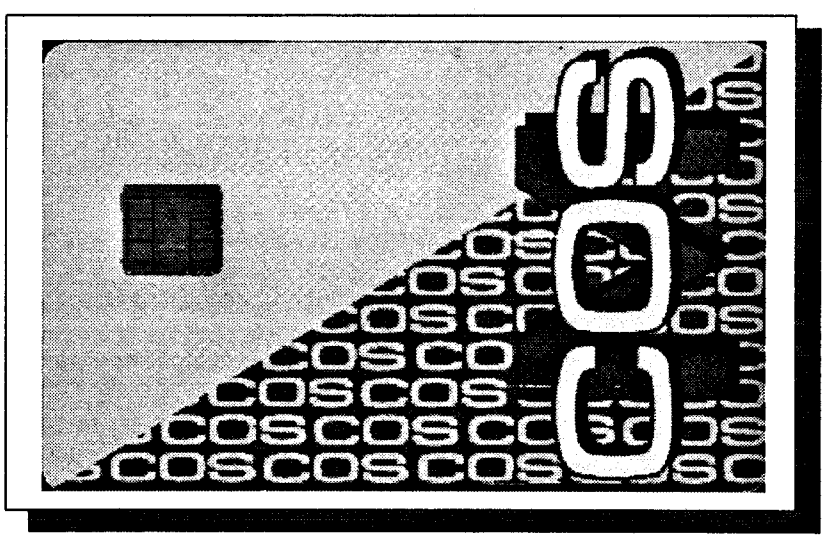
Le masque M4 dispose de la fonction cryptographique TELEPASS [GUQ92]. Il s'agit d'un algorithme à clefs secrètes qui permet de certifier des données issues de la carte.

FIGURE 4 Organisation logique de la mémoire EPROM de la carte CP8



2. COS et MCOS

FIGURE 5 La carte MCOS

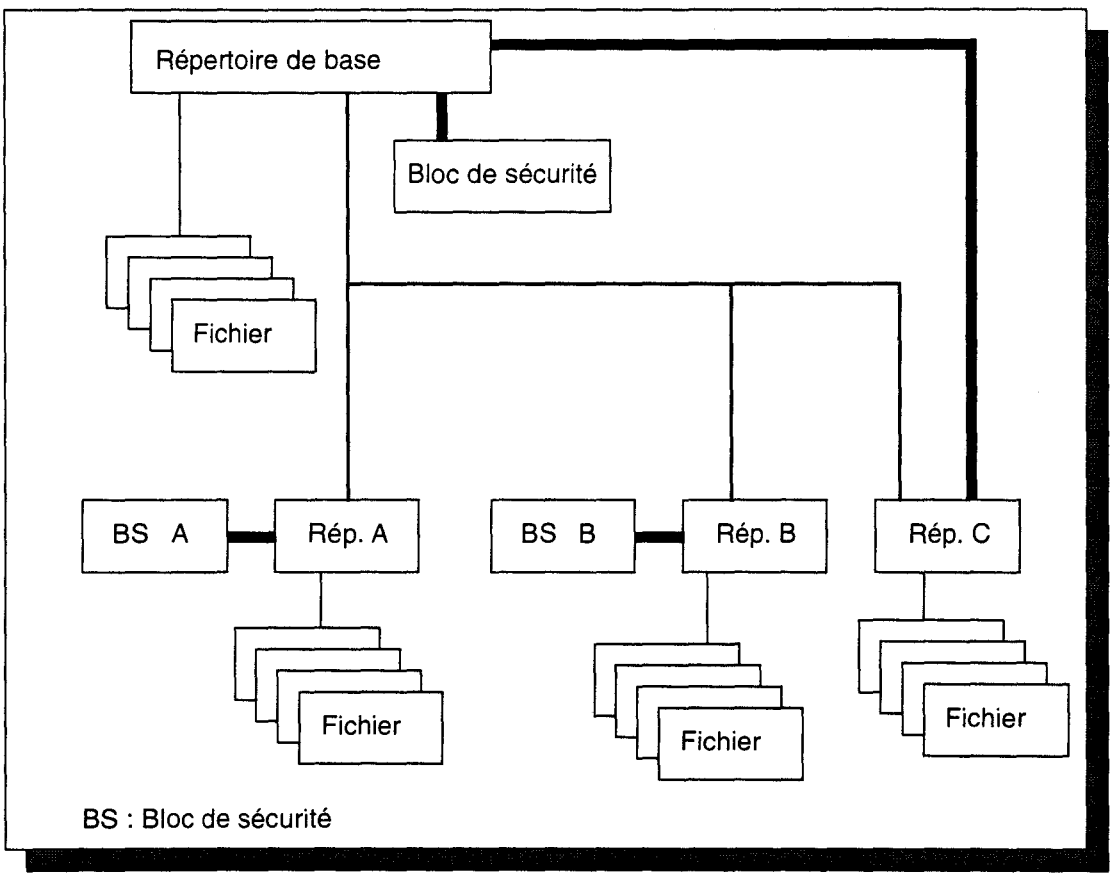


La carte COS (Card Operating System) marque une première évolution dans les masques. D'abord, la mémoire de données est gérée en fichiers indépendants dont le nombre est uniquement limité par la taille de la mémoire, comme sur un disque. L'accès au fichier s'opère par sélection du fichier puis par déplacement dans ce fichier. Une carte COS peut gérer jusqu'à huit codes secrets différents qui servent indifféremment de clef «émetteur de

carte», «porteur» ou «utilisateur». La présentation de ces codes ouvre des droits d'accès à la carte, et en particulier des droits en lecture et écriture sur les fichiers. Un des avantages de cette carte est qu'un même droit peut être obtenu par présentation de plusieurs codes.

La carte MCOS pour Multi COS impose une seconde évolution. Elle ajoute la notion de répertoires qui reproduisent chacun la structure d'une carte COS. Chaque répertoire peut contenir sa propre application avec ses fichiers et ses 8 clefs d'accès (maintenues dans le bloc de sécurité du répertoire). MCOS distingue d'une part, l'émetteur de carte qui peut créer des nouvelles applications et donc des sous-répertoires, et d'autre part, des émetteurs d'application qui organisent les fichiers et leurs 8 utilisateurs. De plus, MCOS est la première carte à avoir intégré de la mémoire EEPROM électriquement reprogrammable. Cela lui permet donc de remettre un fichier à zéro et de le reprogrammer. D'ailleurs, MCOS possède une nouvelle commande : la mise à jour (ou effacement puis écriture) et un nouveau droit d'accès aux fichiers : le droit de mise à jour. Un utilisateur peut alors avoir le droit d'écriture sans avoir le droit de mise à jour et donc d'effacement.

FIGURE 6 Organisation logique de l'espace utilisateur de la carte M.C.O.S.



De plus, dans la carte MCOS, les droits sont soit locaux à une application, dans ce cas, ils sont perdus quand un autre répertoire est sélectionné, soit globaux lorsque ces droits sont acquis jusqu'à la mise hors tension ou la remise à zéro de la carte.

Les fichiers et les répertoires possèdent chacun un descripteur qui occupe de la mémoire. Mais seuls les fichiers réservent de la mémoire à leur création pour leurs données. La mémoire est donc allouée à la demande, quelle que soit l'application concernée.

Les cartes COS et MCOS sont aussi les premières à autoriser l'ajout de **code exécutable, appelé filtre**, dans la carte. Le filtre sert normalement à étendre les fonctions du masque en y ajoutant ou en modifiant des commandes, notamment des commandes applicatives. Le code est écrit dans un fichier qui est ensuite désigné comme filtre en phase de personnalisation. En phase d'utilisation, le masque exécute le filtre juste après la réponse à remise à zéro. Le filtre ne peut donc pas modifier la réponse à remise à zéro, ce qui est important pour la sécurité car elle donne des informations sur l'émetteur de carte qui a installé le filtre. Le filtre peut utiliser les routines du masque, notamment celles d'entrées/sorties. Le fonctionnement du filtre consiste à analyser une commande avant le masque pour détecter une nouvelle commande qu'il exécutera ou une commande existante dont il modifiera le traitement. Sinon il redonne la main au masque initial pour continuer la recherche. Mais le filtre peut entièrement détourner le masque en ne rendant pas la main au masque initial et donc peut contrecarrer la sécurité dynamique (protection des fichiers). Cependant la sécurité physique ne change pas, et le filtre ne pourra jamais lire le code du masque en ROM [Pey90].

Certaines versions des cartes MCOS disposent de l'algorithme cryptographique DES et d'un ensemble de commandes permettant de l'utiliser. Ainsi, les entrées/sorties de la carte peuvent être chiffrées. L'authentification est aussi permise.

3. TB100

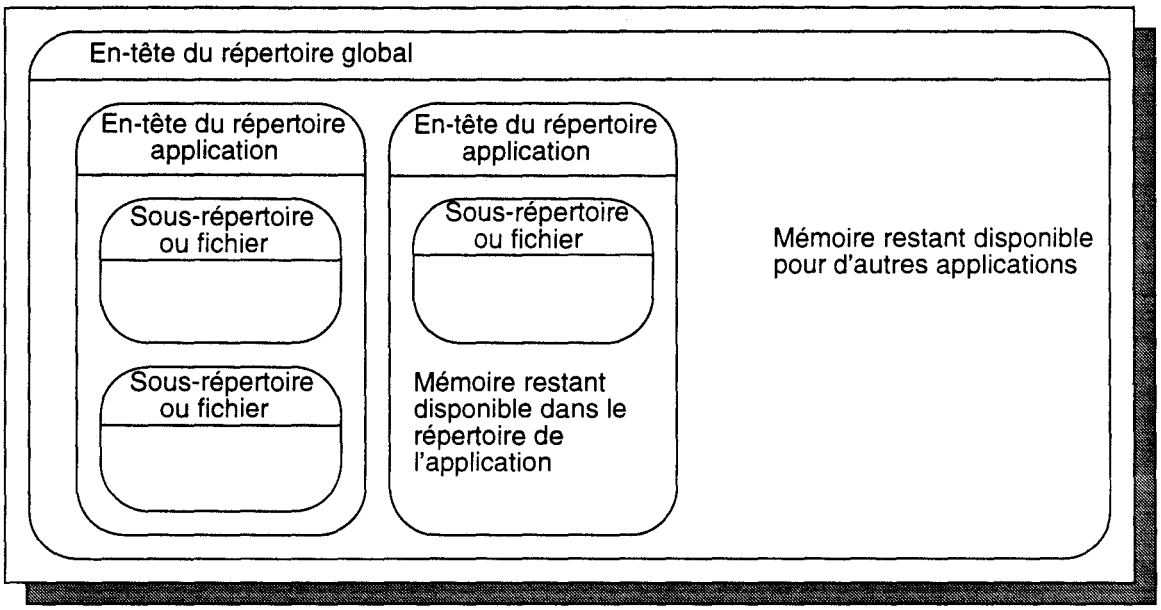
La TB100 est la plus évoluée des cartes Philips. La mémoire de travail est structurée en une arborescence de répertoire à deux niveaux de profondeur. Chaque répertoire doit spécifier sa taille lors de sa création. Les sous-répertoires sont inclus dans la mémoire réservée au répertoire père. De même, les fichiers de données, appelés zones de travail, sont de taille fixée à leur création et occupent la mémoire réservée au répertoire. Le choix d'une taille optimum pour les répertoires et les fichiers, conduit l'émetteur de carte à trouver un compromis entre l'évolution possible de la carte, en laissant possible l'ajout d'une application, et l'évolution de chacune des applications, car celles-ci sont fonction de la mémoire restant disponible à chaque niveau de l'arborescence.

Les répertoires et les fichiers comportent chacun un en-tête comprenant les attributs de sécurité. Les droits de création de sous-répertoire et de fichier, les droits d'accès en lecture, écriture et effacement sont spécifiés dans cet en-tête. Ils font référence à la présentation d'une ou plusieurs clefs utilisateurs.

Il existe de nombreux types de clefs correspondant chacun à un usage particulier. Le nombre de clefs de chaque type est limité. Mises à part les clefs de fabrication et de personnalisation qui ne sont utilisées qu'avant la phase d'utilisation, les clefs sont situées dans des fichiers secrets et agissent dans le répertoire auquel elles appartiennent.

La carte TB100 dispose de nombreuses commandes cryptographiques utilisant différentes clefs de la carte. La présentation de certaines clefs, notamment la clef de l'émetteur de carte, consiste en une authentification cryptographique avec émission d'un nombre aléatoire pour éviter le re-jeu. Par contre, la présentation d'autres clefs, comme celle du porteur, consiste à fournir le code secret qu'est cette clef.

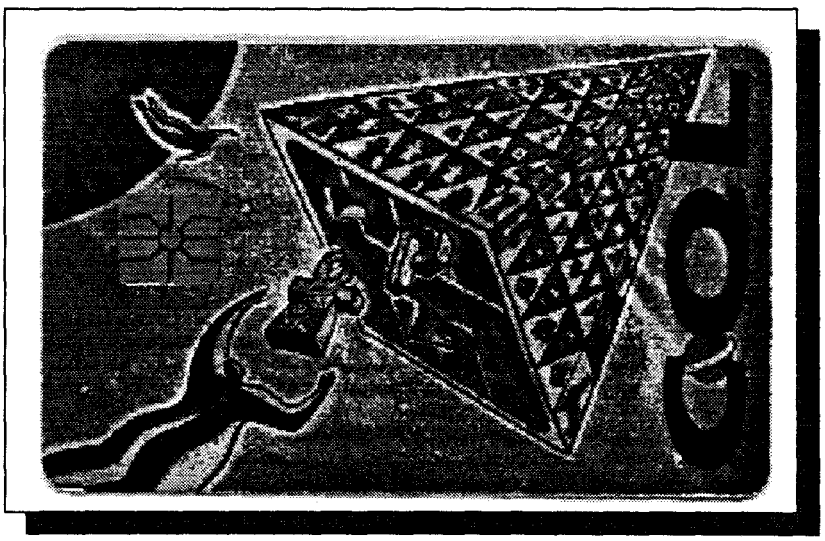
FIGURE 7 Organisation et occupation de la mémoire dans la carte TB100



4. CQL

La carte CQL [Gordon92] [Grimonprez92] a été conçue dans le laboratoire RD2P en 1992 puis réalisée industriellement par la société Gemplus. Elle intègre les concepts des S.G.B.D. (Systèmes de Gestion de Base de Données) relationnelles et un sous-ensemble entièrement compatible du langage standard d'interrogation de base de données SQL (Structured Query Language).

FIGURE 8 La carte CQL



La gestion de sa mémoire de données comme une base de données relationnelle offre de nombreux avantages. Tout d'abord, il n'y a plus de zones système ni de fichiers en mémoire mais seulement des tables. Les tables système contiennent les attributs nécessaires à la sécurité du système et sont accessibles uniquement par le masque. Les autres

tables sont créées au fur et à mesure des besoins des intervenants. Ensuite, la notion de vue sur une table permet d'augmenter la granularité de la protection tout en conservant la solidarité des données appartenant à une même structure.

Les intervenants, au sens de CQL, sont :

- l'émetteur d'applications qui est l'administrateur de la base de données. Son rôle est d'autoriser la venue d'une nouvelle application en créant un nouveau gestionnaire d'application
- les applications : on appelle applications les gestionnaires d'un ensemble de tables de données. Ils sont donc habilités à créer des tables. Ils ont le droit de déclarer de nouveaux utilisateurs et de donner certains privilèges concernant leurs tables, à n'importe quel autre intervenant de la carte
- les utilisateurs qui ne peuvent opérer sur les tables qu'en fonction des privilèges qui leur sont accordés par les applications

Les intervenants sont identifiés par un nom et authentifiés par présentation d'un code secret.

Il existe un intervenant prédéfini PUBLIC qui ne demande pas d'authentification. Cet intervenant est actif quand aucun autre ne s'est présenté. De plus, lui accorder un privilège signifie l'accorder à tous.

Les privilèges sont accordés ou retirés, à tout moment, à un intervenant pour une table ou une vue, et sont :

- la consultation de données
- l'ajout de données
- la suppression de données
- la modification de données

La carte CQL est multi-applicative du fait qu'elle peut gérer plusieurs applications indépendantes entre elles. De plus, elle est évolutive. La taille d'une table du système ou d'une application n'est pas fixée à sa création et peut évoluer au cours du temps. Il est donc toujours possible d'ajouter une nouvelle application, de nouvelles tables et de nouveaux partenaires tant que la mémoire de la carte le permet.

5. ETSI

L'ETSI (European Telecommunications Standards Institute) propose une norme sur les cartes à micro-processeur [ETSI] visant à l'uniformisation des cartes et des terminaux pour la télécommunication.

Cette norme est échelonnée en cinq niveaux. Nous nous situerons au niveau supérieur qui apporte le plus de nouveautés dans le monde de la carte. Mais il n'existe, à ce jour, aucune carte obéissant complètement à un quelconque niveau de cette norme.

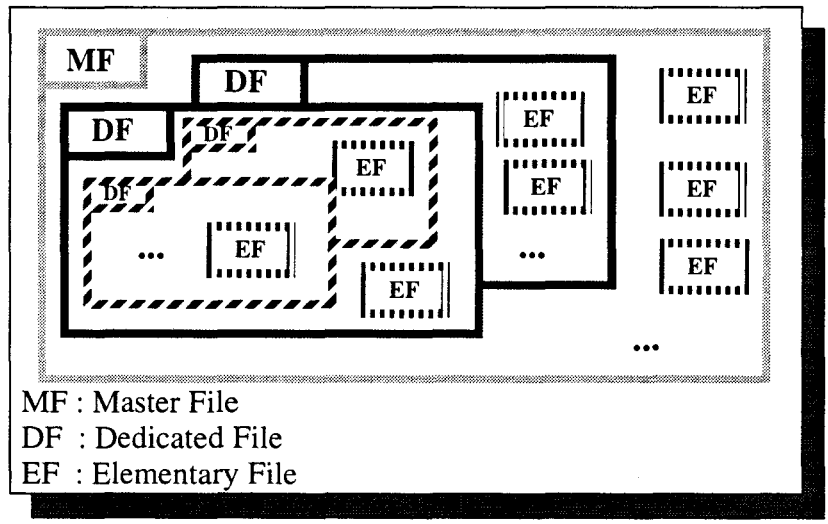
Les grandes lignes de cette norme sont :

- l'espace utilisateur est géré en arborescence de répertoires à niveau de profondeur 3. Le répertoire racine appelé MF (Master File) contient les informations concernant la carte, le fabricant, l'émetteur de carte et les références du porteur. Ses répertoires fils DF (Dedicated File) sont les applications. Ils contiennent la clef de l'émetteur

d'application et celle permettant d'authentifier le porteur dans cette application. Les DF peuvent contenir des sous-applications avec leur propre jeu de clefs et de fichiers

- les fichiers (EF : Elementary File) ne sont plus seulement binaires mais peuvent être, par exemple, structurés en enregistrements séquentiels ou cycliques. Toutes les commandes d'accès propres à ces structures sont spécifiées dans la norme
- il existe des fichiers contenant un programme qui peut être exécuté. Ce fichier est créé en même temps que l'application-répertoire qu'il concerne sous le contrôle de l'émetteur de carte
- la carte possède une fonction d'authentification du terminal auquel elle est connectée
- la carte utilise des algorithmes cryptographiques à clefs secrètes qui permettent l'authentification des différents intervenants et l'accès encodé aux fichiers

FIGURE 9 Organisation logique de la «carte ETSI»



2.3.3.2 Masques spécifiques à une application

Les masques spécifiques à une application sont des masques qui possèdent des caractéristiques et des commandes applicatives. Les commandes opérationnelles sont toujours présentes mais moins nombreuses, en raison des contraintes de taille du composant et en particulier de la mémoire ROM.

Toutefois, nous remarquons que ces masques spécifiques sont souvent obtenus par spécialisation d'un masque à usage général.

1. Banque

   Les cartes bancaires sont des cartes de type M4-CP8 spécialisées. On parle de masque M4-B0 et B0' pour les plus récentes [Guez88]. Ce masque fournit des fonctions propres au service et à la sécurité bancaires.

2. PCOS

   La carte PCOS est une carte Porte-Monnaie électronique. Elle a d'abord été réalisée à partir d'une carte MCOS en lui ajoutant un filtre applicatif, puis ces fonctions ont été fixées à la fabrication. Le filtre ajoute des commandes applicatives comme le débit et le

crédit d'un compte et s'approprie des numéros de type de fichiers MCOS pour y stocker des structures de données relatives au compte, comme un plafond de retrait.

3. SIM

La carte SIM (Subscriber Identifier Module) [SIM94] est utilisée dans les systèmes de radiotéléphone GSM et sert à identifier le numéro d'abonnement du porteur. Cette carte répond en partie aux caractéristiques de la norme ETSI niveau 0.

2.3.4 Modélisation fonctionnelle des masques

La description ci-dessus des masques ne montre pas comment les cartes traversent les étapes du cycle de vie pour arriver à l'objet personnel sécurisé dans les mains d'un porteur. Bien que le cycle de vie d'une carte soit une caractéristique générale à toutes les cartes, les moyens ou méthodes de configuration de la carte sont spécifiques à chaque type de masque. Dans [Cordonnier92], nous proposons un modèle fonctionnel des cartes qui permet de décrire toutes les cartes actuelles selon une même formulation montrant ainsi les particularités de chaque masque et de chaque personnalisation.

En d'autres termes, ce modèle montre la sécurité logicielle apportée par le masque ainsi que la mise en place de cette sécurité par les intervenants sur la carte. La sécurité concerne la protection de la mémoire vis à vis des différents utilisateurs légitimes ou fraudeurs. Cette mémoire est divisée en domaines qui sont des zones, des répertoires, des fichiers ou des tables contenant soit du code, soit des données et auxquels une protection particulière peut être appliquée. La protection est un ensemble de droits d'accès octroyés à tel et tel utilisateur. Lors du processus de personnalisation, les utilisateurs vont accorder à d'autres utilisateurs des privilèges, c'est à dire des droits, qu'ils ont eux-mêmes reçus préalablement. Ces droits ne sont pas explicités dans le modèle car ils sont spécifiques à chaque masque, à l'exception du privilège CREATE nécessaire au modèle.

Le modèle repose sur les 4 règles suivantes :

- un privilège accordé à un partenaire est explicitement relatif à un ou plusieurs domaines
- le fabricant est le premier partenaire à intervenir sur la carte et il est supposé posséder tous les privilèges sur un unique domaine qui est la mémoire totale de la carte
- un partenaire peut donner tout ou partie de ses privilèges sur un domaine à un autre partenaire sur le même domaine ou sur l'une ou plusieurs de ses subdivisions. Il ne peut en aucun cas accorder plus de privilèges qu'il n'en possède, c'est à dire qu'il n'en a reçu
- s'il a reçu le privilège CREATE sur un domaine A, un partenaire peut créer un ou plusieurs domaines à l'intérieur de A. Ce partenaire dispose des mêmes privilèges sur les sous-domaines que sur A. Il peut donc accorder ces privilèges à d'autres partenaires

Une représentation naturelle de ce modèle est un graphe où chaque noeud est un partenaire et où chaque branche montre l'octroi d'un privilège sur un domaine d'un partenaire à un autre. Cependant, une représentation syntaxique nous paraît plus intéressante. D'une part, elle permet d'être plus lisible et plus précise dans la description. D'autre part, cette représentation présente trois avantages d'importance :

- elle facilite la comparaison par ordinateur de deux cartes en vue d'un classement
- elle peut servir à la personnalisation des cartes

- elle peut être soumise à un interpréteur pour une vérification automatique des privilèges

Nous avons donc défini un langage dont la syntaxe est une suite ordonnée de descriptions de partenaires qui montre l'ordre chronologique dans lequel les partenaires personnalisent la carte. La description d'un partenaire se compose de quatre champs :

- le champ Nom du Partenaire
- le champ Actions : il comprend les actions que mène le partenaire pour personnaliser la carte, conformément aux privilèges qu'il possède. Ces actions sont spécifiques à chaque masque.
- le champ Accord de privilèges à d'autres partenaires
- le champ optionnel de commentaire

La syntaxe d'une description de partenaire est la suivante :

```
NOM_DE_PARTENAIRE : {

( [Action sur Domaine [, Action sur Domaine]* ] ) ,

( [ Privilège [, Privilège]* : Domaine [, Domaine]* -> Partenaire [, Partenaire]*

[; Privilège [, Privilège]* : Domaine [, Domaine]* -> Partenaire [, Partenaire]* ]* ] )

[/* commentaire */]

}
```

[...] signifie que l'intérieur des crochets est optionnel.
[...]* signifie que l'intérieur des crochets peut apparaître zéro ou plusieurs fois.
Les caractères **gras** sont les séparateurs de la syntaxe.

Nous présentons l'exemple de la description d'une carte du type le plus répandu : la carte mono-applicative. Cet exemple montre bien la complexité des étapes de personnalisation d'une carte.

```
MANUFACTURER : {
(CREATE D1,D2 in domain),
(EXECUTE D1 -> CARD ISSUER ; CREATE, READ, WRITE D2 -> CARD
ISSUER)
/* D1 est le domaine du code en ROM, le domaine D2 est pour les
données */
}

CARD ISSUER : {
( CREATE DA, DB in D2 ; WRITE DA ) ,
```

```
/* DA est un domaine réservé à l'émetteur de carte où il inscrit
les codes d'accès des partenaires suivants */
( EXECUTE D1 -> APPLICATION ISSUER ; CREATE, READ, WRITE DB ->
APPLICATION ISSUER )
/* Le domaine DB est la zone où l'émetteur d'application pourra
organiser ses données */
}

APPLICATION ISSUER : {
( CREATE DX, DY, DZ in DB ; WRITE DX, DY, DZ) ,
/* L'émetteur d'application crée trois sous-domaines pour les
données */
( EXECUTE D1 -> SERVICE PROVIDERS X, Y, Z ;
READ, WRITE DX, DY -> SERVICE PROVIDER X ;
READ, WRITE DY, DZ -> SERVICE PROVIDER Y ;
READ DX -> SERVICE PROVIDER Z )
}

SERVICE PROVIDER X : { () , () }

SERVICE PROVIDER Y : { () , () }

SERVICE PROVIDER Z : { () , () }
```

Cette description de carte relate les parties du cycle de vie les plus importantes, celles où la sécurité est mise en place : les phases de fabrication et de personnalisation de la carte. La chronologie est respectée puisqu'un partenaire ne peut entreprendre ses actions et accorder ses privilèges qu'après avoir reçu les siens des partenaires précédemment décrits.

La modélisation des cartes fait apparaître clairement les caractéristiques propres à chaque masque de carte. D'une part, les partenaires qui se succèdent donnent un aperçu du type de cycle de vie. D'autre part, les actions et les droits sur les domaines sont spécifiques au type de masque.

Le langage permet de décrire toutes les cartes actuelles. Dans [Cordonnier92], d'autres exemples sont donnés. L'article propose également la description de cartes futuristes qui permettraient le téléchargement temporaire de code exécutable dans la carte et le partage d'informations entre les applications d'une même carte. Le modèle nécessite alors quelques extensions.

Enfin, la modélisation fonctionnelle des cartes à micro-processeur va permettre leur classement suivant un critère particulièrement important pour les cartes à puce : leur cycle de vie et les fonctions visant à la sécurité. Dans la littérature, seules deux classifications des cartes à micro-processeur sont employées. Il s'agit du classement par type d'application (voir paragraphe 2.2.3 : Les applications des cartes) et du classement matériel (type de processeur, qualité et capacité des différentes mémoires). La taxonomie fonctionnelle issue de la modélisation comblera le manque de classement par rapport aux masques.

2.4 Conclusion

Les cartes à micro-processeur sont avec les ordinateurs portables et les notebooks, les précurseurs d'une nouvelle ère informatique, celle de la «nomadique» ou informatique nomade. Bien que toutes ces machines aient été développées en réponse au besoin de mobilité, seules les cartes à micro-processeur permettre de considérer que la mobilité demande beaucoup de sécurité.

Nous voulons conclure par une présentation des éléments des cartes à micro-processeur qui nous semblent importants pour un objet nomade et sa sécurité, mais aussi de ceux qui nous paraissent contraignants dans un usage personnel nécessairement évolutif.

2.4.1 Les atouts

Tout d'abord, nous pouvons remarquer que les cartes à micro-processeur ont été à la fois adoptées par de nombreuses sociétés ou collectivités et acceptées par un large public. Pour les premières, la carte représente un niveau de sécurité sans lequel leur application ne pourrait exister, et ceci pour un faible coût. De plus, la carte étant un produit standard, les applications sont facilement mises en oeuvre et leur intégration sur tout système d'information est aisé, puisqu'il suffit de développer la communication entre le système d'information et la carte à travers un lecteur de carte. Pour le Public, la carte apporte un service supplémentaire qui se caractérise notamment par une plus grande plage horaire d'utilisation et une rapidité du service demandé. Ensuite, elle se transporte facilement et est simple à utiliser : il suffit d'entrer la carte dans un lecteur pour obtenir le service (c'est le Plug and Play).

La sécurité se situe au niveau à la fois du matériel et du logiciel :

La sécurité matérielle est assurée par :

- un composant monolithique qui rend difficile la distinction des différents éléments
- l'entrelacement des adresses de la mémoire ROM pour la rendre inintelligible même à partir d'un microscope électronique
- des détecteurs d'utilisation frauduleuse
- la seule interface d'entrée/sortie est sa pastille de contact et plus particulièrement son contact d'entrée/sortie

La sécurité logicielle est assurée par le cycle de vie de la carte et son masque. Le cycle de vie est composé d'importantes étapes répondant aux besoins de sécurité du nomadisme. D'abord, la phase de fabrication donne les caractéristiques du masque et inscrit des références non altérables dans la carte permettant de la distinguer d'entre toutes les autres. Ensuite la phase de personnalisation fixe définitivement le comportement de la carte vis à vis de l'extérieur. Comme pendant ces deux phases, la carte est encore configurable et est donc pour cela moins sécurisée, ces deux phases sont effectuées dans un milieu protégé. C'est au passage à la phase d'utilisation que la sécurité de la carte, organisée dans les phases précédentes, devient irréversiblement opérationnelle. Enfin la dernière phase est propre au nomadisme où la carte peut être perdue, volée ou détruite. La carte peut détecter l'un de ces cas et se rendre irrévocablement inopérante. Le cycle de vie fait apparaître différents intervenants typiques qui sont le fabricant, l'émetteur de carte, le ou les émetteurs d'application, le porteur et les prestataires de services.

Le masque impose les caractéristiques de la carte. En effet, toutes les entrées/sorties, notamment les commandes, sont sous son contrôle. De plus, le code du masque étant inscrit en mémoire ROM, il est impossible de le modifier et donc de détourner son contrôle. Le masque protège l'accès aux données de la carte par les différents utilisateurs. L'authentification d'un utilisateur s'effectue par présentation d'un code secret ou par des méthodes cryptographiques. Dans le cas d'une présentation de code secret, un mécanisme de ratification permet d'empêcher la recherche exhaustive de ce code secret. Le masque utilise également la cryptographie pour décoder les données en entrée ou coder celles en sortie en vue de contrer l'espionnage de la ligne de communication et rendre les messages confidentiels.

2.4.2 Les insuffisances

Même si les cartes à micro-processeur ont répondu aux besoins des premières applications, elles ont commencé à montrer des insuffisances pour les applications actuelles. Ces applications sont de plus en plus complexes. Elles nécessitent des algorithmes cryptographiques plus performants et ont besoin de traiter des données multimedia [Ale95]. Les cartes actuelles ne peuvent supporter ces caractéristiques et donneraient des temps de réponse trop longs. Les applications futures vont nécessiter des ressources matérielles et logicielles supplémentaires. D'autre part, la rigidité du processus de création d'une carte applicative jusqu'à son arrivée dans les mains d'un porteur doit être remise en cause. En réalité, ces insuffisances, ou plutôt ces limitations, sont le résultat des choix pour la sécurité de la carte, compte tenu des contraintes technologiques. Les normes ISO 7816 qui, pour une partie, en découlent, contribuent au maintien de ces limitations.

Commençons par les limitations dues au matériel :

- La taille des différentes mémoires devient insuffisante pour les applications les plus évoluées. Malgré les progrès espérés en matière d'intégration, la limite des 25 mm2 de la puce de silicium en raison des contraintes de flexion restera un frein à l'évolution des cartes. Ne faudrait-il pas changer de support ?
- Le micro-processeur est de faible puissance. Des fonctions cryptographiques plus performantes, sont appelées à remplacer celles existant sur les cartes, mais le micro-processeur 8 bits ne pourra pas supporter la surcharge de calculs que cela implique. Le projet européen CASCADE, en cours de développement, a pour but de mettre un processeur plus puissant dans une carte [Cas94]
- Les entrées/sorties sont entièrement gérées par le micro-processeur qui passe beaucoup de temps à scruter la ligne, à émettre bit à bit les messages ou à les reconstituer en mémoire. Il serait intéressant de décharger le micro-processeur par une unité d'entrée/sortie qui réalise indépendamment les tâches précédentes. Il permettrait, de plus, des communications plus évoluées et multiples
- La structure physique des cartes les rend sensibles à leur environnement. En particulier, ces cartes fonctionnent mal dans des situations de haute humidité et de haute température. Cela a conduit à la suggestion d'une version «haute humidité/température» de la norme ISO 7816 à Singapour, intitulée 7816-1(S)

Ensuite le masque est un système d'exploitation très rudimentaire, encore une fois limité par la taille de la mémoire ROM. Ses points faibles sont :

- La gestion de l'espace mémoire utilisateur est simple. Toutes ou certaines zones de mémoire sont prédéfinies à la fabrication. Leur taille peut parfois être ajustée à la personnalisation. D'autres zones (par exemple des fichiers) sont allouées séquentiellement dans la mémoire restante. Mais une fois allouée, cette mémoire est irrécupérable, même si la zone n'est plus utilisée. L'évolution d'une application qui demande une zone plus grande, doit invalider la zone précédente et s'allouer une nouvelle zone. Dans les cartes contenant seulement de l'EPROM, la technologie imposait cette limitation. Mais il est dommage que les cartes à EEPROM aient d'abord conservées cette limitation. La carte CQL et la norme ETSI ne l'ont plus
- La granularité de la protection des données au niveau du fichier dans les systèmes à arborescence de fichiers pose des problèmes d'intégrité et de consistance des informations. Une information est un enregistrement logique que l'émetteur d'application est contraint de scinder en plusieurs fichiers pour appliquer des droits différents sur des groupes distincts de champs de son enregistrement. Il semble donc qu'un système qui utilise des tables à la façon des bases de données relationnelles est préférable car il permet, grâce au mécanisme de vue, de conserver l'unité d'une information dans une table et d'offrir des privilèges différents à des sous-ensembles de champs au travers des vues
- Lors de la mise en place d'une nouvelle application dans la carte, l'émetteur définit et initialise la structure de données de son application mais il ne peut pas donner du code qui sera exécuté par le processeur de la carte. Seul le masque peut contenir quelques commandes applicatives qui sont donc inscrites en phase de fabrication. Cependant, la carte MCOS permet d'ajouter du code en phase de personnalisation, qui doit servir à l'extension du jeu de commandes de la carte [Pey91] mais qui peut tout aussi bien passer outre le masque et ses protections d'accès

  Certains traitements d'informations s'effectuent donc à l'extérieur de la carte. Le fait de sortir la donnée de la carte, puis de la ré-écrire, crée une faille dans la sécurité. La donnée peut être copiée par le système hôte ou être interceptée sur la ligne série. Il serait plus important, d'un point de vue sécurité, d'accroître le traitement de données à l'intérieur de la carte, surtout pour des traitements simples ou des données confidentielles
- Chaque masque est spécifique à son constructeur. Un émetteur d'application doit ré-écrire son application s'il veut changer de support. Cette spécificité représente un handicap pour la mise en place d'une application sur des cartes multi-applicatives d'origine diverse dans le futur

La rigidité du cycle de vie rend la carte peu évolutive et plutôt mono-applicative. L'évolution de la carte se situe à deux niveaux.

- Une nouvelle application ne peut être ajoutée que pendant la phase de personnalisation. Même si l'initialisation et l'usage de l'application peuvent être différés, les structures de données sont définies pendant la personnalisation pour des raisons d'organisation de la mémoire
- Une application elle-même ne peut pas évoluer pour les raisons de gestion de la mémoire citées plus haut. Elle ne peut être structurellement mise à jour, ni supprimée physiquement (sauf carte CQL). Une carte dont l'application est obsolète ou doit évoluer, termine son cycle de vie. La vie d'une carte dépend de son application

Les cartes possédant des répertoires sont théoriquement multi-applicatives. Mais, pendant la phase de personnalisation qui est généralement réalisée dans un milieu protégé, les dispositifs de sécurité du masque ne sont pas encore activés. La sécurité de la carte repose seulement sur le contrôle de l'émetteur de carte. Pour que la carte devienne multi-applicative, il faut que :

- soit les émetteurs d'application mettent en place leur application à tour de rôle. Dans ce cas, l'indépendance des applications n'est plus assurée car un émetteur peut connaître et modifier les applications précédemment installées
- soit les émetteurs d'application fournissent indépendamment leur application à l'émetteur de carte qui se charge de la mise en place globale de la carte. Mais ils n'ont aucune certitude de l'impartialité de l'émetteur de carte

En fin de personnalisation, l'émetteur de carte a pour rôle de faire entrer la carte dans sa phase d'exploitation, après laquelle il est impossible de vérifier les droits attribués aux différents partenaires et de les modifier. En conséquence, ces cartes sont composées soit d'applications provenant du même émetteur qui plus est, est l'émetteur de carte, soit de sous-ensembles de la même importante application. Si nous ne pouvons pas dire que ces cartes sont mono-applicatives, nous dirons qu'elles sont mono-émetteurs. Cependant, un autre type de carte multi-applicative, la carte CQL, approche un début de solution à ce problème.

La carte est un objet personnel, attaché à un individu appelé porteur, et pourtant le porteur n'en est pas le propriétaire (celui-ci est l'émetteur de carte). Le plus souvent, il n'a pas non plus d'accès privilégié à sa carte. Il n'a donc aucun droit particulier sur la carte, pas même le droit de consulter les informations le concernant contenues dans sa carte. L'application est installée avant que l'émetteur de carte ne la remette au porteur et celui-ci peut utiliser ses services tant que l'émetteur ne la lui reprend pas. Son rôle est réduit à entrer sa carte dans un lecteur de carte pour obtenir un service.

En résumé, les cartes à micro-processeur actuelles ont besoin de contenir le code exécutable de leurs applications, d'être plus évolutives aussi bien au niveau de la carte (entrée de nouvelles applications et retrait des applications devenues inutiles ou obsolètes), qu'au niveau d'une application (mise à jour) et d'être telle que le porteur de la carte soit un partenaire pouvant s'impliquer dans l'évolution de sa carte. c'est ce à quoi s'est intéressé le projet de Carte Blanche.

Chapitre 3
La Carte Blanche : Les Concepts

3.1 Introduction

Nous avons présenté dans les chapitres précédents, différents objets nomades et plus particulièrement la carte à micro-processeur. Ces matériels ont chacun des caractéristiques fonctionnelles intéressantes pour une utilisation individuelle mobile.

Mais l'engouement pour tous ces équipements conduit à deux difficultés. La première est l'encombrement. Chaque objet nomade est individuellement facilement portable et d'une utilisation confortable. Mais un individu qui voudrait disposer de toutes les fonctions du nomadisme se retrouverait, par exemple, avec un téléphone mobile, un Newton, un ordinateur portable, un radio-messager, des cartes à micro-processeur, etc, ce qui finirait par faire un certain volume.

L'autre difficulté est le choix de l'objet nomade parmi ceux que l'on porte. Un individu qui dispose de plusieurs cartes à micro-processeur mono-applicatives, même si le problème de l'encombrement se pose moins, doit rechercher la carte adéquate pour obtenir le service demandé. En cas de multiplicité de ces cartes, cette recherche peut devenir fastidieuse.

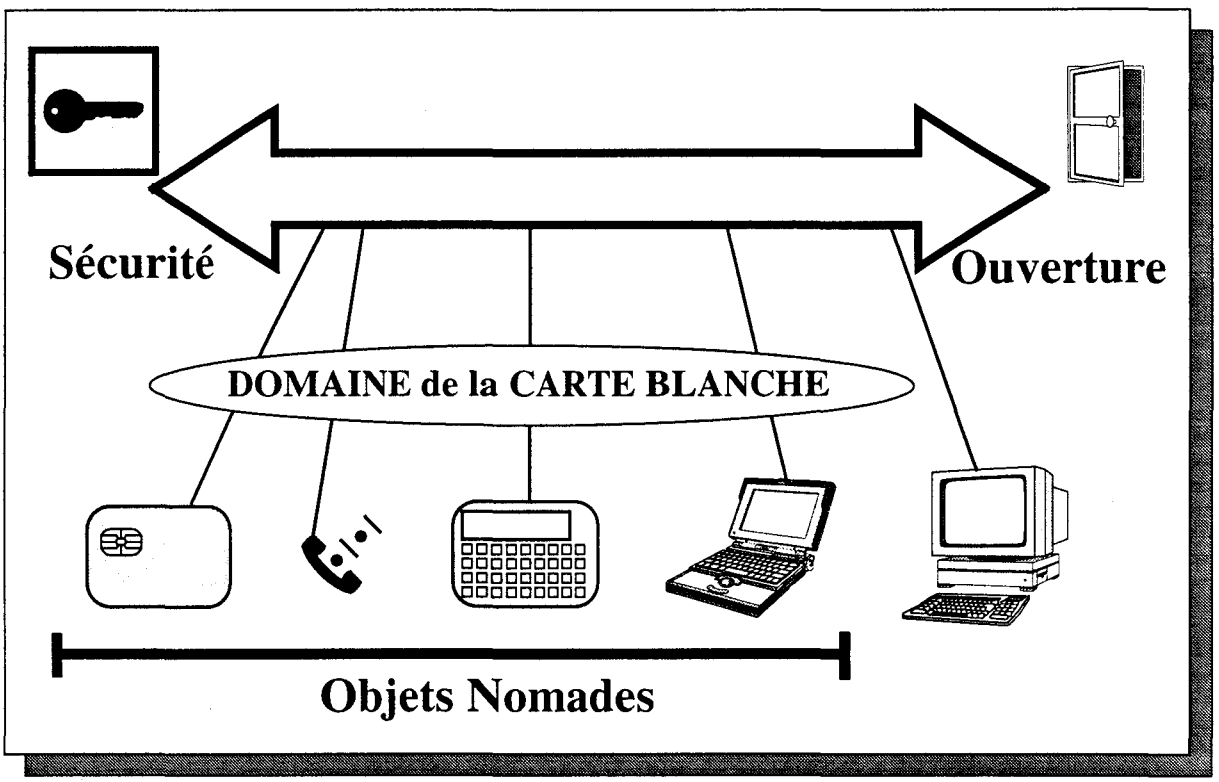
D'autre part, certains objets nomades peuvent nécessiter une utilisation conjointe. Par exemple, beaucoup d'applications nomades peuvent recourir à un paiement de leurs prestations par carte bancaire. La coopération de deux applications sur des supports hétérogènes doit être prévue par avance. La généralisation de lien de coopération entre des applications développées sur des matériels différents et à des périodes distinctes semble impossible.

Le regroupement de différentes fonctions d'objets nomades dans un même matériel nous semble souhaitable, d'une part pour alléger l'encombrement de l'utilisateur mais aussi pour lui offrir une plus grande convivialité de l'objet nomade pour l'obtention des services qu'il demande.

Notre étude nous a également conduits à classer les objets nomades sur une échelle à deux graduations inverses. L'une est le degré de sécurité des objets nomades, l'autre est le degré d'ouverture [PPT95]. La carte à micro-processeur est très sécurisée mais au dépens d'une évolution structurelle, impossible après sa personnalisation. Par contre, l'ordinateur portable, très proche de l'ordinateur de bureau, n'est pas ou peu sécurisé mais il est personnalisable dans le temps.

Pour un objet nomade, il est évident que la sécurité est très importante. Tout d'abord, il contient des informations personnelles, voire confidentielles. D'autres informations sont des données critiques qu'il est impératif de protéger, comme le solde d'un Porte-Monnaie Electronique. Ensuite, il ne bénéficie pas d'un environnement sécurisé comme les locaux où sont placés les ordinateurs de bureau. D'ailleurs, il peut facilement être perdu ou volé. L'objet nomade ne doit pas être utilisable par n'importe qui et il doit se protéger des fraudes. Pour les émetteurs d'application, leur application doit être protégée contre le vol.

FIGURE 10 La Carte Blanche et les objets nomades



Cependant, les nouveaux besoins des cartes à micro-processeur comme ceux du nomadisme en général, requièrent une ouverture plus grande de l'objet nomade aussi bien sur le plan qualitatif que quantitatif. Notamment, les besoins de l'utilisateur pouvant évoluer au cours de sa vie, il est nécessaire que les services rendus par son objet nomade puissent varier au cours du temps.

Les futurs objets nomades devront allier à la fois la sécurité des cartes à micro-processeur et l'ouverture des ordinateurs portables.

Nous proposons donc un nouvel objet nomade, la Carte Blanche, capable de fournir différents services à son utilisateur et d'évoluer selon ses besoins, avec toute la sécurité requise pour un objet nomade. Le domaine de la Carte Blanche doit couvrir les applications des cartes à micro-processeur comme celles des autres objets nomades. Mais la Carte Blanche doit également faciliter l'implantation de nouvelles applications.

Dans ce chapitre, après avoir défini les objectifs de ce travail, nous présentons le modèle de fonctionnement de la Carte Blanche. Ensuite, chaque nouveau concept est développé et argumenté.

3.2 Les Objectifs

L'objectif de ce travail est de définir une nouvelle génération d'Objets Nomades proches des cartes à micro-processeur mais qui s'en distingueront. Ces objets devront montrer une sécurité au moins comparable aux cartes mais ils doivent pallier leurs insuffisances. Par exemple, la spécification d'Eurocard-MasterCard-Visa (EMV) a été créée pour de multiples applications dans les industries de la banque et de la finance. Celle-ci, cependant, envisage encore que les multiples applications sont incorporées au moment de la fabrication et ne sont pas chargées dynamiquement dans la carte. De plus, les applications sont seulement créées pour une unique industrie, en l'occurrence la banque. Pour les nouveaux objets nomades, il faut alors diminuer les contraintes trop fortes au niveau de la personnalisation des cartes. Les objets nomades devront être multi-applicatifs et les applications pourront provenir d'émetteurs indépendants. De plus, aucune application ne peut en général être ajoutée, modifiée ou supprimée physiquement. Ces objets nomades doivent bénéficier d'un cycle de vie plus souple.

Outre les enjeux techniques et économiques d'un nouvel objet nomade, ce cycle de vie doit prendre en compte une composante sociale. Le porteur attitré de l'objet nomade doit être son propriétaire et non plus l'émetteur de carte. Ensuite, ce porteur doit pouvoir choisir les services qu'il voudra trouver sur son objet nomade de même qu'il aurait choisi les matériels distincts dont il avait besoin.

Le peu de traitement des données par les cartes elles-mêmes oblige à sortir les données parfois confidentielles ou critiques. Notre objet nomade doit donc intégrer du code pour chaque application. La présence de plusieurs applications comprenant leur propre code doit permettre la coopération de ces applications à l'intérieur de l'objet nomade.

Pour sa sécurité, notre objet nomade est doté tout d'abord de mécanismes de protection physique et de contrôle d'accès équivalents à ceux employés dans les cartes à micro-processeur. Cependant, le cycle de vie entraîne la disparition du super-émetteur sur lequel reposait beaucoup de privilèges dans les cartes à micro-processeur. D'autre part, les caractéristiques évoluées nécessitent des protections supplémentaires. Le système et les applications doivent être protégés contre les accès par le code exécuté d'une application. De plus, la coopération entre applications doit préserver l'indépendance de chaque application.

3.3 Le Modèle de la Carte Blanche

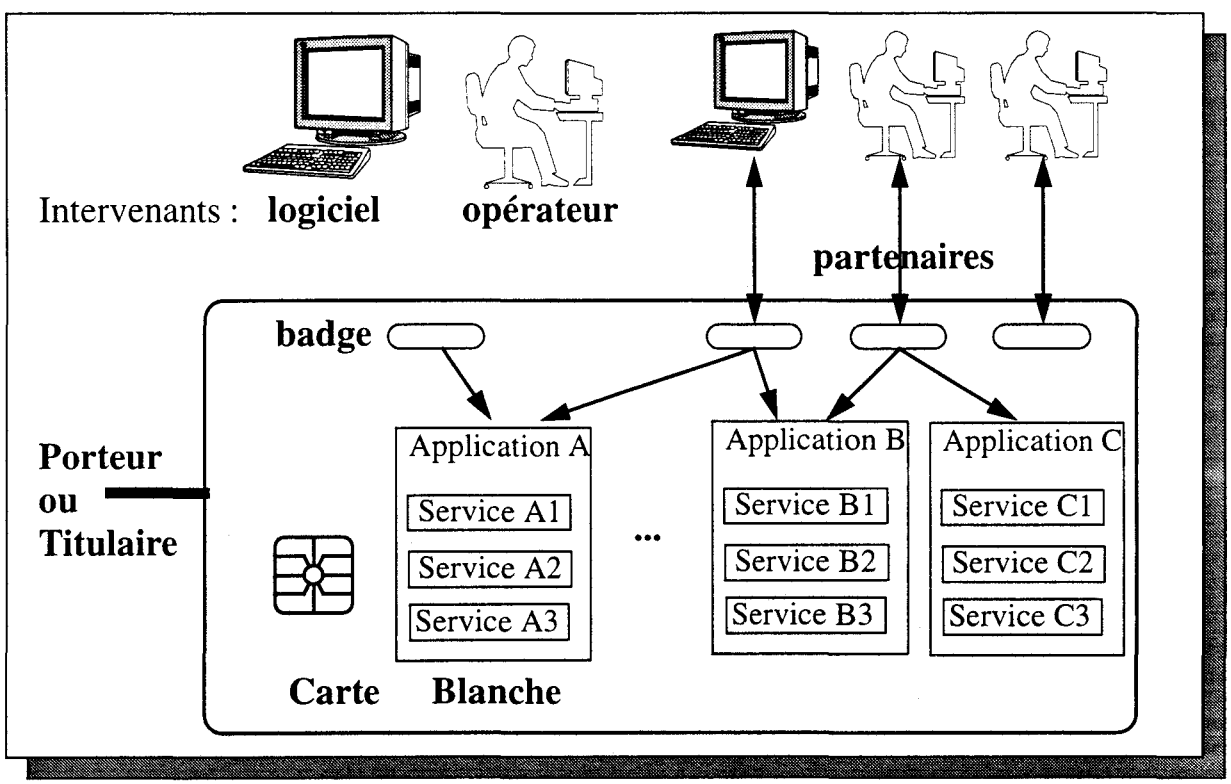
Nous présentons les concepts de la Carte Blanche qui tente de préfigurer les objets nomades de demain. Mais avant d'expliciter les concepts, nous donnons quelques définitions nécessaires à la compréhension des termes employés.

3.3.1 Définitions Préliminaires

Les définitions données dans ce paragraphe sont suffisamment générales pour être comprises individuellement. Les notions qu'elles supportent seront, pour certaines, affinées tout au long du chapitre.

- *Le Porteur*
  Le porteur est la personne ou l'objet qui détient la Carte Blanche à un moment donné. Il peut donc y avoir des porteurs différents mais il n'y a en général qu'un seul porteur principal. C'est grâce au déplacement du porteur que l'objet est nomade. Le porteur est en général à l'initiative de l'utilisation de sa Carte Blanche.
- *Le Titulaire*
  Le titulaire est le porteur principal et légitime de la Carte Blanche. Ce terme n'est employé que s'il existe, dans l'objet nomade, des informations permettant de l'identifier
- *Une Application*
  Une application de la Carte Blanche est un ensemble de moyens logiciels interne à l'objet nomade, capable de réaliser un certain nombre de tâches appelées *services* concourant à la même finalité. La mise en place de ces moyens logiciels s'appelle l'*installation d'application*.

FIGURE 11 Représentation des différentes entités mises en oeuvre dans et autour de la Carte Blanche



- *Un service*
  Un service est l'une des tâches qu'une application peut réaliser individuellement sur demande d'un utilisateur de l'objet nomade.
- *Un Intervenant*
  Un intervenant est une entité externe à la Carte Blanche qui participe à son utilisation. En

effet, les objets nomades sont faits pour communiquer avec l'extérieur et un intervenant est donc un interlocuteur potentiel de la Carte Blanche. Il dispose pour cela des moyens matériels et logiciels pour envoyer et recevoir des commandes et des données. Il existe deux sortes d'intervenants : les opérateurs et les logiciels.

- *Un Opérateur*
  Un opérateur est un *intervenant* humain. Il dialogue avec la Carte Blanche au moyen d'un terminal disposant d'une interface homme-machine. Le porteur peut se comporter en opérateur lorsqu'il désire consulter les informations que contient son objet nomade.
- *Un logiciel applicatif*
  Un logiciel applicatif est un intervenant logiciel, c'est à dire un programme externe à l'objet nomade et qui dialogue automatiquement avec la Carte Blanche et une ou plusieurs de ses applications.
- *Un Emetteur d'Application*
  Un émetteur d'application est un *intervenant* susceptible d'installer une *application* dans la Carte Blanche.
- *Un Partenaire*
  Un partenaire est un intervenant qui possède des droits personnalisés dans la Carte Blanche et qui, pour pouvoir en jouir, doit être reconnu par celle-ci, par l'intermédiaire d'un badge.
- *Un badge*
  Un badge est une structure interne à la Carte Blanche qui, d'une part, permet de reconnaître un intervenant, et d'autre part, contient les droits que celui-ci possède
- *Un Système Hôte*
  Un système hôte est un système d'information qui reconnaît et utilise la Carte Blanche et quelques unes de ses applications. Il héberge des *logiciels*.

3.3.2 Un Modèle de Fonctionnement pour les Objets Nomades

La Carte Blanche est universelle. Sa conception ne donne aucune orientation pour les futures applications que chaque Carte Blanche pourra exploiter. A sa sortie de fabrication, la Carte Blanche ne possède que le logiciel de base servant à la gestion du système et l'unique application système qui constitue l'interface utilisateur du système d'exploitation pour les intervenants. Cette première application rend la Carte Blanche opérationnelle dès sa fabrication.

Ensuite, les Cartes Blanches sont directement mises en vente par les grands réseaux de distribution à destination de leur utilisateur final. Tout le monde peut donc acquérir une Carte Blanche. Sauf pour des raisons commerciales qui consisteraient, par exemple, à pré-installer une application «promotionnelle», la distribution doit être effectuée sans modification du fonctionnement de la Carte Blanche. La distribution de la Carte Blanche doit également être anonyme : le nom du porteur ne doit pas être inscrit dans l'objet nomade. Suivant ce que veut en faire le porteur, sa Carte Blanche peut rester anonyme ou être titularisée dès que le porteur en aura besoin. Tous les porteurs de Carte Blanche à l'issue de la distribution, possèdent un objet nomade fonctionnellement identique.

Cependant, comme tout objet construit en grand nombre, des informations individuelles, comme un numéro de série, sont ajoutées sur chaque Carte Blanche, ne serait-ce que par soucis de

comptage. De plus, elles permettent de distinguer deux Cartes Blanches identiquement vierges ou deux Cartes Blanches dont les applications ne sont pas discriminantes. Pour la sécurité, ces informations sont des éléments sûrs pour retrouver un objet nomade particulier.

La Carte Blanche est multi-applicative. Elle peut contenir plusieurs applications. Les applications peuvent être de tous types au sens où ils sont définis dans le chapitre 2. De même, les applications pourront être d'un nombre quelconque, de complexité et de degré de sécurité différents. Chaque application peut provenir d'émetteurs d'application différents. Les applications d'une Carte Blanche possèdent leurs données, et doivent être capables d'effectuer leurs propres traitements à l'intérieur de l'objet nomade.

Chaque application doit pouvoir fonctionner comme si elle était seule dans la Carte Blanche. Mais pour profiter de la présence des diverses applications sur le même support, elles doivent également pouvoir coopérer entre elles. Une application pourra utiliser les services d'une autre application sans sortir d'information de la carte. Par exemple, le porteur achètera des jetons téléphoniques de l'application «PubliPhone» en payant avec l'application «Banque» ou «Porte-Monnaie Electronique» cohabitant sur la même carte. Cette coopération doit être sécurisée et doit conserver l'indépendance des applications.

La Carte Blanche est évolutive. De nouvelles applications peuvent être installées à tout moment. Ces applications peuvent également être mises à jour, voire même supprimées pour laisser la place à d'autres. Dès qu'une application est installée, elle est de suite utilisable. Ces opérations peuvent se répéter aussi souvent que le porteur le désire, dans la mesure des ressources mémoire disponibles.

La Carte Blanche est ouverte à son porteur. La Carte Blanche ne possède au départ aucune autre application que l'application système. Cette caractéristique est une première explication du nom de Carte Blanche.

Après sa distribution, la Carte Blanche peut évoluer selon les besoins de son porteur. Le porteur choisit chaque application qu'il veut faire installer ou retirer de son objet nomade et le moment où il désire le faire. De même, suivant l'usage qu'il voudra en faire, le porteur pourra définir le degré de sécurité de son objet nomade. Ces caractéristiques sont la deuxième raison du nom du modèle dans lequel le porteur a «Carte Blanche» pour installer et retirer des applications sur son objet nomade personnel.

Le porteur doit avoir un droit de regard sur les applications de sa Carte Blanche. Il doit pouvoir consulter les applications pour obtenir les renseignements qui le concernent directement, par exemple le solde de son compte bancaire. La Carte Blanche doit pour cela être interrogée par le porteur, c'est à dire un opérateur, à partir d'un terminal domestique disposant d'une interface homme-machine.

3.4 Les Partenaires

La Carte Blanche est susceptible d'être utilisée par de nombreux intervenants : les émetteurs d'application et des utilisateurs, en particulier le porteur. Elle doit donc obtenir une sécurité d'accès équivalente à celle des cartes à micro-processeur. D'une part, tous les intervenants d'une

Carte Blanche ne doivent pas pouvoir utiliser toutes les applications de l'objet nomade, ni même tous les services d'une même application. D'autre part, l'accès aux commandes administratives, telles que la commande qui autorise un intervenant à installer une application ou la commande d'installation elle-même, doit être réglementé.

La notion de partenaire répond à ces besoins. Pour avoir accès aux applications auxquelles il a droit, un intervenant doit être reconnu comme étant un partenaire de la Carte Blanche. Grâce à la notion de partenaire, les commandes administratives, accessibles uniquement à un super-émetteur dans les cartes à micro-processeur, reviennent de droit, pour certaines, au porteur qui définit les besoins de sa Carte Blanche, tandis que d'autres peuvent être attribuées à des émetteurs qui installent une application. Les droits d'accès à ces commandes seront donc répartis entre les différentes classes d'intervenants auxquelles ils se rapportent.

Nous allons décrire comment le porteur et les émetteurs d'application qu'il choisit peuvent devenir partenaire de la Carte Blanche alors qu'ils ne sont pas et ne doivent pas être connus par avance. Nous définirons les différentes classes d'intervenants et les droits administratifs dédiés aux partenaires correspondant à ces classes.

3.4.1 La sécurité d'accès

La sécurité d'accès à la Carte Blanche est obtenue au moyen des badges. Un badge permet, d'une part, de reconnaître un intervenant, et d'autre part, de vérifier les droits que cet intervenant possède. La reconnaissance d'un intervenant est régie par le principe de présentation.

3.4.1.1 Le principe de présentation (ou de Login)

Le principe de présentation impose que pour qu'un intervenant puisse mettre en oeuvre la Carte Blanche, il existe un badge qui permette de le reconnaître comme un partenaire. Lorsque qu'un partenaire est reconnu, toutes les commandes qui s'en suivront seront faites en fonction des droits de ce partenaire.

La Carte Blanche pouvant posséder plusieurs applications, chacune d'elle pouvant être utilisées par plusieurs intervenants, le nombre d'intervenants peut être important. De plus les intervenants potentiels de toutes les applications des cartes blanches peuvent être nombreux. La présentation doit permettre de distinguer tous ces différents intervenants. En outre, elle doit pouvoir être effectuée par un intervenant de type opérateur, comme le porteur par exemple. L'utilisation de simples codes secrets comme dans les cartes à micro-processeur paraît insuffisante. Une présentation en deux étapes : identification et authentification, comme par exemple dans le Login d'UNIX [UNIX], est une solution intéressante. Cependant, la Carte Blanche doit autoriser des authentifications plus sophistiquées, à base d'algorithmes cryptographiques, qui font des cartes à micro-processeur, un outil très sécurisé.

3.4.1.2 Le mécanisme de ratification

Pour renforcer la sécurité, un mécanisme de ratification peut être mis en place lors de l'authentification. La Carte Blanche possède un mécanisme de ratification comme celui des cartes à micro-processeur. Il permet de protéger un partenaire de la recherche exhaustive de son authentifiant. En effet, les objets nomades pouvant être facilement perdus ou volés, cette occasion laisserait beaucoup de temps à un intervenant malhonnête pour chercher l'authentifiant dans le but de frauder.

Le mécanisme de ratification permet de limiter le nombre de présentations fausses consécutives d'un partenaire avant le blocage de ce partenaire. Le nombre d'essais autorisés peut être défini pour chaque partenaire. Ce mécanisme souvent associé à une authentification par code secret peut se généraliser à d'autres modes d'authentification.

3.4.2 Le choix des émetteurs d'application par le porteur

Lorsque le porteur vient d'acquérir sa Carte Blanche, elle ne contient aucune application. Elle ne possède donc aucun badge correspondant à un intervenant émetteur ou utilisateur d'applications. Le porteur de la Carte Blanche doit avoir le choix des applications dont il veut disposer sur son objet nomade. Il a donc le choix des émetteurs d'application et doit leur permettre d'installer leur application. Le principe de la présentation impose qu'un émetteur voulant installer une application (ou effectuer toute autre opération) se présente préalablement comme partenaire de la Carte Blanche. Tous les émetteurs devant être indépendants, chacun d'eux doit avoir son propre badge. Le choix par le porteur d'un émetteur d'application revient donc à l'autoriser à devenir partenaire de sa Carte Blanche, c'est à dire à créer un badge.

Mais la création d'un badge doit être effectuée par un partenaire (principe du Login). Or la Carte Blanche étant anonyme à l'achat, le porteur n'a pas de badge spécifique et n'est donc pas partenaire. Il doit donc exister, dans la Carte Blanche, un premier partenaire prédéfini : le partenaire PUBLIC [Gri92]. Ce partenaire permet à tout intervenant, y compris le porteur, d'effectuer les commandes publiques sur la Carte Blanche et certaines commandes nécessaires pour que les émetteurs choisis par le porteur puissent devenir partenaires de la Carte Blanche.

La Carte Blanche propose **trois niveaux d'association des émetteurs** suivant la finalité d'utilisation de l'objet nomade définie par le porteur et la sécurité requise.

1. Le premier niveau est celui dans lequel est l'objet nomade lorsque le porteur en fait l'acquisition. Comme le porteur ne possède pas d'accès privilégié, la création de nouveaux partenaires est effectuée à partir du partenaire PUBLIC. Le choix de l'émetteur et l'autorisation du porteur pour l'association de cet émetteur se limitent à remettre la Carte Blanche à l'émetteur pour qu'il y crée son badge. Le problème de ce niveau est qu'il est possible alors à l'émetteur de créer autant de badges qu'il le désire, donc éventuellement d'en créer plus que ce qui était prévu et autorisé par le porteur. De même, lorsque l'objet nomade est perdu, n'importe quel intervenant peut devenir partenaire en créant son badge.
2. Le niveau suivant peut être atteint lorsque le porteur désire plus de protection dans l'association d'émetteur, tout en restant anonyme. Pour cela, il doit obtenir un accès privilégié à sa Carte Blanche, en créant à partir du partenaire PUBLIC, un badge particulier. Par la suite, seul le partenaire ainsi défini, pourra autoriser l'association d'un émetteur en effectuant lui-même l'ordre de création d'un badge. Un seul partenaire de ce type peut être défini. Il est donc évident qu'il est nécessaire au porteur principal d'être le premier à effectuer cette opération. Il n'est alors plus possible de créer des partenaires à partir de PUBLIC.

   L'identifiant de ce partenaire est librement choisi par le porteur et ne peut donc servir d'identité du porteur. La Carte Blanche reste donc encore anonyme.
3. Le niveau de sécurité suivant est atteint quand la Carte Blanche est titularisée. Cette fois-ci, l'objet nomade est nominatif et possède l'identité de son porteur légitime. Un parte-

naire TITULAIRE est créé. Ce partenaire peut également autoriser l'association d'émetteur.

3.4.3 Association d'un Nouvel Intervenant

L'association d'un nouvel intervenant permet au porteur d'autoriser des émetteurs d'application à devenir partenaires de sa Carte Blanche. Ensuite, ces émetteurs peuvent avoir eux-mêmes besoin de définir d'autres partenaires nécessaires à l'usage spécifique de leur application.

3.4.3.1 Association des émetteurs d'application à la Carte Blanche

Quel que soit le niveau d'association des émetteurs d'application, les partenaires qui en résultent doivent tous être indépendants les uns des autres. Le porteur ayant donné l'autorisation d'association, il laisse l'émetteur d'application définir lui-même son identifiant et son authentifiant, car il n'a pas à les connaître. Chaque émetteur est libre de les choisir à condition qu'il n'y ait pas d'ambiguïté. Il est donc le seul à connaître ces deux éléments et à pouvoir se présenter sous ce partenaire. Ces partenaires sont ainsi indépendants.

Les émetteurs que le porteur a choisis, sont libres de s'associer ou non à la Carte Blanche. Ils peuvent, par exemple, exiger que la Carte Blanche soit titularisée s'ils veulent être sûrs de s'associer à l'objet nomade d'un porteur en particulier. Il résulte que l'association d'un émetteur à une Carte Blanche constitue un accord entre le porteur, qui autorise la création, et l'émetteur, qui accepte en définissant son identifiant et son authentifiant.

L'autorisation d'association des émetteurs inclut le droit d'installer une application. Mais des intervenants peuvent ne pas avoir besoin d'installer une application, ou ne pas être autorisés par le porteur à en installer une. Pourtant ils peuvent demander à avoir accès à la Carte Blanche pour utiliser des services d'applications et souhaiter être aussi indépendants que les émetteurs. Le porteur doit pouvoir autoriser l'association de ces intervenants par la même procédure que pour les émetteurs.

3.4.3.2 Association d'intervenant spécifique à une application

Les émetteurs d'application peuvent avoir besoin de créer des partenaires qui vont utiliser spécifiquement certains services de leur application. Des intervenants doivent pouvoir utiliser certains services particuliers d'une application sans pour autant avoir accès à tous les services et sans avoir les privilèges de l'émetteur. L'émetteur ne peut donc fournir son identifiant et son authentifiant à tous les intervenants susceptibles d'utiliser un ou plusieurs services de l'application. Il a alors le droit de créer de nouveaux partenaires pour l'utilisation spécifique de certains services de son application. Ce droit lui est attribué en même temps que le droit d'installer une application. L'émetteur définit l'identifiant et l'authentifiant de chacun de ces nouveaux partenaires, et, pour chacun de ces partenaires, les communique aux intervenants concernés et à eux seuls. Ces partenaires ainsi créé par un émetteur, ne sont donc plus indépendants car ils dépendent de l'émetteur. L'émetteur d'application peut donner en toute confiance le droit d'utiliser certains des services de son application aux partenaires qu'il a ainsi créés puisqu'il peut choisir lui- même les intervenants qui pourront utiliser ces services par l'intermédiaire de ces partenaires.

Les partenaires ainsi créés sont en général de simples utilisateurs de services et ne possèdent donc pas le droit d'installer une application. Cependant, un émetteur d'application peut très bien décomposer une importante application en plusieurs autres de taille plus petite, et les faire instal-

ler par différents partenaires qui dépendent de lui. La Carte Blanche ne doit donc pas interdire l'association d'émetteurs dépendants par un autre émetteur d'application.

3.4.4 Les autres opérations sur les partenaires

La Carte Blanche étant évolutive, elle permet évidemment de modifier certaines caractéristiques des partenaires existants ou de les supprimer. Lorsqu'un partenaire est bloqué par le mécanisme de ratification, la Carte Blanche doit également permettre de le réhabiliter.

3.4.4.1 Modification des caractéristiques d'un partenaire et suppression

Certaines caractéristiques de partenaire peuvent être modifiées. Il s'agit de l'authentifiant et du nombre de ratifications. En effet, pour renforcer la protection d'un partenaire, il peut être intéressant de changer régulièrement son authentifiant ou d'ajuster le nombre de ratifications que le partenaire peut supporter avant de se bloquer. L'identifiant d'un partenaire ne peut pas être modifié car il sert de référence pour l'attribution des droits d'utilisation des services par les émetteurs.

Les partenaires de la Carte Blanche doivent pouvoir être supprimés. En effet, le porteur peut ne plus avoir besoin d'une application, et l'émetteur n'a plus de raison d'être associé à la Carte Blanche lorsque l'application est retirée. De même, les partenaires dépendant de cet émetteur devraient également être supprimés.

Les partenaires indépendants, qu'ils soient créés par PUBLIC, par le partenaire «porteur» ou par le partenaire TITULAIRE, peuvent être modifiés ou supprimés, mais uniquement par eux-mêmes. Les partenaires PUBLIC, «porteur» et TITULAIRE qui ont permis de les créer ne peuvent en aucun cas les modifier. C'est évident pour PUBLIC, car n'importe quel intervenant pouvant accéder à la Carte Blanche par PUBLIC, pourrait modifier et supprimer des partenaires qui ne seraient donc plus indépendants. Pour les partenaires créés avec l'autorisation du porteur ou du titulaire, ceux-ci n'ayant pas connaissance des caractéristiques des partenaires, ne peuvent donc pas les modifier et les supprimer. De plus, la suppression d'un partenaire par le porteur ou le titulaire pourrait être utilisée comme moyen de fraude : la suppression du partenaire entraînerait la suppression de toute trace d'utilisation des services.

Dans le cas d'un partenaire dépendant d'un émetteur, c'est cet émetteur qui peut modifier les attributs du partenaire et/ou le supprimer.

3.4.4.2 Réhabilitation de partenaire bloqué

Un partenaire peut être bloqué par un fraudeur avant que la Carte Blanche ne soit restituée à son titulaire, ou plus simplement par une erreur de manipulation. Dans ce cas, il est nécessaire de pouvoir débloquer ce partenaire pour lui permettre d'utiliser à nouveau ses services. Cette opération s'appelle la réhabilitation d'un partenaire. Il n'y a plus dans la Carte Blanche, un super-émetteur qui possède ce droit pour tous les partenaires.

La réhabilitation d'un partenaire doit être effectuée par un autre partenaire ayant l'autorité de décider si ce partenaire peut être réhabilité ou pas.

- Pour les partenaires dépendant d'un émetteur, cet émetteur possède naturellement le droit de réhabiliter les partenaires qu'il a lui-même créés.

- Pour les partenaires indépendants qui ont accepté de s'associer par les partenaires PUBLIC ou «porteur», ils doivent demander à être réhabilités par le partenaire «porteur» ou le partenaire TITULAIRE, s'ils existent. Si aucun d'eux n'existe, le partenaire ne peut pas être réhabilité.
- Pour les partenaires indépendants qui ont demandé à s'associer à partir du titulaire seulement, ils doivent être réhabilités uniquement par le partenaire correspondant à un organisme officiel dans lequel tout intervenant peut avoir confiance : le partenaire OFFICIEL qui est prédéfini dans la Carte Blanche. Le choix d'association de ces partenaires à partir du titulaire est dû au fait que leur application a un caractère confidentiel ou critique. Ces partenaires demandent donc plus de protection et le porteur n'offre plus suffisamment de garanties.

Ni le porteur, ni le partenaire OFFICIEL ne connaissent l'identifiant et l'authentifiant du partenaire. Ils sont seulement capables de vérifier que l'intervenant qui leur demande une réhabilitation est effectivement un intervenant associé au partenaire bloqué et d'envoyer l'ordre de réhabilitation. Ensuite, ils laissent l'intervenant fournir son identifiant et son authentifiant.

3.5 Les Applications

Les applications sont des éléments essentiels pour la Carte Blanche car la quantité et la qualité des applications qui la composent contribuent à l'intérêt de l'utilisation de l'objet nomade par son porteur.

3.5.1 Composition Générale d'une Application

Nous allons définir ce qu'est une application dans un objet nomade évolué. En informatique, une application est un logiciel qui gère des données pour effectuer un ensemble de tâches contribuant à un même but. Une application peut se nomadiser pour deux raisons. Tout d'abord, elle manipule des données qui sont propres à l'individu qui la transporte et qui sont protégées par l'objet nomade. D'autre part, ces données doivent être disponibles et manipulables facilement en différents endroits.

3.5.1.1 Les Données

Les données d'une application sont des informations qui sont mémorisées dans l'objet nomade. Ces données peuvent évoluer pendant l'utilisation de l'application mais elles restent constantes entre deux utilisations de l'application, et sont conservées quand l'objet nomade n'est plus actif, ni même alimenté.

Les informations concernent directement l'application et/ou le porteur de l'objet nomade. Elles peuvent être constantes (un nom), rarement modifiables (une adresse), ou régulièrement mises à jour (une table de tarifs). Elles peuvent avoir un caractère confidentiel (un état de santé) ou critique (une somme d'argent).

3.5.1.2 Le Code

Les applications nécessitent également des traitements. Ces traitements utilisent les données de l'objet nomade pour réaliser le travail demandé. Ils peuvent servir à présenter les données à l'extérieur (l'identité du porteur, les soldes du porte-monnaie électronique, ...), les faire évoluer en

échange d'un résultat (diminuer le solde du compte bancaire en échange de quelques billets). Le caractère confidentiel de certaines données et le gain d'efficacité obtenu, nous conduit à inclure le code réalisant les traitements de l'application à l'intérieur de l'objet nomade.

L'implantation du code dans l'objet nomade peut être temporaire ou permanent.

- Application à code temporaire
  Les données de l'application sont et restent dans l'objet nomade. Lorsque l'application est utilisée, le code est chargé temporairement en mémoire, y est exécuté puis est finalement supprimé. Les avantages sont, d'une part, l'économie de place en mémoire EEPROM puisque le code n'est pas conservé de façon permanente, d'autre part, la mise à jour plus facile de ce code. Il reste cependant deux difficultés. La première est de garantir l'authenticité du code importé, puis d'assurer le lien entre les données dans la carte et ce code qui peuvent tous deux être alloués différemment sur deux cartes distinctes. La seconde est la diffusion du code sur un grand nombre de bornes de connexion de l'objet nomade pour que l'application bénéficie d'une grande couverture géographique d'utilisation.
- Application à code résident
  Les données et le code de l'application sont chargés en même temps en mémoire EEPROM et y restent. Il n'y a plus de problème de lien code-donnée puisque le code et les données sont conçus pour fonctionner ensemble, et le code connaît la structuration en mémoire des données, qui est invariable. Le principal avantage de cette implantation est la disponibilité du code avec moins de contrainte d'environnement. Il est donc utilisable à partir de nombreuses bornes de connexion. De plus, l'objet nomade est plus autonome et ce code permet une utilisation domestique sur un terminal léger tel qu'un minitel ou un des futurs postes de télévision. Ainsi le porteur peut utiliser facilement, de chez lui, les traitements de l'application qui lui sont destinés (par exemple les services de consultation de ses comptes bancaires). Quel que soit l'environnement d'utilisation de la Carte Blanche, les intervenants et les applications coopérantes utilisent toujours une application de la même manière puisque le code exécuté est toujours identique. De plus, les mises à jour d'application peuvent être faites individuellement compte tenu des besoins réels du porteur. Par exemple, certaines options d'une application ne seront installées que sur l'objet nomade des porteurs qui en ont réellement la nécessité.

Pour valider le concept de la Carte Blanche, nous étudions dans un premier temps le fonctionnement d'application à code résident. De plus, ces applications nous semblent plus proches de l'esprit de la Carte Blanche où le porteur doit disposer d'une grande souplesse d'utilisation des applications qu'il a choisies. Cependant, des applications à code temporaire pourront fonctionner également. Les problèmes liés à ces applications sont étudiés par ailleurs [PV94].

Les futurs objets nomades multi-applicatifs sont destinés à recevoir des applications très variées venant d'horizons différents. La plupart de ces applications sont indépendantes et possèdent alors leur propre code et leurs propres données. Ce sont les applications spécifiques. Parmi les applications installées dans un objet nomade, certaines peuvent appartenir à une même grande classe d'applications nomades, et présenter un fonctionnement similaire. Par exemple, dans la famille des applications de pré-paiement, nous trouvons toujours l'achat et la consommation de jetons. Il serait intéressant d'utiliser le même code sur des données différentes dans le cadre de deux applications distinctes. De même, une application peut avoir un but utilitaire et donner la

possibilité d'utiliser des outils logiciels sur des données appartenant à un utilisateur différent de l'émetteur d'application. Un exemple d'utilitaire est un gestionnaire de fichier ou de base de données. Cela nous amène à réfléchir sur l'opportunité d'insérer dans notre objet nomade en plus des applications spécifiques, des applications génériques.

- Application spécifique
  Les applications spécifiques répondent à des besoins individuels en termes de traitements et de données. Elles n'ont donc pas de points communs entre elles et sont indépendantes. Le code et les données de chaque application sont émis par le même émetteur respectif. Le code de l'application ne se rapporte qu'à ses données, et ses données qu'à son code. L'émetteur d'une application ne veut pas que l'on utilise son code sur d'autres données que celle de son application, ni que l'on puisse modifier ses données autrement que par le code de l'application. L'indépendance de ces applications demande donc leur cloisonnement par le système d'exploitation de l'objet nomade
- Application générique
  Les applications génériques ont un code qui peut s'appliquer sur des ensembles différents de données. Ces ensembles comprennent toujours un noyau identique : les données propres de l'application et des compléments distincts pour le compte de différents utilisateurs. Le choix du complément de données lors de l'exécution du code confère à l'application un caractère non plus générique mais individuel. Les avantages des applications génériques sont l'économie de mémoire grâce à la factorisation de code d'applications de même type, et la plus grande facilité d'ajouter un nouvel ensemble de données utilisant un code déjà existant.

  Une application générique doit être capable de distinguer les différents utilisateurs pour sélectionner le jeu de données correct. L'idéal serait de profiter de l'identification des utilisateurs par le système. Cela permet à l'émetteur de ne pas définir un mode d'identification et aux utilisateurs de ne mémoriser qu'un seul identificateur. D'autre part, le propriétaire du code et du noyau de données est à priori différent des propriétaires des données spécifiques. Ces derniers doivent être assurés que le code ne donnera pas accès à leur données propres, à d'autres utilisateurs. Il est difficile de définir quels droits ce propriétaire peut avoir sur les données spécifiques, et si le propriétaire des données spécifiques peut ou ne peut pas manipuler ses propres données autrement que par le code de l'application générique.

Dans la Carte Blanche, la sélection des jeux de données doit être pris en charge par le système car le code des applications chargées sous le seul contrôle de l'émetteur, ne peut pas être considéré à 100% fiable. La complexité des mécanismes nécessaires aux applications génériques ajouterait trop de difficulté dans la protection des multiples applications qui peuvent exécuter du code dans la Carte Blanche. Dans notre étude actuelle, nous avons mis de côté les applications génériques pour nous focaliser seulement sur la protection des applications en dynamique.

3.5.2 Architecture Interne d'une application dans la Carte Blanche

Les applications pour la Carte Blanche sont dans cette étude des applications spécifiques à code permanent. Elles sont composées de code et de données fournis en même temps par l'émetteur de l'application. Nous allons décrire plus précisément l'organisation du code et des données des applications pour Carte Blanche et comment ils s'articulent.

3.5.2.1 Les Services de l'application

Le code doit permettre de réaliser chacune des tâches que propose l'application. Ces tâches sont appelées les services de l'application et doivent pouvoir être exécutées individuellement. D'ailleurs, il ne doit pas être possible d'exécuter le code de l'application sans débuter au niveau d'un service.

La Carte Blanche n'impose pas que le code de chaque service soit distinct. Au contraire, pour faciliter le développement d'une application par un programmeur, et pour optimiser la taille du code de l'application, la Carte Blanche permet que le code de l'application soit structuré en programmes - les services - et sous-programmes que tous les services pourront partager.

3.5.2.2 Les Données de l'Application

Les données de l'application sont des informations que l'application utilise et peut modifier, et qui doivent conserver leur valeur d'une utilisation à l'autre. Il ne faut donc pas les confondre avec les variables dynamiques qui n'existent que lors de l'exécution d'un service, mais il faut surtout que l'émetteur d'application fournisse leur valeur initiale lors de l'installation. Ces données peuvent être constituées de valeurs numériques ou alphanumériques, structurées ou non, de fichiers ou de tables. Cependant, pour ne pas restreindre les possibilités pour le programmeur d'une application, la Carte Blanche ne prend pas en charge la représentation des données. Le type et la sémantique des données sont concrétisés par leurs traitements. Par exemple, le solde du compte bancaire est une donnée numérique partagée par les services `Débiter`, `Créditer` et `Consulter un compte` de l'application `Banque`. Les services `Débiter` et `Créditer` vont modifier le solde.

Toutes les données doivent être directement accessibles à partir de n'importe quel service de l'application. La Carte Blanche n'a pas à protéger les données d'une application par rapport au code de cette même application. Le programmeur de l'application peut et doit gérer lui-même l'accès service-données.

3.5.2.3 Accès à l'application par les partenaires

L'accès à l'application et à ses éléments par les différents partenaires doit être protégé. Nous allons déterminer quels droits peuvent être attribués aux partenaires sur le code et sur les données d'une application.

Le code de l'application ne doit servir qu'à exécuter des services de l'application. L'émetteur d'application peut attribuer ou non un droit d'exécution de chaque service de son application à différents partenaires.

La Carte Blanche ne permet pas l'accès direct aux données et ne possède d'ailleurs pas de commandes de manipulation des données des applications. Par contre, le droit d'exécution d'un service donne le droit implicite de manipulation des données que ce service utilise, et la protection de ces données est garantie par le traitement du service qui est supposé être défini correctement par l'émetteur. Dans les concepts de la Carte Blanche, les applications possèdent leur propre code pour permettre, le plus souvent possible, le traitement des données en interne, ce qui évite de sortir les données de l'objet nomade. Mais, tous les traitements ne pourront être effectués par l'objet nomade, et dans ce cas, un service de l'application peut être prévu pour fournir une donnée à l'extérieur afin qu'elle y soit traitée, puis pour récupérer la donnée modifiée que ce service

pourra également contrôler. Le code des services permet une grande diversité des contrôles sur les données.

La Carte Blanche n'utilise la protection classique des données par l'attribution des droits de lecture, d'écriture, de suppression ou de mise à jour, pour deux raisons. D'une part, la vérification de tels droits sur les données ne s'appliquerait pas au code de l'application, car les données doivent être accessibles sans restriction par le code de l'application. D'autre part, donner des droits sur les données, à des partenaires, signifierait que ces partenaires pourraient effectuer sur ces données des traitements autres que ceux définis dans les services, et sans que l'application puisse les contrôler.

Une application est accessible uniquement à travers ses *services*. On ne peut jamais accéder directement à ses données. Les services forment donc l'interface de l'application vis à vis des partenaires.

3.5.3 Gestion des Applications

La gestion des applications est l'ensemble des opérations qui permet à un émetteur d'application de mettre en oeuvre une application sur la Carte Blanche puis de la contrôler. Elle comprend l'installation d'une nouvelle application, la mise à jour et la suppression de cette application par l'émetteur mais aussi la configuration des droits d'accès de cette application aux différents partenaires de la Carte Blanche.

3.5.3.1 Installation d'une application

L'installation d'application est l'une des plus importantes caractéristiques de la Carte Blanche. Elle permet d'ajouter de nouvelles applications provenant d'émetteurs indépendants, tout au long du cycle de vie de l'objet nomade. D'après les concepts de la Carte Blanche, le choix des applications doit revenir au porteur, et la procédure d'installation des applications doit assurer leur indépendance.

Le porteur choisit les applications qu'il veut installer sur son objet nomade en les faisant charger sur sa Carte Blanche par les émetteurs respectifs de ces applications. Pour chaque nouvelle application qu'il souhaite installer, le porteur autorise l'émetteur de l'application à devenir partenaire de sa Carte Blanche en créant un badge, et lui donne le droit d'installer une application. Cette procédure montre comment les charges de personnalisation qui pesaient, dans les cartes à micro-processeur, sur le seul émetteur de carte sont réparties sur différentes entités de la Carte Blanche. Tout d'abord, le choix des applications sur la Carte Blanche revient de droit à celui qui utilise à tout instant son objet nomade, c'est à dire au porteur. Ensuite, ce n'est plus une unique entité qui possède et charge toutes les applications dans la Carte Blanche. Chaque émetteur autorisé charge lui-même son application, sans que le porteur en connaisse le code et les données (sinon il pourrait les exploiter commercialement ou les modifier à son profit). Ainsi, la Carte Blanche permet l'installation d'applications provenant d'émetteurs indépendants.

L'installation d'une application consiste à charger dans la Carte Blanche tous les éléments constitutifs d'une application et à construire les structures de sécurité nécessaires à l'utilisation protégée de l'application.

Pour chaque application, l'émetteur d'application doit fournir à la Carte Blanche :

- le code permettant d'effectuer les services et la liste de ces services
- l'ensemble des données persistantes de l'application. Ces données sont fournies sous la forme d'une valeur initiale pour la première utilisation de l'application

Après installation, le partenaire correspondant à l'émetteur est le seul à connaître les services de son application. Par conséquent, il est le seul à pouvoir utiliser ces services et doit être tenu comme seul responsable de son application.

Le porteur qui n'a pu exercer de contrôle des applications à priori doit posséder un moyen d'action à posteriori sur l'émetteur et son application. Il s'agit de la *neutralisation* d'application (cf. 3.5.3.5 : Neutralisation d'application).

3.5.3.2 Attribuer des droits d'exécution

Après installation d'une application, seul l'émetteur a le droit d'utiliser les services de son application. Or les services qui composent une application peuvent être destinés à être utilisés par différents intervenants. En effet, en prenant l'exemple d'une application bancaire, certains services sont réservés :

- à l'émetteur d'application pour le suivi de l'application de l'extérieur de la Carte Blanche. Nous trouvons dans cette catégorie le service de collecte des dernières transactions
- à des intervenants indépendants qui peuvent avoir besoin de ces services. Le service `Débiter` peut être utilisé par les intervenants commerçants, et le service `Consulter le solde` par le porteur de la Carte Blanche
- à des intervenants spécifiques à l'application qui sont les seuls à pouvoir utiliser ces services. Le service `Retrait d'Espèces` est réservé aux intervenants de type `Distributeur Automatique de Billets`

L'émetteur d'application ne peut pas divulguer son identifiant et son authentifiant à tous les intervenants susceptibles d'utiliser un service particulier car non seulement ils auraient accès à tous les services de l'application, mais ils pourraient également modifier, voire supprimer l'application. A chaque intervenant doit correspondre un badge. L'émetteur permet à chaque partenaire correspondant aux intervenants précédents, d'accéder seulement à des services bien déterminés. Chaque autorisation concerne précisément un seul service pour un seul partenaire.

Si l'intervenant est indépendant de l'émetteur, l'attribution des droits ne pourra se faire qu'après un accord extérieur à la Carte Blanche entre l'émetteur et l'intervenant avec communication, par ce dernier, de son identifiant.

Pour les intervenants spécifiques à l'application, la Carte Blanche permet à l'émetteur de créer lui même les partenaires correspondants, donc de connaître leur identifiant. Il lui est alors facile de leur attribuer les droits d'exécuter les services particuliers qu'il leur destine. Le fait que l'émetteur peut à tout moment modifier l'authentifiant de ses partenaires dépendants et même les supprimer, confère un caractère de sûreté au droit d'exécution qu'il leur attribue.

L'attribution du droit d'exécution d'un service ne peut se faire qu'à un partenaire qui existe. En effet, l'attribution à des partenaires inexistants, mais qui seraient susceptibles de s'associer plus tard, pose plusieurs problèmes. D'abord, le droit occupe de la mémoire dans la Carte Blanche

alors qu'il n'est pas encore fonctionnel et qu'il ne le sera peut-être jamais. Ensuite, l'identifiant du partenaire bénéficiaire de ce droit peut être choisi volontairement ou involontairement par un autre intervenant qui s'associe à la Carte Blanche et qui disposerait illégitimement du droit d'exécuter ce service.

Voici un autre exemple qui illustre l'attribution de droits à des partenaires spécifiques : la délégation des services à d'autres partenaires. Le partenaire `Compagnie de Transport` a installé l'application `Transport` qu'il est le seul à gérer. Il associe trois nouveaux partenaires `Composteur`, `Revendeur` et `Contrôleur` auxquels il attribue respectivement des droits d'exécution de ses services `Composter`, `Acheter` et `Contrôler`.

L'attribution des droits d'exécuter des services d'une application ne peut pas être effectuée seulement lors de l'installation mais pendant toute la «vie» de la Carte Blanche car de nouveaux partenaires peuvent être créés dans l'objet nomade. Les droits peuvent être également retirés à tout moment de l'exploitation de la Carte Blanche, par exemple, quand un partenaire est supprimé de la Carte Blanche ou que l'émetteur constate une utilisation douteuse de son service par un partenaire.

3.5.3.3 Suppression d'application

La Carte Blanche est évolutive. Des applications peuvent donc être nouvellement installées mais aussi être supprimées. Il existe de nombreuses raisons de supprimer une application :

- l'émetteur retire son application du monde nomade
- l'émetteur ne souhaite plus mettre son application à la disposition d'un porteur particulier
- la durée d'utilisation de cette application expire pour le porteur
- le porteur n'a plus besoin de cette application

De plus, les applications devenues obsolètes doivent pouvoir être supprimées pour laisser la place à de nouvelles applications.

La suppression d'application retire de la mémoire de la Carte Blanche les données, le code et la liste des services de l'application ainsi que toutes les structures qui servaient à maintenir la sécurité. Tous les droits d'exécution encore attribués à des partenaires sont automatiquement détruits.

Il parait normal qu'une application puisse être supprimée uniquement par le partenaire qui l'a installée. Si la volonté de supprimer l'application provient du porteur, ce dernier devra demander à l'émetteur de l'application de le faire. Il ne peut évidemment pas le faire lui-même car il pourrait également chercher à tromper l'émetteur en supprimant l'application immédiatement après une opération dans l'espoir que cette opération ne lui sera pas comptabilisée. Bien sûr, le porteur doit pouvoir exercer une autorité sur l'émetteur qui refuse d'obtempérer. Il s'agit de la neutralisation de l'application. Si l'émetteur ne retire toujours pas son application ou en cas de disparition de l'émetteur, le porteur pourrait avoir recours au partenaire de confiance pour une suppression exceptionnelle de l'application.

3.5.3.4 Mise à Jour d'application

La Carte Blanche est évolutive aussi parce qu'une application n'est pas figée de son installation à sa suppression. En effet, une application a parfois besoin d'être mise à jour pour :

- améliorer le fonctionnement ou corriger des erreurs de services existants
- ajouter des caractéristiques à l'application en termes de données, services nécessaires à l'administration de l'application
- ajouter de nouvelles options de l'application sur demande du porteur

La Mise à Jour d'application de la Carte Blanche permet d'ajouter, modifier ou retirer des *services* ou *données* de l'application. Elle doit reprendre les valeurs des données de l'application dans l'état au moment de la mise à jour. Elle doit aussi conduire à une interface compatible pour les autres utilisateurs de l'application, autant que les modifications le permettent. Cela lui permet de conserver également les droits d'exécution déjà attribués à des partenaires.

La Mise à Jour d'une application est réalisable par son propriétaire seulement, éventuellement sur demande du porteur.

3.5.3.5 Neutralisation d'application

Le porteur doit posséder un moyen de pression sur les applications et leur émetteur. Ce moyen est nécessaire si l'application chargée par l'émetteur ne correspond pas à la demande du porteur, et également lorsque le porteur souhaite supprimer cette application alors que l'émetteur refuse de le faire. Le porteur doit pouvoir neutraliser une application. La neutralisation consiste à rendre inopérants tous les services de cette application, aussi bien pour le partenaire de l'émetteur que pour tous ceux auxquels il a attribué le droit d'exécution. Les commandes administratives de l'application restent opérantes, notamment la commande de suppression d'application.

La neutralisation est une action réservée au porteur et peut être mise en oeuvre uniquement par lui. Il faut donc obligatoirement qu'il se soit associé comme partenaire porteur ou qu'il ait titularisé sa Carte Blanche (cf paragraphe 3.6 : La Titularisation).

3.5.4 Coopération d'applications dans la Carte Blanche

La Carte Blanche est multi-applicative. Certaines applications peuvent avoir besoin d'utiliser d'autres applications. Le cas le plus courant est l'utilisation d'une application `banque` ou `porte-monnaie électronique` pour racheter des jetons dans les applications de prépaiement. Par exemple, le service `Achat de tickets` de l'application `Transport` payera à l'aide du service `Débiter` de l'application `Banque`. On peut également imaginer des applications demandant des informations à une autre, par exemple l'application `Sécurité Sociale` demande un certificat de scolarité à l'application `Etudiant`. Il est souhaitable de profiter de la présence de plusieurs applications sur le même support informatique pour qu'elles puissent coopérer entre elles sans sortir d'information de l'objet nomade.

La coopération avec une autre application est obtenue en autorisant l'utilisation d'un service d'une application, à l'intérieur d'un service d'une autre application. Pour la coopération entre applications, l'émetteur de l'application utilisée permet au propriétaire de l'application coopérante d'appeler le service concerné. De même que pour l'attribution des droits d'exécution, les deux émetteurs indépendants ne connaissent pas leur identifiant de partenaires mutuels. L'autori-

sation de coopération est donc l'aboutissement d'un accord externe à la Carte Blanche entre les deux émetteurs.

Si un émetteur d'application A donne à un autre émetteur B le droit de coopérer avec un service S_A de son application, ce partenaire B pourra utiliser le service S_A et tous les services S_B de son application y compris chaque service S_{BA} qui utilise le service S_A. Ce partenaire B peut être naturellement amené, en qualité d'émetteur, à autoriser l'utilisation des services S_B de sa propre application à un troisième partenaire C, mais l'exécution des services S_{BA} nécessite l'utilisation du service S_A.

Le droit de coopération donné par l'émetteur A à l'émetteur B, peut alors être donné avec ou sans délégation.

- Le droit de coopération sans délégation oblige le partenaire C à demander à A l'autorisation d'utiliser son service S_A pour pouvoir faire fonctionner correctement les services S_{BA}. Dans ce cas, tout utilisateur d'un service nécessitant la coopération devra lui-même posséder les droits d'exécution de tous les services mis en jeu.
- Le droit de coopération avec délégation autorise le partenaire B à attribuer au partenaire C, le droit d'exécuter, sans restriction, tous ses services S_B, y compris les services S_{BA}. Le service S_A pourra être utilisé dans l'exécution du service S_{BA}, mais ne pourra jamais être appelé directement par C. Ce type de coopération facilite la gestion des droits pour la coopération, mais demande que le premier émetteur puisse avoir confiance dans les attributions des droits, par le second émetteur, à d'autres partenaires, car cela permet d'effectuer facilement un «Cheval de Troie» entre un émetteur et un autre partenaire.

Lors de la coopération entre applications, l'indépendance des deux applications doit être conservée. L'application qui appelle un service d'une autre application ne doit pas pouvoir accéder n'importe où dans le code ni à toutes les données de cette autre application. Elle ne peut que lancer le service autorisé et obtenir les résultats de ce service.

Les mêmes services sont utilisés par un logiciel sur un système hôte et par une autre application de la Carte Blanche. Le fonctionnement de ce service doit être identique aussi bien en mode coopératif qu'en mode communication. Lors de la coopération, les deux applications dialogueront entre elles de la même façon qu'elles le feraient avec un logiciel externe. Les résultats du service appelé seront obtenus par ce moyen.

3.6 La Titularisation

La Carte Blanche est conçue comme un objet nomade personnel. L'acquéreur d'une Carte Blanche peut décider de l'utilisation de son objet nomade. En général, il devient le porteur principal et légitime et décide des applications qu'il fait installer sur son objet nomade pour ses besoins.

Certains émetteurs n'accepteront d'installer leur application que sur l'objet nomade d'un ayant-droit. Par exemple, une banque n'installera son application que sur la Carte Blanche d'un titulaire d'un compte dans cette banque. Les émetteurs des applications contenant des données personnelles concernant le porteur seront certainement amenés à effectuer un tel contrôle. D'autres émetteurs peuvent avoir simplement besoin de connaître à qui vont être réellement desti-

nés les services de leurs applications. La Titularisation répond alors au besoin d'identifier le porteur légitime ou plus généralement le propriétaire de l'objet nomade. Elle consiste à inscrire des informations permettant l'identification de ce propriétaire. Le titulaire de la Carte Blanche est donc le propriétaire identifié de l'objet nomade.

D'autre part, cela permettrait à une Carte Blanche retrouvée après sa perte ou son vol d'être restituée à son propriétaire. Ensuite, un fraudeur voulant se substituer au porteur légitime pourrait ainsi être confondu.

La Carte Blanche dissocie les notions de titulaire et de porteur. En effet, le porteur n'est pas toujours le titulaire. Le titulaire peut simplement prêter sa carte à une autre personne pour utiliser l'une des applications non relatives au titulaire, par exemple, l'application Télécarte. Ensuite, le titulaire peut être une personne morale, par exemple une société. Le porteur peut être alors un employé de cette société ou un objet mobile appartenant à celle-ci. De même, la Carte Blanche peut être titularisée au nom d'une famille complète, et le porteur est alors n'importe quel membre de cette famille. Le porteur peut également être un fraudeur. En effet, l'individu qui retrouve une Carte Blanche perdue ou volée peut chercher à exploiter illégitimement cet objet nomade. Enfin, le porteur n'est pas toujours une personne mais peut aussi être un objet. Le titulaire peut ne pas exister ou alors être son propriétaire (personne physique ou morale).

Contrairement aux cartes à micro-processeur, la Carte Blanche n'impose pas la titularisation au moment de la distribution. En effet, l'ouverture de la Carte Blanche permet une utilisation moins personnelle de l'objet nomade. L'acquéreur peut ne faire installer que des applications qui ne traitent aucune donnée personnelle. Dans ce cas, leurs émetteurs n'obligent pas la Carte Blanche à être titularisée. De même, l'acquéreur de la Carte Blanche peut la destiner à suivre des produits à travers une chaîne de fabrication où elle est réinitialisée à chaque nouveau cycle. L'identification du titulaire peut être parfois inutile. Pour ces raisons, sa Titularisation est facultative.

La Carte Blanche étant évolutive, le porteur légitime peut être amené à installer d'autres applications, cette fois-ci personnelles. Il est alors nécessaire d'autoriser la titularisation avant l'installation. La Titularisation doit donc être possible à tout moment à partir de la distribution.

Le titulaire de la Carte Blanche est unique, d'abord parce qu'il ne peut y avoir qu'un seul propriétaire, ensuite parce que le contrôle des autorisations à installer une application ne peut être donné qu'à un seul individu pour des raisons de sécurité. Les informations d'identification du titulaire inscrites dans la Carte Blanche doivent permettre à un émetteur de connaître le titulaire ou à un individu de lui rendre sa Carte Blanche perdue. Les applications précédemment installées ne doivent pas être modifiées par cette titularisation. Par contre, toutes les applications installées nominativement, font référence au titulaire. Il est souhaitable qu'elles partagent la même identification du titulaire pour ne pas la dupliquer et éviter les problèmes liés à la redondance. Par conséquent, l'information d'identification du titulaire doit être publique et indépendante de toute application. De plus, elle ne doit plus pouvoir être modifiée car les données des applications nominatives ne seraient plus en rapport avec le titulaire. La Titularisation de la Carte Blanche n'est donc réalisable qu'une seule fois.

La référence à un même titulaire favorise la coopération des applications nominatives. En effet, le degré de confiance mutuelle des émetteurs nécessaire à l'établissement de liens de coopération de telles applications, ainsi que l'intérêt que cette coopération suscite, est plus important par le fait que les applications nominatives sont généralement sûres et qu'elles concernent le même individu.

L'identification du titulaire doit être une information en laquelle un émetteur doit pouvoir avoir confiance. Contrairement aux cartes à micro-processeur pour lesquelles l'émetteur de cartes se portait garant de l'information qu'il inscrivait en phase de personnalisation, c'est le porteur désirant se légitimer qui décide de la titularisation de sa Carte Blanche. Il ne peut enregistrer lui-même son identification car celle-ci n'aurait aucun crédit auprès des émetteurs. Il faut donc que la titularisation soit effectuée par un partenaire qui peut vérifier que l'information qu'il inscrit dans la Carte Blanche correspond bien au porteur supposé légitime qui vient lui demander de titulariser son objet nomade. Ce partenaire va dépendre de la nature de l'information permettant l'identification du porteur.

3.6.1 Enregistrement du Titulaire de la Carte Blanche

Il existe plusieurs façons d'identifier un titulaire. Il peut s'agir du numéro de compte client d'une banque ou d'un numéro d'adhérent. Mais ces identifications ne sont valables que dans un contexte applicatif. Par contre, une identification plus universelle d'un titulaire est son identité. De plus, la première installation d'application nominative ne peut vraiment se référer qu'à cette identification. La Carte Blanche retient donc l'identité du titulaire comme moyen de l'identifier. Les autres moyens dépendraient des applications que le titulaire fait installer sur son objet nomade et ne pourraient tous être pris en charge par la Carte Blanche. Par la suite, les applications pourront ajouter dans leurs propres données l'identification usuelle du titulaire pour chacune d'elles. Ensuite elles pourront rendre publique et partager leur identification par un de leurs services.

L'identification du titulaire de la Carte Blanche est donc composée de son **nom complet** (patronyme et prénom pour une personne physique, raison sociale intégrale pour une personne morale). Ces informations ne sont pas suffisantes car il faut distinguer les homonymes. Il est indispensable d'ajouter un moyen de les différencier, déjà utilisé par les administrations, comme **un numéro de pièce d'identité**. L'ensemble est naturellement invariable et ne doit donc pas pouvoir être modifiée même frauduleusement. La référence du titulaire peut comprendre également d'autres données complémentaires qu'il est intéressant de partager entre les différentes applications. L'adresse du titulaire (son siège social pour une personne morale) en est un exemple. La référence du titulaire permet alors de connaître les coordonnées complètes du titulaire pour pouvoir, en cas de perte, lui rendre sa Carte Blanche. Le titulaire pouvant déménager, la Carte Blanche doit autoriser la modification de cette partie de la référence du titulaire. La mise à jour de l'adresse du titulaire se fait en une unique opération, sans problème d'homogénéité dû à la redondance de l'information. La modification est répercutée sur toutes les applications nominatives qui utilisent les coordonnées du titulaire

Qui mieux que les organismes déjà habilités à prouver l'identité d'un individu, tels que **mairies, préfectures, ambassades**, seraient capables de fournir les informations correctes sur le porteur ? La Titularisation est donc autorisée à un unique partenaire particulier, le partenaire

prédéfini OFFICIEL sous lequel seuls les organismes ci-dessus peuvent se présenter. Le porteur se rendra donc à l'un de ces organismes pour que celui-ci présente le partenaire OFFICIEL et titularise la Carte Blanche.

3.6.2 Création d'un partenaire TITULAIRE

La titularisation s'accompagne de la création d'un partenaire particulier : le partenaire TITULAIRE. Le choix de l'authentifiant est laissé au porteur présumé légitime qui vient titulariser son objet nomade. Ainsi, seul ce porteur pourra présenter ce partenaire à la Carte Blanche. Le partenaire TITULAIRE ainsi créé, étant le propriétaire de la Carte Blanche, a naturellement le droit d'installer ses applications personnelles. Parce qu'il décide des applications qu'il veut faire installer, le partenaire TITULAIRE possède le droit d'associer un nouveau partenaire indépendant. Si cela n'est pas déjà fait par la création du partenaire de catégorie porteur, des intervenants ne peuvent plus s'associer à partir du partenaire PUBLIC. Ce privilège peut être supprimé pour le partenaire porteur ou conservé pour les associations d'émetteurs d'applications non nominatives.

L'existence de ce partenaire est bien utile pour les applications nominatives qui ont souvent besoin d'offrir certains de leurs services au titulaire lui-même, comme par exemple le service de consultation du compte bancaire. Le titulaire ayant un partenaire associé peut donc accéder à des services. Grâce à l'identifiant prédéfini de ce partenaire, les émetteurs de ces applications peuvent facilement lui donner les droits d'utilisation des services qui le concernent. De plus, la procédure de titularisation les assure que ce partenaire correspond bien au titulaire de la Carte Blanche.

3.7 Conclusion

Les concepts de la Carte Blanche répondent aux objectifs de cette étude. La Carte Blanche est un objet nomade universel et évolutif dans lequel peuvent cohabiter plusieurs applications provenant d'émetteurs indépendants. La sécurité équivalente à celle des cartes à micro-processeur est obtenue par un contrôle d'accès aux applications basé sur l'identification et l'authentification des intervenants. La souplesse d'évolution a été rendue possible par la disparition du super-émetteur. Les privilèges qu'il détenait sont redistribués en fonction des aptitudes inhérentes à chacun. Le porteur ou le titulaire autorise des émetteurs à installer les applications dont il a besoin. Chaque émetteur gère ensuite son application. Seule l'une des prérogatives du super-émetteur n'a pu être généralisée : c'est la confiance qu'il apportait dans le nom du titulaire et dans le pouvoir de réhabiliter un partenaire. Le partenaire prédéfini OFFICIEL tient ce rôle.

Cependant, les caractéristiques nouvelles de la Carte Blanche ont besoin de nouvelle sécurité pour protéger les applications. Notamment, la possibilité d'exécuter du code importé par des émetteurs indépendants à l'intérieur de l'objet nomade demande des mécanismes de protection dynamique de la mémoire de l'objet nomade. Ces mécanismes ne peuvent être pris en charge que par un véritable système d'exploitation.

Le chapitre suivant donnera les spécifications du système d'exploitation qui supportera le modèle de la Carte Blanche. Les contraintes que ce modèle entraîne y seront également traitées.

Chapitre 4
La Carte Blanche : Spécifications et Contraintes

4.1 Introduction

Nous présentons dans ce chapitre, les spécifications du système d'exploitation de la Carte Blanche. Ces spécifications permettent de mettre en oeuvre les concepts établis au chapitre 3. Ce chapitre décrit d'abord le nouveau cycle de vie engendré par le modèle de fonctionnement de la Carte Blanche. Ensuite, nous présentons les différents objets - les partenaires et les applications - manipulés par le système d'exploitation. Après, nous abordons le langage de commandes du système d'exploitation. Enfin, nous faisons le point sur les contraintes engendrées par le modèle de la Carte Blanche ou par les choix d'implémentation de celui-ci.

4.2 Le Cycle de Vie

Le cycle de vie est l'ensemble des états que prend la Carte Blanche ainsi que les règles de passage d'un état à l'autre. Ce cycle de vie, beaucoup plus souple que celui des cartes à micro-processeur, est la conséquence directe du modèle de fonctionnement de la Carte Blanche.

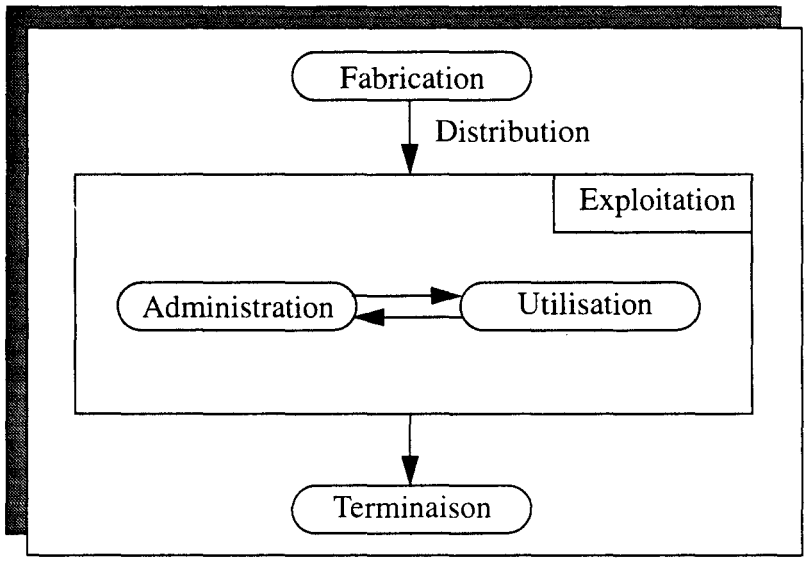
La principale différence avec la carte à micro-processeur, est la disparition de la phase de personnalisation effectuée une fois pour toute avant la distribution au porteur. Les fonctions de cette phase sont reportées après la distribution, au niveau de la phase d'exploitation de la Carte Blanche pour laisser libre n'importe quel porteur potentiel d'acquérir une Carte Blanche et d'y installer les applications dont il a besoin.

4.2.1 Fabrication

La fabrication est la phase durant laquelle le *système d'exploitation* est implanté dans la carte. Pendant cette phase, des informations propres à chaque carte et qui permettent de la distinguer des autres, sont placées dans la carte. Ces informations comprennent la référence du fabricant et un identificateur unique qui permettra, le cas échéant, de retrouver une carte particulière (par exemple, si elle avait fait l'objet d'une fraude). A la suite de la fabrication, la Carte Blanche est prête à l'emploi.

Le passage de la fabrication à l'exploitation est la **distribution** de la Carte Blanche à un individu qui devient ainsi son porteur.

FIGURE 11 Le cycle de vie de la Carte Blanche



4.2.2 Distribution

La distribution est la vente des cartes de façon anonyme par les grands réseaux de distribution. Dans ses concepts, la Carte Blanche est déjà prête à l'emploi et sa distribution ne change pas son état. Aucune information n'est inscrite dans l'objet nomade, pas même une information concernant son nouveau porteur.

4.2.3 Exploitation

L'exploitation est la phase de fonctionnement courant de la Carte Blanche dans laquelle l'utilisation des applications alterne avec l'administration de l'objet nomade. La possibilité d'accéder à des fonctions d'administration à tout moment de l'utilisation habituelle des objets nomades, confère à la Carte Blanche son caractère évolutif.

4.2.3.1 Utilisation

L'utilisation de la Carte Blanche comprend la mise en oeuvre des applications existantes dans l'objet nomade à un moment donné. Elle correspond à la phase d'utilisation des cartes à microprocesseur. Elle ne change donc en rien la structure de la Carte Blanche, particulièrement en nombre d'applications et de partenaires.

4.2.3.2 Administration de la Carte Blanche

L'administration de la Carte Blanche concerne toutes les commandes qui changent la structure fonctionnelle de l'objet nomade. Elle comprend essentiellement la gestion des partenaires et des applications, et la titularisation. Elle comprend donc l'installation de nouvelles applications, leur mise à jour ou destruction éventuelle, l'association de nouveaux partenaires, la configuration des droits d'utilisation de ces applications par les différents partenaires, ainsi que la définition du titulaire qui sert à rendre nominative la Carte Blanche. Cette phase correspond à la phase de person-

nalisation des cartes à micro-processeur. Elle regroupe l'utilisation des commandes administratives (cf. chapitre 2) de la Carte Blanche.

4.2.4 Terminaison

La terminaison est une phase spécifique aux objets nomades. La Carte Blanche entre dans cette phase quand elle ne peut plus être utilisée normalement. Le passage est involontaire de la part du porteur quand il s'agit de la destruction accidentelle, de la perte ou du vol. Dans ce dernier cas, la Carte Blanche détecte l'utilisation frauduleuse et se bloque. Le porteur peut également invalider volontairement son objet nomade, parce qu'il n'en a plus besoin, après quoi aucune application ne peut être utilisée. Cependant, les émetteurs d'applications pourront encore récupérer l'état des données de l'application.

4.3 Les Partenaires

Un partenaire correspond à un intervenant dont les références et les droits sont notés dans un badge à l'intérieur de la Carte Blanche. La notion de partenaire dans la Carte Blanche est très importante car les droits que possède chaque intervenant connu sur l'objet nomade y sont directement attachés.

4.3.1 Les Catégories de Partenaires

La Carte Blanche distingue trois catégories de partenaires auxquelles sont attachés des droits administratifs. Selon que le partenaire correspond à un simple utilisateur, à un émetteur d'application ou au porteur légitime de la Carte Blanche, celle-ci ne doit lui donner que les droits administratifs qui lui sont nécessaires.

4.3.1.1 Les Clients

Les partenaires Clients correspondent aux simples utilisateurs de la Carte Blanche. Ils peuvent utiliser les commandes opérationnelles du système et les services publics d'applications ainsi que les services qu'ils sont spécifiquement autorisés à utiliser.

4.3.1.2 Les Serveurs

Les partenaires Serveurs correspondent aux émetteurs d'application. Ils ont le droit d'installer une application dans la Carte Blanche puis de la gérer. Ils peuvent autoriser ou non d'autres partenaires à l'utiliser. Ils peuvent également créer d'autres partenaires, dépendant du serveur, auxquels ils pourront donner l'exclusivité d'utilisation d'une partie de leur application. Ces partenaires se comportent donc comme des serveurs d'application de la Carte Blanche, auprès des autres partenaires. Ils peuvent également utiliser leurs propres applications et celles que leur ont autorisées d'autres serveurs.

Remarque : Tant qu'un serveur n'a pas installé sa propre application, il se comporte comme un client. Mais ce client est évolutif puisqu'il peut à tout moment installer son application.

4.3.1.3 Le porteur

Cette catégorie correspond au porteur légitime de la Carte Blanche. Bien que celui-ci soit unique, il peut être représenté pour la Carte Blanche par deux partenaires : l'un anonyme, l'autre nominatif. Il ne peut donc y avoir au plus, que deux partenaires de cette catégorie. Ces partenaires

reprennent les caractéristiques des serveurs mais ils permettent également l'association d'émetteurs (et plus généralement d'intervenants) indépendants.

4.3.2 Les Partenaires prédéfinis du système

Il existe dès la sortie de la fabrication de la Carte Blanche des partenaires prédéfinis. Ces partenaires sont étroitement liés au système et sont nécessaires à sa gestion : le partenaire SYSTEME lui-même et les partenaires PUBLIC et OFFICIEL. Le quatrième partenaire du système est quant à lui créé dynamiquement au cours de l'exploitation de la carte : le TITULAIRE.

4.3.2.1 Le partenaire SYSTEME

Le partenaire SYSTEME est le serveur principal du système d'exploitation. Il est le propriétaire de l'application SHELL qui permet de commander le système. Tous les partenaires prédéfinis du système sont dépendants de ce partenaire.

Pour la sécurité de la Carte Blanche, il n'est pas possible d'avoir accès directement à ce partenaire par présentation, ce qui interdit la modification de l'application SHELL et les partenaires du système.

4.3.2.2 Le partenaire PUBLIC

Le partenaire PUBLIC est un client particulier de la Carte Blanche. Le partenaire PUBLIC ne demande pas d'authentification et il est le partenaire présenté par défaut à chaque connexion de la Carte Blanche avec l'extérieur. Ce partenaire permet l'accès aux commandes de la Carte Blanche dès sa distribution alors qu'il n'existe encore aucun partenaire serveur ou porteur. Ce partenaire PUBLIC permet l'arrivée sur l'objet nomade d'un ou plusieurs serveurs pour qu'ils installent leurs applications et du partenaire porteur anonyme.

Ensuite, il permet également à tout intervenant de consulter les informations publiques de la Carte Blanche. Par exemple, un guichet multi-usages peut consulter la liste des applications actuellement disponibles sur l'objet nomade. Ainsi, il peut les comparer avec celles qu'il est capable de mettre en oeuvre et offrir au porteur de la Carte Blanche le choix optimum des services qu'il peut rendre. D'autre part, une personne retrouvant la Carte Blanche perdue, peut, grâce à cet accès libre à PUBLIC, consulter le nom et l'adresse du titulaire afin de lui restituer son objet nomade.

4.3.2.3 Le partenaire OFFICIEL

Le partenaire OFFICIEL est un partenaire serveur. L'existence de ce partenaire est due à la suppression de l'émetteur de carte dont Toutes les prérogatives ont pu être réparties entre les différents intervenants courants de la Carte Blanche (son porteur et les multiples émetteurs d'application) sauf une : la confiance que tout intervenant ou tout partenaire peut lui porter. Cette confiance intervenait à deux niveaux. Le nom du porteur inscrit à la personnalisation au porteur était considéré juste. La réhabilitation des partenaires bloqués par ratification était confiée à l'émetteur de carte.

Le partenaire OFFICIEL est créé pour illustrer cette confiance. Ce partenaire a la charge importante de la titularisation. Son accès est d'ailleurs réservé aux organismes nationaux (commissariats, mairies, préfectures) et internationaux (ambassades) déjà habilités à faire foi de l'identité d'une personne morale ou physique, c'est à dire un organisme de confiance. Ce partenaire est

serveur car il représente l'émetteur idéal pour installer des applications comme une pièce nationale d'identité, un passeport, un permis de conduire ou une carte grise.

De plus, il permet de réhabiliter les partenaires qui ont demandé à s'associés à la Carte Blanche à partir du partenaire TITULAIRE. En effet, les partenaires autonomes étant responsables hiérarchiques d'eux-même, ne peuvent pas se réhabiliter puisqu'ils sont bloqués, et ils ont suffisamment confiance dans le partenaire OFFICIEL pour lui autoriser leur réhabilitation.

4.3.2.4 Le partenaire TITULAIRE

Le partenaire TITULAIRE est créé à la titularisation de la Carte Blanche par le système d'exploitation. Il est un partenaire serveur, pour lui permettre d'évoluer avec la Carte Blanche et d'installer par la suite des applications qui lui sont propres. La concrétisation du titulaire en partenaire permet aux applications nominatives de connaître un partenaire particulier qui accédera à certains de leurs services qui le concernent directement. Par exemple, le partenaire TITULAIRE est le seul autre partenaire que la BANQUE à accéder au service Consultation du compte.

4.3.3 Le Principe de Présentation

Le principe de présentation impose qu'il n'y ait qu'un seul partenaire actif à la fois lors de l'utilisation de la Carte Blanche. Toutes les commandes qui s'en suivront seront faites au nom du partenaire et en fonction de ses droits. Par défaut, le partenaire prédéfini PUBLIC est actif automatiquement à la mise sous tension de la Carte Blanche. De PUBLIC, tout intervenant peut présenter ses références à la Carte Blanche pour être reconnu partenaire et ainsi disposer des applications qui lui sont autorisées.

Un intervenant est reconnu comme étant un partenaire de l'objet nomade, s'il présente à celui-ci la référence exacte du-dit partenaire. celle-ci est composée d'un identificateur, en général le nom de l'intervenant, et d'un authentifiant. L'authentification peut se faire classiquement par un code secret (l'authentifiant est alors ce code), ou par des modes plus sophistiqués tels que des protocoles d'authentification à base d'algorithmes cryptographiques.

4.3.4 Caractéristiques des partenaires liées au «login»

Chaque partenaire possède des caractéristiques autres que son identificateur, son mode d'authentification, son authentifiant et sa catégorie. Elles sont liées essentiellement à la présentation de l'intervenant et permettent soit de faciliter l'utilisation de la Carte Blanche pour des partenaires non expérimentés, soit de renforcer la protection d'accès.

4.3.4.1 L'auto-démarrage

L'auto-démarrage est l'exécution automatique d'un service d'une application dès que l'intervenant est reconnu partenaire ayant le droit d'exécuter ce service. Cette caractéristique est intéressante lorsqu'un partenaire utilise toujours le même service (le seul qu'il puisse exécuter), ou lorsque l'un des services fait office de menu d'accueil accédant aux autres services, et facilitant l'utilisation de l'application par un intervenant opérateur.

L'auto-démarrage peut prendre différents aspects. Lorsque le service exécuté automatiquement se termine, le partenaire peut rester actif ou être rendu inactif (le partenaire PUBLIC devient alors actif). Dans ce dernier cas, le partenaire ne pourra donc jamais accéder à un autre service, pas

même aux commandes générales de la Carte Blanche. Cette caractéristique est plutôt réservée aux partenaires clients dépendants dont la principale activité est d'utiliser des services d'une application non-système. Elle semble inadaptée aux serveurs car ils ont besoin d'accéder aux services du système pour gérer leur application. De plus, il est préférable de ne pas donner cette caractéristique aux partenaires autonomes, elle serait irréversible car ils ne pourraient plus modifier eux-même cette caractéristique.

4.3.4.2 Le mécanisme de ratification et la réhabilitation de partenaire

Le mécanisme de ratification de la Carte Blanche permet de rendre inopérante toute tentative de présentation d'un partenaire après N présentations fausses consécutives de son authentifiant. Le partenaire est alors dans un état bloqué. Le fraudeur ne peut donc plus savoir si les authentifiants qu'il fournit à l'objet nomade sont bons ou mauvais et ne peut donc pas faire de recherche exhaustive. Le nombre N d'essais autorisés est défini par le responsable de l'association du partenaire. Ce mécanisme souvent associé à une authentification par code secret peut se généraliser à d'autres modes d'authentification.

Un partenaire peut être bloqué par un fraudeur avant que la Carte Blanche ne soit restituée à son titulaire, ou plus simplement par une erreur de manipulation. Dans ces cas, il est nécessaire de pouvoir débloquer ce partenaire pour lui permettre d'utiliser à nouveau ses services. Cette opération s'appelle la réhabilitation d'un partenaire et n'est autorisée qu'à des partenaires ayant un droit de regard sur le partenaire à réhabiliter. Pour les partenaires dépendants, le partenaire responsable de leur association peut décider de leur remise en activité. Par contre, pour les partenaires autonomes, il n'existe pas de telle hiérarchie. Il ne peuvent pas non plus se réhabiliter eux-même, car ils ne peuvent pas exercer leur responsabilité, puisqu'ils sont bloqués. La tâche de réhabiliter revient donc au partenaire de confiance de la Carte Blanche : le partenaire OFFICIEL.

4.3.5 Création d'un Nouveau Partenaire

L'association d'un nouvel intervenant à la Carte Blanche, émetteur ou non, par la création d'un nouveau partenaire. Il existe quatre procédures de création d'un nouveau partenaire, selon que le partenaire porteur ou TITULAIRE est défini, et selon que le partenaire à créer doit être indépendant ou dépendant, client ou serveur. Dans tous les cas, la catégorie du partenaire doit être spécifiée. Des créations de nouveau partenaire peuvent intervenir à tout moment de la phase d'exploitation de la Carte Blanche.

4.3.5.1 Association par le partenaire prédéfini PUBLIC

Cette procédure est la seule utilisable dès la phase d'exploitation de la Carte Blanche. En effet, ni le porteur, ni les émetteurs ne possèdent encore de partenaire sur l'objet nomade. Seul le partenaire PUBLIC qui est actif automatiquement à la mise sous tension de l'objet nomade permet d'exécuter la commande de création de partenaire. Cette procédure permet la création de partenaires indépendants et du partenaire porteur anonyme. Après que le porteur anonyme soit créé ou que la Carte Blanche soit titularisée, cette procédure n'est plus valide.

Le porteur remet sa Carte Blanche à l'intervenant qui peut ainsi exécuter lui-même la commande de création quand PUBLIC est actif. Ensuite cet intervenant définit lui-même dans un badge l'identifiant et l'authentifiant de ce partenaire. Il est donc le seul à connaître ces références et à pouvoir présenter ce partenaire par la suite. Ce partenaire est ainsi indépendant.

La création du partenaire porteur anonyme s'obtient en précisant la catégorie porteur dans la commande de création. Le porteur supposé légitime qui effectue le premier cette commande, est donc libre de définir l'identifiant (et l'authentifiant) du partenaire. Il peut donc rester anonyme et posséder les droits qui reviennent au porteur.

4.3.5.2 Association par le partenaire porteur

Cette procédure n'est possible qu'à partir du moment où le partenaire porteur anonyme existe. Dans ce cas, il remplace la procédure précédente pour la création de partenaire indépendant. Cependant, la commande de création est effectuée par le porteur qui définit lui-même la catégorie. Ensuite, pour que le partenaire créé soit indépendant, le partenaire actif devient alors PUBLIC pour permettre à l'intervenant d'entrer l'identifiant et l'authentifiant de son propre partenaire. Lorsque la commande est réalisée, PUBLIC est toujours actif et une commande supplémentaire donnée par l'intervenant ne bénéficiera pas des droits du porteur.

La conséquence de cette procédure est que la réhabilitation de ce partenaire ne peut être effectuée que par le partenaire porteur. Si ce degré de sécurité ne lui suffit pas, il doit être créé par la troisième procédure.

4.3.5.3 Association par le partenaire TITULAIRE

Cette procédure n'est possible qu'à partir du moment où la Carte Blanche est titularisée et donc que le partenaire TITULAIRE existe. Ensuite, la procédure est identique à la précédente mis à part que la commande est effectuée par le TITULAIRE. De plus, le partenaire porteur anonyme peut encore être créé par cette procédure.

La conséquence de cette procédure est que la réhabilitation de ce partenaire est exclusivement réalisable par le partenaire OFFICIEL. Les serveurs d'applications importantes ou critiques manipulant de l'argent en particulier, devront être créés par cette procédure.

4.3.5.4 Association par un partenaire serveur

Cette procédure permet à tout serveur de créer de nouveaux partenaires spécifiques pour leur application. Dans ce cas, leur identifiant, leur mode d'authentification et leur authentifiant sont définis par le serveur. Pour qu'un intervenant autre que l'émetteur d'application correspondant au serveur puisse présenter ce partenaire, il faut que le serveur lui délègue lui-même l'identifiant et l'authentifiant. Les partenaires associés par un serveur ne sont donc plus autonomes mais ils dépendent du serveur.

4.3.5.5 Responsabilité des partenaires

La responsabilité d'un partenaire influe sur les droits de modification des caractéristiques de ce partenaire, de réhabilitation et de suppression de ce partenaire.

La responsabilité d'un partenaire revient à l'intervenant qui l'a associé. Ce principe hiérarchique est projeté dans la Carte Blanche sur l'ensemble des partenaires. Le responsable hiérarchique d'un partenaire associé par un serveur est naturellement ce serveur. Par contre, un partenaire associé sous PUBLIC est son propre responsable. En effet, la responsabilité de tous les partenaires indépendants ne peut être remise au partenaire PUBLIC auquel tout intervenant a accès. Le principe de la responsabilité hiérarchique fait que les partenaires indépendants sont aussi autonomes.

4.3.6 Modification des attributs et Suppression d'un partenaire

La Carte Blanche étant évolutive, elle permet évidemment de modifier les attributs des partenaires existants ou de les supprimer.

4.3.6.1 Modification des attributs

La modification des attributs concerne le mode d'identification, l'authentifiant, l'auto-démarrage et les restrictions d'accès. Seul l'identificateur, la codification et la catégorie d'un partenaire ne sont pas variables.

4.3.6.2 Suppression de partenaire

La suppression de partenaire intervient lorsqu'un serveur veut retirer l'accès à la Carte Blanche à un intervenant auquel il a permis de se présenter, ou à un émetteur d'application qui enlève son application et qui n'a donc plus besoin d'être partenaire.

4.3.6.3 Droits de modification et de suppression

Le droit de modification et de suppression d'un partenaire revient au partenaire responsable de l'association de ce partenaire. Dans le cas d'un partenaire associé par un serveur, c'est ce serveur qui peut modifier les attributs du partenaire et/ou le supprimer. Dans le cas des partenaires autonomes, il est impensable que PUBLIC puissent modifier leurs attributs, et encore moins les supprimer. Cela signifierait que n'importe quel intervenant pourrait le faire. Donc, les partenaires associés par PUBLIC étant autonomes et responsables d'eux-même, peuvent s'auto-modifier et se supprimer.

4.4 Les Applications

4.4.1 Composition Générale d'une Application

Les applications nomades possèdent des caractéristiques et des contraintes particulières.

Le code des applications nomades est mémorisé dans une mémoire non-volatile exécutable. Il peut donc être exécuté sur place (code XIP : eXecute In Place) sans avoir à le charger en mémoire de travail. De même, les données des applications nomades se trouvent en mémoire non volatile adressable. Elles sont donc immédiatement adressable par le code de l'application et il est également inutile de constituer ces données sous la forme d'un fichier.

Les applications nomades ont des contraintes de temps et de ressources. Chaque service est souvent de courte durée pour ne pas pénaliser l'utilisateur dans ses déplacements. De plus, la mémoire de travail des objets nomades restant limitée, les applications ne devront allouer que le minimum de mémoire. Une application nomade n'est donc pas constituée d'un unique programme mais de plusieurs programmes correspondant aux différents services de l'application qui pourront être appelés séparément. Chaque programme n'utilisera que les ressources qui lui sont strictement nécessaires.

Le temps acceptable pour un traitement est variable suivant le service qu'il va rendre à l'utilisateur.

Les applications nomades ont leur code et leurs données inscrits et utilisés directement en mémoire non volatile. La protection doit donc être assurée de la même manière en statique comme en dynamique par des mécanismes de gestion de la mémoire. Le code de l'application n'est autorisée à modifier que la mémoire contenant ses données et ses variables. Il ne peut donc endommager aucune autre application, ni son propre code. Par contre, les données doivent être protégées par des mécanismes de commit et rollback [MR94] car un arrêt anormal d'un service peut laisser l'ensemble des données de l'application dans un état incohérent.

4.4.2 Architecture Interne d'une application dans la Carte Blanche

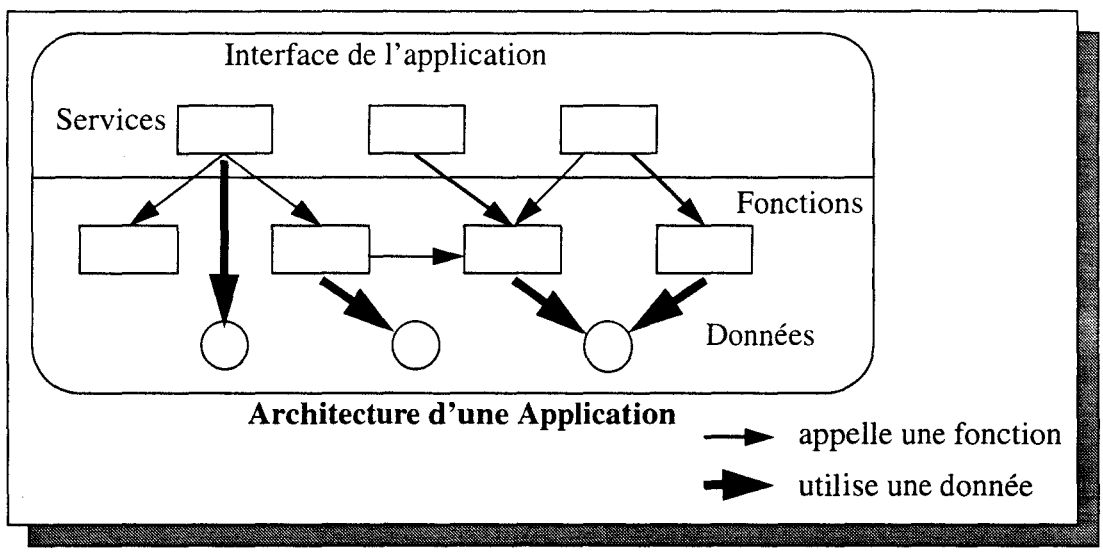
Les applications pour la Carte Blanche sont composées de plusieurs services que l'on peut appeler individuellement, et de données sur lesquelles travaillent ces services. Les services d'une même application peuvent avoir besoin de traitements communs. La Carte Blanche doit permettre la définition de fonctions partagées entre tous les services d'une application. Il est donc nécessaire d'ajouter ce nouvel élément d'application qui est compris dans le code de l'application.

4.4.2.1 Les Services de l'Application

Un service est un programme que l'on peut exécuter indépendamment des autres. Il peut appeler des fonctions de l'application. Il utilise des données particulières à l'application.

Une application est accessible uniquement à travers ses *services*. On ne peut jamais accéder directement ni à ses données, ni à ses fonctions. Les services forment donc l'interface de l'application.

FIGURE 12 Architecture d'une application pour la Carte Blanche



4.4.2.2 Les Fonctions de l'Application

Une fonction est un traitement indivisible d'une *application*. Elle est mise en oeuvre par un *service* ou une autre fonction de la même *application*.

Une fonction peut être partagée entre plusieurs services et autres fonctions de la même application. Les fonctions ont accès aux *données de l'application*.

4.4.2.3 Les Données de l'Application

Une donnée est une information appartenant à une *application*. Toutes les données d'une application sont communes à tous les *services* et *fonctions* de l'application et accessibles par eux seuls.

L'autre particularité de ces données est la suivante. Elles conservent leur valeur entre deux utilisations d'un service de la même application. La valeur initiale de ces données doit être fournie à l'installation de l'application.

4.4.3 Gestion des Applications

La gestion des applications comprend l'installation, la mise à jour et la suppression de l'application mais aussi l'établissement et le retrait des autorisations d'utiliser des services de l'application aux partenaires de la Carte Blanche.

4.4.3.1 Installation d'une application

L'installation d'une application consiste à charger dans la Carte Blanche tous les éléments constitutifs d'une application et à construire les structures de sécurité nécessaires à l'utilisation sécurisée de l'application.

Deux des composantes de l'application, les services et les fonctions, sont du code exécutable. Les fonctions sont utilisables par tous services ou fonctions de la même application. Les services sont alors des programmes qui sont indépendants mais pas distincts. Pour faciliter le développement de l'application d'une part, et simplifier la protection de celle-ci dans l'objet nomade d'autre part, le code de l'application entière est fourni en un seul bloc de code pour lequel l'émetteur fournira le point d'entrée de chaque programme correspondant aux services.

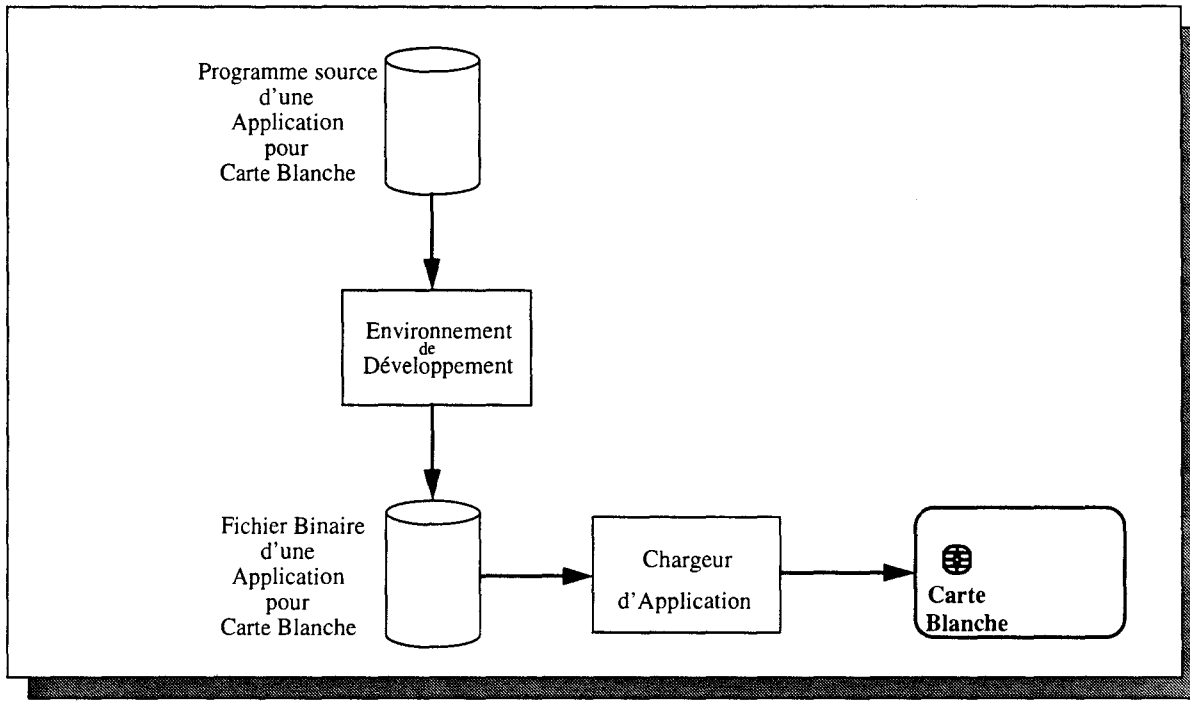
Pour chaque application, l'émetteur d'application doit fournir à la Carte Blanche :

- l'ensemble des données persistantes de l'application. Ces données ont besoin d'une valeur initiale pour la première utilisation de l'application. La Carte Blanche ne considérant pas le type de chaque donnée, les données sont fournies sous la forme d'une suite d'octets dont on précise le nombre
- le code de l'application. Chargé sous forme d'un seul bloc binaire, il comprend les services et les fonctions de l'application
- pour chaque service, le nom d'appel du service et son point d'entrée dans le bloc de code

D'autres éléments sont à fournir au système de la Carte Blanche mais nous rentrerons dans les détails de l'installation d'une application dans le chapitre 5 : Le Système d'Exploitation pour la Carte Blanche.

Toutes ces informations doivent être écrites dans un fichier binaire qui sera ensuite transmis à chacune des Cartes Blanches dont le porteur souhaite disposer de l'application. En effet, la Carte Blanche est un environnement de programmes et non pas de programmation et l'application doit être développée à l'extérieur de la Carte Blanche par l'émetteur. La constitution de ce fichier peut être aidée par un compilateur croisé qui permet à l'émetteur de développer son application dans un langage évolué (figure 13 de la page 69)

FIGURE 13 Développement et Installation d'une application pour la Carte Blanche



La procédure d'installation commence par la remise de la Carte Blanche, par son porteur, à l'émetteur d'application de son choix. Il peut ainsi présenter l'identifiant et l'authentifiant de son partenaire serveur lui-même et, puisqu'il est serveur, il peut installer son application. Il devient responsable et possède le contrôle de son application. Ses droits sur cette application sont régis par le principe de la Propriété.

Principe de la Propriété :

- Le partenaire serveur qui a installé l'application est le propriétaire de cette application. Il en est responsable
- Ce partenaire décide d'autoriser ou non l'accès aux différents services de son application aux autres partenaires et/ou applications de la Carte Blanche
- Il est le seul à pouvoir supprimer ou mettre à jour son application
- Il peut toujours accéder à tous les services de son application. Sitôt l'installation, il est le seul à pouvoir le faire

Ce principe permet de séparer le concepteur de l'application qui peut donc gérer son application comme il l'entend, des différents utilisateurs de l'application qui ne pourront qu'utiliser tout ou partie de l'application.

4.4.3.2 Suppression d'application

La suppression d'application retire de la mémoire de la Carte Blanche les données, le code et la liste des services de l'application ainsi que toutes les structures qui servaient à maintenir la sécurité. Tous les droits d'utilisation encore attribués à des partenaires sont automatiquement détruits.

4.4.3.3 Mise à jour d'application

La mise à jour d'une application est réalisable par son propriétaire seulement, éventuellement sur demande du porteur.

La Mise à Jour d'application de la Carte Blanche permet d'ajouter, modifier ou retirer des *services*, *fonctions* ou *données* de l'application. Du fait de l'entrelacement des services et des fonctions dans le code de l'application et de l'accès direct à toutes les données par ce code, l'ajout et le retrait d'un élément de l'application ne peut s'effectuer que par un remplacement total de l'application. De plus, cela permet de ne pas fragmenter l'application dans la mémoire pour qu'elle se présente comme une application que l'on installe pour la première fois et pour ne pas compliquer la protection dynamique de l'application. Il faut donc récupérer la valeur courante des données de l'application pour les restituer dans la nouvelle version.

La Mise à Jour d'application est différente d'une suppression suivie d'une nouvelle installation. Elle conserve les valeurs courantes des données restant valides. Elle ne supprime pas les droits d'utilisation précédemment attribués à des partenaires. Elle doit donc conserver les mêmes nom d'appel des services encore valides pour que l'interface de l'application vis à vis des utilisateurs de la Carte Blanche soit compatible.

4.4.4 L'offre : attribution de droits d'exécution et de coopération

L'attribution de droits d'exécution et de coopération d'application dans la Carte Blanche est le complément du principe de propriété. Ce principe détermine qui gère l'application et qui est le seul à pouvoir l'utiliser par défaut, alors que les attributions permettent au serveur propriétaire d'autoriser ou non l'utilisation de tout ou partie de l'application individuellement à différents partenaires ou applications de la Carte Blanche.

4.4.4.1 Offre de service

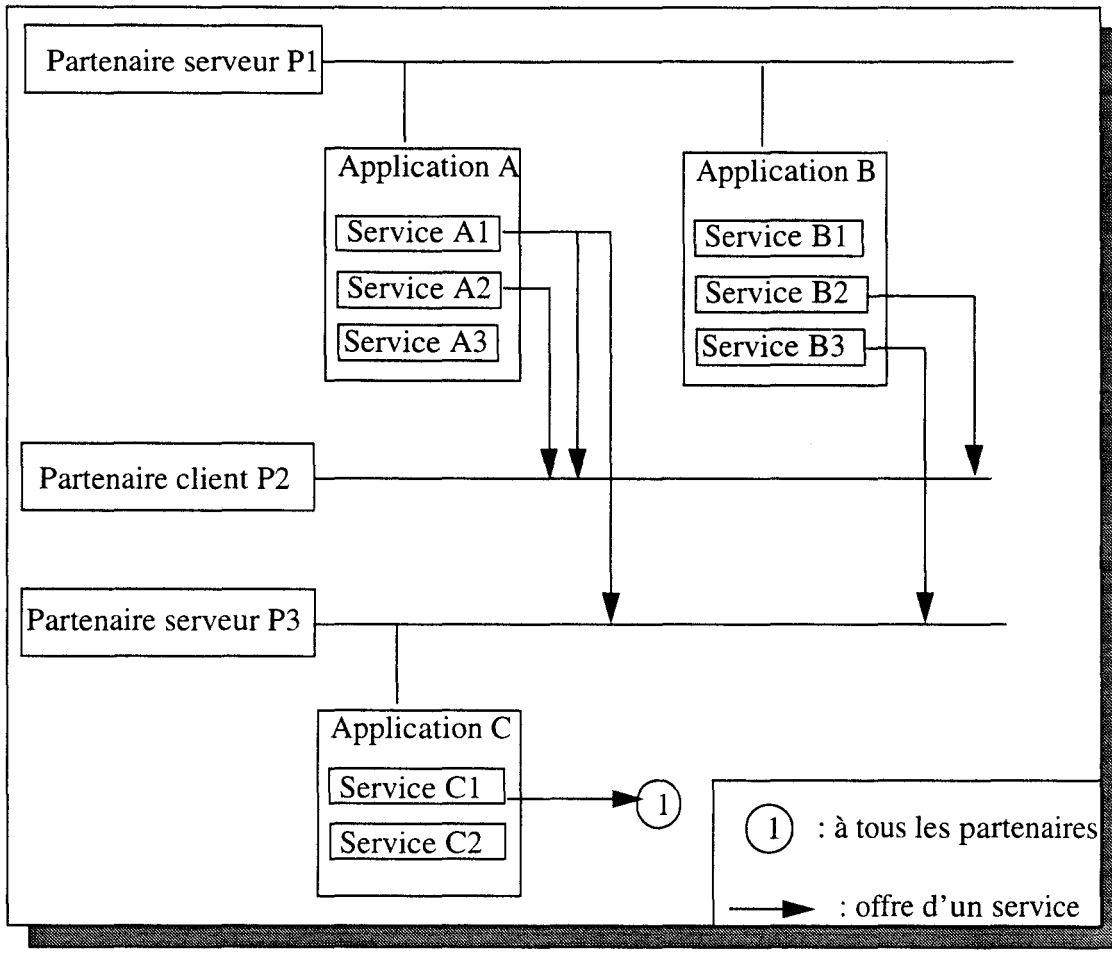
Un même mécanisme permet à la fois de donner un droit d'exécution à un partenaire et un droit de coopération à une application. Il s'agit de l'offre de service. Une offre est le droit d'utiliser un service que le propriétaire de l'application concernée donne à un autre partenaire. Chaque service est individuellement offert à un seul partenaire à la fois. Ce partenaire est obligatoirement existant.

L'offre d'un service permet au partenaire bénéficiaire d'appeler directement ce service comme s'il s'agissait d'un service de sa propre application. Ensuite, un serveur bénéficiaire peut également utiliser ce service par l'intermédiaire d'un service de son application. L'offre autorise donc aussi la coopération entre applications.

Une offre peut être attribuée à tout moment de l'exploitation de la Carte Blanche. Elle peut être également retirée à tout moment de l'exploitation de la Carte Blanche.

L'offre n'est pas transmissible : le partenaire bénéficiaire de l'offre d'un service ne peut lui-même offrir ce service. Seul le propriétaire de l'application à laquelle appartient ce service peut décider d'offrir ou non ce service à un partenaire.

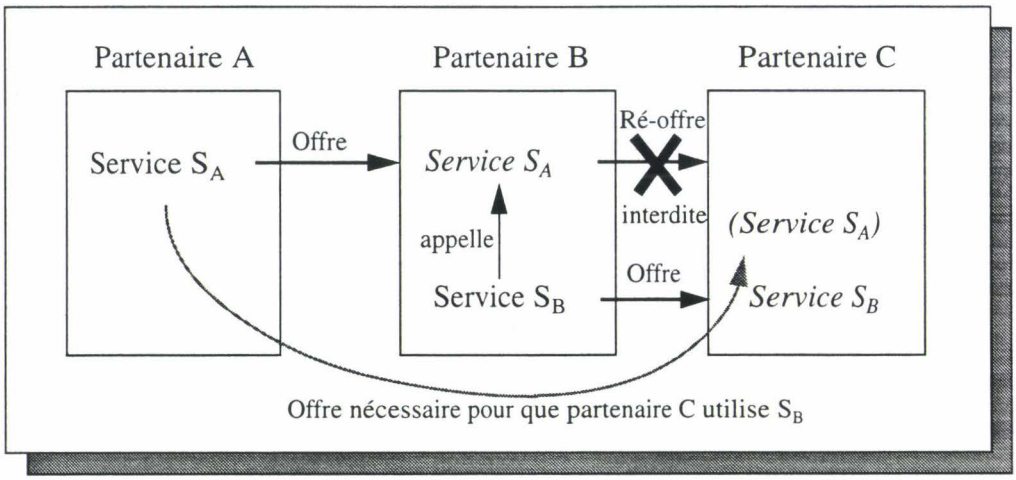
FIGURE 14 Le mécanisme d'offre de service



Par contre, dans le cas où un service offert est utilisé dans une autre application, le propriétaire de cette application a logiquement le droit d'offrir l'un de ses services à un autre partenaire. Prenons l'exemple de la figure suivante. Le partenaire A est propriétaire du service S_A qu'il offre au partenaire B. B peut donc utiliser S_A mais ne peut le ré-offrir au partenaire C. B, également serveur, possède un service S_B qui appelle S_A. B peut exécuter complètement son service car il a aussi le droit d'utiliser S_A. Maintenant B, propriétaire de son service S_B, offre celui-ci au partenaire C. Cette seconde offre donne la possibilité au partenaire C d'appeler le service S_B mais ne doit pas l'autoriser à exécuter le service S_A ni directement, ni indirectement. Lorsque C appellera S_B, ce service commencera à s'exécuter normalement mais produira une erreur à l'appel du service S_A. Ainsi cette méthode ne permettra pas de constituer facilement un cheval de Troie entre un émetteur d'application et un quelconque partenaire. Pour que le partenaire C puisse exécuter entièrement le service S_B, il faudra que le partenaire A, propriétaire de S_A, lui offre personnellement ce service. Ce mécanisme assure donc le contrôle complet des offres des services par le propriétaire de l'application même dans le cadre d'une coopération entre applications.

L'offre entre les deux serveurs concrétise précisément l'accord de coopération entre applications. Les deux autres offres servent uniquement de simples droits d'exécution de service à des partenaires.

FIGURE 15 Mécanismes d'offre de service pour la coopération



Pour obtenir ce fonctionnement, la Carte Blanche vérifie dynamiquement que le partenaire dispose réellement de l'offre du service à chaque fois qu'il l'appelle directement ou à travers un autre service.

4.4.4.2 La Modification des Offres dans le Temps

Des offres de services peuvent être ajoutées ou retirées à un partenaire à tout moment de la phase d'exploitation de la Carte Blanche. Par exemple, on peut avoir besoin d'offrir un service à un partenaire qui n'existait pas à l'installation de l'application, ou à un partenaire nouvellement associé. On peut aussi vouloir en retirer une offre à un partenaire en fin de contrat avec l'émetteur d'application.

Or, l'offre et le retrait ne peuvent être effectués que par le propriétaire du service concerné. Cela signifie que ce propriétaire doit se présenter lui-même à chaque fois qu'une modification des offres de ses applications intervient. Le problème est qu'il faut que le porteur vienne à lui pour qu'il puisse le faire. Pour cette raison, les applications doivent être conçues de la manière la plus attractive possible pour le porteur afin que ce dernier utilise le plus possible sa Carte Blanche. Cela aura pour effet d'augmenter la probabilité d'intervention possible des divers émetteurs.

Les offres à un partenaire pouvant être retirées à tout moment, la vérification, par le serveur, de la possession de l'offre des services que va utiliser une application, parait inutile au moment de l'installation. De plus, le serveur pourra obtenir ces offres par la suite. Néanmoins, l'émetteur d'application doit prévoir dans son application qu'une demande de service offert peut échouer. Pour cela, la Carte Blanche lui donne le moyen de vérifier dynamiquement la disponibilité d'une offre pour le partenaire utilisant le service en cours.

4.4.4.3 Une autre approche de l'offre de service

Dans un projet commun de notre laboratoire de recherche RD2P, nous avons été amené à intégrer les concepts du S-Shell dans la Carte Blanche [PPT95]. Le S-Shell est un moyen de décrire et de mettre en oeuvre un système de contrôle d'accès performant et évolué pour tout système informatique [Trane95]. Pour la Carte Blanche, le S-Shell permet de donner la description de la sécurité d'une application par son propriétaire au moment de l'installation.

Le S-Shell nous a permis de pallier l'inconvénient des applications coopérantes. En effet, le propriétaire du service intermédiaire ne peut pas offrir ce service aussi librement qu'il le veut, car pour l'utiliser complètement, l'utilisateur final doit également obtenir l'offre du service appelé. La procédure de l'offre de service coopérant paraît complexe et il pourrait être utile d'avoir confiance dans le propriétaire de l'application intermédiaire pour les offres qu'il promulgue. Les bénéficiaires de ces offres pourraient utiliser le service intermédiaire mais également le service appelé sans offre de la part de son propriétaire. Nous pensons que les deux modes d'offres doivent exister et que le choix de l'offre revient au propriétaire du service appelé. Dans notre étude, nous avons défini deux modes d'offre :

- **l'offre sans délégation** qui est du même ordre que l'offre définie initialement dans la Carte Blanche. Seul le partenaire bénéficiaire peut utiliser ce service directement ou par l'intermédiaire d'un service de ses applications ou d'un autre service offert. Il ne peut pas transmettre cette offre à un autre partenaire
- **l'offre avec délégation**. Le bénéficiaire d'une telle offre peut utiliser ce service directement ou par l'intermédiaire d'un service de ses applications ou d'un autre service offert. Il ne peut toujours pas offrir explicitement ce service à un autre partenaire. Par contre, lorsqu'il offre l'un des services de son application à un partenaire et que ce service appelle le service offert avec délégation à son propriétaire, il offre en même temps le service qu'il a lui même reçu mais uniquement si il est utilisé à travers le service qu'il a explicitement offert. Le serveur qui accorde ce mode d'offre délègue partiellement le droit d'offrir le service correspondant à un autre serveur en qui il a suffisamment confiance.

4.5 La Titularisation

La Titularisation doit être réalisée par le partenaire prédéfini OFFICIEL. Elle consiste à créer dans la Carte Blanche, une structure de données contenant les coordonnées du titulaire. Cette structure est composée de deux parties. L'une est déterminée à la titularisation et ne peut plus être modifiée par la suite. Cette partie est donc destinée à recevoir le nom et le prénom, ou la raison sociale du titulaire. L'autre partie peut, quant à elle, être modifiée après la titularisation mais uniquement par l'émetteur. Elle est donc destinée à recevoir l'adresse ou le siège social du titulaire.

Cette structure est une information globale à la Carte Blanche et peut donc être lue par tous les partenaires, et particulièrement par PUBLIC, grâce à une commande du système. Elle est indépendante de toute application, donc la titularisation ne change rien aux applications précédemment installées. Cependant, toutes les applications peuvent utiliser ces informations mais uniquement pour les lire, et jamais pour les modifier. C'est notamment le cas des applications nominatives.

La Titularisation définit un partenaire particulier lié au système, le partenaire TITULAIRE, en créant son badge. Ce partenaire est de la catégorie «porteur». Quoiqu'étant un partenaire dépendant du système, le choix de l'authentifiant est laissé au soin du porteur présumé légitime qui vient titulariser son objet nomade.

La prise en charge du titulaire par le système d'exploitation a plusieurs raisons. La première est que le titulaire est une information générale à la Carte Blanche. Elle est indépendante de toute application. La seconde est le besoin de confier l'inscription du titulaire dans la Carte Blanche à

une autorité officielle. Enfin, la troisième est d'assurer le mécanisme unique d'inscription du titulaire dans la Carte Blanche.

4.6 L'Application Système

L'application «système» est le SHELL ou interpréteur de commandes de la Carte Blanche. Elle n'est pas tout à fait une application telle que nous les avons décrites dans les paragraphes précédents. D'abord, cette application est résidente dans la Carte Blanche. Elle existe dès la fin de la fabrication. Ensuite, son code est étroitement lié au système d'exploitation qui fonctionne en permanence dès que la Carte Blanche est alimentée et ses services constituent l'interface de commandes de l'objet nomade. Son propriétaire est le partenaire SYSTEME. Ce partenaire ne pouvant jamais être présenté, il est impossible de modifier ou de supprimer ces applications, et d'attribuer de façon anormale, à un quelconque partenaire, tout service particulier de cette application.

Toutes les commandes du système d'exploitation sont des services de l'application SHELL du partenaire SYSTEME. En principe, ces services ne sont accessibles qu'au propriétaire. Pour que chacun des partenaires de la Carte Blanche, suivant qu'il est client ou serveur, partenaire porteur ou partenaire TITULAIRE, dispose réellement des fonctionnalités décrites dans les paragraphes précédents, le partenaire SYSTEME lui offre les services adéquats de son application SHELL. Cette offre est faite statiquement pour les autres partenaires du système déjà présents à l'installation, ou dynamiquement, à l'association de nouveaux partenaires. Nous allons décrire maintenant la liste des services de chaque application système ainsi que les offres de ces services aux divers partenaires. De plus, cette description est également un bon récapitulatif des caractéristiques fonctionnelles de la Carte Blanche.

Nous avons classé ces commandes en trois catégories de services non exclusives. Ce découpage se prête bien à la répartition des services en fonction de la catégorie des partenaires auxquels ils sont offerts dynamiquement par le partenaire SYSTEME.

4.6.1 La gestion des applications

Ces services permettent à l'émetteur d'application de manipuler son application au niveau de sa structure (données, code, services) et de ses liens de coopération. Pour cette raison, ces services sont offerts dynamiquement à tous les partenaires serveurs. Ces services sont :

- *Installer une application* : ce service charge dans la Carte Blanche les éléments constitutifs d'une application (ses données, son code, la liste de ses services). Il est offert dynamiquement à tout nouveau partenaire serveur pour lui permettre d'installer son application
- `Supprimer une application` : ce service retire de la Carte Blanche les éléments constitutifs d'une application (les données, le code et les services). Il est offert dynamiquement à tout nouveau partenaire serveur pour lui permettre de supprimer sa propre application
- `Mettre à jour une application` : ce service lit la valeur actuelle de chaque donnée d'une application puis remplace globalement les données, le code et les servi-

ces de cette application par une nouvelle version. Ce service est offert dynamiquement aux partenaires serveurs pour qu'ils puissent faire évoluer leur application

- `Offrir un service à un partenaire` : ce service permet à un émetteur d'application d'autoriser un partenaire et ses applications à utiliser un service de son application. Ce service est offert dynamiquement à tous les serveurs pour qu'ils puissent définir les liens de coopération de leur application
- `Retirer l'offre d'un service à un partenaire` : ce service permet à un émetteur d'application de ne plus autoriser un partenaire et ses applications à utiliser un service de son application. Ce service est offert dynamiquement à tous les serveurs pour qu'ils puissent modifier les liens de coopération de leur application

4.6.2 La gestion des partenaires

Ces services sont d'une part, complémentaires aux services précédents pour l'administration d'une application car ils permettent aux serveurs de déterminer des partenaires spécifiques à leur application, auxquels ils offriront des services particuliers. D'autre part, ces services sont des services d'ouverture car ils permettent à des intervenants non référencés par avance de devenir partenaires et de pouvoir, s'ils sont serveurs, installer de nouvelles applications en toute indépendance. Ces services sont :

- `Associer un partenaire` : ce service permet de créer un nouveau partenaire client ou serveur. Ce service est offert dynamiquement à tous les serveurs pour qu'ils puissent associer les partenaires nécessaires à leur application et qui sont ainsi dépendants du serveur. Ce service est momentanément offert au partenaire PUBLIC pour permettre la création de partenaires indépendants, jusqu'à la création d'un partenaire de catégorie serveur. Ensuite cette offre est réservée aux partenaires «porteur» et TITULAIRE.
- `Dissocier un partenaire` : ce service permet de supprimer un partenaire de la Carte Blanche. Ce service est offert dynamiquement à chaque serveur pour lui donner la possibilité de supprimer un partenaire qu'il a lui même créé, parce qu'il n'est plus utile à son application. Ce service est offert dynamiquement à tous les partenaires indépendants pour qu'ils puissent se retirer de la Carte Blanche
- `Modifier les attributs d'un partenaire` : ce service permet de changer le mode d'authentification, l'authentifiant, et autres caractéristiques liées au login. Il est offert dynamiquement aux serveurs qui peuvent ainsi influer sur le comportement de ce partenaire dont ils ont la responsabilité. Ce service est offert dynamiquement à tous les partenaires indépendants pour qu'ils puissent modifier eux-même leurs attributs
- `Réhabiliter un partenaire` : ce service permet de débloquer un partenaire préalablement rendu indisponible par le mécanisme de ratification. Il est offert dynamiquement à tous les partenaires serveurs qui, étant responsables hiérarchiques des partenaires qu'ils ont créés, sont compétents pour décider de la réhabilitation ou non de ces partenaires. Ce service est également offert aux partenaires «porteur» et «TITULAIRE» pour qu'il puisse particulièrement réhabiliter les partenaires bloqués et qui ont été créés à partir de PUBLIC ou du partenaire «porteur», et au partenaire OFFICIEL pour ceux créés à partir du TITULAIRE

4.6.3 Les services de sécurité

Les services de sécurité sont des services qui vont permettre au porteur de contrer des utilisations frauduleuses de sa Carte Blanche. Ces services sont :

- `Purger les partenaires natifs` : ce service permet de retirer de la Carte Blanche tous les partenaires qui ne possèdent ni application ni offre de service car ils sont ainsi inutiles et encombrent la mémoire.

  Ce service est offert à PUBLIC pour qu'il puisse libérer de la mémoire inutilement occupée afin d'associer un nouveau partenaire autonome.
- `Neutraliser un application` : ce service permet d'interdire l'utilisation de tous les services d'une application.

  Ce service est offert dynamiquement aux partenaires «porteur» et TITULAIRE lorsque ceux-ci est créé
- `Bloquer la Carte Blanche` : ce service permet de terminer le cycle de vie de la Carte Blanche en interdisant toutes utilisations de services d'applications dans la Carte Blanche.

4.6.4 les services généraux d'utilisation du système

Ce sont les services de l'application SHELL qui interviennent en phase d'utilisation et non pas dans la phase d'administration de la Carte Blanche.

Ces services sont offerts dynamiquement à tous les partenaires de la Carte Blanche. Ce sont :

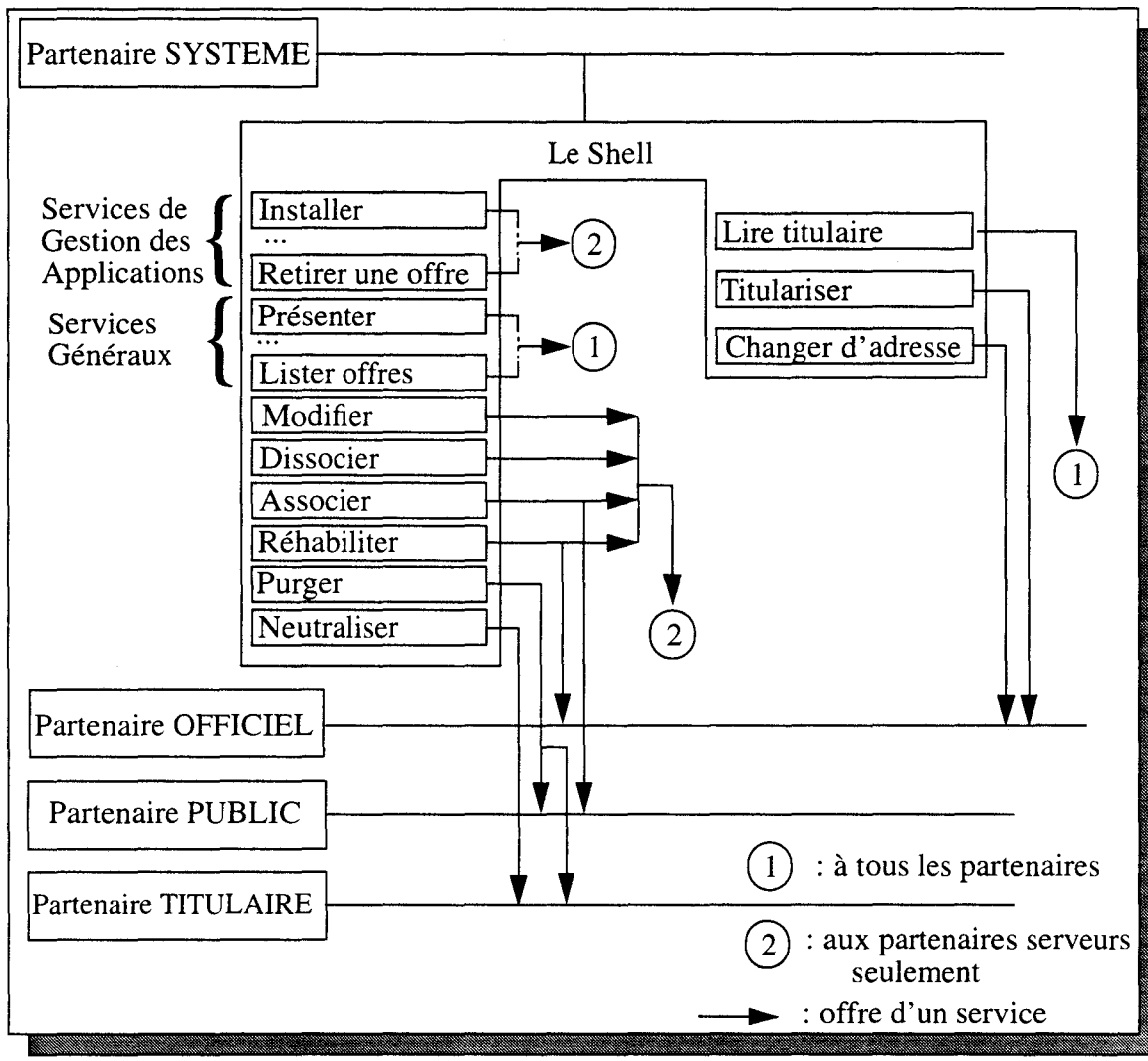
1. **les services de positionnement de l'intervenant**
   - `Présenter un partenaire` : ce service permet de changer de partenaire actif (particulièrement quand PUBLIC est déjà présenté);
   - `Qui suis-je` : ce service permet de vérifier l'identifiant du partenaire actif à un moment donné.
2. **les services de consultation**

   Ils permettent aux partenaires de connaître ce dont ils disposent individuellement.
   - `Lister les applications disponibles` : ce service permet de connaître précisément les applications actuellement disponibles sur la Carte Blanche, à savoir, pour chacune d'elles, son nom, sa codification et la signature de son émetteur. Par contre, il n'indique pas la façon d'y accéder pour des raisons de sécurité.

     Ce service est offert dynamiquement à tous les partenaires de la Carte Blanche car il peut servir aux clients pour qu'ils recherchent les services d'application qu'ils pourraient obtenir auprès des émetteurs d'application, aux serveurs pour qu'ils vérifient l'environnement dans lequel ils vont installer et faire configurer leur application, et particulièrement au partenaire PUBLIC pour qu'un guichet multi-applications puissent connaître les services qu'il va pouvoir utiliser sur une Carte Blanche donnée.
   - `Lister les services offerts` : ce service permet au partenaire actif de connaître l'ensemble des services qui lui sont actuellement offerts. Ainsi, ce partenaire peut savoir les offres nouvellement attribuées ou retirées. Les services des applications système dont l'offre est invariable dans le temps ne sont pas pris en compte par ce service.

Ce service est dynamiquement offert à tous les partenaires car ils peuvent tous être bénéficiaires de l'offre d'un service

FIGURE 16 Offres des principales commandes aux partenaires



4.6.5 La Gestion du Titulaire

Les services de gestion du titulaire sont Titulariser, Lire titulaire, et Changement d'adresse.

- *Titulariser* : Ce service permet d'inscrire le titulaire dans la Carte Blanche. Les références du titulaire sont son nom, son adresse et un numéro de pièce d'identité pour une personne ou sa raison sociale, son siège social et un numéro de SIRET pour une société. Il associe également le partenaire particulier TITULAIRE.

  Le service Titulariser est utilisable une seule fois pendant la phase d'exploitation de la Carte Blanche

  Ce service est offert au partenaire OFFICIEL uniquement.
- *Lire titulaire* : Lire titulaire est un service qui permet à tous les partenaires de connaître les références du titulaire. Elle permet de retrouver le titulaire pour lui rendre sa Carte Blan-

che. Elle permet à un émetteur d'application nominative de vérifier la concordance du titulaire de la Carte Blanche avec le destinataire de l'application, avant l'installation.

Ce service est donc offert dynamiquement à tous les partenaires de la Carte Blanche

- *Changer d'Adresse* : Le service Changer d'Adresse donne la possibilité de modifier la partie adresse ou siège social de la référence du titulaire.

  Ce service est offert au partenaire OFFICIEL seulement

4.7 Les Contraintes Induites par le Modèle

Le modèle de la Carte Blanche apporte une grande souplesse et une largesse d'utilisation. Nous allons montrer dans ce paragraphe comment la sécurité est assurée dans ce modèle dans lequel un partenaire peut librement s'associer, installer son application et coopérer avec d'autres applications, sans que soit vérifiées la loyauté de ses objectifs et la correction de son code.

4.7.1 La Sécurité

Les fondements de la sécurité de la Carte Blanche sont identiques à ceux des cartes à microprocesseur. Ils concernent d'une part les fonctions de base du système, et d'autre part les moyens de communication.

4.7.1.1 Les fonctions de base

Les fonctions de base sont les services de l'application système. Cette application est installée avec le système d'exploitation en phase de fabrication, donc dans l'environnement sécurisé du fabricant. Ensuite, elle n'est plus modifiable, car elle est inscrite dans la mémoire ROM de la Carte Blanche. De plus, le serveur de cette application, le partenaire SYSTEME, ne peut pas se présenter pour attribuer des droits supplémentaires à un partenaire en particulier. La sécurité apportée par le système ne peut donc être altérée.

4.7.1.2 Protection des applications

Les aspects multi-applicatif et multi-émetteur de la Carte Blanche signifient que les applications sont indépendantes. L'indépendance des applications est obtenue par le cloisonnement des applications. L'exécution d'un service d'une application ne peut pas sauter du code de l'application à celui d'une autre ni accéder directement aux données d'autres applications. Cette protection est effectuée au niveau du système d'exploitation. C'est pourquoi elle sera traitée dans le chapitre 5 : Le système d'exploitation de la Carte Blanche.

Par conséquent, l'assurance du cloisonnement minimise le problème de l'installation d'application libre. L'honnêteté de l'application et la qualité de son code ne sont pas contrôlées pour ne pas alourdir la procédure d'émission. L'application peut mal fonctionner ou peut être destinée à frauder, le cloisonnement empêchera tout accès irrégulier à l'ensemble de la Carte Blanche. Nous pouvons espérer que la facilité d'émettre une application incitera au développement de nombreuses applications pour la Carte Blanche, ce qui augmentera l'intérêt et le rendement du concept de Carte Blanche.

La Carte Blanche permet à tout partenaire de connaître les applications déjà installées et d'obtenir des informations permettant de les authentifier si nécessaire. Cependant, la connais-

sance d'une application ne permet pas d'y accéder. Il faut encore connaître un des partenaires autorisés à l'utiliser. C'est le cas des intervenants dédiés tels que les distributeurs automatiques de billets. Il n'est pas possible d'obtenir la liste de tous les partenaires et d'essayer de la corréler avec celle des applications. Si par hasard, un fraudeur trouvait le nom d'un partenaire, il lui faudrait encore réussir l'authentification.

L'implantation de l'application est quant à elle entièrement gérée par la Carte Blanche. L'emplacement de chaque service et de chaque donnée d'une application n'est connu que du système seul et varie d'une Carte Blanche à une autre suivant son ordre d'arrivée. L'unique moyen d'accéder à un service pour un intervenant est donc de le demander au système.

4.7.1.3 Interface contrôlée

Ensuite, la seule interface de la Carte Blanche est son (ou ses) port(s) de communication. Elle est entièrement sous le contrôle du système d'exploitation. Ce dernier autorise les entrées/sorties sur les ports uniquement pour demander un service et échanger des données pendant l'exécution de ce service. Lorsqu'une demande de service arrive à l'objet nomade, elle est faite pour le compte d'un seul partenaire (principe du login). La Carte Blanche vérifie, dans l'ordre de priorité :

- si ce service fait partie des services du système offerts à la catégorie du partenaire
- si il s'agit d'un service d'une application appartenant au partenaire (principe de la propriété)
- si ce service lui est personnellement offert

Le service ainsi reconnu peut être correctement lancé par le système qui connaît son emplacement en mémoire. Le système d'exploitation étant le passage obligé pour accéder à un service, il n'est donc pas possible d'exécuter d'autres services que ceux précisés ci-dessus, ni d'exécuter un programme d'une autre façon.

Lors de l'exécution d'un service, le (ou les) port(s) de communication ne permet(tent) pas de violer les ressources de l'objet nomade, notamment sa mémoire. Le service ne lit pas ni n'écrit directement sur les ports. Il demande au système s'il y a des données en entrée pour lui ou d'envoyer des données à l'intervenant. Le système d'exploitation assure que les données reçues seront retransmises au service en cours d'exécution, et que les données émises par le service seront envoyées à l'intervenant. La Carte Blanche interdit l'espionnage et la modification des données et du code des autres applications via les entrées/sorties puisque d'une part, l'application ne peut émettre que des informations lui appartenant, et d'autre part, l'application peut uniquement mémoriser les informations qu'elle reçoit dans son espace de données. De ce fait, même les entrées/sorties d'une application installée librement et donc dont la finalité n'est pas déterminée, n'est pas source d'affaiblissement de la sécurité du système.

L'unique mode de communication de la Carte Blanche ne permet donc pas de court-circuiter la protection du système d'exploitation.

4.7.1.4 Sécurité relative aux offres de service

L'offre de service requiert du point de vue du serveur comme de celui du bénéficiaire un certain nombre de garanties. Un serveur peut offrir à tous les partenaires de la Carte Blanche un ser-

vice banal, mais il veut n'offrir un service important qu'à des partenaires de confiance. Dans ce cas, la décision de coopérer provient de l'établissement d'un contrat de collaboration entre l'émetteur de l'application d'une part, et un intervenant ou un autre émetteur d'autre part, correspondant à un partenaire. Ce contrat externe à la Carte Blanche est à l'origine de la confiance entre les partenaires. La Carte Blanche ne permet pas, pour des raisons de sécurité, de connaître des informations sur les partenaires qu'elle détient. Le contrat externe permettra donc de livrer l'identificateur du partenaire à indiquer lors de l'offre du service.

Le serveur peut vouloir que son service soit utilisé directement par un partenaire et ne soit pas inclus tôt ou tard dans une autre application. Dans ce cas, il peut offrir son service uniquement à des partenaires clients.

Si le serveur exige une confiance absolue dans le bénéficiaire de l'offre de son service, la Carte Blanche lui permet d'associer ses propres partenaires, clients ou serveurs. Comme il en sera le responsable hiérarchique, il connaîtra son identificateur et son authentifiant, et pourra le gérer comme il le désire. Offrir un service à un tel partenaire est donc très rassurant.

Ensuite, que le service soit offert à un partenaire de confiance ou à tous les partenaires, le serveur peut souhaiter suivre la bonne ou mauvaise utilisation de ses services, notamment pour détecter leur utilisation frauduleuse. Pour cela, un service peut demander à la Carte Blanche quel partenaire est en train de l'utiliser directement ou indirectement, et dans ce dernier cas, par quelle application coopérante il a été appelé.

Le fait d'être bénéficiaire de l'offre d'un service n'oblige en rien de l'utiliser. Ce partenaire doit vérifier l'origine du service et peut obtenir les informations disponibles sur l'application de ce service.

Une application utilisant un service offert doit prévoir que ce service peut ne pas lui être offert dès l'installation de l'application ou qu'il peut lui être retiré inopinément entre deux utilisations. La Carte Blanche permet donc à une application de vérifier la disponibilité d'une offre avant d'appeler réellement le service correspondant. Lors de la coopération, le service appelé est exécuté séparément du service appelant afin de préserver l'indépendance et la confidentialité de chaque application. Le résultat est transféré par l'intermédiaire du système, comme si les applications communiquaient avec l'extérieur.

4.7.2 Le Nommage des Eléments Manipulés par le Système

La Carte Blanche manipule des partenaires, des applications et des services. Certains d'entre eux existent dès la fabrication et sont identiques dans toutes les Cartes Blanches. Mais tous les autres ne sont pas connus d'avance et varient d'un objet nomade à l'autre.

4.7.2.1 Identification d'un partenaire

Les partenaires sont soumis au principe du login. Il leur faut donc un identifiant et un authentifiant. L'identifiant est un nom, en général le nom de l'intervenant qui l'utilise (ex. : Dupond, Banque, Billetterie). Tous les partenaires d'une même Carte Blanche doivent avoir un nom différent, qu'ils soient autonomes ou dépendants, qu'ils soient clients ou serveurs. En effet, le mécanisme de login accède à tous les partenaires sur un même plan et il lui faut donc les distinguer tous.

Ce nom est donné librement à l'association. Ainsi, n'importe quel intervenant a le droit d'appeler son partenaire «Banque». On ne peut donc avoir confiance dans le nom d'un partenaire. Ce problème est déjà pris en compte dans le principe du login où tout partenaire est authentifié après identification. Le véritable problème est qu'un authentique établissement bancaire ne pourrait plus utiliser ce nom lorsque le porteur lui demandera d'installer son application. Par la suite, suivant la Carte Blanche du porteur qui veut utiliser un service bancaire, l'intervenant de la banque devra utiliser des noms différents pour se logguer.

Pour ne pas alourdir la procédure d'association de la plupart des partenaires, le libre choix des identifiants est conservé. Mais pour offrir une facilité et une sécurité supplémentaires, un partenaire peut être associé officiellement avec une codification unique. Dans ce cas, pour qu'un intervenant soit sûr d'être identifier, il fournira la codification plutôt que l'identifiant du partenaire. Cette codification est gérée par un organisme indépendant en qui tout le monde a confiance (Trusted Third Party) et qui délivre ce deuxième identifiant unique : l'**organisme certificateur**. La codification est intégrée optionnellement lors de l'association du partenaire au choix de l'intervenant ou du partenaire qui réalise l'association. Parce qu'elle doit être unique, la Carte Blanche doit s'assurer que la codification provient de l'organisme certificateur. Celui-ci remet à l'intervenant une carte d'habilitation de la technologie des cartes à micro-processeur qui contient et protège la codification. La procédure d'association de partenaire nécessitant une codification nécessite la présence de cette carte qui est reconnu par la Carte Blanche et lui fournit elle-même la codification.

4.7.2.2 Identification d'une application

L'identificateur d'une application sert à deux choses. La première est inhérente à un système multi-applicatif évolutif. Un intervenant doit pouvoir connaître les applications que contient une Carte Blanche donnée. En fonction de cela, un intervenant qui ne sait utiliser qu'une seule application, peut ou ne peut pas fonctionner avec cette Carte Blanche. Ce mécanisme montre comment l'accès à un service peut être «spontané», c'est à dire automatiquement retrouvé par rapport au système hôte sur lequel le porteur a choisi de la connecter. Un intervenant multi-applicatif peut quant à lui, faire le tri des applications de la Carte Blanche en fonction de ses possibilités et le proposer au porteur pour qu'il choisisse le service qu'il désire. L'identificateur d'application est en deuxième lieu nécessaire aux offres de services. Un partenaire ne va pas utiliser un service qu'il lui est offert sans en connaître l'application d'origine.

Chaque application possède un nom. C'est un identificateur intelligible qui permet au porteur ou à un opérateur de repérer facilement les applications de la Carte Blanche. C'est pourquoi ce nom doit être unique dans chaque Carte Blanche. De même que pour les partenaires, cela oblige parfois une application à être qualifiée de noms différents suivant l'objet nomade utilisé. Il devient difficile aux intervenants de retrouver des applications rien que par ce nom. Nous avons eu encore recours à une codification optionnelle et unique gérée par un organisme indépendant. Une application peut être référencée par cette codification. Mais une telle codification pourrait permettre d'obtenir plus d'informations sur l'application. Elle serait divisée en plusieurs champs. Un champ type indiquerait la catégorie d'application. Un champ origine donnerait une référence de l'émetteur d'application. Un champ langue nationale spécifierait la langue utilisée par la Carte Blanche pour une application à usage international. En effet, la codification n'a pas seulement un rôle d'identificateur unique, mais aussi permet un référencement international des applications car leur nom seul, bâti dans la langue d'origine, n'est pas compréhensible à l'étranger. Par exemple, une

application Santé doit être reconnue dans tous les pays visités par le porteur, et y être utilisable en cas d'urgence.

Autant le nommage des applications est relativement libre, autant l'installation d'une application avec codification doit être protégée. Une codification ne peut être introduite que si elle est totalement sûre et unique. La procédure d'installation d'application certifiée sera détaillée plus loin dans ce chapitre.

4.7.2.3 Identification des services

Les services sont les éléments de la Carte Blanche dont le nommage est très important puisqu'ils sont très souvent appelés. Dans les cartes à micro-processeur, les services sont identifiés par un code d'instruction. Cela ne posait pas de problème puisque chaque carte avait son propre jeu d'instructions et ne pouvait en changer. Pour les services des applications système de la Carte Blanche, nous pourrions utiliser le même procédé. Mais il devient très difficile de donner des numéros distincts aux services des applications qui sont installées sans prédétermination ni ordre pré-établi, puis de retrouver à quoi correspond chaque identificateur. Ce point est d'autant plus crucial que les services doivent pouvoir être retrouvés et invoqués non seulement par les intervenants logiciels mais également par le porteur et des intervenants de type opérateur. Pour cette raison, nous avons préféré identifier les services par des noms choisis par l'émetteur d'application, c'est à dire des identifiants intelligibles pour un humain.

Le choix des noms est important. Ils doivent être suffisamment explicites mais la longueur du nom peut rendre contraignante la saisie des commandes par un opérateur. L'utilisation systématique de noms abrégés perdrait en compréhension. Par contre, les services des applications systèmes, fréquemment appelés par tous les utilisateurs, et identiques sur toutes les Cartes Blanches, disposent d'une forme pleine et d'une forme abrégée en trois lettres.

Contrairement aux noms de partenaires et d'applications, les noms de services ne sont pas obligatoirement uniques dans la Carte Blanche. Tout d'abord, ils sont subordonnés à une application. Ensuite, la recherche d'un service appelé dépend du partenaire loggué et donc des droits qu'il possède. Elle commence par scruter parmi les applications dont le partenaire est propriétaire, puis parmi les services qui lui sont offerts. Par conséquent, un même nom pour deux services de deux applications différentes pose peu de problèmes. Côté serveurs, deux applications appartenant au même serveur sont installées par le même émetteur d'application. Cet émetteur peut donc facilement donner des noms différents aux services des applications qu'il va installer chez le même serveur (qu'il a d'ailleurs associé). Côté clients (au sens large), les services offerts proviennent d'une plus large origine d'application. C'est pourquoi l'offre d'un service définit un nom d'alias, nom par lequel le partenaire ou l'application coopérante appellera le service.

Toutefois, il peut arriver que l'appel à un service soit ambigu. Dans ce cas, l'utilisateur peut préciser l'origine du service qu'il demande en faisant précéder le nom du service par le nom de l'application qui, lui, est forcément unique, puis d'un point de séparation :

nom_application. nom_service

Ce nommage explicite est particulièrement utile lorsqu'un nom de service est identique à celui d'un service d'une application système ou de son abréviation. Par défaut, c'est toujours le service du système qui sera exécuté.

4.7.3 Protocole d'installation d'application certifiée

Une application certifiée doit garantir son identité, son intégrité, sa conformité et son origine. Pour cela, elle a été installée en précisant sa **codification** unique et la **signature de son émetteur** suivant un protocole sécurisé. La codification permet aux intervenants de reconnaître l'application quel que soit le nom sous lequel elle a été installée. Elle précise les caractéristiques générales (par exemple le domaine d'activité, la version) de l'application. Le fait que l'application soit signée est un gage de confiance pour l'utilisateur qui pourra ainsi retrouver l'émetteur de l'application en cas de dysfonctionnement par rapport aux caractéristiques escomptées.

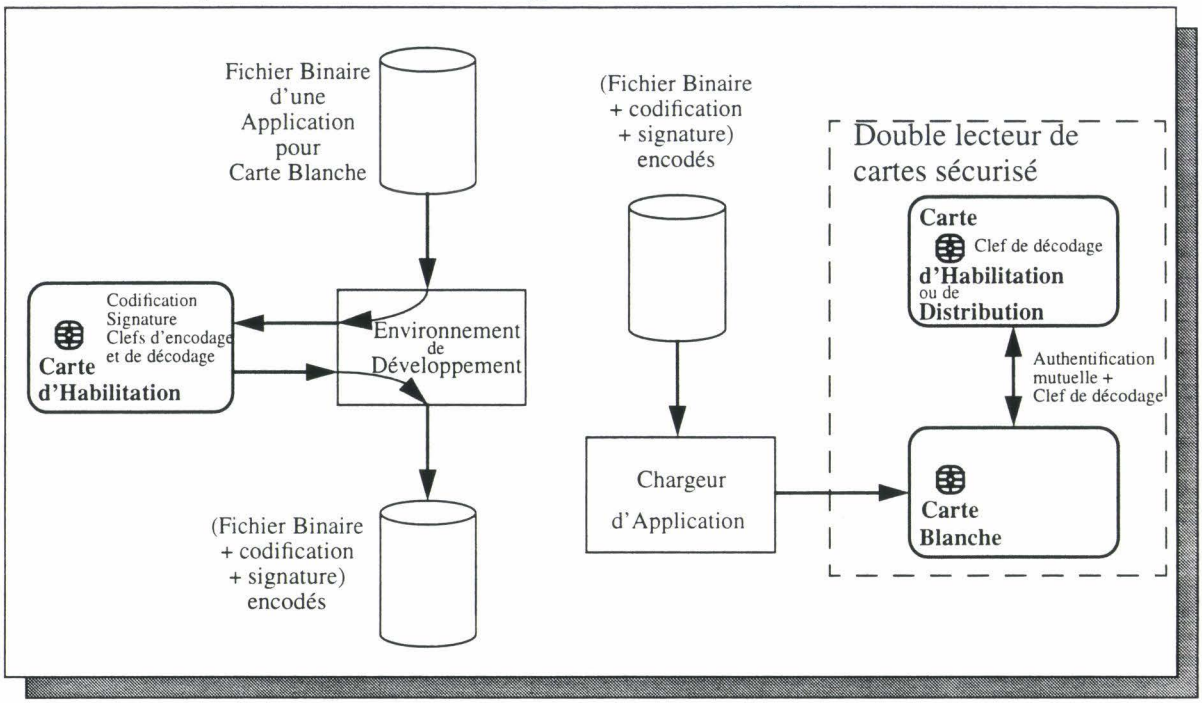
La génération des codifications de toutes les applications à certifier doit être centralisée et est effectuée par un unique **organisme certificateur**. Ensuite la Carte Blanche ne doit accepter une codification lors de l'installation d'une application, que si elle peut être sûre que cette codification provient de l'organisme certificateur. On utilise alors une carte d'habilitation issue de la technologie des cartes à micro-processeur, émise par l'organisme certificateur, qui contient et protège la codification et la signature, et peut être authentifiée par la Carte Blanche.

Le protocole d'installation d'une application certifiée (cf figure 16 de la page 83) est le suivant :

1. L'émetteur d'application qui veut réaliser une application certifiée doit obtenir auprès de l'organisme certificateur une codification et une signature toutes deux uniques. Ces informations lui sont remises sous forme d'une carte d'habilitation.
2. L'émetteur d'application développe son application sur un système extérieur à la Carte Blanche. Lorsque son application est complète et correcte, il génère un fichier binaire contenant les informations nécessaires à une installation d'application normale. Ce fichier contient donc les données, le code de l'application et la liste des services inclus dans ce code.
3. Ce fichier binaire est transmis à la carte d'habilitation pour que celle-ci y ajoute la codification et la signature puis restitue le résultat sous forme chiffrée. La carte d'habilitation utilise une clef d'encodage qui est attribuée à la carte seule par l'organisme certificateur. L'émetteur d'application dispose donc sur son ordinateur de développement, d'un fichier binaire encodé.
4. Lors de l'installation, ce fichier binaire encodé est transmis à la Carte Blanche qui ne peut le déchiffrer qu'en présence de la carte d'habilitation. Cette dernière se connecte donc à l'objet nomade, et après authentification mutuelle, elle transmet à la Carte Blanche, de façon confidentielle, la clef de décodage correspondante qu'elle possède. Il est nécessaire que ce dialogue s'effectue dans un environnement protégée entre les lecteurs de cartes.

La carte d'habilitation issue de la technologie des cartes à micro-processeur permet de détenir la codification, la signature et les clefs d'encodage et de décodage sans qu'il soit possible de les modifier. De plus, les clefs ne sont pas consultables. Ensuite la carte à micro-processeur inclut elle-même la codification et la signature dans le fichier d'installation avant de le chiffrer. Ainsi, la Carte Blanche est certaine que ces informations proviennent de l'organisme certificateur. La carte d'habilitation n'est pas capable de déchiffrer le fichier binaire encodé sinon il serait possible de modifier l'application puis de la chiffrer à nouveau.

FIGURE 17 Préparation et Installation d'une application certifiée



Le protocole assure à l'émetteur d'application que la codification et la signature sont bien associées à l'application qu'il a développée et mise au point car ces informations sont incluses dans le fichier encodé. C'est l'intérêt de l'émetteur d'application car sa signature le désigne comme responsable de l'application chargée en cas d'anomalie. Cette méthode permet également d'assurer l'intégrité et la conformité de l'application qui est installée dans la Carte Blanche. Des modifications du fichier binaire encodé seraient détectées par la Carte Blanche lors du déchiffrement de ce fichier. L'émetteur d'application est ainsi sûr que le code chargé dans une Carte Blanche est son propre code non altéré.

Un autre avantage de cette procédure en plusieurs phases est que l'application peut être chiffrée une seule fois, puis que le même fichier binaire encodé résultant peut servir à l'installation de l'application sur plusieurs Carte Blanche. De plus, ce découpage permet la distribution de l'application par différents distributeurs. Pour cela, l'émetteur d'application remet à chaque distributeur le fichier binaire encodé et une carte de distribution. La carte de distribution, également issue de la technologie des cartes à micro-processeur, est dérivée de la carte d'habilitation. Elle ne contient que la clef de décodage qu'elle est capable de fournir de façon confidentielle à la Carte Blanche tout comme la carte d'habilitation. La carte de distribution étant incapable de chiffrer et de déchiffrer une application, le distributeur est dans l'impossibilité de modifier l'application, et de lui en substituer une autre.

4.7.4 Surpopulation de partenaires natifs

De nouveaux partenaires peuvent s'associer facilement. Ceci permet au porteur de demander à un émetteur d'application de s'associer à la Carte Blanche pour ensuite y installer son application. Mais, rien n'empêche qu'un intervenant n'associe un nouveau partenaire à l'insu du porteur, lorsque celui-ci utilise sa Carte Blanche pour toute autre chose. L'accumulation mal-intentionnée de

tels partenaires pourrait porter préjudice à l'objet nomade en saturant les capacités d'accueil de nouveaux partenaires et applications légitimes voulus par le porteur.

Mais la simple association d'un nouveau partenaire ne diminue pas le niveau de protection de la Carte Blanche. D'abord, ce partenaire ne peut bénéficier d'aucune offre d'application «utilisateur», puisqu'il n'existait pas auparavant. Si, de plus, il est de la catégorie serveur, il ne possède encore aucune application. Même s'il disposait des moyens matériels et logiciels pour développer une application à l'extérieur de la Carte Blanche (voir chapitre 4), et l'installer, cette application serait indépendante et ne pourrait en aucun cas intervenir sur les autres pour les espionner ou les pirater.

Les principaux problèmes restent donc l'occupation de la mémoire par des partenaires inutiles et l'installation d'applications à l'insu du porteur. Des solutions simples sont mises en place. La première consiste à autoriser la suppression de tout partenaire «natif», c'est à dire qui ne bénéficie d'aucune offre en particulier, et qui n'est propriétaire d'aucune application. Ces suppressions s'effectuent sous le contrôle de l'application SHELL, sur demande exclusive du titulaire de la Carte Blanche. La seconde solution est de limiter l'association libre de partenaire à partir de PUBLIC. Cette association doit être validée par le titulaire de la Carte Blanche. La validation consiste à authentifier l'accord du porteur-titulaire. Ces deux protections qui restreignent l'entrée libre de partenaire ne va pas à l'encontre des concepts de la Carte Blanche puisque c'est toujours le véritable propriétaire de la Carte Blanche, c'est à dire son titulaire, qui décide de mettre en oeuvre ou non, ces mécanismes de sécurité.

Chapitre 5
Le Système d'Exploitation pour la Carte Blanche

5.1 Introduction

Le modèle de fonctionnement des Cartes Blanches, décrit au chapitre 3, et sa spécification au chapitre 4, nécessitent de définir un véritable système d'exploitation pour cette nouvelle génération d'objets nomades. Le développement de ce système conduit à effectuer des choix d'implémentation des concepts. Les structures des entités manipulées doivent être définies. Il faut également concevoir tous les mécanismes de fonctionnement et de sécurité.

La Carte Blanche n'est pas destinée à un matériel particulier mais à différents types de machines voulant utiliser les concepts de la Carte Blanche. C'est pourquoi nous donnerons des spécifications générales d'un système d'exploitation standard pour objet nomade. Par contre, nous déduirons des caractéristiques techniques de la machine cible nécessaires à certaines fonctions du système d'exploitation. Chaque fabricant de Carte Blanche aura à moduler et à adapter ce système d'exploitation à l'architecture de l'objet nomade final.

Après avoir décrit la structure du système d'exploitation, ce chapitre développera les primitives du système. L'exécution des services à l'intérieur de la Carte Blanche sera amplement détaillée sur le plan du mécanisme de fonctionnement et de la sécurité.

5.2 Structure du Système d'Exploitation

Le système d'exploitation est développé suivant le principe de la structuration en couche [Tan91]. Cette méthode présente des avantages pour la Carte Blanche des points de vue du développement et de la sécurité.

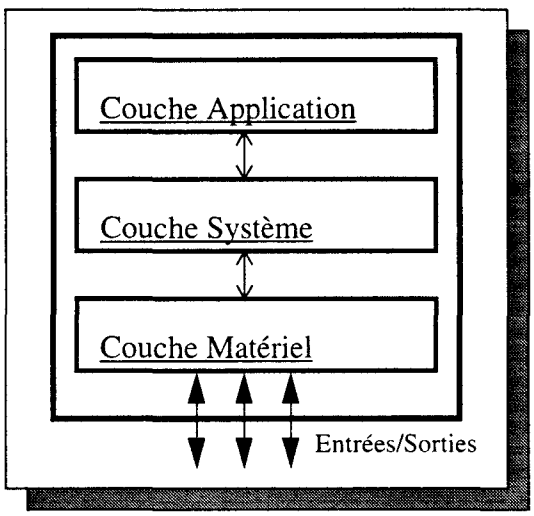
Chaque couche étant de plus faible complexité que le système global, leur développement et leur mise au point sont plus faciles. Cette caractéristique est intéressante dans un système qui doit être modulé en fonction du matériel et des options fonctionnelles choisies par le fabricant. Ensuite, chaque couche fournit à la couche immédiatement supérieure des fonctions de plus haut niveau pour simplifier leurs propres tâches.

La sécurité est renforcée par cette structuration en couche. Une couche ne peut avoir accès qu'aux couches contiguës. Cela signifie que les applications ne peuvent avoir accès au matériel sans le contrôle du système.

Le découpage en couches proposé part des couches les plus proches de la machine aux couches les plus proches de l'utilisateur. Ce découpage est le suivant :

- la couche matériel
- la couche système
- la couche application

FIGURE 19 Structure en couches du système d'exploitation



5.2.1 La couche matériel

La couche matériel gère le matériel de l'objet nomade. Elle fournit à la couche système des primitives fonctionnelles tout en lui cachant les contraintes technologiques et la complexité du matériel.

La couche matériel se charge donc de :

- de la gestion des différentes mémoires de l'objet nomade et de leur protection
- de la gestion des entrées/sorties

5.2.2 La couche système

La couche système détermine les caractéristiques fonctionnelles de la Carte Blanche. Elle gère un ensemble de structures de contrôle des différents objets qu'elle manipule (application, partenaire, etc...). Elle s'appuie sur la couche matériel pour les allocations de mémoire et les entrées/sorties. Elle met en oeuvre l'exécution des services des applications. Elle fournit aux émetteurs d'applications des facilités de développement de leur application.

5.2.3 La couche application

La couche application constitue l'environnement où s'exécute les applications. Une application obtiendra des services de la couche inférieure par des appels système.

Dans la perspective où la Carte Blanche accepterait des application génériques, cette couche devrait être subdivisée en deux parties. La première concerne les applications génériques comme une gestionnaire de tables ou de fichiers. Cette couche est la plus proche de la couche système. La deuxième sous-couche comprend les applications spécifiques indépendantes qui pourront obtenir des services de la couche «application générique».

5.3 La gestion de la mémoire

La couche matériel est chargée de la gestion et de la protection de la mémoire de l'objet nomade. La Carte Blanche est composée de trois types de mémoire réservés à des usages différents. Il s'agit de la mémoire permanente non ré-inscriptible (ROM), de la mémoire non-volatile ré-inscriptible (EEPROM) et de la mémoire de travail, volatile (RAM).

5.3.1 La mémoire permanente non ré-inscriptible

Cette mémoire contient le système d'exploitation. Elle est déterminée en phase de fabrication et ne peut plus être modifiée par la suite. A la mise sous tension ou à la remise à zéro, le micro-processeur commence toujours à exécuter ce logiciel en ROM. Cette caractéristique, héritée des cartes à micro-processeur, assure que le fonctionnement de base d'une Carte Blanche et surtout sa sécurité ne pourront jamais être détournés. Ainsi, la sécurité est préservée lors de l'utilisation des applications installées par divers émetteurs.

La mémoire ROM n'a pas besoin d'être gérée par la couche matériel. Par contre, elle doit être protégée des accès illicites par les applications, notamment pour interdire la copie du code du système d'exploitation.

5.3.2 La mémoire non-volatile ré-inscriptible

Cette mémoire est la mémoire utile de l'objet nomade pour son porteur et les intervenants qu'il choisit, notamment les émetteurs d'application. En effet, elle contient les différentes applications installées dans la Carte Blanche et les références des partenaires.

La technologie de cette mémoire nécessite des procédures d'écriture et de ré-écriture particulières. Ces particularités concernent le temps d'écriture et une phase d'effacement avant la ré-écriture. La couche matériel fournit les primitives d'écriture et de ré-écriture pour les couches supérieures.

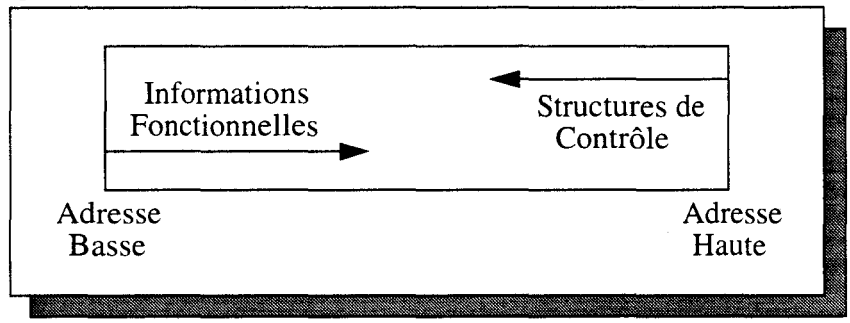
Des applications et des partenaires peuvent être ajoutés ou retirés dans le temps. La couche matériel fournit donc des primitives d'allocation et de libération de la mémoire.

La couche matériel distingue deux catégories d'informations stockées dans cette mémoire :

- les informations fonctionnelles des applications fournies par les émetteurs : les données et le code
- les structures de contrôle des applications et des partenaires uniquement gérées par le système

Ces informations sont de nature très différentes et ne demandent pas le même niveau de protection. La mémoire est alors gérée en deux parties distinctes, l'une pour les informations fonctionnelles, et l'autre pour les structures de contrôle. La répartition de la mémoire globale entre ces deux parties n'est pas évidente. Pour obtenir un remplissage optimum de la mémoire, il est envisageable d'allouer la mémoire à partir de l'adresse le plus basse vers l'adresse la plus haute pour une partie, et inversement pour l'autre partie comme dans la carte MCOS [MCOS]. La mémoire est saturée lorsque les deux parties se rejoignent.,.

FIGURE 20 Séparation et remplissage de la mémoire EEPROM en deux parties



5.3.2.1 Informations fonctionnelles des applications

Il s'agit donc des données et du code des applications. La taille de chaque application est différente des autres et elle est surtout imprévisible. Les applications pouvant être supprimées, il est nécessaire de gérer les espaces libres. Ces espaces libres seront ensuite utilisés par de nouvelles applications de taille inférieure ou égale. La fragmentation qui en résulte, est préjudiciable à la capacité d'accueil de la Carte Blanche. Un mécanisme de ramasse-miettes est indispensable. Le choix d'une stratégie d'allocation et de libération des espaces mémoire s'est basé sur les critères suivants :

- remplir au mieux la partie dédiée aux applications pour laisser libre la partie réservée au système
- le processus de ramasse-miettes doit intervenir le moins souvent possible
- le processus de ramasse-miette doit être très court pour ne pas ralentir l'activité nomade du porteur
- le ramasse-miette doit s'effectuer lors d'une utilisation protégée de l'objet nomade pendant laquelle les risques d'arrachage ou de panne d'alimentation sont nuls, et si le temps nécessaire à ce traitement est acceptable par le porteur

Nous avons choisi la stratégie d'allocation au plus juste. Le petit segment résiduel, si il existe, sera accolé au segment libre le plus proche par le ramasse-miettes. L'allocation d'un nouveau segment dans cette partie de la mémoire correspond à une installation ou une mise à jour d'application, c'est à dire des opérations relativement longues pour lesquelles le porteur est prêt à attendre, par rapport aux services des applications qui sont en général de courte durée.

Pour la sécurité du système, le processus de ramasse-miettes doit garantir la cohérence des informations fonctionnelles lors d'une coupure de l'alimentation de l'objet nomade pendant la procédure. Une parade envisageable est de doter la Carte Blanche d'une alimentation interne autonome.

5.3.2.2 Structures de contrôle du système

Les structures de contrôle permettent au système de gérer les différents objets qu'il manipule (par exemple une application, un partenaire), sous forme de tables. Tous les éléments des tables représentant des objets de même type sont de même format. Cela signifie que des objets supprimés de la Carte Blanche pourront être remplacés par d'autres objets de même type au même emplacement sans fragmentation de la mémoire. La stratégie d'allocation et de libération de la mémoire est alors très simple : le système :

- maintient une liste des espaces libérés par type d'objets
- choisit le premier emplacement libre dans la liste correspondant à l'objet à mémoriser

5.3.3 La mémoire volatile ou mémoire de travail

Cette mémoire sert à maintenir les structures de contrôle du système nécessaire uniquement en dynamique, ainsi qu'à permettre aux commandes du système et aux services des applications de disposer de mémoire pour leurs variables dynamiques.

Cette mémoire est fréquemment sollicitée pour des allocations ou des libérations de segment pour les exécutions de commandes ou de services. L'allocation de mémoire doit donc être rapide. La stratégie la plus rapide est celle de la première zone libre suffisante trouvée. Le changement fréquent des segments alloués implique qu'il y a peu de défaut de place. En outre, des mécanismes de ramasse-miettes sont inutiles dans cette partie de la mémoire qui, régulièrement, perd son contenu à chaque mise hors tension et qui est reprise vide à chaque nouvelle mise sous tension.

5.3.4 La protection des différentes mémoires

La protection des différentes mémoires se situe à deux niveaux. Le premier concerne l'intégrité des informations en mémoire non-volatile ré-inscriptible lors de leur mise à jour. Le second niveau assure la sécurité de tout le système lors de l'exécution d'un service d'une application.

5.3.4.1 Intégrité des informations lors de leur mise à jour

Lorsqu'un ensemble de tables système ou de données d'application est mis à jour dans la mémoire non-volatile ré-inscriptible, la modification partielle des informations, suite à une coupure volontaire ou involontaire de l'alimentation, génère une incohérence de l'ensemble d'information. Ce problème peut provoquer un dysfonctionnement du système.

Pour conserver la cohérence et l'intégrité des informations en EEPROM, la couche matériel fournit des mécanismes déjà utilisés dans les systèmes de base de données [MR94]. Ce sont le "commit" et le "roll back". La couche système invoquera ces mécanismes lorsque ceux-ci seront nécessaires.

5.3.4.2 Sécurité pendant l'exécution d'un service

Lorsque le système exécute un service d'une application, il ne sait pas si le code cherchera à frauder ou non. La couche matériel assure la protection de la mémoire en interdisant, à un service d'une application, l'accès direct au système en mémoire ROM, aux tables système et aux autres applications en EEPROM, ainsi qu'aux segments alloués en RAM pour d'autres traitements en cours.

Cette protection est effectuée par une Unité de Gestion de la Mémoire, matérielle ou logicielle. La couche matériel est chargée de paramétrer cette U.G.M. sur demande de la couche système qui fournit également les paramètres. Les caractéristiques requises pour cette U.G.M. seront précisées dans le paragraphe 5.10 : Protection pendant l'exécution d'un service , après avoir détaillé comment s'exécute un service.

5.4 La gestion des Entrées/Sorties

La couche matériel est chargée de la gestion des Entrées/Sorties. Elle fournit à la couche système les primitives d'émission et de réception de message. Elle libère le système et les applications des contraintes liées au type de port utilisé et au protocole.

L'objet nomade voulant suivre le modèle de fonctionnement de la Carte Blanche peut disposer de plusieurs ports de communication. La première motivation de la prise en compte de cette caractéristique est l'intérêt pour un objet nomade de pouvoir effectuer des communications sans contact en plus des connexions physiques. La seconde est qu'un objet nomade évolué peut avoir besoin d'utiliser des ports de communication performants, tout en conservant le port série pour la compatibilité avec les bornes existantes de connexion des cartes à micro-processeur.

Ne sachant ni le nombre, ni le type des ports qui seront disponibles sur chaque objet nomade, nous ne pouvons pas spécifier toutes les possibilités et nous avons choisi de définir une même interface pour la couche système quel que soit le port utilisé.

Si l'objet nomade ne dispose que d'un port de communication ou s'il ne peut se connecter sur une borne que par un unique port commun, les mécanismes permettant la communication sur plusieurs ports peuvent être étendus à la communication sur plusieurs canaux logiques d'un même port. La notion de canal logique [ETSI] permet donc de gérer plusieurs communications simultanées sur un même port.

5.4.1 Protocole de communication

Le protocole de communication de la Carte Blanche n'est pas complètement spécifié. Si l'on veut conserver une compatibilité avec les cartes à micro-processeur, il faut implanter les protocoles T=0 et T=1 [Tas92]. Par contre, la Carte Blanche qui possède différents ports de communication, nécessite un protocole plus évolué. Ce protocole doit répondre aux critères suivants :

- Le protocole ne doit plus être de type maître-esclave mais bi-latéral

  Deux Cartes Blanches peuvent communiquer entre elles. Une application d'une Carte Blanche peut dialoguer avec une application d'une autre Carte Blanche, tout comme elle le ferait avec un système hôte.

De plus, la Carte Blanche ne doit plus seulement être esclave du système hôte. Lors d'un dialogue, elle peut décider d'interroger inopinément l'intervenant-interlocuteur pour obtenir des compléments d'information. Par exemple, le service de retrait d'espèces de la banque va demander une deuxième authentification du porteur ou une autorisation explicite du siège de la banque, uniquement pour des sommes demandées supérieures à 2 000 francs

- La Carte Blanche doit être capable de prendre l'initiative d'une communication

  En effet, lorsque deux Cartes Blanches communiquent entre elles, l'une des deux doit obligatoirement initialiser la communication.

  Il est également possible et intéressant de le faire vers un système hôte pour obtenir un service particulier. Par exemple, la Carte Blanche se connecte sur un serveur de messagerie électronique pour rapatrier, dans l'objet nomade, les messages du titulaire. Cette caractéristique est beaucoup plus intéressante pour une application médicale URGENCE. Le service MEDICATION qui reçoit les prescriptions du médecin, se connecte à un serveur de base de données médicales pour vérifier s'il n'y a pas de contre-indication avec les antécédents médicaux du titulaire-porteur connus de l'application.
- Il est souhaitable que la Carte Blanche se connecte facilement sur de nombreux réseaux existants pour qu'elle puisse accéder à de nombreux serveurs existants et multiplier le nombre d'applications disponibles sur Carte Blanche

  Cela signifie que le protocole doit être suffisamment simple et proche des plus répandus pour que la mise au format des messages par la borne de connexion soit rapide et efficace.
- La Carte Blanche peut être interrogée à partir d'un terminal domestique. Ceci permet au porteur ou au titulaire d'utiliser plus facilement, à partir de chez lui, les services qui lui sont spécifiquement destinés, tels que la consultation du solde d'une porte-monnaie électronique.

  Les terminaux domestiques sont dits "légers" car ils possèdent un unique logiciel ayant pour fonction la gestion des entrées/sorties et du protocole de communication. La Carte Blanche doit être capable de paramétrer un minimum ce terminal, par exemple pour le mettre en mode de frappe aveugle, lorsque le porteur doit saisir un mot de passe. Le protocole pourrait prendre en compte une telle fonction.

Les protocoles des cartes à micro-processeur ne conviennent évidemment pas. Nous avons défini un protocole d'étude manipulant les éléments nécessaires au système d'exploitation. Le choix plus précis d'un véritable protocole pourra être prévu dans une prochaine étude spécifique aux communications d'une Carte Blanche avec des serveurs à travers des réseaux.

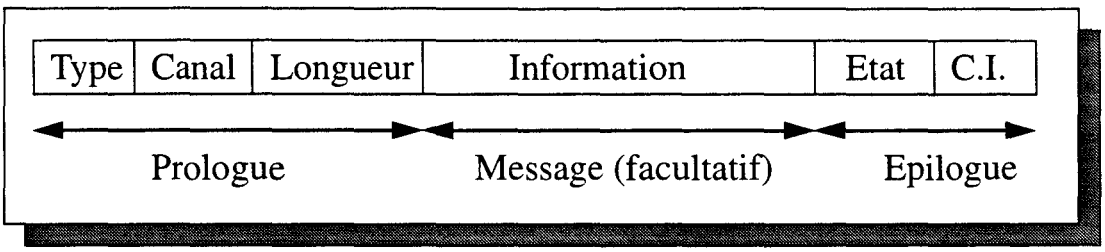
Notre protocole véhicule les informations au moyen de trames. Cette caractéristique répond au troisième critère qui veut que ce protocole soit proche de ceux qui existent. Il existe deux catégories de trames : les trames de message et les trames de services.

5.4.1.1 Les trames de message

Les trames de message sont celles qui véhiculent des informations propres à une commande système ou à un service d'une application.

La forme générale des trames, proche de celles du T=1 (cf chapitre 2), est la suivante

FIGURE 21 Forme générale des trames du protocole



- Le prologue

  Le prologue contient des informations de gestion de la trame nécessaires à la couche matériel. Il n'est jamais connu de la couche système.

  Le champ Type indique s'il s'agit d'une trame de message ou précise de quelle sorte est la trame de service

  Le champ Canal indique le canal logique de référence pour cette trame. Ce champ n'est pas renseigné pour tous les types de trames.

  Le champ longueur indique la taille du message. Cette longueur peut être égale à zéro, particulièrement dans la plupart des trames de services.

- Le message

  Le message est un champ non significatif pour la couche matériel. Le contenu de ce champ n'intéresse que la commande ou le service auquel le message est destiné. Par conséquent, la structure du message est libre pour les applications et la couche système gère ce champ comme une suite d'octets uniquement.

  Ce champ est facultatif. Il est essentiellement renseigné pour les trames de message. Quelques trames de services l'utilisent également.

- L'épilogue

  L'épilogue termine la trame avec des informations de contrôle de trame et de l'état de l'objet nomade.

  Le champ `Etat` peut être vu comme les octets de codes de retour des protocoles de cartes à micro-processeur. Une partie de ce champ donne l'état du protocole tandis qu'une autre partie informe de l'état des applications ou des commandes système en cours d'exécution. Ce champ est consultable par les commandes et services à la réception des messages qui leur sont destinés. Il peut être renseigné par la commande ou le service qui envoie un message. La partie état du protocole peut servir à demander au terminal domestique de se mettre en mode de frappe aveugle.

  Enfin le champ Contrôle d'Intégrité (C.I.) contient une information telle qu'un CRC permettant à la couche matériel de détecter une erreur de transmission.

5.4.1.2 Les trames de services

Les trames de services sont des trames propres au protocole. Quand elles sont reçues, ces trames n'agissent que sur la couche matériel. Seule cette couche peut émettre des trames de services.

Les trames de services se différencient par le champ Type du prologue de la trame. Le protocole étant bi-latéral, ces trames peuvent provenir et arriver à la Carte Blanche. Les trames de services sont les suivantes :

- `Demande de connexion`

  La demande de connexion permet d'ouvrir une communication entre un système hôte et la Carte Blanche. A la mise sous tension de l'objet nomade, aucune communication n'est ouverte sur aucun des ports. La demande de connexion permet de savoir par quel port le système hôte désire dialoguer avec la Carte Blanche.

- `Réponse à connexion`

  La réponse à connexion est une trame d'acceptation de connexion. Le champ Canal indique au demandeur sur quel canal du port il peut dialoguer. La partie Message est renseignée par la couche système qui doit également accepter une nouvelle connexion. Ce message a des fonctions similaires à la réponse à la remise à zéro des cartes à micro-processeur. Il permet d'identifier de façon unique une Carte Blanche particulière en fournissant des références de fabrication. Il fournit au système hôte des informations sur ses capacités en matière de communication (nombre et nature des ports, nombre de canaux logiques par port). Il peut également permettre d'identifier l'application qui a été sélectionnée dans la demande de connexion.

- `Acquittement de trame`

  L'acquittement de trame doit être envoyé à chaque réception de trame correcte et reconnue par le protocole.

- `Non Acquittement de trame`

  Le non-acquittement de trame est envoyé quand une erreur est constatée dans une trame reçue. L'erreur correspond à un contrôle d'intégrité négatif ou à une trame n'étant pas reconnue par le protocole. Le champ état indique le type d'erreur survenue.

  Le Carte Blanche ou le système hôte qui reçoit une trame de non-acquittement doit réémettre la dernière trame qu'il a envoyée.

- `Demande de déconnexion`

  La demande de déconnexion permet de terminer une communication entre la Carte Blanche et un système hôte. Cette trame est nécessaire d'une part pour fermer bi-latéralement la communication, et d'autre part pour libérer un canal logique pour un autre usage de l'objet nomade.

- `Acceptation de déconnexion`

  L'acceptation de déconnexion permet au demandeur de déconnexion d'être sûr que la fermeture du dialogue est effective des deux cotés de la communication.

5.4.2 Les primitives d'entrées/sorties

La couche matériel est composée de deux types de primitives d'entrées/sorties. Celles qui fonctionnent lorsqu'une trame arrive et celles que la couche système peut appeler.

Les primitives de traitement des trames à l'arrivée sont les suivantes :

- Acquisition d'une trame

Cette primitive doit être activée dès qu'une trame arrive sur l'un des ports de la Carte Blanche. Elle vérifie l'intégrité de la trame puis analyse le type de trame. Si il n'y a pas d'erreur elle envoie un acquittement de trame, sinon un non-acquittement.

Ensuite, suivant le type de trame reçue, elle appelle les primitives concernées.

- Attribution d'un numéro de canal logique

  Cette primitive est appelée lorsque qu'une trame de demande de connexion est reçue. Elle attribue le premier canal logique libre sur le port de connexion et avertit la couche système de la demande sur ce numéro de canal. Si aucun canal n'est libre sur le port, la couche système n'est pas sollicitée et une trame de non acquittement est renvoyée avec l'erreur indiquée par le champ état

- envoyer un acquittement ou un non acquittement

  Ces deux primitives peuvent être appelées de n'importe quelle autre primitive d'entrées/sorties de la couche matériel. La primitive d'acquittement permet de prévenir l'émetteur de trame de l'arrivée correcte de sa trame. La primitive de non-acquittement permet de l'avertir de l'erreur rencontrée pour qu'il puisse éventuellement renvoyer sa trame une nouvelle fois.

- attendre l'acquittement ou le non acquittement

  Cette primitive est appelée à la réception d'une trame d'acquittement ou de non acquittement. Le message précédemment envoyé étant mémorisé au niveau de la couche matériel pour une éventuelle retransmission, cette primitive libère la mémoire occupée par le message en cas d'acquittement ou retransmet le message dans le cas contraire.

Les primitives d'interfaces avec la couche système permettent au système de gérer les dimensions multi-port et multi-canal et offrent une facilité de communication pour les commandes et les services de la Carte Blanche en leur cachant la notion de trame et le protocole. Ces primitives sont :

- envoyer une réponse à connexion

  Cette primitive permet à la couche système d'accepter une demande de connexion. Le système doit fournir les informations à faire figurer dans la partie message de la Carte Blanche dans la trame de Réponse à connexion.

- scruter les messages arrivés

  Cette primitive permet à la couche système de prendre connaissance des demandes de connexion et des nouveaux messages arrivés sur chaque canal de chaque port. Ensuite, le système acceptera ou non la connexion et choisira lequel de ces messages il veut traiter en premier

- lire un message

  Cette primitive permet à une commande du système ou à un service d'une application de lire un message arrivé sur le canal du port associé à son exécution. Seul le message extrait de sa trame de type message est fourni, à la couche système, sous la forme d'une suite d'octets

- émettre un message

  Cette primitive permet à une commande du système ou à un service d'une application d'émettre un message sur le canal du port associé à son exécution. La couche système

n'envoie que le message, sous la forme d'une suite d'octets, à la primitive qui construira la trame de type message, avant de l'émettre sur le port

- envoyer une acceptation de déconnexion

  Cette primitive doit être appelée par le système quand tout dialogue est terminé sur le canal de communication et que le système accepte la déconnexion.

5.5 La gestion des sessions

La couche système gère chaque utilisation dynamique de la Carte Blanche sous la forme de sessions.

5.5.1 Définition d'une session

Une session est l'ensemble des ressources de la machine que le système met à la disposition d'un partenaire pour exécuter une commande ou un service dans la Carte Blanche. Nous définissons deux sortes de sessions, les sessions externes er les sessions internes. La distinction réside dans le fait que la commande ou le service dialogue avec un intervenant sur un système hôte, ou avec une autre commande ou un autre service du même objet nomade.

Une session est donc toujours associée à un partenaire conformément au principe du login, et soit à un canal d'un port de communication, soit à un canal interne.

Les sessions sont gérées sous forme de tables système dynamiques. Ces tables sont maintenues en mémoire de travail.

5.5.2 Les sessions externes

Les sessions externes sont des sessions où toutes les entrées/sorties des commandes et des services sont effectuées entre la Carte Blanche et l'extérieur via un port de communication.

A la mise sous tension de la Carte Blanche, il n'y a aucune session ouverte. La création d'une session externe s'effectue de la manière suivante. Lorsque la couche matériel reçoit une demande de connexion, elle détermine un numéro de canal libre sur le port où est arrivée la demande. Lorsque la couche système scrute les ports de communication, elle est avertie de la demande de connexion et elle peut ouvrir une session sur le canal d'affectation. Si la couche système accepte la connexion elle appelle la primitive acceptation de connexion de la couche matériel avec les paramètres qui identifient la Carte Blanche. La session est attachée au canal fourni par la couche matériel.

Si la demande de connexion ne précise pas le partenaire qui veut se connecter, la session est attachée par défaut au partenaire prédéfini PUBLIC. Sinon, si le partenaire est identifié et si sa présentation requiert une authentification, celle-ci est demandée à l'intervenant.

Une session externe peut être fermée. Lorsque la couche matériel reçoit une demande de déconnexion, la couche système en sera informée lors de l'appel à la primitive `Scruter les messages arrivés`. Elle enverra une acceptation de déconnexion et détruira la session si aucune commande et aucun service ne sont en cours d'exécution sur cette session.

5.5.3 Les sessions internes

Les sessions internes sont des sessions où toutes les entrées/sorties des commandes et des services sont effectuées à l'intérieur de la Carte Blanche via un canal interne.

Les sessions internes sont créées lorsqu'un service en cours sur une session externe appelle un service d'une autre application pour coopérer. Un canal interne est également créé. Une des extrémités du canal est attachée à la session interne. L'autre est attachée momentanément à la session externe qui devient pour le temps de la coopération une session interne.

Un canal interne est une structure en mémoire de travail, gérée par la couche système, et qui permet aux deux sessions coopérantes d'échanger des messages dans les deux sens, comme si chacune d'elle communiquait avec l'extérieur.

Le temps de vie d'une session interne est la durée d'exécution du service offert demandé. La terminaison de ce service entraîne la fermeture de la session interne et la restitution de la session d'appel en session externe.

5.5.4 Le besoin d'un fonctionnement multi-session

Le système d'exploitation de la Carte Blanche a besoin d'être multi-session, c'est à dire qu'il doit savoir gérer plusieurs sessions à la fois. D'abord, cette caractéristique est essentielle pour la coopération entre applications. En effet, chaque service participant à la coopération doit être exécuté dans deux sessions différentes afin de préserver l'indépendance des applications. Il est alors impossible à une application d'accéder aux données et au code de l'autre application puisque leur environnement respectif d'exécution sont distincts.

Le fonctionnement multi-session est alors étendu pour profiter de la présence de plusieurs ports sur l'objet nomade et de plusieurs canaux logiques par port. Une session externe pourra être ouverte pour chaque canal de chaque port.

Le système d'exploitation accepte donc plusieurs sessions en cours en même temps, mais une seule session sera active à la fois. La session active sera désignée par un ordonnanceur.

5.5.5 L'ordonnanceur de session

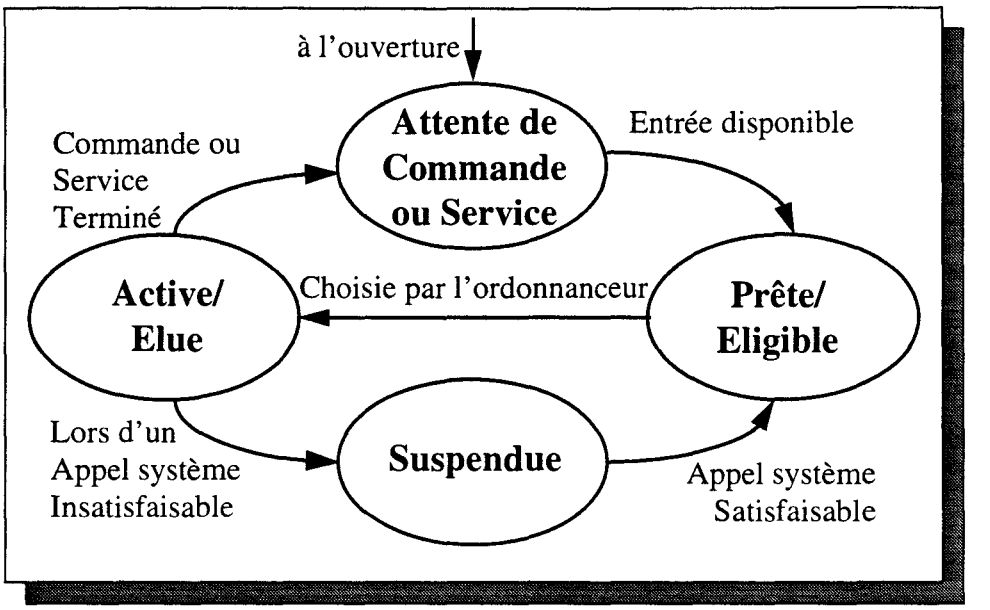
L'ordonnanceur de sessions doit choisir une session à activer parmi toutes les sessions ouvertes. Une session peut prendre quatre états : Attente de Commande ou Service, Prête/Eligible, Active et Suspendue.

5.5.5.1 Session en attente de commande ou de service

Une session en attente de Commande ou de Service est une session libre pour effectuer n'importe quel travail demandé par le partenaire attaché à cette session. A leur ouverture, toutes les sessions sont dans cet état.

Une session revient de l'état actif à cet état lorsque la commande ou le service exécuté se termine. Seules des sessions dans cet état peuvent être fermées par une demande de déconnexion.

FIGURE 22 Diagramme de passage des sessions d'un état à un autre



5.5.5.2 Session prête ou éligible

Une session arrive dans l'état prêt lorsque :

- une entrée est disponible sur le canal attaché à la session alors que la session est en état d'attente
- la cause qui a mis la session en état suspendu, disparaît

L'ordonnanceur décide du passage des sessions de l'état éligible à l'état actif.

5.5.5.3 Session active

Une session devient active lorsque l'ordonnanceur la choisit parmi les sessions éligibles. Une seule session au maximum est active à la fois dans la Carte Blanche. L'activité d'une session correspond à l'exécution d'un service ou d'une commande dans cette session. La session possède donc la ressource du micro-processeur de l'objet nomade pour elle seule. La session reste active jusqu'à :

- sa suspension par le système. Dans ce cas, son état devient suspendu
- ce que le service ou la commande soit terminé. Dans ce cas, elle revient à l'état d'attente de commande ou de service

5.5.5.4 Session suspendue

Une session suspendue est une session dans laquelle une commande ou un service est en cours d'exécution mais qui a été arrêtée par le système lors d'une primitive ou d'un appel système qui n'a pas pu être immédiatement satisfait. Dès que cette primitive ou cet appel système peut enfin être satisfait, la session entre dans l'état éligible pour que l'exécution de la commande ou du service reprenne dès que l'ordonnanceur choisira à nouveau cette session.

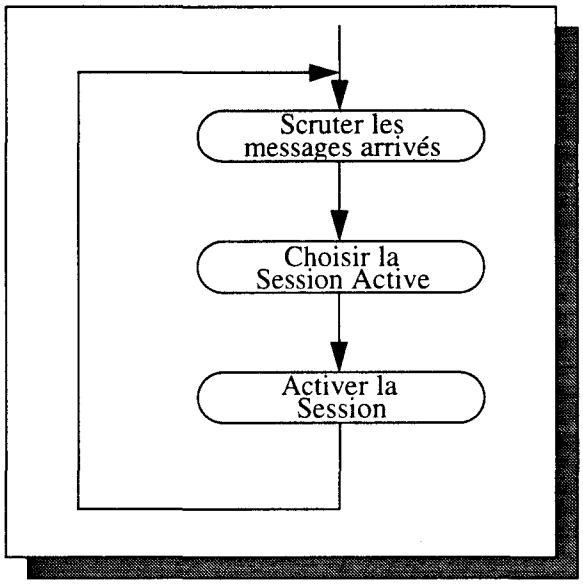
Les causes de suspension de session et les mécanismes de reprise après suspension seront détaillés en fin de ce chapitre après avoir vu toutes les primitives et les appels systèmes.

BU LILLE 1

5.5.5.5 Algorithme de l'ordonnanceur

L'ordonnanceur est le moteur du système d'exploitation. Il interroge régulièrement la couche système sur les nouvelles arrivées de message et active les sessions prêtes à exécuter un service ou une commande. L'algorithme de l'ordonnanceur est constitué d'une boucle infinie. Il est présenté sous la forme d'un organigramme à la figure suivante.

FIGURE 23 Organigramme de l'algorithme de l'ordonnanceur



Le corps de la boucle effectue les traitements suivants dans cet ordre :

- Scruter les messages arrivés

  La primitive Scruter les messages arrivés est appelée. Elle traite les demandes de connexion et crée les nouvelles sessions correspondantes. Puis, pour chaque message arrivé sur un canal, la session correspondant à ce canal est placée en état éligible.

- Choix de la prochaine session active

  L'ordonnanceur parcourt la table des sessions ouvertes de façon cyclique à partir de la session qui suit la session précédemment élue. Il choisit d'activer la première session dans l'état éligible.

- Activation de la session

  S'il s'agit d'une session libre, c'est à dire dans laquelle aucune commande et aucun service ne sont en cours d'exécution, le message est envoyé à l'application système SHELL qui l'analysera comme une ligne de commande. Sinon il s'agit d'une reprise de commande ou de service suspendu, et une primitive de reprise d'exécution est alors appelée.

  La session reste active jusqu'à la prochaine suspension ou à la fin de l'exécution de la commande ou du service.

L'ordonnancement des sessions de la Carte Blanche est donc très simple. Il repose sur les hypothèses qu'il y a peu de sessions ouvertes à la fois et que les services sont en général de courte durée. Les services plus longs sont des services interactifs ou d'interrogation d'une base de données distante. La durée totale d'exécution de ce service comprend beaucoup de temps d'attente de

message de la part d'un opérateur ou d'une réponse à une requête de la part du serveur de base de données. Pendant ces périodes d'attente, la session est alors suspendue, permettant à d'autres sessions d'être activées. Par conséquent, il n'est pas nécessaire d'utiliser des mécanismes plus complexes, comme l'algorithme du tourniquet avec ses quantums de temps et son système de réquisition du processeur. Les sessions n'ont à priori aucun besoin de priorité d'exécution non plus, tous les services en cours étant effectués pour les besoins du même porteur de la Carte Blanche.

5.6 Le shell ou l'interpréteur de commandes

Le Shell est une application système qui reçoit les messages qui arrivent sur une session en attente de commande ou de service. Le Shell est donc un interpréteur de commande qui considère le message comme une ligne de commande.

Cette ligne de commande est analysée lexicalement pour reconnaître les mots clefs ou les identificateurs qu'elle contient, puis est analysée syntaxiquement. Si l'analyseur syntaxique a reconnu une commande correcte, l'application Shell appelle la primitive correspondante en la paramétrant avec les informations reconnues dans la ligne de commande. Si une commande est reconnue mais avec une mauvaise syntaxe, le shell appelle la primitive «émettre un message» qui renverra à l'intervenant une trame contenant le mot d'état indiquant une erreur de syntaxe et un message précisant l'erreur.

Lorsque l'analyseur syntaxique ne reconnaît aucune commande du système, il considère qu'il s'agit d'un appel à un service. Dans ce cas, la ligne de commande est scindée par l'application SHELL de la manière suivante :

- le premier mot de la ligne de commande qui représente le nom d'appel du service invoqué
- la liste chaînée des mots suivants qui représentent chacun un argument destiné à renseigner le service

L'application Shell appelle alors la primitive d'appel de service avec ces deux éléments en paramètre.

5.7 Les commandes du système

Les commandes du système sont les primitives de la couche système qui permettent l'administration de la Carte Blanche. Ces primitives manipulent les différentes tables système comme les tables des partenaires et des applications.

Les primitives de commande du système sont sélectionnées par l'application SHELL suivant la ligne de commande qu'elle reçoit.

Toutes les commandes du système sont exécutées dans une session et donc pour le partenaire attaché à cette session. Elles sont alors capables de vérifier les droits de ce partenaire vis à vis de la commande et par rapport aux objets (partenaire, application, service et offre) qu'elle manipule. Cette caractéristique assure le Principe du Login.

Les commandes du système vont être maintenant décrites en les regroupant par rapport à la table système qu'elles manipulent principalement :

- les commandes relatives aux partenaires
- les commandes relatives aux applications
- les commandes relatives au Titulaire

Les tables système seront décrites en même temps que les commandes qui les créent.

5.7.1 Les commandes relatives aux partenaires

Les commandes relatives aux partenaires sont des commandes qui utilisent essentiellement la table des partenaires. Ces commandes sont :

- Associer un partenaire
- Dissocier un partenaire
- Purger des partenaires natifs
- Modifier des attributs d'un partenaire
- Présenter un partenaire
- Réhabiliter un partenaire
- Lister les partenaires dépendants

La table des partenaires contient d'origine les trois partenaires prédéfinis SYSTEME, OFFICIEL et PUBLIC. Ce dernier est obligatoire pour pouvoir ouvrir la première session sur la Carte Blanche. Un élément de la table des partenaires sera décrit avec la commande d'association de partenaire.

5.7.1.1 Associer un partenaire

La commande d'association d'un partenaire crée un nouvel élément dans la table des partenaires. Cette commande n'est accessible qu'à des partenaires existants, PUBLIC ou serveurs. Pour cela, elle a accès aux informations concernant le partenaire attaché par l'intermédiaire de la session active. Les droits de ce partenaire peuvent être vérifiés par le champ identificateur pour reconnaître PUBLIC, et le champ catégorie pour reconnaître un serveur.

Les paramètres à fournir à la commande pour constituer le nouvel élément dans la table des partenaires sont :

- l'identifiant

  Il correspond au nom du partenaire. Ce nom devant être unique dans la Carte Blanche, la commande doit s'assurer que l'identificateur n'existe pas déjà en parcourant la liste des partenaires existants, avant d'accepter ce nouveau partenaire
- la catégorie

  La catégorie est CLIENT ou SERVEUR. La catégorie CLIENT est retenue par défaut pour ce paramètre
- son mode d'authentification

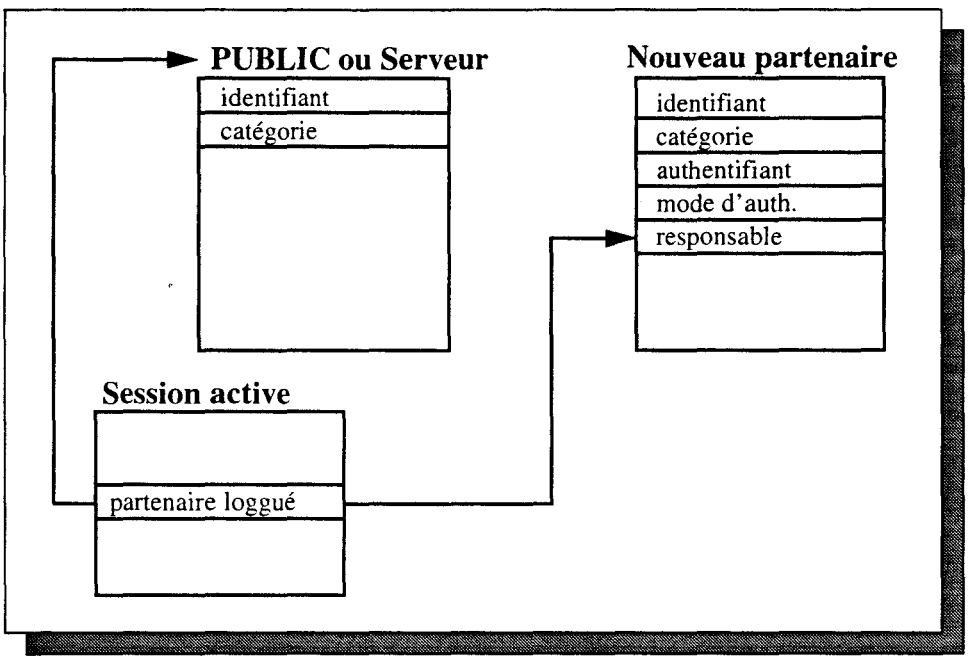
  La Carte Blanche permet d'authentifier les partenaires suivant des modes différents : de la simple authentification par code secret à des protocoles d'authentification à base d'algorithmes cryptographiques. Les modes effectivement implantés sur un objet nomade dépendent du matériel utilisé et de son fabricant. Par conséquent, ce paramètre doit être choisi parmi les modes d'authentification disponibles

- son authentifiant

  La nature de l'authentifiant à fournir dépend du mode d'authentification
- sa limite de ratification

  La limite de ratification peut être choisie par l'intervenant qui associe ce partenaire et correspond au nombre de présentations fausses consécutives que peut supporter le partenaire avant d'entrer dans l'état bloqué

FIGURE 24 Création d'un partenaire



Le champ Responsable hiérarchique est affecté par la commande qui assure que la responsabilité d'un partenaire ne sera pas attribuée à un autre partenaire étranger à cette association. Si ce champ correspond à PUBLIC, cela signifie que le partenaire est son propre responsable. Le champ Indicateur de blocage est placé à faux.

Deux autres champs définissent un partenaire. Ce sont la liste de ses applications et la liste des offres qu'il a reçues. La commande d'association affecte la valeur vide car les partenaires natifs ne peuvent déjà posséder une application, ni avoir reçu des offres de service.

5.7.1.2 Dissocier un partenaire

La commande de dissociation d'un partenaire permet de retirer un partenaire de la table des partenaires. Cette commande ne peut être effectuée que par le responsable hiérarchique de ce partenaire. Si le partenaire à supprimer est le partenaire associé à la session active, la commande peut vérifier l'égalité entre ce partenaire et son propre responsable. Dans ce cas, la session s'attache automatiquement au partenaire PUBLIC. Si le partenaire à supprimer n'est pas le partenaire attaché à la session active, la commande recherche l'existence de ce partenaire puis peut vérifier l'égalité entre le responsable du partenaire à détruire et le partenaire attaché à la session active.

Avant de supprimer réellement un partenaire, la commande doit s'assurer que :

- le partenaire n'est attaché à aucune session ouverte. Il ne doit pas être possible de supprimer un partenaire qui est en train d'effectuer une commande ou un service
- le partenaire ne possède plus aucune application. En effet, la suppression automatique des applications de ce partenaire serait difficile et dangereuse car elle ne concerne pas seulement leur code et leurs données mais aussi les partenaires créés spécialement pour elles et les offres de leurs services. De plus, la suppression de chaque application demande une implication plus importante de la part de l'intervenant responsable de ce partenaire et de ses applications.

Par contre, si le partenaire dispose d'offres de service, il peut être supprimé et la commande répercute la suppression de ces offres au niveau des applications.

5.7.1.3 Purger les partenaires natifs

La commande de purge des partenaires natifs permet de supprimer tous les partenaires qui ne possèdent aucune application, et ne disposent d'aucune offre comme un partenaire qui vient d'être créé. Cette commande est accessible uniquement par le titulaire de la Carte Blanche et donc, uniquement quand il existe après la titularisation. Elle doit donc vérifier que le partenaire attaché à la session active est bien le titulaire. Elle doit également s'assurer que chaque partenaire n'est pas attaché à une session ouverte pour justement installer son application.

5.7.1.4 Modifier des attributs d'un partenaire

La commande de modification des attributs d'un partenaire permet essentiellement de changer l'authentifiant, l'auto-démarrage et la limite de ratification. Seul l'identifiant et la catégorie d'un partenaire ne peut pas changer. Cette commande peut être effectuée uniquement par le responsable hiérarchique du partenaire à modifier. Si le partenaire à modifier est le partenaire associé à la session active, la commande peut vérifier l'égalité entre ce partenaire et son propre responsable. Sinon la commande recherche l'existence du partenaire à modifier puis peut vérifier l'égalité entre le responsable de ce partenaire et le partenaire attaché à la session active.

5.7.1.5 Présenter un partenaire

La commande de présentation de partenaire permet à tout intervenant de se faire reconnaître partenaire de la Carte Blanche. Cette commande peut être effectuée par n'importe quel partenaire de la Carte Blanche. Si la présentation est correcte, la commande attache ce partenaire à la session active et réinitialise le nombre de présentations fausses du partenaire à zéro.

La procédure de présentation est la suivante. La commande vérifie l'existence du partenaire. S'il existe, la commande peut connaître, si sa présentation demande une authentification et suivant quel mode, et également si il est bloqué. Si le partenaire ne demande pas à être authentifié, la présentation est correcte. Sinon et si le partenaire n'est pas bloqué, la commande engage le protocole d'authentification nécessaire. La présentation est acceptée si le protocole utilisant l'authentifiant du partenaire s'est correctement déroulée. Le nombre de présentations fausses est incrémenté dans le cas contraire. Le partenaire est placé dans l'état bloqué si ce nombre atteint la limite de ratification.

Dans le cas où le partenaire n'existe pas, une authentification factice est demandée à l'intervenant. Le mode d'authentification choisi peut être la simple authentification par code secret ou peut être obtenu par un tirage aléatoire des modes disponibles sur l'objet nomade. Le but de cette

authentification est de ne pas fournir trop de renseignements sur les partenaires à un intervenant qui peut être un fraudeur. En effet, cet intervenant ne pourra pas savoir si la présentation a échoué au niveau de l'identification ou au niveau de l'authentification.

De même, si le partenaire est déjà bloqué, la demande d'authentification est quand même effectuée suivant le mode habituel, mais la présentation sera incorrecte quel que soit l'authentifiant présenté par l'intervenant. Le nombre de présentations fausses consécutives n'est plus incrémenté.

Pour ne pas permettre l'interaction d'autres sessions ouvertes sur le partenaire en cours de présentation, la session ne peut pas être suspendue pendant cette commande même pendant le protocole d'authentification qui demande un dialogue entre la Carte Blanche et l'intervenant.

5.7.1.6 Réhabiliter un partenaire

La commande de réhabilitation de partenaire permet de débloquer un partenaire après que celui-ci ait été bloqué par le mécanisme de ratification. Cette commande ne peut être effectuée que par le partenaire OFFICIEL pour les partenaires autonomes ou par le responsable hiérarchique des partenaires dépendants, c'est à dire un serveur.

La commande vérifie d'abord que le partenaire attaché à la session active est soit OFFICIEL, soit un serveur. Seulement après, elle recherche l'existence du partenaire à réhabiliter. Si celui-ci est effectivement bloqué, elle contrôle que le partenaire actif a le droit de réhabiliter le partenaire bloqué et dans ce cas, elle engage l'authentification de ce partenaire. La réussite de l'authentification du partenaire conditionne sa réhabilitation.

Pour ne pas fournir trop de renseignements sur les partenaires à un intervenant qui peut être un fraudeur, la commande ne doit pas informer de la cause exacte du refus de réhabilitation (partenaire inexistant ou non bloqué ou mal authentifié ou n'étant pas sous la responsabilité du partenaire actif).

5.7.1.7 Lister les partenaires dépendants

La commande de listage des partenaires dépendants permet aux serveurs de retrouver l'identifiant des partenaires qu'ils ont eux-mêmes associés, ainsi que de connaître leur état vis à vis de la ratification. Ainsi, le responsable de ces partenaires peut savoir quels sont ceux qui sont bloqués ou qui présentent un nombre élevé de présentations fausses consécutives. Il pourra alors décider de réhabiliter les partenaires bloqués ou ajuster le nombre de ratifications des autres. Cette commande peut être effectuée par les serveurs uniquement.

La recherche des partenaires dépendants n'est pas directe. La commande doit parcourir la table des partenaires à partir du partenaire attaché à la session active pour retrouver les partenaires dont il est le responsable. En effet, ces partenaires se trouvent obligatoirement après leur responsable dans la table des partenaires.

5.7.2 Les commandes relatives aux applications

Les commandes relatives aux applications sont des commandes qui utilisent essentiellement la table des applications et les tables qui sont directement rattachées à cette table. Ces commandes sont :

- Installer une application
- Offrir un service à un partenaire
- Retirer l'offre d'un service
- Supprimer une application
- Mettre à jour une application
- Lister les applications installées
- Lister les offres reçues

La table des applications est initialement vide, l'application système SHELL étant incluse dans le système d'exploitation. Nous détaillons la structure de la table des applications en même temps que la commande qui crée un nouvel élément dans cette table : la commande Installer une application.

5.7.2.1 Installer une application

La commande d'installation d'une application crée un nouvel élément dans la table des applications ainsi que d'autres tables qui sont directement attachées à cet élément. Ces tables additionnelles seront décrites ci-après. Cette commande peut être effectuée par les serveurs uniquement. Elle vérifie donc que le partenaire attaché à la session active est effectivement de cette catégorie.

La procédure d'acquisition d'une application pour la Carte Blanche comprend trois étapes :

- l'acquisition des informations administratives
- le chargement du code de l'application
- l'initialisation des données de l'application
- l'établissement de la liste des services de l'application

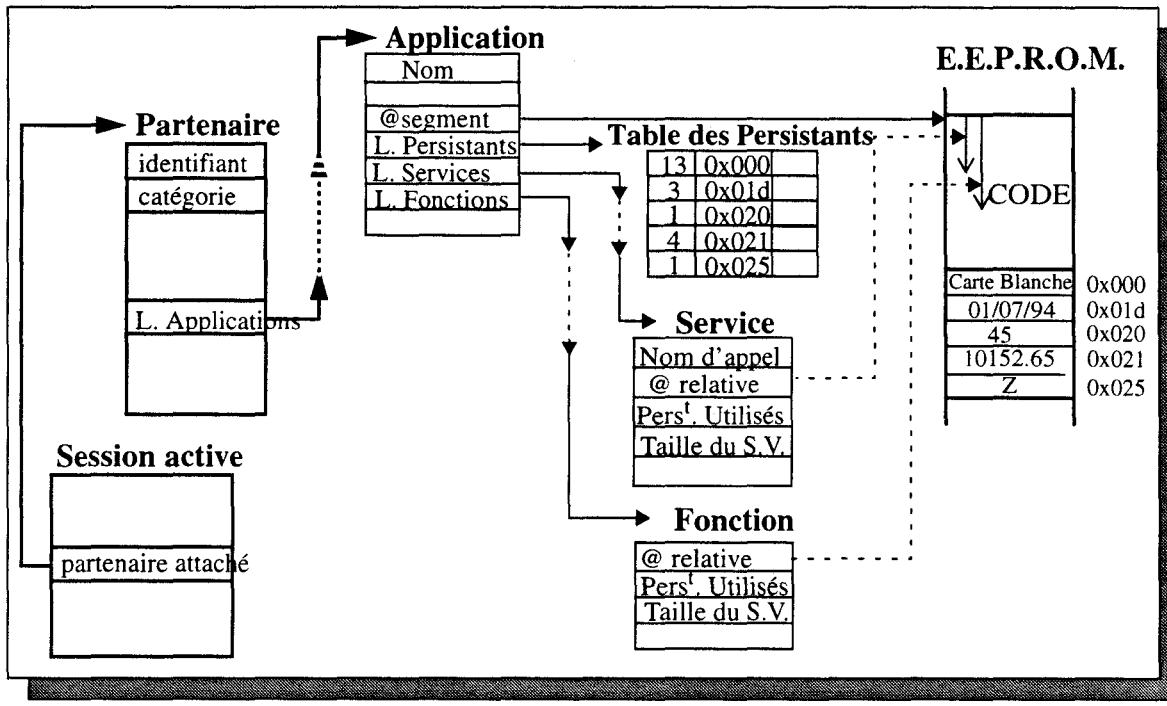
La première étape consiste à fournir le nom de l'application ainsi que l'occupation en mémoire de son code et de ses données. La commande doit contrôler l'unicité du nom de l'application en parcourant la table des applications. Elle vérifie également la suffisance de la mémoire disponible pour cette application. Les étapes suivantes ne sont effectuées qu'à ces conditions.

Le chargement du code de l'application commence par réserver une zone de mémoire non-volatile dont l'adresse initiale est placée dans le champ @segment de l'application. Puis, les octets de code, dont le nombre est spécifié à l'étape précédente, sont lus sur le canal attaché à la session active et, sont copiés consécutivement à partir de l'adresse réservée. L'émetteur d'application ne peut pas connaître ni avant, ni après l'installation, cette adresse et doit fournir un code translatable en mémoire.

L'étape suivante inscrit les valeurs initiales des données de l'application en mémoire non volatile. L'espace mémoire nécessaire aux données lui étant indiqué à la première étape, la commande alloue en une seule fois la mémoire pour l'ensemble des données de l'application. La valeur initiale de chaque donnée doit être fournie sous la forme d'une suite d'octets dont l'émetteur doit préciser la taille, car le système n'a pas à gérer les différents types de données possibles. L'adresse de stockage et la taille de chaque donnée sont mémorisées dans une table particulière

attachée à l'application : la table des persistants, pour leurs manipulations ultérieures. L'utilisation d'une table séparée est rendue nécessaire par l'ignorance du nombre de données persistantes que chaque application peut posséder. Les services et les fonctions de l'application seront également gérés dans des tables attachées à l'application pour les mêmes raisons.

FIGURE 25 Installation d'une application



Enfin, la dernière étape concerne l'interface de l'application, c'est à dire les services de l'application qu'un partenaire peut appeler. La commande fait l'acquisition de chaque service en précisant :

- le nom d'appel du service
- l'adresse relative dans le code de l'application à partir de laquelle le service peut être exécuté. L'adresse réelle peut être retrouvée grâce au champ @segment de la table des applications
- la liste des données utilisées par ce service. Elle est constituée de la suite des numéros d'entrée dans la table des données, des persistants utilisés. Cette liste servira lors de l'appel du service
- la taille du Segment de Variables qu'il est nécessaire d'allouer en mémoire de travail pour l'exécution de ce service

Ces services sont conservés sous la forme d'une table de services attachée directement à l'application.

Etroitement liées à l'utilisation des services, les fonctions sont également spécifiées lors de cette étape. La commande construit une table des fonctions attachée à l'application et qui reprend les mêmes champs que la table des services à l'exception du champ Nom d'appel. En effet, une

fonction n'est pas directement appelable par un utilisateur. Cette fonction ne peut être atteinte que par une instruction d'un service ou d'une autre fonction indiquant son adresse relative dans le code de l'application. C'est cette adresse qui servira à retrouver la fonction dans la table des fonctions attachée à l'application.

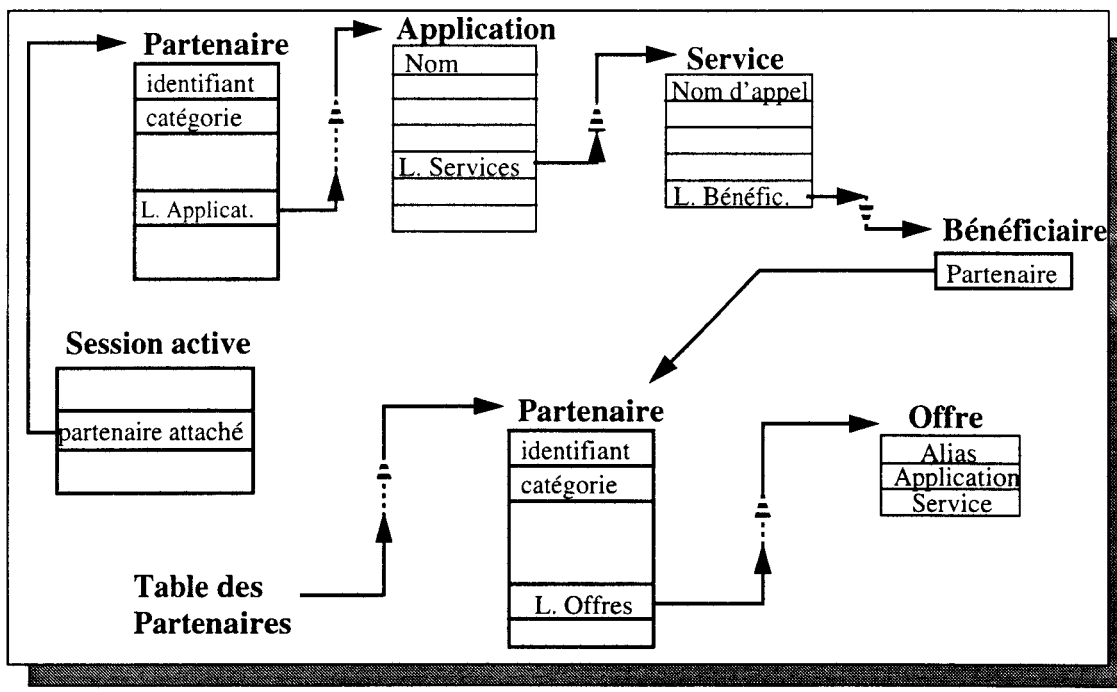
Une fois l'installation complète, l'application est ajoutée à la fin de la table des applications. Elle est également rattachée au partenaire sur la session active qui devient alors le propriétaire. Toutes les applications appartenant au même serveur sont chaînées à partir du champ Liste des applications du partenaire. Le lien de propriété des applications n'est matérialisé que par cette chaîne et il est entièrement géré par la commande pour empêcher l'attribution de cette application à tout autre partenaire non concerné.

5.7.2.2 Offrir un service à un partenaire

La commande d'offre d'un service à un partenaire crée des structures de données système permettant, au partenaire spécifié, d'accéder au service offert, et au serveur de pouvoir lui retirer par la suite cette offre. Ces structures sont mémorisées d'une part dans la table des offres associées à chaque partenaire, d'autre part dans la liste chaînée des bénéficiaires du service qui est offert.

Cette commande n'est autorisée qu'aux seuls serveurs. Elle vérifie donc que le partenaire attaché à la session active est effectivement de cette catégorie. Ensuite, la commande doit contrôler que le service offert appartient à ce partenaire. Pour cela, la commande a accès au champ Liste des applications du partenaire sur la session active et peut parcourir la liste de services de chaque application pour rechercher le service offert.

FIGURE 26 Offre d'un service à un partenaire



Maintenant que le service est trouvé, la commande vérifie l'existence du partenaire bénéficiaire en parcourant la table des partenaires à partir du début. Ensuite, elle ajoute un nouvel élé-

ment à la table des offres de ce partenaire. Les champs qui composent une offre de service sont affectés par la commande de la façon suivante :

- le champ Alias contient le nom que doit utiliser le bénéficiaire pour appeler le service. Si ce nom est différent du nom d'appel du service, il doit être fourni à la commande par l'intervenant. Sinon, le nom réel du service est recopié.
- le champ Application est un pointeur direct sur l'application. Il permettra au système d'accéder rapidement aux composantes de l'application, notamment ses données et ses fonctions, lorsque le service offert de cette application sera appelé
- le champ Service est un pointeur direct sur le service offert. Il permettra au système d'accéder rapidement aux composantes du service (adresse, taille du segment de variables, données persistantes utilisées), lorsque celui-ci sera appelé

Enfin, la commande complète la liste des bénéficiaires attachée au service offert en lui ajoutant un pointeur sur le partenaire bénéficiaire. Cette information sera nécessaire pour supprimer rapidement l'offre de ce service à un partenaire en particulier ou à tous les partenaires.

5.7.2.3 Retirer l'offre d'un service

La commande de retrait de l'offre d'un service permet de supprimer les structures de données système qui permettait à un partenaire autre que son propriétaire d'appeler ce service offert. Cette commande n'est autorisée qu'aux seuls serveurs. Elle vérifie que le partenaire attaché à la session active est bien de cette catégorie.

La commande doit contrôler ensuite que le service appartient à ce partenaire. Pour cela, la commande a accès au champ Liste des applications du partenaire sur la session active et peut parcourir leur liste de services pour rechercher le service offert. De ce service, elle peut retrouver le partenaire bénéficiaire auquel il faut retirer l'offre.

A partir du bénéficiaire, la commande supprime l'offre dans la table des offres attachée à ce partenaire. Puis, à partir du service, ce partenaire est retiré de la liste des bénéficiaires.

5.7.2.4 Supprimer une application

La commande de suppression d'une application permet de retirer une application de la table des applications ainsi que toutes les tables qui lui sont attachées. Cette commande n'est autorisée qu'aux seuls serveurs. Elle vérifie donc que le partenaire attaché à la session active appartient à cette catégorie.

Ensuite, la commande doit contrôler que l'application à supprimer appartient à ce partenaire. Pour cela, la commande a accès au champ Liste des applications du partenaire sur la session active et peut rechercher l'application à supprimer.

Lorsque l'application est trouvée, la commande doit :

- libérer la mémoire non-volatile occupée par les données. Elle possède pour cela de l'adresse et de la taille de chaque donnée dans la table des persistants attachée à l'application
- libérer la mémoire non-volatile occupée par le code. Elle possède pour cela des champs de l'application @segment et Taille du code

- retirer les offres de tous les services à leurs bénéficiaires. La commande utilise la même méthode que la commande de retrait d'une seule offre
- détruire les tables système des Persistants, des services et des fonctions attachées à l'application.

En dernier lieu, la commande supprime l'application de la table des applications.

5.7.2.5 Mettre à jour une application

La commande de mise à jour d'une application permet de remplacer fonctionnellement une application tout en gardant la même interface vis à vis des partenaires bénéficiaires des services de l'application, c'est à dire le nom de l'application et les noms d'alias des services offerts. Cette commande n'est autorisée qu'aux seuls serveurs qui ne peuvent mettre à jour que leurs applications. Elle vérifie donc que le partenaire attaché à la session active appartient à cette catégorie et peut rechercher l'application à mettre à jour à partir du champ Liste des applications.

La mise à jour ne peut avoir lieu que si l'interface de l'application reste inchangée. Pour cela, la commande lit, sur le canal attaché à la session active, la nouvelle liste des services de l'application. S'il manque un service dans la nouvelle liste, ce service ne doit pas être offert dans l'application actuelle. Il peut y avoir des services supplémentaires.

Si ces conditions sont remplies, la commande doit :

- envoyer à l'émetteur d'application l'état actuel de toutes les données de l'application. Elle utilise la table des données persistantes qui contient l'adresse et la taille des valeurs. Ces valeurs sont communiquées, sous la forme d'une suite d'octets et de sa longueur, par le canal associé à la session active
- libérer la mémoire non-volatile occupée par les données. Elle possède pour cela de l'adresse et de la taille de chaque donnée dans la table des persistants attachée à l'application
- libérer la mémoire non-volatile occupée par le code. Elle possède pour cela des champs de l'application @segment et Taille du code
- détruire les tables système des Persistants et des Fonctions attachées à l'application.
- ré-allouer de la mémoire non volatile pour le code et le charger de la même manière et dans les mêmes conditions qu'à l'installation. La nouvelle adresse du code est écrite dans le champ @segment de l'application
- importer le nouveau jeu de données persistantes dans la Carte Blanche comme à l'installation
- mettre à jour la table des services. Les nouveaux services sont insérés dans la table et les services n'existant plus sont détruits. Les champs @relative et Liste des Persistants Utilisés sont mis à jour pour les services restants. Le nom d'appel pourrait être également modifié car le champ Alias des offres est indépendant du nom réel du service et ne change donc pas l'interface de l'application vis à vis des bénéficiaires
- reconstruire la nouvelle table des fonctions de la même façon qu'à l'installation

La commande doit se protéger d'une erreur dans les informations qui lui sont transmises ou d'une panne qui survient pendant la mise à jour des tables attachées à l'application. Le problème

de l'intégrité de l'application survient en cas de modification partielle des tables, ou du code. Deux solutions peuvent être envisagées pour cette commande :

1. L'utilisation des mécanismes de commit et rollback. Elle suppose de recopier dans une partie de la mémoire non-volatile, la totalité de l'application qui pourra ainsi être restaurée en cas de problème. Cette solution paraît difficile à cause du volume de la copie.
2. La perte de l'application. Cette solution est envisageable car la mise à jour est effectuée par l'émetteur de l'application, qui peut donc réinstaller l'application complètement et ré-offrir les services aux partenaires auxquels il les a déjà offert. Cela suppose que l'émetteur conserve l'état des données de l'application que la commande lui fournit dans un premier temps. Avant de modifier quoi que ce soit aux tables et au code, l'application est supprimée logiquement de la table des applications et de la liste des applications de son propriétaire. De plus, seule la table des services est sauvegardée. Si la mise à jour ne peut s'effectuer complètement, l'application n'est plus accessible à la prochaine mise sous tension de l'objet nomade. Mais les bénéficiaires des services supprimés possèdent encore un accès erroné à l'application. Le mécanisme de rollback utilise la copie de la table des services pour supprimer ces liens de coopération avec les bénéficiaires.

5.7.2.6 Lister les applications installées

La commande de listage des applications installées permet à n'importe quel partenaire de connaître le nom des applications qui existent actuellement sur la Carte Blanche. Cette commande n'effectue donc pas de contrôle vis à vis du partenaire attaché à la session active.

La commande parcourt la table des applications et envoie sur le canal associé à la session active le nom de chaque application.

5.7.2.7 Lister les offres reçues

La commande de listage des offres reçues permet à un partenaire de connaître les services offerts dont il dispose et de contrôler l'arrivée ou le retrait d'une offre à son égard. Cette commande étant accessible par tous les partenaires, elle n'a pas à vérifier la nature (identifiant et catégorie) du partenaire attaché à la session active. Néanmoins, elle a accès au champ Liste des offres de ce partenaire. A partir de celui-ci, elle peut envoyer par le canal attaché à la session active le nom d'alias du service dont le partenaire dispose et par indirection (par le champ Application) le nom de l'application d'origine de ce service.

5.7.3 Les commandes relatives au Titulaire

Les commandes relatives au titulaire sont d'un usage moins fréquent et manipulent essentiellement la table du titulaire de la Carte Blanche. Cette table qui n'est constituée que d'un seul élément, n'existe pas à l'achat de l'objet nomade. Elle est créée par la commande Titulariser.

Les commandes sont :

- Titulariser
- Changer la situation du Titulaire
- Obtenir les coordonnées du Titulaire

5.7.3.1 Titulariser

La commande de titularisation permet de créer la table du titulaire ainsi que d'associer un nouveau partenaire particulier au système : le partenaire TITULAIRE. Cette commande peut être effectuée par le partenaire OFFICIEL seulement. Elle vérifie donc que le partenaire attaché à la session active est ce partenaire grâce au champ Identifiant. La Titularisation ne pouvant avoir lieu qu'une seule fois, la commande contrôle l'existence de la table du titulaire.

Ensuite la commande fait l'acquisition des références permanentes, puis des références sujettes à modification. Elle inscrit ces informations dans la table du titulaire qu'elle crée.

Enfin, la commande associe le partenaire d'identifiant TITULAIRE. Elle impose que ce partenaire soit de la catégorie des serveurs et que son responsable hiérarchique soit le partenaire SYSTEME. Le mode d'authentification et l'authentifiant sont définis par l'intervenant.

5.7.3.2 Changer la situation du Titulaire

La commande de changement de la situation du titulaire permet de modifier les champs de référence modifiables dans la table du titulaire. Cette commande peut être effectuée uniquement par le partenaire OFFICIEL. Elle vérifie donc que le partenaire attaché à la session active est ce partenaire grâce au champ Identifiant. La Carte Blanche devant être titularisée, elle contrôle également l'existence de la table du titulaire.

Après avoir fait l'acquisition de la nouvelle situation du titulaire par le canal associé à la session active, la commande met à jour les champs modifiables correspondants.

5.7.3.3 Obtenir les coordonnées du Titulaire

La commande d'obtention des coordonnées du titulaire permet d'envoyer sur le canal attaché à la session active les coordonnées du titulaire si celui-ci existe ou un message informant l'intervenant que la Carte Blanche n'est pas encore titularisée. Cette commande est accessible par tous les partenaires. Elle ne vérifie donc pas l'identifiant, ni la catégorie du partenaire attaché à la session active.

La commande a accès à la table du titulaire pour lire les coordonnées du titulaire et les envoyer à l'intervenant.

5.8 La primitive d'appel de service

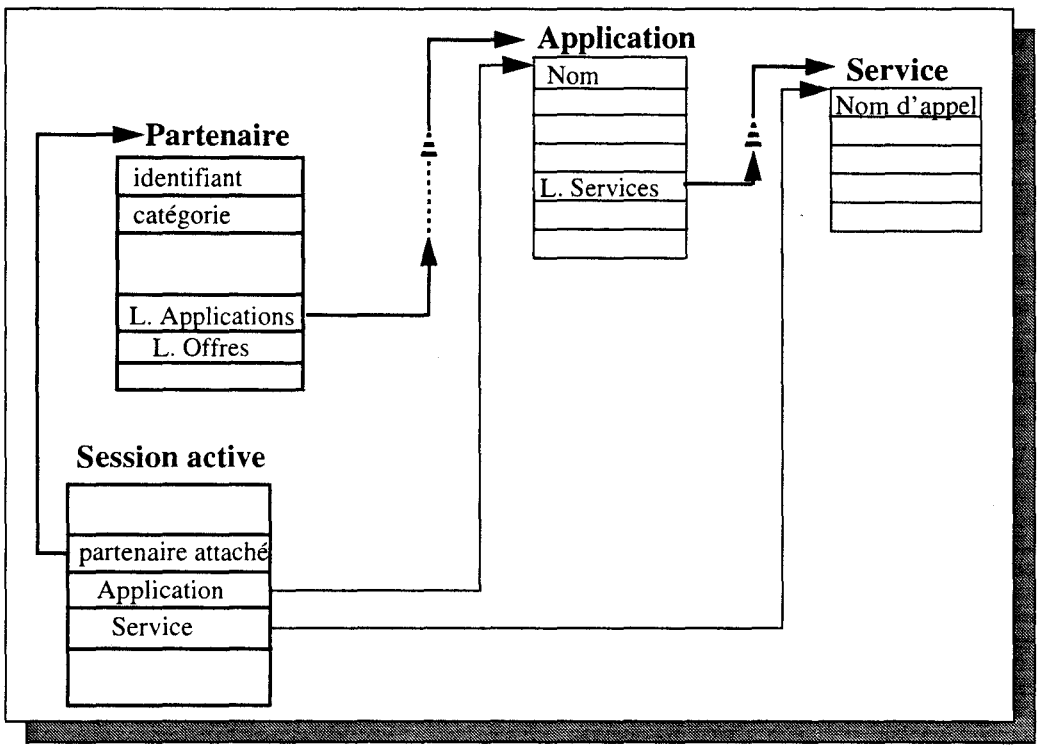
La primitive d'appel de service est appelée par l'application SHELL lorsque l'analyse lexicale et syntaxique n'a reconnu aucune commande de la Carte Blanche. La primitive d'appel reçoit de la part de l'application SHELL le nom du service et la liste de ses arguments. Elle doit rechercher si le service peut être utilisé par le partenaire attaché à la session active. Pour cela, elle recherche dans l'ordre le service

- parmi les applications du partenaire s'il en possède,
- parmi les offres dont dispose le partenaire

5.8.1 Recherche du service parmi les applications que possède le partenaire

A partir du champ Liste des Applications du partenaire attaché à la session active, la primitive peut rechercher le service appelé dans chaque application. Si le nom précisé dans la ligne de commande est de la forme nom_application.nom_service, la recherche est d'abord effectuée au niveau des applications, puis seulement si le nom de l'application correspond, elle est approfondie au niveau des services de cette application. Par contre, si le nom de l'application n'est pas précisé, le nom du service est recherché parmi toutes les applications. Dans ce cas, la première application qui possède un service de ce nom sera retenue. Si des services de même nom existe, il ont tous été installé par le partenaire actif qui est donc capable de les discerner.

FIGURE 27 Recherche d'un service parmi les applications d'un serveur



Quand le service est trouvé, la primitive doit permettre le traitement de ce service. Les informations nécessaires à ce traitement sont réparties dans les éléments application et service des tables correspondantes. La primitive donne l'accès direct à ces éléments à partir de la session pour faciliter :

- la mise en place du service pour qu'il soit traité
- la configuration du mécanisme de sécurité pendant le traitement
- l'accès aux ressources de l'application (données et fonctions)
- la reprise du service après suspension. Quand la session est de nouveau active, elle retrouve directement les mêmes informations

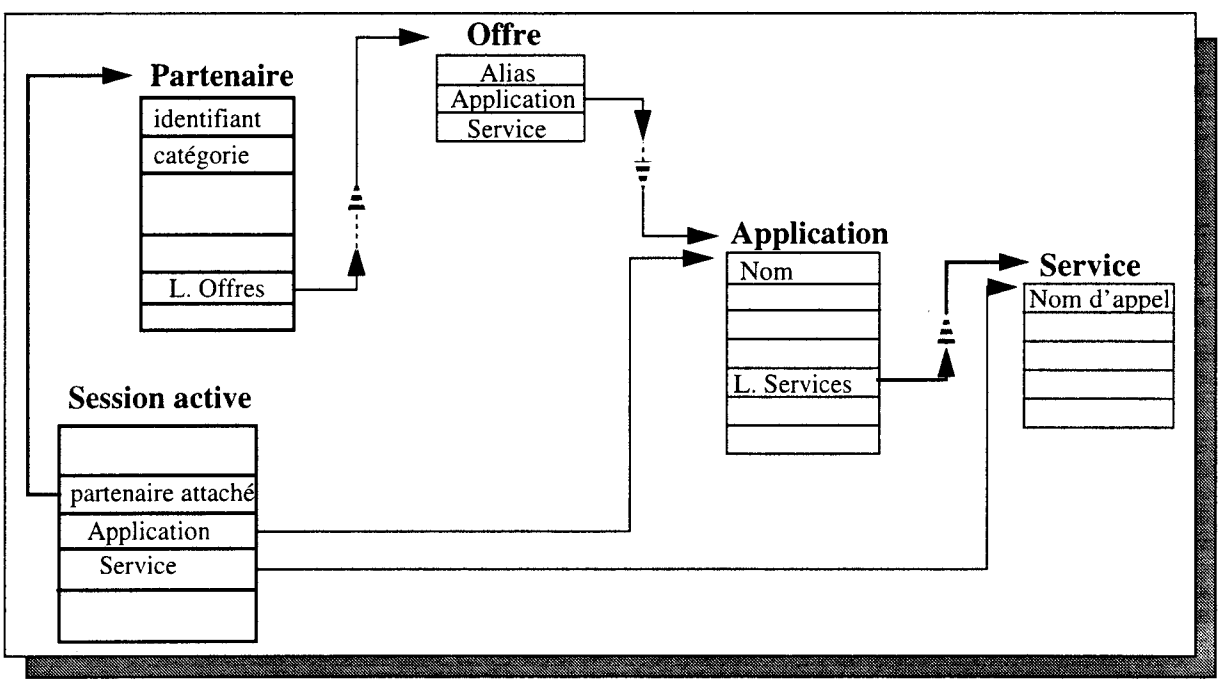
Si le service demandé n'est pas trouvé dans les applications appartenant au partenaire actif, la recherche se poursuit parmi les services offerts à celui-ci.

5.8.2 Recherche du service parmi les offres dont dispose le partenaire

A partir du champ Liste des offres du partenaire attaché à la session active, la primitive peut rechercher le service demandé en le comparant au nom d'alias de chaque offre. Si le nom précisé dans la ligne de commande est de la forme nom_application.nom_service, seule la partie nom du service située après le point est utilisée pour la recherche. Si ce nom est trouvé, la primitive doit contrôler l'égalité des noms d'application, l'un précisé dans la ligne de commande, l'autre obtenu indirectement par le champ Application de l'offre. En cas d'inégalité, la recherche se poursuit. Par contre, si le nom de l'application n'est pas précisé, le premier nom d'alias répondant au service demandé est sélectionné quelle que soit l'application.

Lorsque le service offert est trouvé, la primitive doit permettre de traiter ce service de la même manière que pour les services appartenant au partenaire. Cette fois-ci, les éléments application et service sont obtenus à partir de l'offre retenue et la primitive donne à la session active un accès direct à ces éléments.

FIGURE 28 Recherche d'un service parmi les offres accordées à un partenaire



5.8.3 La liste des arguments

La liste des arguments transmise à la primitive est copiée en mémoire de travail dont l'adresse est mémorisée au niveau de la session. Ces arguments ne sont pas directement accessibles par le service qui les récupérera dans son espace de travail par un appel système. Cet appel système copiera l'argument demandé à l'adresse indiquée en paramètre de cet appel.

5.9 Exécution d'un service

Quand un service est reconnu par la primitive d'appel, la session active possède toutes les informations pour mettre en oeuvre ce service. Elle a accès aux éléments application et service

qui lui permettent de connaître l'adresse du code du service grâce aux champs @segment de l'application et @relative du service, l'adresse de chaque donnée par la table des Persistants liée à l'application.

5.9.1 Mise en Oeuvre du code

La mise en oeuvre du code de l'application dépend de la nature de ce code. Les différentes natures de ce code sont :

- le code natif directement exécutable par le micro-processeur de l'objet nomade
- le code interprété
- une solution mixte : interprétation du code d'une machine virtuelle

Nous présentons les avantages et les inconvénients de ces différentes solutions. Le choix de l'une d'elles dépendra du matériel utilisé sur l'objet nomade, du coût de développement de ce matériel, de la variété des applications qu'il permettra de développer, des performances et de la sécurité.

5.9.1.1 Exécution de code natif

Le code de l'application est du code natif qui est directement exécutable par le micro-processeur de l'objet nomade.

Les principaux avantages de cette solution sont :

- la puissance du micro-processeur est exploitée pleinement par les applications
- le jeu d'instructions de bas niveau permet de développer tout type de traitement, et confère à la Carte Blanche la liberté des applications qu'elle pourra accepter
- les applications sont exécutées aussi rapidement que le système d'exploitation

Pour des raisons de sécurité du système d'exploitation et des autres applications, l'exécution directe nécessite le contrôle par des dispositifs matériels. Ce contrôle intervient à deux niveaux : au niveau du processeur et au niveau de la gestion de la mémoire. Le processeur doit disposer de deux modes d'exécution :

- le mode protégé pour l'exécution des services dans lequel le code n'a pas accès aux instructions d'entrées/sorties et l'accès à la mémoire est limité par une unité de gestion de la mémoire
- le mode superviseur pour l'exécution du système d'exploitation qui a accès à toutes les instructions et à toute la mémoire

La mémoire doit être protégée par une unité de gestion de la mémoire (M.M.U.) pour que le code n'accède qu'aux zones utiles de l'application en mode protégé.

L'inconvénient de l'exécution directe est que le code d'une application développé pour un objet nomade donné n'est pas portable sur toutes les machines. Cependant, si l'émetteur d'application a écrit son application dans un langage évolué, sa compilation pour plusieurs types de machine est facile.

5.9.1.2 Interprétation du code

Le code de l'application est mis en oeuvre par un interpréteur sécurisé d'un langage évolué, par exemple un interpréteur Forth. Les principaux avantages de cette solution sont :

- toutes les instructions sont contrôlées par l'interpréteur
- un gain sur l'occupation de la mémoire par l'application car le code interprété est de taille inférieure au code direct et l'interpréteur est commun à toutes les applications
- les applications sont portables sur tout type d'objet nomade utilisant le même interpréteur

La difficulté est dans le choix du langage à interpréter. Peut-on utiliser un langage existant ou faut-il développer un nouveau langage ? Du point de vue des émetteurs d'application, les critères de choix sont :

- La connaissance et l'utilisation du langage. Un langage connu augmenterait le nombre de développements de nouvelles applications par des émetteurs potentiels. L'idéal pour l'émetteur serait de pouvoir utiliser le même langage pour l'application carte et pour le logiciel correspondant sur le système hôte
- Le niveau d'évolution du langage. Un langage plus évolué peut faciliter le développement d'une application et peut générer un code plus sûr.
- La polyvalence du langage. Le langage doit permettre de développer tous types d'application et de manipuler de nombreuses sortes de données, simples ou complexes

Les critères techniques de choix pour un objet nomade sont :

- La compacité et complexité de l'interpréteur. Du fait des capacités relativement réduites des objets nomades, le code de l'interpréteur doit être compact pour laisser de la mémoire disponible pour les applications. De même, la complexité de l'interpréteur doit être en rapport avec les capacités du micro-processeur de l'objet nomade
- Un langage extensible suivant les besoins des applications. L'interpréteur posséderait un noyau compact dont le jeu d'instructions pourrait être étendu individuellement par chaque application selon ses besoins
- La sécurisation facile des instructions. Suivant la sémantique de chaque instruction et la complexité des informations manipulées, le langage peut être plus ou moins facile à sécuriser
- La performance du code interprété. Le code interprété doit être suffisamment rapide compte-tenu du matériel de l'objet nomade

5.9.1.3 Solution Mixte

Une solution mixte est intéressante à étudier. Le code de l'application est écrit dans le langage d'une machine virtuelle de préférence R.I.S.C. et sera interprété. Cette solution présente à la fois les avantages du mode direct et ceux du mode interprété :

- le code virtuel est proche du code réel. Le traitement de l'application peut donc être aussi libre que pour la machine réelle. La correspondance entre les instructions virtuelles et les instructions réelles est simple
- la protection des instructions virtuelles est facile. La sémantique des opérandes est entièrement spécifiée. Il s'agit soit d'adresse dans le code, soit d'adresse directe ou indirecte dans

le segment de données, ou de constantes. La protection est identique à celle pour le mode direct et peut être assurée par une unité de gestion mémoire (M.M.U.) logicielle

L'intérêt d'une machine R.I.S.C. est que son jeu d'instructions est réduit et que toutes les instructions sont au même format. L'interpréteur répond positivement aux critères de choix cités précédemment car il est facile à réaliser, compact et rapide.

Cette solution peut être vue, comme une simulation des modes protégé et superviseur, sur une machine qui n'en possède pas. L'exécution du système d'exploitation en code natif correspond au mode superviseur tandis que l'interprétation des applications en code virtuel s'apparente au mode protégé.

5.9.2 Accès à la mémoire par le code

Quelle que soit la nature du code des applications, celui-ci aura besoin d'accéder aux données de l'application mais aussi à des variables de travail. Les données persistantes sont déjà présentes en mémoire non volatile tandis qu'un segment de mémoire de travail doit être alloué à l'appel du service pour ses variables dynamiques. Nous allons maintenant discuter des différentes solutions d'accès aux données et aux variables de l'application.

5.9.2.1 Accès aux données

L'accès aux données par le code de l'application pose deux types de problèmes. Le principal est la protection de l'intégrité de ces données en cas d'arrêt anormal d'un traitement. La solution proposée résout également le second problème de la différenciation des accès à la mémoire par le code pour les données et pour ses variables.

Les données persistantes d'une application se situent en mémoire non-volatile qui nécessite une procédure d'écriture différente de celle d'une mémoire de travail pour les variables. Le traitement d'une application utilise non seulement les données de l'application mais aussi des variables dynamiques qui sont situées en mémoire de travail. Le code devrait distinguer pour chaque instruction si les opérandes proviennent de la mémoire non-volatile ou de travail. Cette distinction devrait se faire par le développeur de l'application si le code est natif, par l'interpréteur si le code est interprété. Dans tous les cas, cette différenciation est contraignante et il serait intéressant d'uniformiser les accès à la mémoire en copiant les données persistantes nécessaires au traitement dans la mémoire de travail.

La modification des données persistantes directement en mémoire non-volatile peut générer un problème d'intégrité en cas d'arrêt anormal du traitement car les données auraient été partiellement mises à jour en mémoire. Des mécanismes de commit et rollback peuvent être mis en oeuvre. Mais il est plus simple de conserver en mémoire non volatile un jeu de données cohérent et d'effectuer les modifications pendant toute la durée d'un service sur une copie en mémoire de travail. Cette copie sera ré-écrite en mémoire non volatile à la fin du service. Par contre, si le service est interrompu anormalement, les modifications sont perdues, et l'application conserve un jeu de données cohérent en mémoire non-volatile. La mise à jour des données nécessite l'utilisation du commit et du rollback. Mais ce mécanisme n'interviendrait qu'une seule fois en fin d'exécution d'un service pour l'ensemble des données utilisées et les risques de dysfonctionnement, pendant la mise à jour de plus courte durée, seront proportionnellement plus faibles.

La copie des données persistantes en mémoire de travail peut suivre différentes stratégies :

- copie intégrale du jeu de données

  Toutes les données de l'application sont chargées en mémoire de travail à l'appel du service. L'avantage est que le mécanisme de copie intervient une fois, pour toute la durée du service, quel que soit le déroulement du service et les fonctions qu'il sera amené à appeler. Par contre, le service peut n'utiliser qu'une partie des données et la copie intégrale peut paraître longue et inutile. Cette remarque est également valable pour le mécanisme de commit et rollback.

  En outre, cette solution interdit l'exécution de plusieurs services d'une même application simultanément sur des sessions différentes.

- copie partielle des données nécessaires au service globalement

  Seules les données directement nécessaires au code du service et, à celui des fonctions que ce service appelle, sont copiées en mémoire de travail. Le mécanisme de copie n'intervient encore qu'une seule fois mais la durée de copie est optimisée.

  Cette solution autorise l'utilisation simultanée de deux services d'une même application n'utilisant pas les mêmes données, sur différentes sessions.

- copie partielle des données pour chaque service et chaque fonction

  Seules les données nécessaires directement par le code du service sont copiées en mémoire de travail. Chaque appel de fonction copie les données, nécessaires à la fonction, et non encore chargées. Le principal intérêt de cette solution est l'appel de fonction conditionnel. Si une fonction n'est appelée que dans certaines conditions d'utilisation du service, les données nécessaires seulement à cette fonction ne sont pas copiées inutilement en mémoire de travail et restent disponibles pour d'autres services de l'application dans d'autres sessions.

  Cependant, cette solution pourrait amener à des situations d'interblocage. Par exemple, deux services fonctionnant en même temps appellent chacun une fonction utilisant une donnée du service exécuté sur l'autre session.

  La mise à jour des données en mémoire non-volatile doit généralement être effectuée à la fin du service pour la cohérence des données. Mais une mise à jour au retour d'une fonction est envisageable uniquement si l'émetteur de l'application considère que les modifications apportées par cette seule fonction génèrent un jeu de données cohérent pour l'application. Dans ce cas, l'émetteur précisera ce fonctionnement au système d'exploitation lors de l'installation de l'application.

Pour améliorer la disponibilité des données persistantes des applications, le programmeur pourrait également préciser si les données utilisées par chaque service et chaque fonction sont en lecture seule ou en lecture et écriture. Plusieurs services utilisant les mêmes données en lecture seulement pourraient fonctionner simultanément.

5.9.2.2 La structure du segment de variables

Quelle que soit la nature du code de l'application, il a besoin de variables dynamiques pour son exécution. Pour conserver l'indépendance des services en cours d'exécution directe par le microprocesseur ou d'interprétation, chaque session possédera son segment de variables individuel.

Le code de l'application ne peut pas connaître l'adresse réelle des variables dans le segment puisque celle-ci peut changer d'une exécution à l'autre. Il utilise un adressage relatif à ce segment. Les fonctions sont obligées de posséder leur propre segment de variables car elles peuvent être appelées de différents services et même d'autres fonctions. Les adresses relatives ne peuvent donc plus être identiques.

L'organisation des variables dynamiques à l'intérieur des segments de variables est transparente au système d'exploitation qui ne fait que leur allouer de la mémoire. Cette organisation revient au programmeur, au compilateur ou à l'interpréteur.

Le code de l'application doit également accéder aux copies des données persistantes. Le code ne connaît pas non plus l'adresse réelle de ces copies et doit donc recourir à un adressage relatif. Pour obtenir une continuité des accès à la mémoire par le code, les données et les variables doivent utiliser le même espace d'adresses relatives.

Cependant, les données ne peuvent pas être copiées directement dans le segment de variables du service car les fonctions appelées par le service doivent utiliser la même copie des données et le code de l'application doit toujours accéder de la même façon à ces données, que ce code soit celui d'un service ou d'une fonction. De plus, les fonctions ne vont pas toujours trouver la même configuration des données en mémoire suivant qu'elles seront appelées d'un service ou d'un autre, ou encore de telle ou telle autre fonction.

Les données, qui ont provoqué la suspension de l'exécution d'un service car elles étaient utilisées sur une autre session, possèdent déjà une copie en mémoire de travail. Lorsqu'elle se termine, l'autre session recopiera ces données en mémoire non volatile. Mais, pour éviter trop de manipulations des données persistantes entre la mémoire non-volatile et la mémoire de travail, leur copie sera conservée en mémoire de travail pour la session en attente. De ce fait, les services également ne trouveront pas la même configuration des copies des données.

Par conséquent, les données persistantes sont chargées en mémoire de travail, indépendamment des segments de variables. Ensuite, le système doit permettre au code de l'application d'y accéder à partir de son segment de variables. Pour cela, il inscrit consécutivement à partir de la première adresse du segment de variables du service ou de la fonction, l'adresse réelle de chaque donnée nécessaire dans un ordre spécifique à chaque service et à chaque fonction. Le micro-processeur ou l'interpréteur devront posséder un adressage indirect ou alors le système devra disposer d'un adressage virtuel en mode protégé cachant totalement l'indirection au code de l'application.

5.9.3 Préparation de l'environnement d'exécution du service

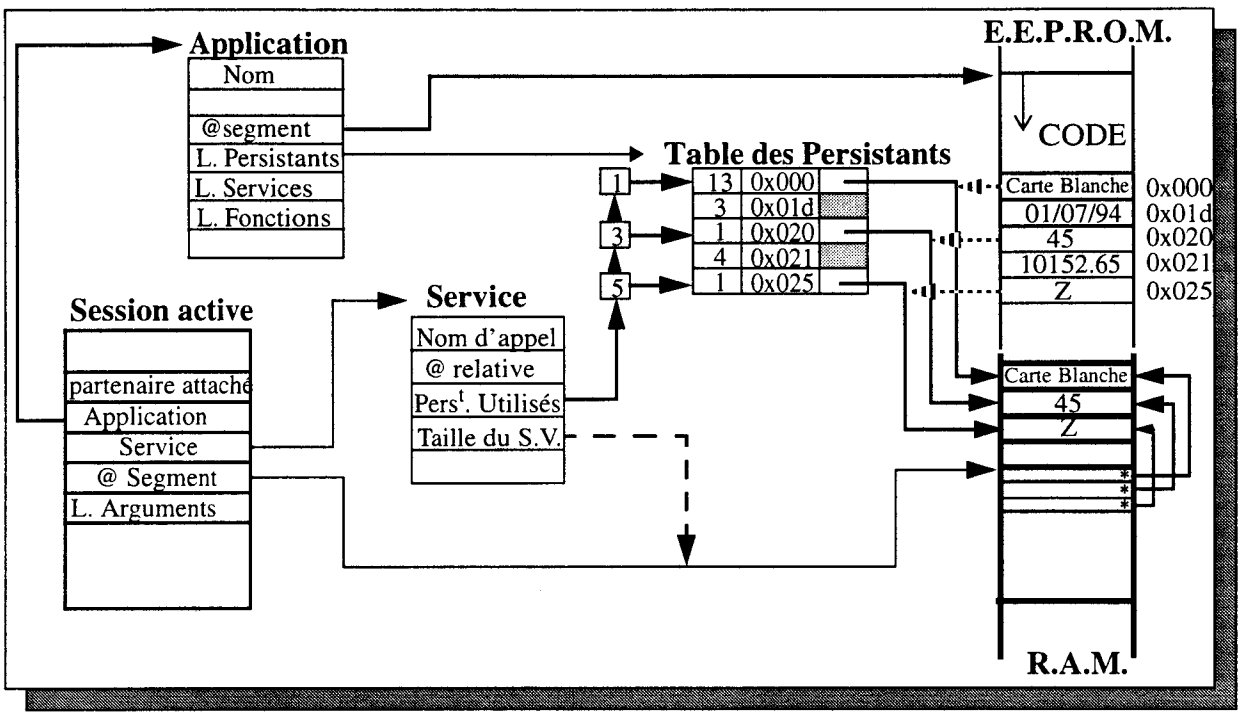
Avant de traiter le code du service, la primitive doit copier les données persistantes en mémoire de travail et allouer de la mémoire de travail pour le segment de variables.

L'adresse des copies des données persistantes est mémorisée au niveau de la table des Persistants afin que les fonctions qui seront appelées par la suite puissent accéder à la même copie et que d'autres sessions n'y accèdent pas en même temps mais puissent les utiliser par la suite.

Toutes les données de l'application ne sont pas copiées en mémoire de travail. Cela permet d'utiliser sur une autre session d'autres services de la même application qui ne manipulent pas le même jeu de données persistantes.

L'adresse de segment de variables est mémorisée au niveau de la session afin que celle-ci puisse la retrouver après une suspension.

FIGURE 29 Exécution d'un service



5.9.4 Appel à une fonction

L'exécution du service peut conduire à un appel de fonction, c'est à dire un saut à une sous routine de l'application qui retourne au programme appelant quand elle se termine. Cette fonction ne partage pas les variables du service appelant puisque ce service peut ne pas toujours être le même. Il possède donc son propre segment de variables. Par contre, il peut utiliser tout ou partie des mêmes données persistantes et également en utiliser d'autres. Le service (ou une autre fonction) appelant peut passer des paramètres à cette fonction. La fonction peut renvoyer une valeur quand elle retourne au programme appelant.

5.9.4.1 Le passage de paramètres et la valeur de retour

Le passage de paramètres entre un programme et une sous-routine, ainsi que le retour de valeur, utilisent généralement une pile ou une mémoire commune. Ne connaissant pas le matériel utilisé pour la Carte Blanche, nous ne pouvons pas baser ces mécanismes sur une pile physique. Les segments de variables du service (ou de la fonction) appelant(e) et de la fonction appelée étant disjoints, la deuxième méthode n'est pas non plus utilisable. Les mécanismes de passage de paramètres et de retour d'une valeur des fonctions doivent être pris en charge par le système d'exploitation de la Carte Blanche ou l'interpréteur, qui simuleront une pile logiciellement.

Les différents modes de passage de paramètres doivent être disponibles. Il s'agit soit du passage de constante, soit du passage du contenu d'une variable du programme appelant par valeur ou par variable. Le passage par valeur consiste à effectuer une copie de la variable dans le segment de variables de la fonction. Cette copie peut évoluer pendant le service mais la variable d'origine conservera sa valeur initiale. Par contre, le passage par variable consiste normalement à utiliser la même variable dans la fonction, ce qui a pour effet que l'évolution de cette variable pendant la fonction est automatiquement répercutée au retour au programme appelant. Les deux premiers modes peuvent être implanter dans le système d'exploitation de la Carte Blanche mais le dernier ne peut pas être à proprement parler un passage par variable. En effet, la fonction ne doit pas avoir accès aux variables du programme appelant et ne peut accéder qu'à celles de son propre segment (protection de la mémoire). Pour obtenir le même fonctionnement, le système devra recopier les paramètres passés par variables à l'adresse des variables du programme appelant.

Il est inutile de fournir des données de l'application en paramètre puisque la fonction peut y avoir accès autrement.

La méthode proposée pour ces passages de paramètres est la suivante. Elle repose sur le fait que le segment de variables de la fonction n'existe pas encore lorsque le programme appelant prépare les paramètres. Ce segment n'est crée qu'au moment du saut à la sous-routine. Le programme utilise un appel système qui crée une table dynamique de passage de paramètre. Cette table contient l'adresse, la taille et le type de chaque paramètre. A l'appel de la fonction, cette table servira à copier, dans le segment de variables de la fonction, les constantes situées dans le code de l'application, et les variables situées dans le segment de variables du programme appelant. Ensuite, au retour de la fonction, cette table permettra de recopier la valeur des paramètres passés par variables dans le segment de variables du programme appelant.

La valeur retournée par la fonction doit être écrite dans le segment de variables du programme appelant. Le programme doit préciser, avant l'appel, à quelle adresse la valeur doit être inscrite. Cette adresse est ajoutée à la fin de la table des paramètres. La fonction indique au système l'adresse où se trouve la valeur à retourner. Le système recopiera cette valeur à l'adresse précédente en même temps que les paramètres passés par variables.

La table dynamique des paramètres est détruite avant de reprendre l'exécution du programme appelant.

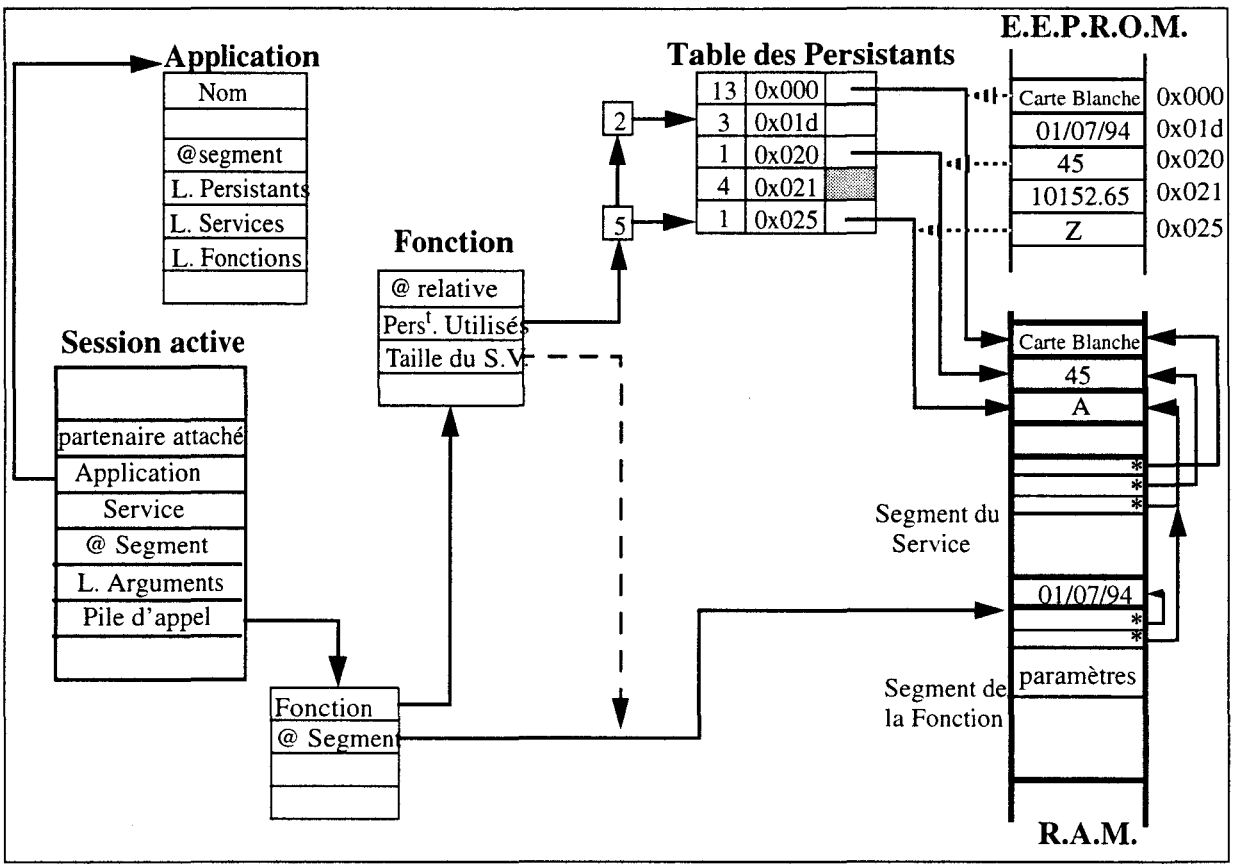
5.9.4.2 Mécanisme de déroutement de l'exécution

Le saut à une fonction doit être détecté en mode protégé par le système ou par l'interpréteur. Les informations concernant cette fonction sont retrouvées par la recherche dans la table des fonctions de l'application sur l'adresse relative d'appel.

Les données persistantes utilisées et non encore utilisées sont alors copiées en mémoire de travail. Si l'une d'elle est utilisée sur une autre session, la session active est suspendue. Le risque d'interblocage est possible mais il reste peu probable pour la Carte Blanche. Si, le cas survenait, il faudrait couper l'alimentation de la Carte Blanche, et la cohérence des applications serait assurée par les mécanismes de copie des données en RAM.

Un nouveau segment de variables, de la taille spécifiée dans la table des fonctions de l'application, est alloué. L'adresse des copies des données persistantes est écrite en tête de ce segment.

FIGURE 30 Appel d'une fonction



Les appels de fonction consécutifs sont maintenus dans une pile d'appel liée à la session afin de retrouver les informations du programme appelant à chaque retour de fonction.

Le code de la fonction peut alors être exécuté jusqu'à l'instruction de retour au service (ou à la fonction) appelant(e). Les données persistantes utilisées seulement par cette fonction sont recopiées en mémoire non-volatile si cette option est choisie à l'installation de l'application. Les autres restent dans leur état pour la suite du service. D'ailleurs, dans la figure précédente, la copie de la cinquième donnée de la table des persistants a changé de valeur au cours de la fonction. Le segment de variables est libéré et le service peut reprendre grâce aux informations maintenues au niveau de la session et de sa pile d'appel.

5.9.5 Les appels système

Les appels système permettent aux applications d'avoir accès à des ressources dont elles ne disposent pas car elles sont protégées ou partagées. Ils fournissent également aux applications les moyens de contrôler leur accès.

Les appels système sont demandés par une interruption logicielle (réelle ou interprétée), et la primitive correspondante est exécutée en code natif (mode superviseur ou code réel).

Les paramètres des appels système peuvent être fournis soit par les registres du micro-processeur, soit en utilisant une zone mémoire particulière appelé «page zéro». Les applications ont

accès à cette page pour écrire les paramètres à fournir avant d'effectuer l'appel système correspondant.

Les appels systèmes essentiels pour les applications d'une Carte Blanche sont ceux concernant les entrées/sorties. Ils permettent d'une part aux programmeurs de ne pas se soucier de la constitutions des messages en trames, et au système d'assurer le cloisonnement des communications sur les différents canaux. Ces appels sont :

- envoyer un message
- recevoir un message

Ils sont utilisés indifféremment pour les canaux internes et pour les canaux externes pour obtenir le même fonctionnement d'un service utilisé de l'extérieur ou dans le cadre d'une coopération.

Les appels système sont nécessaires à la coopération entre applications. L'appel d'un service offert demande l'ouverture d'une session interne et la vérification des droits d'utilisation de ce service par le partenaire attaché à la session active. De même, la fin de la coopération doit être demandée pour que la session interne soit fermée et que la session appelante retrouve son canal de communication initial avec l'extérieur. Ces appels système sont donc :

- appel d'un service offert
- fin de coopération

Enfin, des appels système vont permettre aux applications de contrôler leur environnement d'exécution. Ce sont :

- lire un argument

  cet appel permet de récupérer un argument en le copiant dans le segment de variables du service
- qui est l'utilisateur du service en cours ?

  cet appel permet de référencer les différentes utilisations d'un service par les différents bénéficiaires du service
- un service est-il offert ?

  cet appel est très important car il permet de savoir si un service est effectivement offert avant d'essayer de l'utiliser pour ne pas provoquer d'erreur.

5.10 Protection pendant l'exécution d'un service

L'exécution de code importé à l'intérieur d'un objet nomade multi-applicatif pose le problème de la protection des autres applications vis à vis de ce code. Pendant l'exécution d'un service (ou d'une fonction) le système doit assurer que :

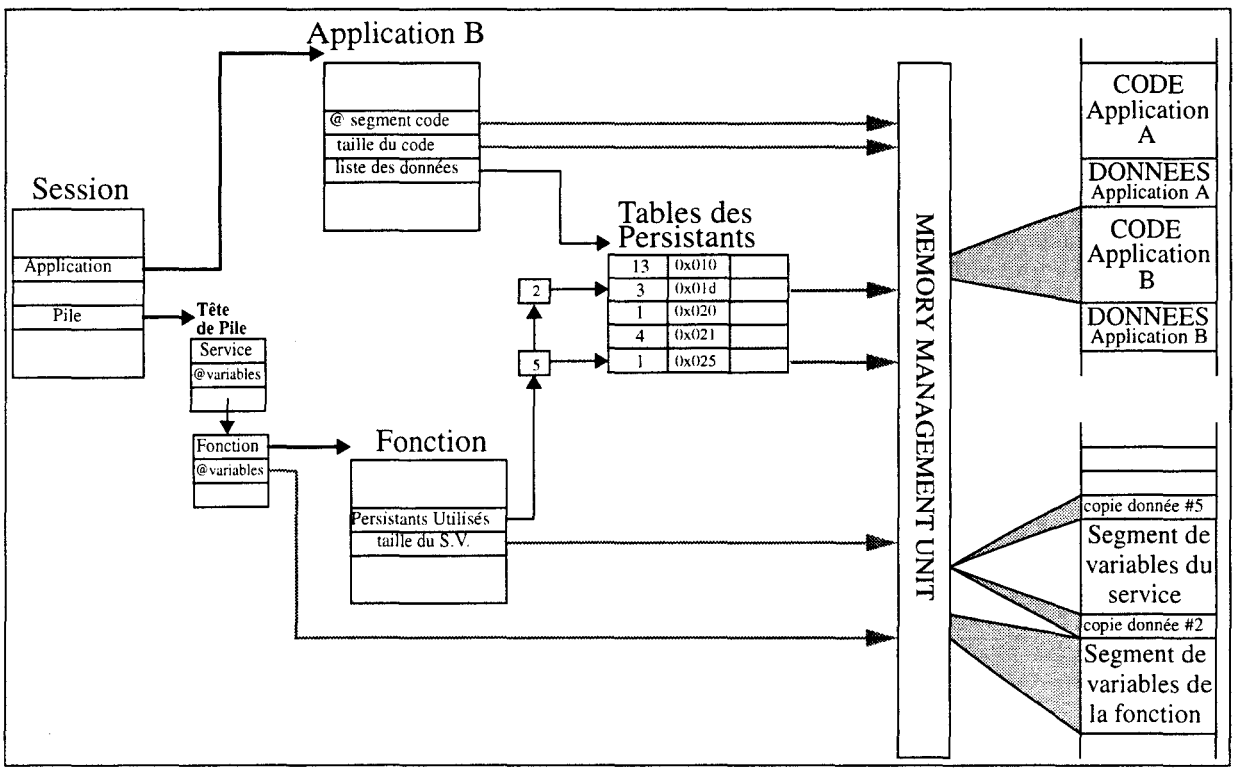
- chaque instruction exécutée est incluse dans le code de l'application
- ces instructions ne puissent manipuler que les informations concernant l'application et le traitement en cours

Dans la Carte Blanche, ces informations se situent :

- pour les constantes, dans le segment de code en mémoire non-volatile
- pour les variables, dans le segment de variable en mémoire de travail
- pour les copies des données persistantes de l'application, en mémoire de travail également

Les services et les fonctions des applications ont aussi accès à la page zéro en mémoire de travail pour les échanges de paramètres avec les appels système.

Une unité de gestion de la mémoire (M.M.U. : Memory Management Unit) permettrait de protéger la mémoire lors de l'exécution des services et fonctions dans la Carte Blanche. Cette M.M.U. doit être logicielle dans le cas du code interprété, et matérielle pour le code natif.



Le cahier des charges de cette unité pour la Carte Blanche est le suivant :

- elle ne doit être activée qu'en mode protégé du micro-processeur ou lors de l'interprétation
- elle doit protéger, en lecture et exécution, le code de l'application
- elle doit protéger, en lecture et écriture, le segment de variables et les copies des données
- elle doit donner l'accès libre, en lecture et écriture, à la page zéro
- elle doit interdire l'accès au reste de la mémoire

La M.M.U. doit être paramétrée par le système d'exploitation avant d'exécuter un service et à chaque changement de contexte d'exécution. Ces changements sont l'appel d'une fonction et l'élection d'une nouvelle session active. A partir de la session active, le système dispose de toutes les informations pour paramétrer l'unité. Par son accès à l'application, il obtient l'adresse et la

taille du segment de code. Il accède également à la table des Persistants qui, conjuguée à la liste des Persistants Utilisés obtenue par le service, fournit l'adresse et la taille des copies de données. Enfin, l'adresse du segment de variables est obtenue dans la session active ou dans sa pile d'appel et sa taille se situe au niveau du service ou de la fonction en cours.

5.11 Les suspensions de session

Une session élue par l'ordonnanceur peut être suspendue par le système car son activité ne peut pas être momentanément poursuivie. Une suspension peut intervenir soit lors de l'appel d'un service ou d'une fonction, soit lors d'un appel système.

Une session peut être reprise quand la cause de sa suspension disparaît. Dans ce cas, la session est placée dans l'état prêt pour que l'ordonnanceur puisse à nouveau l'élire. Ce mécanisme doit être pris en charge par les primitives qui tendent à supprimer chaque cause de suspension.

Les causes de suspension et les mécanismes de reprise correspondants sont décrits ci-après.

5.11.1 suspension à l'appel d'un service ou d'une fonction

Lorsque la primitive d'appel de service a reconnu un service et le droit d'utiliser ce service par le partenaire attaché à la session active, elle doit être en mesure de mettre en oeuvre ce service. Les raisons de suspension de la session et les mécanismes de reprise correspondants sont :

- un service de la même application est en cours d'exécution sur une autre session si le système a choisi l'option d'interdire l'exécution multiple de services d'une même application. La session est ajoutée à une file d'attente de sessions suspendues propre à l'application. Lorsque qu'un service de cette application se termine, la première session de cette file d'attente est placée dans l'état prêt
- une des données nécessaires au service est déjà utilisée sur une autre session si l'exécution multiple de service de la même application est autorisée. La session est ajoutée à une file d'attente de sessions suspendues propre à l'application. Lorsqu'un service ou une fonction de cette application libère des données, toutes les sessions de cette file d'attente sont placées dans l'état prêt
- il n'y a plus de mémoire de travail disponible pour le segment de variables du service. La session est ajoutée à une file d'attente propre au défaut de mémoire. Dès que de la mémoire est libérée, toutes les sessions de cette file d'attente sont placées dans l'état prêt

La primitive d'appel de fonction peut également provoquer la suspension de la session pour ces deux dernières raisons.

5.11.2 suspensions par les appels système

Un appel système suspend la session active lorsque celui-ci ne peut pas être satisfait. Les appels système pouvant conduire à une suspension, les causes de ces suspensions et les mécanismes de reprise correspondants sont les suivants :

- l'appel système lire un message provoque la suspension de la session s'il n'y a aucun message disponible sur le canal attaché à la session active. La session est à nouveau placée

dans l'état prêt par l'algorithme d'ordonnancement, qui détecte la présence d'un message sur le canal attaché à la session, lorsqu'il scrute les ports de communication

- l'appel de service offert suspend la session lorsque le service appelé est lui même suspendu pour une raison exposée au paragraphe précédent. La reprise d'une session interne provoque le placement de la session d'appel dans l'état prêt
- l'appel système Fin de coopération provoque la suspension de la session si le service appelé n'est pas terminé. La primitive de fermeture de la session interne qui clôture l'exécution du service offert, place la session d'appel en état prêt

5.12 Conclusion

Ce chapitre montre comment peuvent être implémentés les concepts de la Carte Blanche. Le système d'exploitation qui en résulte n'est pas compliqué mais il est très volumineux et les mécanismes de protection sont assez nombreux. L'objet nomade qui utilisera ce système d'exploitation devra avoir des ambitions à la mesure des possibilités de la Carte Blanche. Cet objet ne sera certainement pas une carte à micro-processeur telle qu'elle est définie dans la norme ISO 7816.

Une simulation de ce système d'exploitation sur station SUN a été réalisée pour éprouver les concepts de la Carte Blanche et montrer la faisabilité du système d'exploitation décrit dans ce chapitre. Cette réalisation est décrite au chapitre 6.

De plus, pour que ce système soit opérationnel, il faut pouvoir alimenter la Carte Blanche en applications. Le chapitre 6 propose également une chaîne de développement d'application pour la Carte Blanche.

Chapitre 6
Réalisation d'un prototype et Développement d'applications

6.1 Introduction

Après avoir défini un nouveau modèle de fonctionnement pour objet nomade et déterminé les caractéristiques du système d'exploitation nécessaire à l'obtention de ce fonctionnement, il nous reste à évaluer et à valider le modèle de la Carte Blanche. Nous avons alors développé un prototype de système d'exploitation pour la Carte Blanche. Ce prototype se base sur le cahier des charges que forment les chapitres 4 et 5 de ce mémoire. Il doit montrer la faisabilité de ce modèle, y apporter des solutions techniques en matière de matériel, système et logiciel.

Certains points sont encore volontairement non spécifiés car la définition de ces points est laissée à chaque constructeur de Carte Blanche. Il s'agit essentiellement des caractéristiques du matériel. Certaines fonctions du système d'exploitation découlent directement de ces choix. La façon de mettre en oeuvre le code des applications en est une. Pour notre prototype, nous avons développé un interpréteur sécurisé.

Enfin, pour que le prototype soit complet, il faut être en mesure de l'alimenter en applications. Afin d'inciter des émetteurs à produire des applications pour la Carte Blanche, il est souhaitable de leur faciliter le développement en leur fournissant des outils spécifiques de développement dans un langage évolué. Comme la Carte Blanche n'est pas un environnement de programmation mais plutôt un environnement de programme, l'application est développée à l'extérieur de l'objet nomade et est fournie à la Carte Blanche sous une forme directement exploitable par elle.

6.2 Le prototype

Le prototype doit avoir à la fois des objectifs de démonstration et d'évaluation. Du point de vue démonstratif, le prototype doit montrer l'intérêt du modèle de fonctionnement de la Carte Blanche, c'est à dire, principalement, :

- les différents niveaux de création de nouveaux partenaires sous le seul contrôle du porteur «légitime»

- l'installation de nouvelles applications indépendantes par des partenaires serveurs différents qui sont individuellement les seuls à gérer leur application respective. Cette installation assure la protection des autres applications présentes et à venir.
- les offres de services et la coopération
- la titularisation

Le prototype doit également permettre d'évaluer le système d'exploitation au niveau de ses mécanismes mis en oeuvre et de la sécurité. Les principaux sont :

- l'exécution des services et fonctions d'application assurant la confidentialité et la cohérence des applications
- Le fonctionnement multi-session
- La gestion et la protection de la mémoire
- les jeux de commandes et de primitives du système

6.2.1 Plate-forme de simulation

Il nous faut réaliser un prototype de système d'exploitation. Nous devons choisir la plate-forme de simulation et donner des réponses techniques.

Nous n'avons pas de véritable machine cible pour accueillir le système d'exploitation. Nous avons juste mis à jour quelques caractéristiques de cette machine cible. Il faut écrire le système d'exploitation sur un autre système (système hôte) sans utiliser les caractéristiques de ce système hôte.

Le choix d'une plate-forme de simulation doit permettre de mettre en oeuvre le système d'exploitation de la Carte Blanche d'une part, et différentes bornes de connexion ou guichets, qui seront pilotés soit par un opérateur, soit par un logiciel applicatif externe.

Les caractéristiques recherchées sont :

- la disponibilité du ou des matériels pour le développement
- la dotation d'un environnement de développement en C complet, facile et efficace (éditeur, compilateur et déboggueur)
- la capacité à faire communiquer des programmes entre eux (le SE d'une part avec les guichets d'autre part)
- la mise en place facile du Système d'Exploitation et de son environnement (les guichets)

Deux plate-forme conviennent à ces caractéristiques : une station UNIX avec OpenWindows et un PC sous MS-DOS avec Windows.

MS-DOS avec WINDOWS

La bonne connaissance des PC et de MS-DOS est intéressante pour développer un nouveau système d'exploitation et pour éviter le piège d'utiliser le système hôte. De plus, la présence d'une unité de gestion des Entrées/Sorties permettrait de réellement gérer les communications mais elle présente la difficulté de trouver autant de ports et de nécessiter plusieurs PC.

L'interface graphique WINDOWS donne la possibilité de faire tourner le SE et les guichets sur le même PC et offre l'intérêt du multi-fenêtrage pour les démonstrations. Cependant, il nécessitait un PC à base de 386 ou 486. Or, à l'époque où a débuté le développement du prototype, une telle machine n'était pas disponible et il n'y avait pas dans notre laboratoire de réelle connaissance sur le nouveau système d'interface graphique qu'était WINDOWS 3.1 et notamment sur les communications entre processus.

UNIX et OPENWIN

Tous les programmes tournent sur une même machine. Une seule machine est donc suffisante et elle était disponible lors du développement. La communication entre programme est facile par l'utilisation sockets. De plus, un pré-développement de la communication multi-port entre carte et guichets avait déjà été développé [Vast92].L'environnement graphique OpenWindows est mieux connu dans notre laboratoire et sa convivialité offre des intérêts pour des démonstrations. Par contre, la moins bonne connaissance d' UNIX nécessite une attention particulière pour éviter les pièges du développement d'un système d'exploitation sur un autre.

Remarque : Une autre plate-forme de simulation-démonstration intéressante
L'idée est de brancher les contacts d'une fausse carte sur un port série d'un PC. Celui-ci simule ainsi les contacts d'une véritable carte à puce. Le PC joue alors le rôle de masque et de mémoire. L'avantage de ce système est la liberté de développement du masque, la diminution des contraintes, sa facilité de déboggage et son impact démonstratif. Malheureusement, les caractéristiques recherchées par le prototype de la Carte Blanche ne peuvent pas être atteints, à savoir, le multi-port, et la disponibilité de matériel.

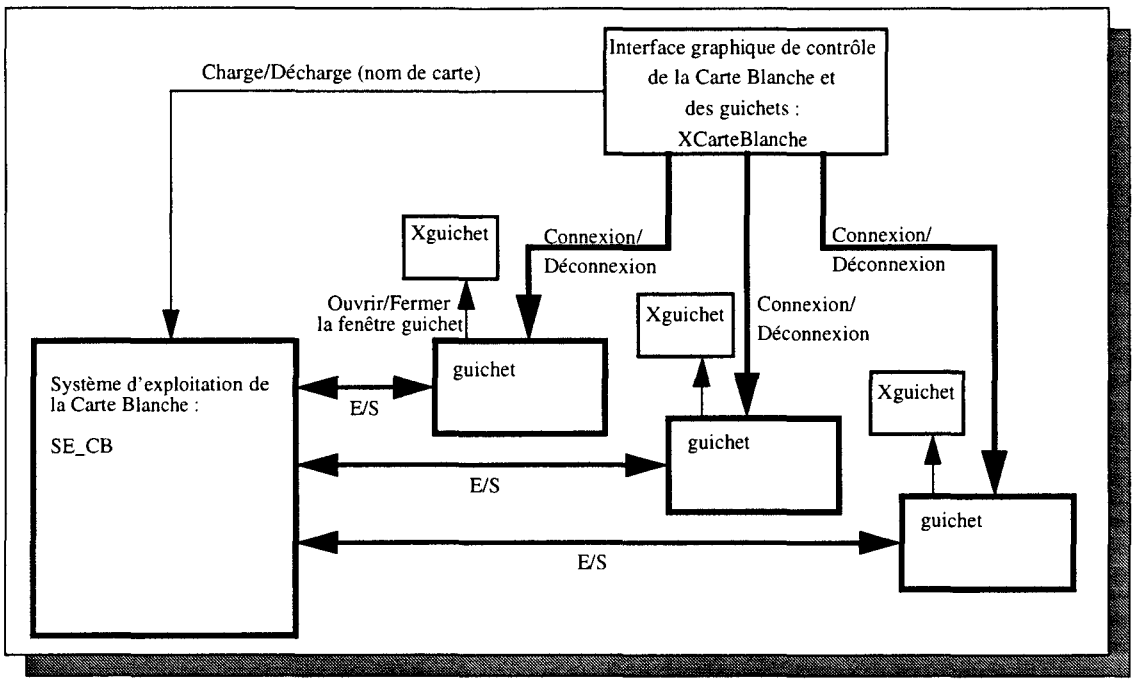
6.2.2 Structure du Prototype

Nous voulons montrer que le programme du Système d'Exploitation est le plus possible indépendant du système hôte et encore plus de l'environnement OPENWIN. Pour cela, nous allons différencier le plus possible d'une part, les programmes qui concernent la Carte Blanche proprement dite et les logiciels qui communiquent avec elle, d'autre part, les programmes qui font l'interface avec UNIX et OpenWindows.

Les programmes d'interface XWindows gèrent le fenêtrage, la création/destruction des sockets, le lancement des programmes de Système d'Exploitation et de guichet proprement dits, déclenche le chargement des paramètres carte au commencement de la session carte, et la sauvegarde de ces paramètres à la fin de cette session.

Les programmes Système d'exploitation. et guichet s'exécutent comme si ils étaient seuls sur la machine et ne connaissent de leur environnement réel que des sockets qu'ils considèrent comme des ports d' Entrées/Sorties. D'autres sockets (ou pipes) sont cependant ajoutées pour le contrôle des programmes par l'interface XWindows. Elles correspondent en fait à la perception que le système d'exploitation peut avoir des contacts alimentation, masse et horloge d'une carte qui induit ainsi la mise en route du Système d'Exploitation, le chargement et la sauvegarde des paramètres carte.

FIGURE 31 Structure du prototype



Les programmes XWindows sont :

- `XCarteBlanche`
- `XGuichet`

Les programmes principaux :

- `SE_CB`
- `guichet`

`XCarteBlanche` lance `SE_CB` et 3 fois `XGuichet` et attribue à chacun l'extrémité des sockets qui leur sont nécessaires pour communiquer. Chaque `XGuichet` lance `guichet`.

Tous ces programmes ont été réalisés par mes soins en langage C. A titre indicatif, le programme qui réalise le système d'exploitation SE_CB comporte environ 5000 lignes de programme. Ce code n'ayant subi aucune optimisation, aucune mesure de taille ou de temps n'est vraiment significative.

6.2.3 Le système d'exploitation: SE_CB

Le système d'exploitation `SE_CB` est indépendant de son environnement à l'exception de son interface avec le programme de contrôle XCarteBlanche. En fait, cette interface se réduit à fournir le nom du fichier contenant l'état de la carte insérée dans le guichet juste avant et juste après la session carte, au moyen de pipes.

Pendant la session carte, comme il se doit, le système d'exploitation ne possède comme E/S que les sockets. Les entrées/sorties standards d'UNIX sont ignorées à l'exception des affichages de

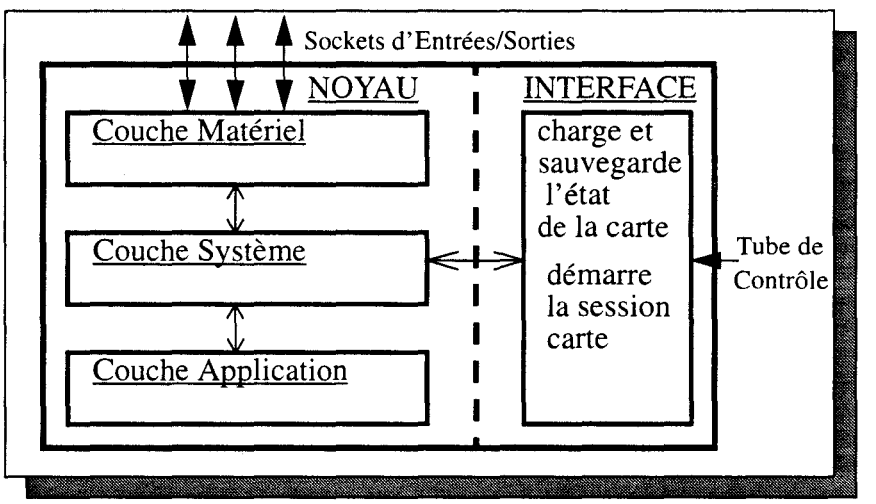
message pour les seuls besoins de traçage, déboggage, d'espionnage et de démonstration du programme. Cet affichage est produit dans une fenêtre du programme de contrôle XCarteBlanche.

La session carte est une boucle composée en deux phases :

- acquisition des entrées/sorties sur les ports que ce soit des messages de services (demande de connexion/déconnexion) ou des messages à destination du Système d'Exploitation. ou des applications.
- ordonnancement des sessions à l'état prêt.

Le système d'exploitation est construit sur le principe des couches : une couche ne peut demander un service qu'à une autre couche immédiatement contiguë. Comme précisé dans le cahier des charges, SE_CB est divisé en trois couches : la couche matériel, la couche système et la couche application

FIGURE 32 Structure du programme SE_CB du prototype



6.2.3.1 La couche matériel

Il est bien évident que la couche matériel est la couche la moins indépendante du système hôte puisqu'elle gère les E/S (les sockets UNIX) et la mémoire.

Pour la gestion de la mémoire, une zone mémoire est allouée au système d'exploitation qui la gère et la protège entièrement lui-même en deux parties distinctes, l'une représentant la RAM, l'autre l'EEPROM. Ainsi, tout accès à la mémoire passe par cette gestion de la mémoire

La gestion des entrées/sorties utilise les mécanismes de sockets pour simuler une unité de communication indépendante de type UART. Cela permet de recevoir des messages en asynchrone et d'être averti qu'il y a au moins un message d'arrivée sur un port de communication. La gestion des entrées/sorties gère trois ports de communication et jusqu'à trois canaux logiques par port. En outre, elle implémente le protocole préconisé dans le chapitre Système d'exploitation de la Carte Blanche. Tout message entrant ou sortant du système passe par cette gestion.

6.2.3.2 La couche système

La couche système est totalement indépendante du système hôte. Toutes les primitives du système décrites dans le chapitre précédent ont été implémentées à l'exception des primitives de coopérations qui ne sont pas terminées à ce jour. Dans le prototype, les tables système ont la particularité d'être constituées sous forme de liste chaînée. Seul le mode d'authentification par code secret existe pour l'instant.

L'introduction de nouvelles applications

La Carte Blanche étant un environnement de programmes et non pas un environnement de programmation, les applications pour la Carte Blanche sont développées à l'extérieur de l'objet nomade. L'installation se fera en envoyant une suite d'octets, représentant l'application, au système d'exploitation qui pourra ainsi générer les tables système relatives à cette application et d'inscrire en EEPROM son code et ses données.

Le système informatique sur lequel l'application va être développée prépare la suite d'octets sous forme de fichier binaire. Nous choisissons l'extension `.install` pour ce fichier.

Le fichier `application.install` est composé de deux parties :

- un en-tête de description de l'application
- le code machine monobloc de l'application

L'en-tête du fichier contient d'une part :

- le nom de l'application
- sa codification (facultative)
- la signature électronique de l'éditeur de l'application (fournie de façon sécurisée par des algorithmes cryptographiques)
- sa visibilité (facultative)
- la liste des services avec leur nom d'appel et leur codification
- la taille de code

et d'autre part, pour chaque service et chaque fonction :

- leur adresse relative dans le segment de code
- la liste des données persistantes qu'ils utilisent
- et la taille du segment de variables qu'il faut leur allouer pour les exécuter

Le code généré par le compilateur est :

- monobloc, pour faciliter l'allocation et la protection de l'EEPROM
- translatable, car on ne sait pas d'avance où va être implanté le code
- et XIP (eXecutable In Place) parce que le code n'est pas chargé en RAM pour être exécuter

Le compilateur de l'application doit donc collecter les informations dans les fichiers sources de l'application pour constituer la partie en-tête du fichier `application.install`.

L'exécution de services dans les sessions

L'exécution des services d'application est effectuée par un interpréteur sécurisé qui simule l'exécution de code d'un machine virtuelle. Cet interpréteur est décrit au paragraphe 6.3 : Interpréteur sécurisé .

6.2.3.3 La couche application

La couche application est indépendante du système hôte mais elle dépend comme nous le verrons par la suite beaucoup de l'implémentation des applications à l'intérieur du prototype.

L'application système SHELL ou le langage de commandes

L'application SHELL a été conçu à l'aide des utilitaires LEX et YACC d'UNIX [LM90]. Ces utilitaires permettent l'analyse syntaxique et lexicale des commandes envoyées au système d'exploitation. Elle reçoit les messages arrivant sur les sessions à l'état «Attente de commande ou de service», les analyse comme des lignes de commande puis appelle, si une commande est reconnue, la primitive correspondante de la couche système, sinon la primitive d'appel de service.

La syntaxe des principales commandes est la suivante.

Pour la Gestion des Partenaires :

- `create partner <nomPartenaire> [server|client|carrier]`
- `present partner <nomPartenaire>`
- `unlock partner <nomPartenaire>`
- `modify attributes ...`
- `delete partner <nomPartenaire>`
- `list dependant partner`

Pour la Gestion des Applications :

- `create application <nomApplication>`
- `offer function <nomService> to <nomPartenaire>`
- `delete application <nomApplication>`
- `retire offer <nomService> from <nomPartenaire>`
- `lock application <nomApplication >`
- `list application`
- `list service`
- `list offer`
- `list recipient`

Et pour l'Appel de service :

- `<nomService> <listeArguments>`

6.3 Interpréteur sécurisé

Le prototype de la Carte Blanche est réalisé sur des stations de travail UNIX sous OpenWindows. Le système d'exploitation est un processus UNIX qui ne peut donc pas recevoir dynamiquement du code et l'exécuter. De plus, le code exécuté sous UNIX ne pourrait pas être contrôlé par notre prototype. La solution simple et efficace est de charger le code dans la mémoire de données du processus du prototype et de l'interpréter.

Le prototype utilise la solution mixte de mise en oeuvre du code des applications proposée au chapitre 5. Cette solution consiste à interpréter le code en langage machine d'un processeur virtuel. Elle est particulièrement intéressante pour le prototype puisqu'elle montre les mécanismes de protection de la mémoire pour un interpréteur mais aussi elle simule ces mécanismes dans le cas d'une exécution en code natif.

6.3.1 Le choix d'un langage interprété

L'utilisation d'un interpréteur de code intermédiaire dans les cartes à micro-processeur a déjà été pratiquée à RD2P. Les masques de carte et les applications étant de plus en plus gros et complexes, l'idée est d'écrire ces programmes dans un langage évolué et de compiler ces applications dans un langage intermédiaire pour que le code généré soit plus compact que le code réel. Il faut ensuite interpréter le code intermédiaire et il est nécessaire que l'interpréteur soit de petite taille. Ainsi, à partir d'un certain seuil de complexité, la taille du code généré plus celle du code de l'interpréteur, est inférieure à la taille du code généré par un compilateur traditionnel. Cet avantage prend de l'importance dans une carte multi-applicative où chaque application profitera du même interpréteur. L'interprétation de code intermédiaire présente deux autres avantages. Le premier est la portabilité des applications. Le code intermédiaire, quel que soit le type de processeur, est identique. Le changement de processeur implique uniquement la réécriture de l'interpréteur. Le second avantage est la simulation possible du code généré par le compilateur sur une station de travail.

Un premier interpréteur CAVIMA (CArd VIrtual MAchine) [Gordons91] est conçu pour les micromodules actuels d'architecture CISC 8 bits. Le code CAVIMA est généré par un compilateur C particulier, le compilateur C_CARD, bien adapté aux contraintes des cartes à micro-processeur.

Une deuxième version de compilateur et de code intermédiaire a été réalisée : le C_CARD 2.0 qui génère du code pour une machine virtuelle appelé QUAD. Le développement des applications se faisant en langage C, les instructions de cette machine sont définies pour y retrouver la richesse des opérateurs du C. Cette caractéristique rend la génération aisée et surtout compacte de code intermédiaire par le compilateur C-Card. C-Card 2.0 est également conçu pour développer des masques et des applications sur de nouvelles architectures de cartes à micro-processeur. Pour cette raison, C_CARD se base sur un modèle de programmation et une architecture minimum pour générer du code intermédiaire qui soit facilement portable sur différentes machines réelles. En effet, ne connaissant pas le processeur utilisé (machine à pile, nombre de registres, ...), l'allocation des variables, les mécanismes d'appel de fonction, etc... ne peuvent être définis à cette étape. La machine virtuelle QUAD est d'une architecture non spécifiée complètement. C-Card 2.0 produit un programme en mnémoniques, où tous les opérandes sont des symboles. L'étape suivante

est une traduction du code mnémonique connaissant le processeur cible et son modèle de programmation.

Le fait que l'architecture ne soit pas complètement spécifiée, et que le modèle de programmation du processeur final puisse être défini par nous-mêmes, est intéressant pour le prototype. Il permet de découvrir des mécanismes qu'il serait souhaitable de voir apparaître dans le matériel sur lequel s'implantera le système d'exploitation définitif.

Le compilateur C-Card a particulièrement été développé pour permettre l'étude d'architectures RISC (Reduced Instructions Set Chip) pour les cartes à micro-processeur. La machine virtuelle QUAD est pseudo-RISC. Il n'y a que 32 instructions et toutes les instructions sont de même format. La traduction du code intermédiaire QUAD pour un processeur cible réel ou virtuel est donc relativement facile à concevoir. La traduction dans un code réel n'assure pas de conserver la compacité du code obtenu. Par contre, le code QUAD mnémonique doté d'un modèle de programmation peut aussi être traduit en code QUAD virtuel en vue d'être interprété. Dans ce cas, le traducteur produit bien-sûr un code tout aussi compact. De plus, l'interpréteur est également simple à réaliser et est de petite taille. Par conséquent, le code de l'interpréteur plus le code virtuel de l'application restent plus compacts que celui par la traduction en code réel. Cette machine a été définie dans le projet OCEAN (Outils de Conception et d'Evaluation d'Architecture Nouvelle) [Caron94].

Le choix d'une machine virtuelle s'est naturellement porté sur ce produit de notre laboratoire car il intègre déjà des contraintes des cartes à micro-processeur et présente des caractéristiques intéressantes pour notre prototype. De plus, les outils développés autour de cette machine sont réutilisables et adaptables aux spécificités de la Carte Blanche.

6.3.2 la sécurité de l'interpréteur

L'utilisation d'un interpréteur de code QUAD permet de montrer comment contrôler totalement le processus d'exécution des services d'application dans le mode interprété bien-sûr, mais aussi dans une simulation du mode direct. Dans ce dernier cas, l'exécution du système d'exploitation sous UNIX et l'interprétation des applications simulent les deux modes de fonctionnement de la Carte Blanche. Le système d'exploitation peut accéder à toute la mémoire du processus UNIX tandis que l'accès aux mémoires, virtuellement RAM et EEPROM, par les applications, sera entièrement contrôlé par le système via l'interpréteur.

Dans tous les cas, l'interpréteur simule l'Unité de Gestion de la Mémoire ou M.M.U. (Memory Management Unit). et cela est facilité par le fait que le QUAD est un langage machine de type R.I.S.C. Le format fixe des instructions permet de vérifier aisément que l'instruction en cours et la prochaine instruction se trouvent dans le segment de code de l'application. De plus, les opérandes des instructions sont de nature connue et il est facile de vérifier qu'elles appartiennent au segment correspondant.

Le paramètrage de la MMU simulée est effectué à l'appel d'un service grâce aux informations maintenues dans les tables session, application et service. Il est modifié à chaque changement de contexte d'exécution de l'interpréteur. Ces changements sont provoqués par les appels et retours de fonction, les suspensions et reprises de session.

Les appels et retours de fonctions interviennent sous la forme des instructions respectives CALL et RET du processeur QUAD. L'instruction CALL doit provoquer le déroutement du processus d'interprétation pour invoquer la primitive d'appel de fonction en précisant l'adresse relative dans le code de la fonction. La primitive retrouve donc la fonction dans la table des fonctions et peut allouer le segment de variables nécessaire, prendre en compte les paramètres de la fonction et copier les données utilisés en RAM. La pile d'appel de la session est mise à jour et la primitive se termine par la modification des paramètres de la MMU. L'interprétation poursuit alors son cours. De même, l'instruction RET doit provoquer le déroutement de l'interprétation pour appeler une primitive système qui permet de retrouver le contexte précédent et de reparamétrer la MMU.

Ce mécanisme de déroutement est réalisé facilement par un interpréteur, mais il n'est pas évident que cela soit réalisable sur un processeur réel.

Les suspensions et les reprises de session sont facilitées par l'utilisation d'un interpréteur. Il suffit de mémoriser au niveau de la session et de sa pile d'appels, le compteur ordinal de l'interpréteur et le contexte d'exécution. Ainsi, à chaque reprise, la MMU peut retrouver le paramètrage initial et l'interpréteur peut reprendre l'exécution là où elle s'était arrêtée.

Les appels système se traduisent en QUAD par une demande d'interruption logiciel, c'est à dire par l'instruction INT. Cette instruction redonne la main au système en invoquant la primitive correspondant à l'appel système demandé.

Cependant, le processeur QUAD est basé sur une architecture minimum et ne possède donc ni registre, ni pile. Préalablement à l'interruption, les paramètres des appels système doivent être écrits dans un segment de mémoire réservé et accessible à tous, appelé la page zéro. La primitive système récupère ses paramètres à des adresses de la page zéro qui lui sont spécifiquement attribuées.

Le problème se retrouve pour le passage de paramètres et la valeur de retour d'une fonction. Nous utilisons des interruptions logiciels pour indiquer au système les paramètres avant l'appel d'une fonction, et la valeur de retour avant l'instruction RET.

Le processeur QUAD ne dispose pas de l'adressage indirect pour accéder aux données de l'application, ni de la distinction de l'adressage indexé pour le segment de variable et l'adressage direct pour la page zéro et les données. Le compilateur ne peut donc pas suivre le modèle de programmation du chapitre 5. La MMU doit donc simuler un adressage virtuel pour que le code n'utilise qu'un espace continu de données. Cet adressage virtuel continu est présenté à la figure 33 de la page 137. Le compilateur pour le prototype utilise donc ce modèle de programmation.

6.4 Chaîne de compilation

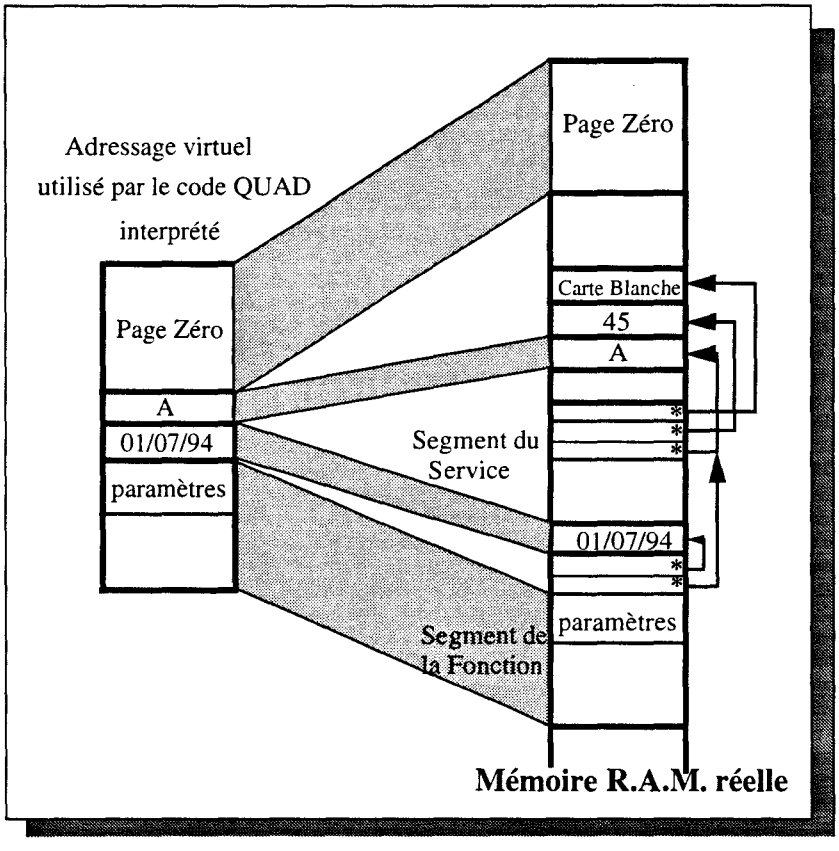
Les Cartes Blanches, toutes initialement dans le même état, doivent être chacune chargée avec les applications nécessaires à leur porteur. Il faut donc donner, à des émetteurs potentiels, les moyens de développer de telles applications pour la Carte Blanche.

Les contraintes de la plupart des objets nomades, dues à la taille de la mémoire, leurs spécificités et leur sécurité, ont souvent conduit à écrire, en assembleur ou en langage machine, leur système d'exploitation et quelques fonctions applicatives.

Cette méthode permet en effet d'obtenir la meilleure compacité et la meilleure efficacité de code, mais elle présente en fait des inconvénients :

- l'écriture est difficile ce qui augmente les temps de conception
- la portabilité des programmes est impossible puisque chaque processeur a son propre jeu d'instructions
- la complexité croissante des applications rend cette méthode à court terme obsolète

FIGURE 33 Adressage virtuel continu pour le code QUAD interprété



L'intérêt de la Carte Blanche est la multiplicité des applications existantes. Cette multiplicité ne peut être atteinte que s'il est facile de développer des applications. Nous préconisons donc le développement des applications pour Carte Blanche dans un langage évolué, plutôt qu'en assembleur. De plus, cela permet, à l'émetteur, d'écrire l'application de la Carte Blanche et le logiciel correspondant pour le système hôte, dans ce même langage.

Le langage C nous paraît bien approprié pour écrire des applications pour la Carte Blanche car il est universel. Nous allons utiliser, pour le prototype, le langage C-CARD pour les qualités de son compilateur et son adéquation avec le langage QUAD et l'interpréteur sécurisé étudiés précédemment dans ce chapitre.

6.4.1 Le compilateur et le langage C_CARD

Devant l'accroissement de l'importance et de la complexité des masques des cartes à microprocesseur et des quelques fonctions applicatives que l'on peut y ajouter, le compilateur C-CARD a été conçu par RD2P pour permettre le développement de ces programmes dans un langage évolué : le langage C-CARD. Le langage C_CARD est un langage compatible C_ANSI. Le langage C_Ansi a été choisi pour ses aptitudes à réaliser un système d'exploitation et à fournir du code compact. Mais sa puissance ne garantissant pas toujours un code conforme, C_CARD ajoute des contraintes d'écriture permettant un contrôle plus précis des variables, des expressions et des structures de contrôle.

Deux versions de ce compilateur ont été réalisées. La seconde version, C-CARD 2.0, génère un code intermédiaire correspondant à une machine QUAD. Ce code intermédiaire est constitué de quadruplets où chaque instruction et ses trois opérandes sont sous forme mnémonique. Aucun modèle de programmation n'est donc considéré, ni pour l'allocation des variables, ni pour le passage de paramètres.

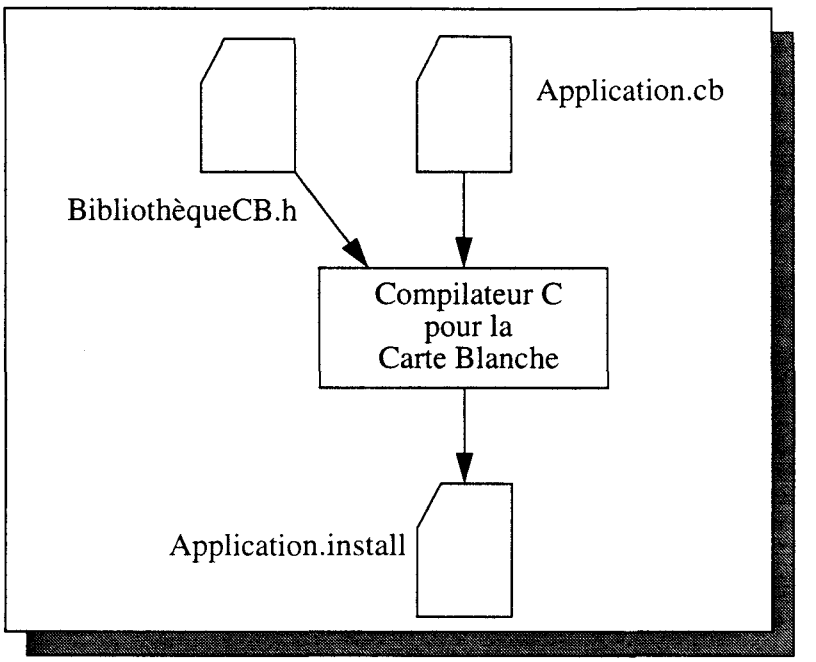
Ce code doit être ensuite traduit en un code pour une machine donnée et en considérant cette fois-ci un modèle de programmation. Le projet OCEAN, pour lequel C-CARD 2.0 a été élaboré, a développé un traducteur de code QUAD intermédiaire vers un code machine équivalent, correspondant à une machine virtuelle QUAD. Dans ce code, les instructions mnémoniques sont transformées en code-opération et les variables suivent un modèle de programmation particulier.

Pour notre projet, nous conservons tel quel le compilateur C_CARD et nous adaptons le traducteur-assembleur QUAD aux exigences de la Carte Blanche, pour qu'il génère le code de l'application et les informations qui s'y rapportent.

6.4.2 Caractéristique du compilateur d'application pour Carte Blanche

Le compilateur que nous avons conçu, génère, à partir d'un fichier de description d'une application, un fichier d'installation de cette application pour la Carte Blanche. Les noms de ces fichiers ont respectivement les extensions `.cb` et `.install` (par exemple, figure 34 de la page 139, les fichiers `application.cb` et `application.install`). Le compilateur utilise également une bibliothèque de fonctions C, qui facilite la programmation des accès aux primitives du système de la Carte Blanche, nécessaires à toute application. Cette bibliothèque se nomme `bibliothequeCB.h`.

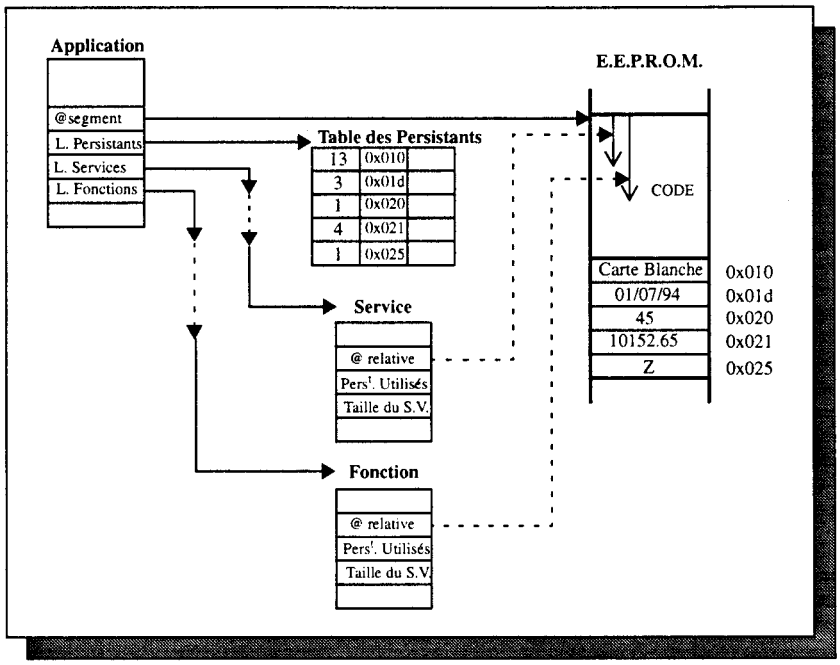
FIGURE 34 Compilateur d'application pour Carte Blanche



Le fichier `.install` permet de fournir à la Carte Blanche tous les éléments nécessaires à l'installation de l'application. Ces éléments servent à construire l'ensemble des tables système, le segment de code et le segment de données persistantes, tels qu'il sont représenter à la figure 35 de la page 140. Nous n'avons représenté dans ce schéma qu'un seul service de la table Services et qu'une seule fonction de la table Fonctions.

Le fichier .install contient donc, en plus du code, d'autres informations que ne peut pas fournir le compilateur C-CARD à partir d'un programme écrit en C. Une partie de ces informations est générée par le traducteur en langage QUAD (cf. 6.4.6 : Le Traducteur de la Carte Blanche), et l'autre doit être fournie dans le fichier .cb.

FIGURE 35 Représentation statique d'une application



La description d'une application est en effet plus complète qu'un simple programme en C. D'abord, il faut y ajouter des éléments sémantiques propres aux applications de la Carte Blanche. Ensuite, la composition particulière des applications pour la Carte Blanche a des conséquence sur le programmation en C.

Le fichier `application.cb` possède donc les caractéristiques suivantes :

- le nom de l'application doit être précisé
- il n'y a pas de fonction `main()` dans le programme
- Il faut distinguer les fonctions C qui sont des services, et dans ce cas préciser leur nom d'appel, de celles qui représentent des fonctions de l'application
- les variables déclarées en globale dans le programme n'ont pas cette signification puisqu'il n'y a pas de `main()`. Elles représentent les données persistantes de l'application, accessibles par tous les services et toutes les fonctions, et pour cette raison, elles possèdent toutes une valeur initiale
- le programme peut utiliser des fonctions de la bibliothèque `bibliothequeCB.h`

D'autres attributs d'application et de services peuvent être précisés. Nous avons choisi d'inclure ces éléments sémantiques sous forme de directives. En voici la liste :

- `#APPLICATION <nom_application>`
  donne le nom de l'application. Cette directive est obligatoire.
- `#EMETTEUR <nom_emetteur>`
  associe le nom de l'émetteur à l'application
- `#CODIFICATION <code_alphanumérique>`

précise la codification de l'application quand elle se trouve après la directive #APPLICATION. Elle est facultative.

- `#VISIBILITE OUI | NON`

  indique au système d'exploitation si cette application doit apparaître ou pas dans la liste des applications dans la Carte Blanche.

- `#SIGNATURE`

  cette directive indique par sa présence au système d'exploitation que l'application n'est valide qu'après vérification du code et des données inscrits dans la Carte Blanche et après inscription de la signature électronique de l'émetteur d'application, le tout par des procédés cryptographiques.

- `#DONNEES`

  marque le début de la déclaration des données persistantes. Plusieurs sections de déclarations peuvent survenir dans le source de l'application : l'une dans la partie déclaration des variables globales d'un programme C classique, l'autre dans la partie déclaration des variables locales d'une fonction C si la ou les données persistantes ne sont utilisées que par cette fonction. Cette dernière possibilité est une facilité de programmation, puisque toutes les données persistantes seront gérées de la même manière.

- `#DONNEES_FIN`

  marque la fin de la déclaration des données persistantes.

- `#SERVICE <nom_service>`

  précise que la fonction C suivante est un service de l'application et donne son nom d'appel.

- `#CODIFICATION <code_alphanumérique>`

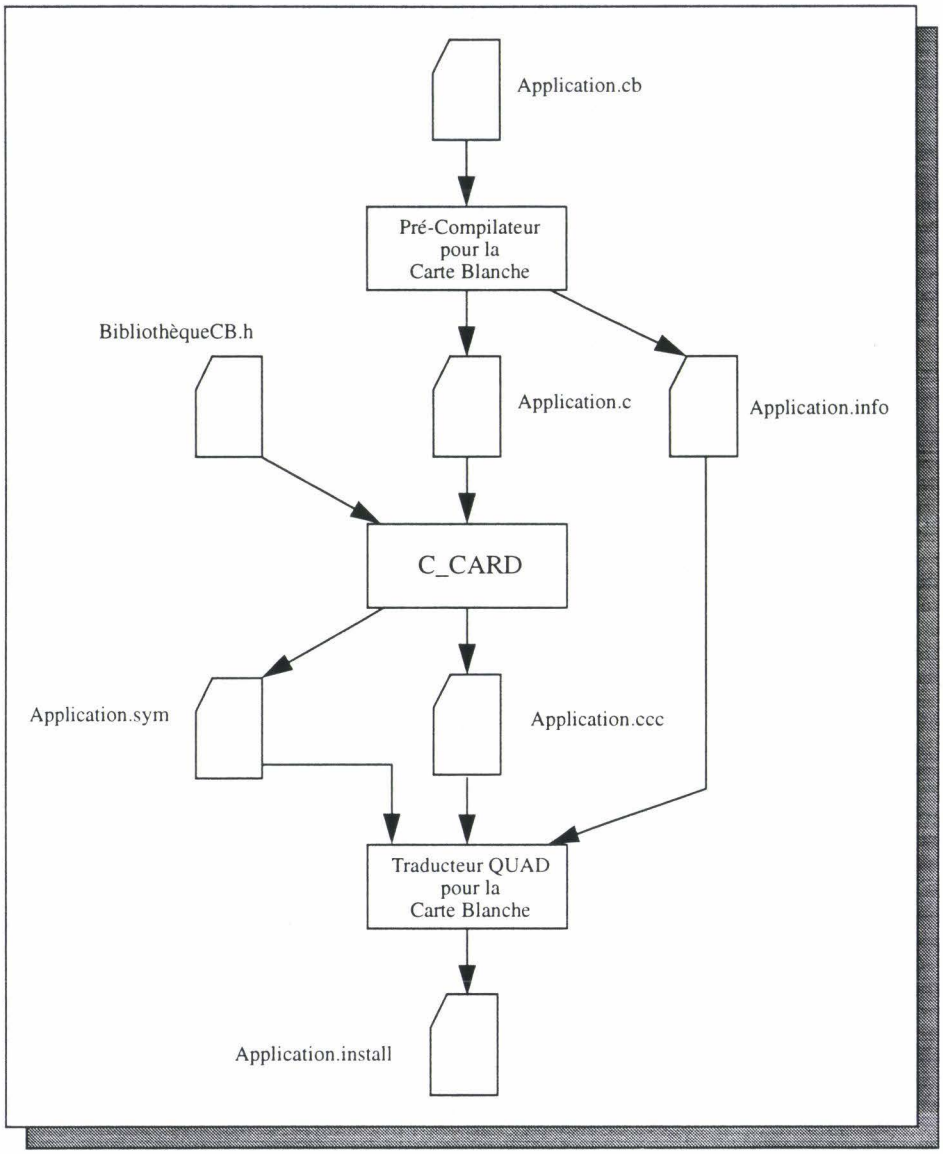
  précise la codification du service quand elle se trouve après la directive #SERVICE. Elle est facultative.

Cependant, la compilation des services et des fonctions est réalisée par le compilateur C-CARD que nous voulons utiliser tel qu'il est. Or, ce compilateur ne peut pas comprendre, ni accepter, les directives dans le fichier application.cb. Dans le processus de génération du fichier .install, il faut donc faire précéder la phase de compilation proprement dite, par une pré-compilation du fichier `.cb`. Ce pré-compilateur permet de présenter à C-CARD un fichier épuré des directives et ne contenant que le programme en C, d'une part, et de collecter l'ensemble des informations qui sont nécessaires au traducteur pour générer le fichier `.install`, d'autre part. La compilation d'une application pour la Carte Blanche est donc une chaîne de compilation que nous présentons au paragraphe suivant.

6.4.3 La Chaîne de compilation

Pour générer le fichier `application.install`, nous partons du source Carte Blanche de l'application `application.cb`. Le schéma ci-dessous présente la chaîne de compilation dans son ensemble.

FIGURE 36 La chaîne de compilation pour la carte Blanche



`Application.cb` est donc le fichier source de l'application que l'on veut installer dans la Carte Blanche. Il contient donc :

- les attributs de l'application, comme son nom, sa codification et sa visibilité
- la valeur initiale des données de l'application qui seront partagées entre les services et les fonctions;
- pour chaque service, ses attributs, c'est à dire, son nom d'appel et sa codification, suivis de son source en C
- le source C de toutes les fonctions de l'application.

Le pré-compilateur de la Carte Blanche a pour objet de collecter les informations nécessaires à la constitution des tables système d'application, d'extraire les valeurs initiales des données de l'application et de placer le tout dans un fichier de travail `application.info`, et enfin d'obtenir un fichier source C épuré de l'application compilable par C_CARD : le fichier `application.c`.

`BibliothèqueCB.h` est le fichier de fonctions C utilisables par le programmeur de l'application pour avoir accès aux différentes primitives du système mises à sa disposition.

`Application.ccc` est le fichier de code QUAD symbolique que C_CARD génère en même temps qu'il construit la table des symboles du programme dans le fichier `application.sym`.

Enfin, le traducteur de la Carte Blanche construit le fichier d'installation `application.install` à partir des informations fournies par le pré-compilateur et du code symbolique généré par C_CARD.

6.4.4 Le pré-compilateur

Le pré-compilateur a pour but de fournir un fichier compilable par C-CARD 2.0, à partir du fichier de description de l'application. Il faut donc supprimer les directives et collecter les informations importantes pour la Carte Blanche et qu'il faut retrouver dans le fichier .install final.

Le pré-compilateur va donc lire le fichier source ligne par ligne et détecter celles qui possèdent une directive. Le pré-compilateur effectue en même temps un contrôle lexical des directives dans le programme source. Il signale précisément, au programmeur, toute erreur comme par exemple l'oubli d'une directive obligatoire, la non spécification d'un nom ou d'un argument, des définitions multiples, ou une mauvaise définition des données.

A la lecture d'une directive `#application`, `#service`, ou `#donnees`, le traitement du groupe correspondant est effectué. Pour la première directive, il s'agit de récupérer le nom et les attributs de l'application. Le traitement de la directive `#service` consiste à extraire le nom d'appel et les attributs du service, mais aussi le nom de la fonction C qui s'y rapporte pour que le traducteur puisse associer le code généré à ce service. Pour les données de l'application, seuls leur nom sont retenus. Le codage numérique de leur valeur initiale est effectué, par le traducteur, en même temps que celui des constantes contenues dans le corps des fonctions C. Toutes ces informations recueillies sont inscrites dans un fichier `application.info` et ce, dans un format bien défini afin de faciliter leur récupération plus tard par le traducteur.

Du fait du contrôle important effectué par le compilateur C-CARD, celui-ci ne permet pas l'affectation des variables au niveau des déclarations. De plus, il vérifie que chaque fonction C définie, est au moins utilisée à partir de la fonction `main()` qui doit donc également exister. Le pré-compilateur ajoute à la fin du fichier épuré des directives, une fonction main() fictive qui affecte les valeurs initiales à chaque donnée persistante et utilise une fois chaque service de l'application. Les fonctions de l'application ne figure pas dans la fonction `main()` car elles sont en principe utilisées par les services de l'application.

Ce traducteur a fait l'objet d'un stage de D.U.T. et est présenté dans [Bro94].

6.4.5 La bibliothèque d'appels système de la Carte Blanche

Les applications ont besoin d'accéder aux ressources partagés du système, en particulier aux entrées/sorties, et à ses caractéristiques fonctionnelles comme la coopération. Elles ont donc accès à des primitives système qui sont mises en oeuvres par des interruptions logicielles et qui permettent d'assurer la sécurité du système et des autres applications vis à vis de l'application qui les uti-

lisent. Les paramètres sont passés à la primitive par un segment de mémoire réservé et accessible à tous : la page zéro. Pour faciliter la programmation des appels système par les émetteurs d'application, nous avons constitué une bibliothèque de fonction d'appel système. Cette bibliothèque se nomme bibliothequeCB.h. Pour le prototype, cette bibliothèque comprend, pour le moment, les appels système suivants.

Pour les primitives d'Entrées/Sorties :

- `ecrire(char *Message)`
- `sysRecevoir(char *Message, int *taille)`

Pour la primitive de récupération des arguments :

- `char *sysArgument(int NiemeArgument)`

Pour les primitives de Service Offert :

- `sysAppelOffre(char *nomService)`
- `unsigned char offreDisponible(char *nomService)`

6.4.6 Le Traducteur de la Carte Blanche

La traduction est la phase la plus importante de la chaîne de compilation. Le traducteur doit générer le fichier d'installation `application.install` à partir du code symbolique QUAD, de la table des symboles et du fichier d'informations. Ce fichier est constitué en deux parties : l'en-tête et le code monobloc de l'application.

L'en-tête contient, dans l'ordre :

- le nom de l'application, la taille de mémoire nécessaire pour son code et ses données
- les attributs de l'application
- les valeurs de données persistantes
- la liste des services avec, pour chaque service, son nom, ses attributs, son adresse relative dans le segment de code, les données qu'il utilise et la taille de son segment de variables
- la liste des fonctions avec, pour chaque fonction, son adresse relative dans le segment de code, les données qu'elle utilise et la taille de son segment de variables

Pour le nom et les attributs de l'application, le traducteur les trouve directement dans le fichier d'informations. Pour les autres, il faut d'abord obtenir des renseignements à partir du code symbolique généré par C-CARD.

Le code symbolique est constitué d'instructions et de pseudo-instructions QUAD. Les instructions ont un code-opération mnémonique et des opérandes symboliques. Il n'y a pas d'allocation des variables, mais on connaît la classe (GLOBAL, LOCALE, PARAMETRE, TRAVAIL) et la taille en octets de tous les identificateurs. Ces informations sont constituées par C-CARD sous forme de table des symboles. Des pseudo-instructions QUAD sont ajoutées pour faciliter le portage du code QUAD sur une machine réelle. Cette composition du code symbolique nous permet

d'obtenir les renseignements qu'il nous faut, puis de traduire l'application. Pour cela, la traduction procède à deux lectures du fichier de codes symboliques.

La première lecture permet :

- le repérage des données persistantes utilisées par chaque service et chaque fonction grâce à leur nom symbolique collecté par le pré-compilateur
- le repérage des paramètres pour les fonctions
- le repérage des variables locales pour chaque service et chaque fonction

Grâce à ces renseignements, le traducteur peut organiser le segment de variables de chaque service et de chaque fonction. Cette organisation suit le modèle de programmation de la Carte Blanche qui est donné au paragraphe "la sécurité de l'interpréteur" à la page 135. Elle permet d'affecter à chaque identificateur du code symbolique, une adresse relative au segment. De plus, elle donne la taille du segment, qui est nécessaire pour constituer l'en-tête du fichier `.install`.

A la deuxième lecture, le traducteur dispose de tous les éléments pour construire le code final de l'application. Chaque instruction symbolique lue est remplacer par une instruction QUAD de format fixe, dans laquelle le code-opération mnémonique devient numérique, les adresses relatives au segment de variables sont substituées aux variables symboliques, et les valeurs immédiates sont numérisées. A chaque fois qu'il rencontre une pseudo-instruction indiquant une nouvelle fonction C, le traducteur mémorise l'adresse relative de cette fonction C dans le code. Le nom symbolique de la fonction permet de retrouver dans le fichier d'information s'il s'agit d'un service ou d'une fonction de l'application et de lui associer l'adresse relative courante. Lorsqu'arrive la fonction main() fictive, le traducteur numérise les valeurs initiales des données de l'application mais n'inclut pas le code de cette fonction. Il connaît à ce point la taille du code, qui est nécessaire à l'installation de l'application.

Enfin, le traducteur peut construire l'en-tête du fichier `application.install` en utilisant les informations collectées lors de la pré-compilation et les renseignements obtenues de la traduction. Ensuite il y ajoute le code de l'application.

Cependant, deux points n'ont pas encore été abordés : les appels système et le passage de paramètres à une fonction.

6.4.6.1 Les appels système

Les appels système sont obtenus en utilisant une fonction de la bibliothèque de fonctions C `bibliothequeCB.h`, dans le corps des programmes des services et des fonctions de l'application. Ces fonctions sont directement compilées par C-CARD en une suite d'instructions terminée par l'instruction mnémonique `INT` *<numéro_d'interruption_logicielle>*. Les premières instructions sont des instructions d'écriture des paramètres en page zéro. La structure de la page zéro étant définie dans la bibliothèque, les instructions mnémoniques apparaisse avec les noms symboliques de la page zéro. Le traducteur reconnaît ces noms (ils commencent par `page0_`), et ne les comptabilise pas dans les variables locales de la fonction en cours de traduction.

6.4.6.2 Le passage de paramètres à une fonction et la valeur de retour d'une fonction

L'appel d'une fonction est compilée en QUAD mnémonique par l'instruction CALL, précédée de pseudo-instructions QUAD précisant les paramètres à passer. L'interpréteur de QUAD (cf. paragraphe 6.3.2 page 135) préconise l'utilisation d'une interruption logicielle pour indiquer au système les paramètres à passer à une fonction.

Le traducteur transforme donc chacune de ces pseudo-instructions en un appel à la primitive «d'enregistrement d'un paramètre». Cet appel est alors traduit par une instruction d'écriture dans la page zéro, de la nature du paramètre (valeur ou variable) puis du type de passage (par valeur ou par variable), suivies de l'instruction INT correspondant à la primitive. Lors de l'exécution de l'instruction CALL par l'interpréteur, le système retrouve ces enregistrements pour transmettre ces paramètres à la fonction appelée.

De même, avant l'instruction RET de retour de fonction, le traducteur insère des écritures dans la page zéro et un appel à la primitive d'enregistrement de valeur de retour, pour que le système transmette au programme appelant la valeur de retour.

6.5 Conclusion

Le prototype que nous avons développé, permet de mettre en évidence les caractéristiques originales du modèle de fonctionnement de la Carte Blanche. L'écriture de ce prototype a également montré la faisabilité logicielle du système d'exploitation. Les mécanismes que nous avons mis en oeuvre dans le prototype se sont avérés efficaces.

L'interpréteur sécurisé de code QUAD montre l'intérêt de la solution mixte. il donne des éléments sur la nature du matériel nécessaire au système d'exploitation de la Carte Blanche pour obtenir le fonctionnement et la sécurité recherchée. Ces éléments sont les caractéristiques du micro-processeur (mode protégé et mécanisme d'interruption) et de l'Unité de Gestion de la Mémoire ou MMU.

Cependant, la procédure de changement de contexte à l'appel d'une fonction ne peut être implémentée sur les micro-processeurs actuels. La définition d'un processeur répondant aux critère de l'exécution de code en mode direct, n'est peut être pas intéressant, comparé à un interpréteur de code d'une machine virtuelle.

La plate-forme sur laquelle a été bâti le prototype, ne permet malheureusement pas d'estimer la taille du système d'exploitation. Si un tel système d'exploitation apparaît difficilement portable sur les cartes à micro-processeur actuelles en raison de sa complexité, il semble concevable d'envisager la réalisation d'un prototype minimum sur les futures générations de carte. En effet, il sera possible de disposer prochainement de cartes comportant un micro-processeur RISC 32 bits [Pey94] capable de supporter de tels systèmes d'exploitation. D'ailleurs, une telle carte est actuellement à l'étude dans le projet européen CASCADE (EP8670).

Enfin, la chaîne de développement d'application pour la Carte Blanche montre qu'il est facile pour un émetteur de définir une nouvelle application. Cette chaîne est longue car nous avons voulu utilisé des outils existants, mais elle montre que la réalisation d'un compilateur n'est pas difficile et que l'on pourrait développer des compilateurs d'autres langages de haut niveau.

Conclusion

Dans ce mémoire, nous avons effectué l'étude critique des objets nomades, et plus particulièrement des cartes à micro-processeur. A partir de cette étude, nous avons établi un nouveau modèle de fonctionnement qui nous semble idéal pour les objets nomades : le modèle de la Carte Blanche. Le porteur principal choisit lui-même les applications qu'il veut trouver sur sa Carte Blanche. Ces applications sont indépendantes et peuvent provenir d'émetteurs différents. Des applications peuvent être ajoutées ou retirées tout au long du cycle de vie de l'objet nomade. Les autres originalités sont que les applications peuvent effectuer en interne leurs propres traitements et que des applications distinctes peuvent coopérer. Toutes ces caractéristiques font de la Carte Blanche, un objet nomade résoluement ouvert.

Pour obtenir ce fonctionnement avec toute la sécurité nécessaire aux objets nomades et à notre modèle en particulier, nous avons défini les spécifications d'un système d'exploitation. Dans ce travail, nous avons volontairement fait abstraction des contraintes matérielles car le modèle de la Carte Blanche doit pouvoir s'appliquer sur plusieurs types de matériels. De ce fait, le système d'exploitation établit un standard pour objets nomades qu'il faut alors adapter à l'équipement final.

La sécurité de la Carte Blanche se situe à deux niveaux. D'abord, le premier niveau est le contrôle d'accès à l'objet nomade. Bien que le nombre d'applications à protéger varie et, en conséquence, le nombre d'intervenants qui vont les utiliser, le système d'exploitation assure une sécurité d'accès au moins équivalente à celle des cartes à micro-processeur. Le deuxième niveau, inexistant dans les cartes à micro-processeur, concerne l'exécution et la coopération d'applications. L'exécution du code d'une application doit préserver l'intégrité du système d'exploitation et des autres applications. Nous avons proposé trois méthodes différentes pour protéger l'ensemble du système. L'indépendance des applications pendant la coopération doit également être garantie. Nous donnons une solution qui tire partie de la nature des services des applications.

Le S-SHELL permet aux émetteurs d'exprimer facilement des besoins évolués de sécurité pour leur application, tels que les droits de coopération avec ou sans délégation. Il serait intéressant de l'adapter à la Carte Blanche.

Bien que très ouverte, la Carte Blanche nous semble avoir un niveau de sécurité suffisant, mais le rapport ouverture-sécurité pourrait ne pas convenir à certains émetteurs d'application. Des versions moins ambitieuses pourraient alors être développées.

Cependant, certaines caractéristiques peuvent encore être améliorées. D'abord, il faudrait déterminer la meilleure façon de créer le partenaire OFFICIEL. Ensuite, le fait que ce partenaire est identique sur toutes les Cartes Blanches est un point faible dans la sécurité. Il faudrait que ce partenaire possède un mode d'authentification sûr. Nous étudierons prochainement différents modes d'authentification, adaptés à divers partenaires, qui pourraient être basés sur des protocoles à apport nul de connaissance.

La réalisation du prototype nous a permis de valider le modèle de fonctionnement de la Carte Blanche et de contrôler les spécifications et les mécanismes du système d'exploitation. Cependant, la nature de la plate-forme de simulation ne permet une évaluation ni de la taille du système d'exploitation et des applications, ni de leurs performances. Il serait intéressant de porter le système d'exploitation sur une machine réelle. Il nous faudrait d'abord choisir un ou plusieurs matériels sur lesquels on adapterait le système d'exploitation. Le projet OCEAN pourrait ensuite nous aider à optimiser et à évaluer les systèmes d'exploitation pour chaque équipement.

Une autre perspective importante de la Carte Blanche est d'introduire la notion de Carte Active. L'objet nomade ne va plus être seulement réactif mais pourra prendre des initiatives. La Carte Blanche pourra intégrer des applications fonctionnant sur le principe de l'abonnement ou du paiement à l'utilisation. De plus, elle pourra interroger elle-même des banques de données pour obtenir un renseignement complémentaire. Elle autorisera l'automatisation des tâches journalières du porteur. Le protocole de communication proposé dans ce mémoire prend déjà en compte certaines de ces fonctions.

D'autres projets en cours aujourd'hui, notamment le projet européen CASCADE, ont des objectifs communs à ceux de la Carte Blanche quoique leur démarche soit différente. Les analyses des problèmes et les solutions proposées dans ce mémoire pourront peut-être apporter une contribution à l'avancement de leurs travaux.

Annexe A
Caractéristiques physiques des Cartes à Micro-Processeur

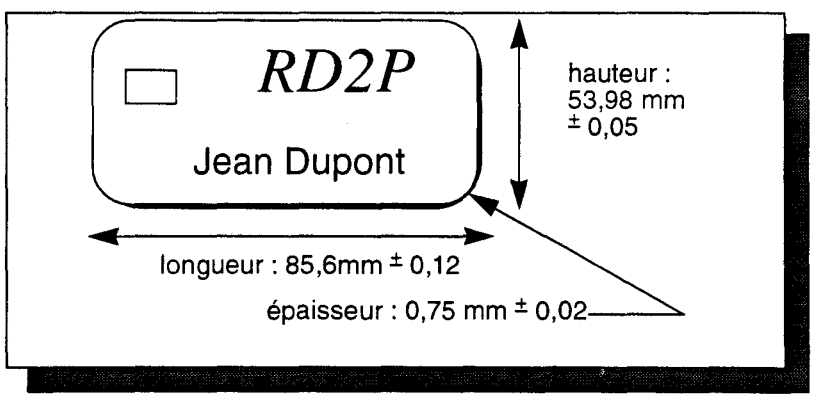
Cette annexe regroupe les caractéristiques physiques de contact et de communication des cartes à micro-processeur.

A.1 Caractéristiques physiques

Les caractéristiques physiques de la carte à micro-processeur sont spécifiées dans la norme ISO 7816-1. Nous présentons cette norme sous trois aspects : le support, le profil de la surface de contact, et les contraintes matérielles [ISO1].

Tout d'abord, le support des cartes à puce reprend les normes des cartes à bande magnétique pour la dimension et l'épaisseur du plastique. D'ailleurs, des emplacements sont réservés sur le support pour une bande magnétique et pour l'embossage.

FIGURE 37 Le support du composantt



Ensuite, la carte à puce possède une surface de contact pour l'utilisation du micromodule et la norme indique le profil de cette surface de contact par rapport au support de la carte.

Enfin, du fait de l'existence d'un composant électronique dans la carte, et de sa cohabitation avec une bande magnétique, la norme précise les contraintes relatives aux points suivants :

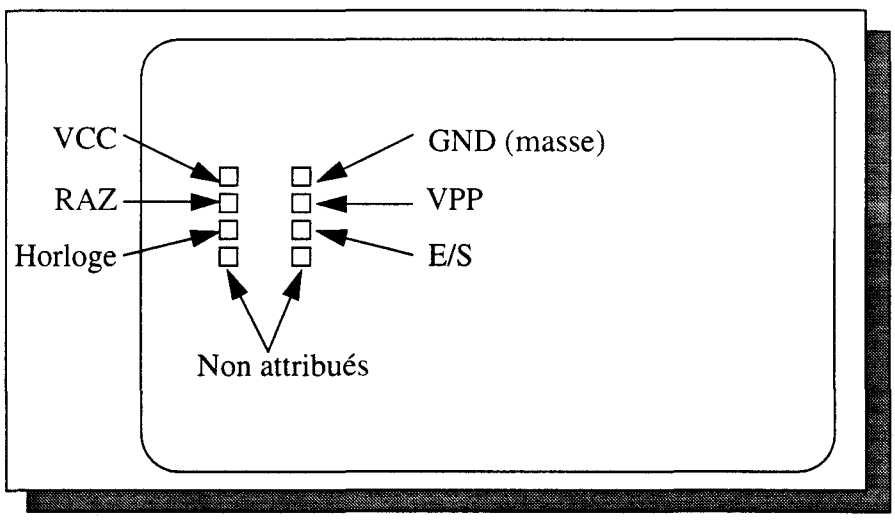
- protection aux ultraviolets et sensibilité aux rayons X : ces sources électromagnétiques peuvent altérer le contenu des mémoires de la carte
- résistance mécanique de la carte et des contacts : ce sont les contraintes de torsion dont nous reparlerons ci-après auxquelles il faut ajouter celles concernant l'insertion de la carte dans un lecteur et l'établissement des contacts
- interférence électrique entre bande magnétique et circuit : l'utilisation du composant électronique peut induire un champ magnétique nuisible
- influence d'un champ magnétique dû au lecteur de carte
- niveau d'électricité statique supporté : il peut causer un dérèglement ou une destruction du circuit intégré
- niveau de dissipation thermique accepté : dû au fonctionnement du composant

A.2 La pastille de contact

La pastille de contact est spécifiée dans la norme ISO 7816-2, en termes de nombre de contacts, leur position sur le support, leur dimension et leur attribution [ISO2].

L'emplacement de la pastille de contact sur le support plastique tient compte de deux critères : ne pas empiéter sur la piste magnétique ni sur la zone d'embossage et respecter des contraintes de torsion. En effet, la carte doit tenir dans une poche où elle peut subir des torsions que le composant électronique doit supporter. C'est pourquoi l'emplacement se trouve proche d'un coin de la carte, moins flexible, et la taille du composant ne doit pas dépasser les 25 mm^2. La France, précurseur dans le domaine des cartes, avait choisi un emplacement moins central bien avant cette norme.

FIGURE 38 Attribution des contacts de la carte



Les contacts sont au nombre de huit dont seulement six sont actuellement utilisés (voir figure). Ce sont :

- la tension d'alimentation VCC : c'est la tension continue qu'il faut appliquer aux composants du micromodule pour leur bon fonctionnement

- la masse (tension de référence)
- la remise à zéro (reset) : ce contact permet de (ré)initialiser le micro-processeur correctement
- le signal d'horloge : car il n'y a pas d'horloge dans la carte pour des raisons de taille du composant
- le contact d'entrée/sortie : tous les ordres et toutes les données passent par ce contact
- la tension de programmation VPP : Elle sert à écrire les mémoires EPROM qui nécessitent une tension d'écriture supérieure à VCC suffisante pour la lecture. Du fait de la diminution de cette tension de programmation et de la consommation électrique correspondante pour les nouveaux composants de type EEPROM, les micromodules actuels produisent cette tension à partir de VCC et le contact VPP n'est alors plus nécessaire.

A.3 Signaux électroniques et protocole d'échange

La norme ISO 7816-3 fixe les tensions d'alimentation et de programmation, les procédures de fonctionnement et les protocoles de transmission [ISO3].

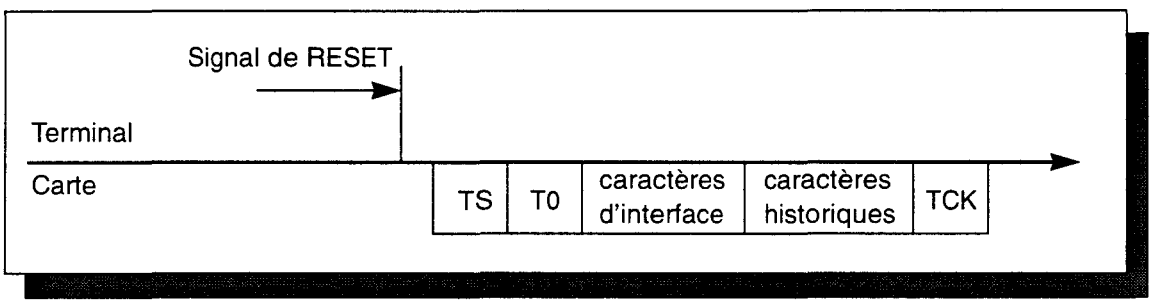
La tension d'alimentation est de 5V conformément à la technologie TTL, mais on trouve de plus en plus de composants fonctionnant sous 3V pour des raisons d'économie d'énergie. La tension de programmation dépend des composants mémoires utilisés et peut aller de 5V à 21V.

Les procédures de fonctionnement concernent l'ordre d'application des différents signaux au démarrage de la carte, l'ordre de désactivation de ces signaux en fin d'utilisation et la «réponse à remise à zéro». La «réponse à remise à zéro» est la suite d'octets de format bien déterminé envoyée par la carte lors de sa mise sous tension ou d'un redémarrage à chaud (remise à zéro). Elle commence toujours par le caractère TS suivi dans l'ordre du caractère de format T0, des caractères d'interface et des caractères historiques. Elle se termine facultativement d'un caractère de contrôle et le tout fait au plus 32 caractères. La signification de ces caractères est la suivante :

- TS : le caractère de synchronisation
  Ce caractère doit permettre au dispositif interfacé à la carte de détecter le début de la trame de la réponse au reset. TS précise également la convention de codage, inverse ou directe. La convention inverse signifie que le UN logique est un niveau bas et que le bit le plus significatif des octets est envoyé en tête. La convention directe signifie que le UN logique est un niveau haut et que le bit le moins significatif est envoyé en tête. La transmission de TS est indépendante du protocole et la convention choisis
- T0 : le caractère de format
  Ce caractère précise l'organisation de la suite de la réponse. Il indique la présence de tout ou partie des caractères d'interface ainsi que le nombre éventuellement nul de caractères historiques
- les caractères d'interface
  Ces caractères indiquent le ou les types de protocoles utilisés ainsi que des paramètres spécifiques à chaque protocole. Les protocoles se nomment T=0 et T=1 qui sont normalisés et dont nous reparlerons plus loin, et T=2 à T=15 pour des applications futures. Le nombre et la nature de ces octets dépendent de T0.

- les caractères historiques
  Au nombre de 15 maximum, ils donnent des informations générales issues de la mémoire de la carte telles que, par exemple, le fabricant de la carte, le type de puce insérée dans la carte, l'émetteur de carte, et l'état de la vie de la carte. Leur nombre est indiqué dans T0. Leur contenu est déterminé par chaque fabricant
- TCK : le caractère de contrôle
  Ce caractère doit être émis si et seulement si l'un des protocoles spécifiés dans les caractères d'interface est différent du T=0

FIGURE 39 Réponse à la remise à zéro



Les protocoles normalisés actuellement utilisés sont le T=0 et le T=1. Le protocole ouvert T=14 est également exploité par IBM et les Japonais, mais nous ne le traiterons pas ici.

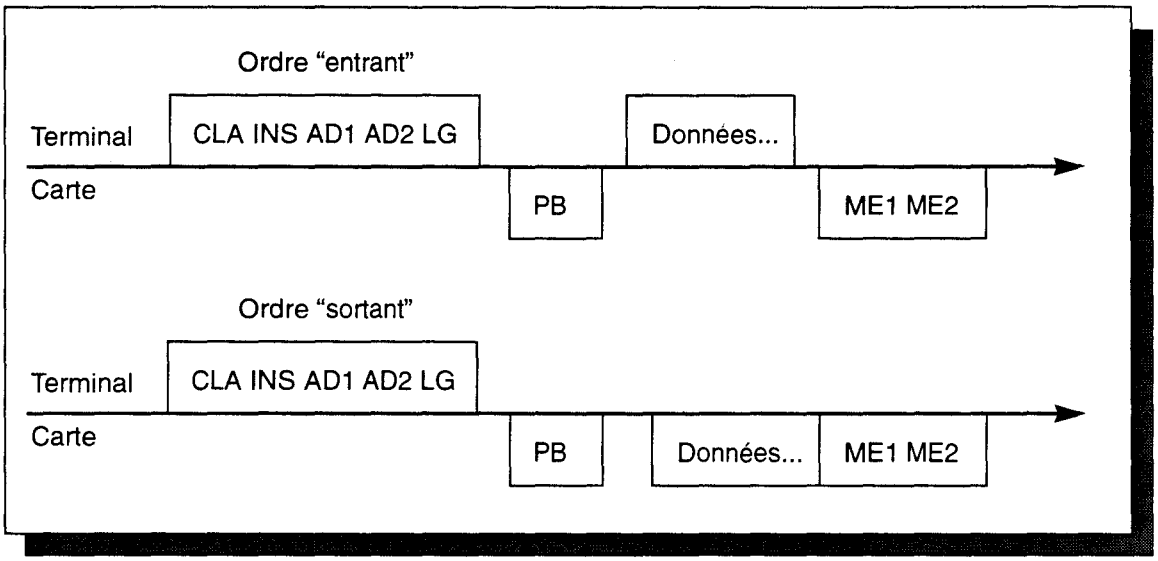
A.3.1 Le protocole T=0

Le protocole T=0 est une transmission de caractères asynchrone en mode semi-duplex. Dans ce protocole, la carte est esclave du système hôte : l'échange de données commence toujours par l'envoi d'un ordre du système hôte vers la carte. L'ordre est soit entrant quand les données sont fournies à la carte pendant l'exécution, soit sortant quand la carte délivre les données pendant l'exécution.

L'ordre est formé d'une suite de 5 octets : l'octet CLA indique la classe de l'instruction (mais est peu utilisé aujourd'hui), l'instruction à exécuter est donnée par l'octet INS, suivi de deux octets AD1 et AD2 qui servent de paramètres ou d'adresse. Le dernier octet LG spécifie le nombre d'octets de données qui doivent être échangées. A la réception de l'ordre, la carte renvoie un octet PB (Procedure Byte) comme accusé de réception de l'ordre. Il permet d'une part, que le système hôte vérifie la compréhension de l'ordre demandé à la carte, et d'autre part, que la carte indique au système hôte si les données doivent être envoyées globalement ou octet par octet séparées par un nouveau PB, et lui demande d'appliquer ou non la tension de programmation VPP. Le nombre d'octets équivalent à LG transite par la ligne série. Enfin, une fois l'instruction exécutée, la carte renvoie deux octets ME1 et ME2 (SW1 et SW2 en anglais) comme mots d'état qui renseigneront le système hôte du bon déroulement ou non de l'instruction aussi bien au niveau du protocole qu'au niveau de l'exécution de l'ordre lui-même. Ce protocole possède un paramètre dans la réponse à la remise à zéro, qui est le temps d'attente entre deux émissions de caractères, ou temps de travail ou temps de garde. La carte est considérée muette si un caractère attendu n'arrive pas au-delà du temps de garde pendant une instruction. Si le temps d'exécution de l'instruction par la

carte dépasse le temps de garde, la carte doit envoyer un PB particulier au système hôte pour prolonger le délai

FIGURE 40 Ordres «entrant» et «sortant» du protocole T=0



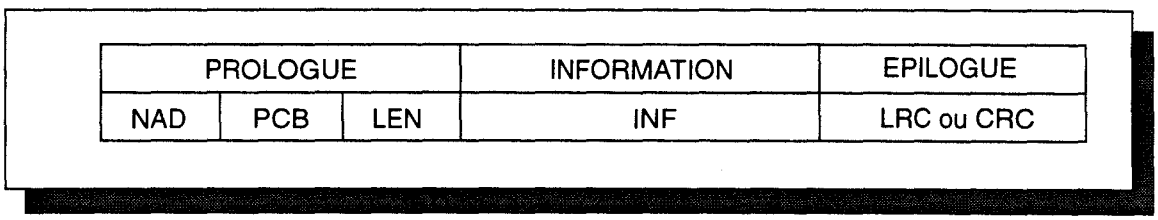
A.3.2 Le protocole T=1

Le protocole T=1 est une transmission par blocs en mode semi duplex. Dans ce protocole, le plus petit élément qui transite sur la ligne est un bloc. Ce protocole permet un flux de données dans les deux sens, le chaînage de blocs et la correction d'erreurs. Le dispositif d'interface envoie toujours le premier bloc à la carte, puis lui et la carte se transmettent tour à tour un bloc.

Les blocs sont formés en trame de deux ou trois champs :

- le champ prologue
- le champ d'information facultatif
- le champ épilogue.

FIGURE 41 Trame de message du T=1



Le prologue est constitué des trois octets NAD, PCB et LEN. NAD (node address) identifie les adresses de la source et de la destination de la trame. Il permet donc d'utiliser de multiples canaux logiques, et sert également au contrôle de la tension de programmation VPP. PCB (protocole control byte) permet de distinguer trois types de blocs :

- les blocs d'informations (I-bloc)

- les blocs «prêt à recevoir» (R-bloc)
- les blocs superviseurs (S-bloc)

Les blocs d'informations servent à la transmission de commandes applicatives ou de données entre la carte et le dispositif d'interface. Le protocole n'interdit pas à la carte d'envoyer une commande au dispositif d'interface au moyen d'un I-bloc. Les blocs prêt à recevoir sont utilisés comme accusé de réception de chaque bloc chaîné avec un bloc suivant ou pour indiquer une erreur dans le protocole. Dans le cas d'un bloc non chaîné, un bloc d'informations sert également d'accusé de réception. Les blocs superviseurs sont utilisés pour établir des paramètres de contrôle, pour effectuer une synchronisation ou pour rétablir l'état produit après une erreur.

Le champ LEN indique le nombre d'octets contenus dans le champ d'information, s'il existe. Ses valeurs vont de 0 à 254 caractères, la valeur 255 étant réservée pour une utilisation future.

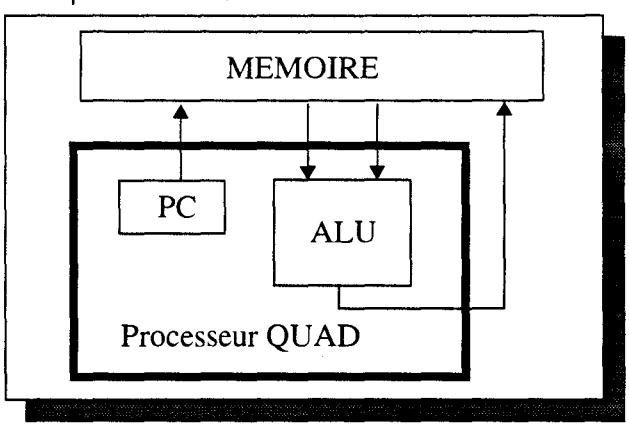
Le champ épilogue a pour rôle la détection d'erreur sur le bloc. Il s'agit d'un LRC (longitudinal redundancy check) ou d'un CRC (cyclic redundancy check).

Annexe B
La machine virtuelle QUAD

Cette annexe présente la machine virtuelle QUAD et son jeu d'instructions qui sont utilisés par le compilateur C_Card qui génère un langage intermédiaire correspondant à cette machine et à l'interpréteur sécurisé du système d'exploitation réalisé (cf. chapitre 6). Je remercie Olivier Caron qui m'autorise à reproduire cet extrait de sa thèse [Caron94].

Le processeur QUAD est une machine virtuelle dont l'architecture n'est pas spécifiée complètement. Ainsi, il n'est établi aucune hypothèse relative à l'existence d'une pile, de registres ou système de fenêtres de registres... Naturellement, le compilateur C_Card doit utiliser un modèle de programmation afin de pouvoir générer du code QUAD. La connaissance de ce modèle est indispensable pour porter le code QUAD sur un processeur cible. Le compilateur C_Card, ne disposant d'aucune information sur l'architecture réelle du processeur cible, considère que l'architecture minimum est conforme à la figure suivante :

FIGURE 42 Architecture du processeur QUAD



Le registre unique PC (Program Counter) contient l'adresse de la prochaine instruction à exécuter. L'unité arithmétique et logique (ALU) accède directement à la mémoire pour acquérir les opérandes

de l'instruction et stocker le résultat. La mémoire est partitionnée en différentes classes afin de faciliter le portage sur un processeur cible. Les classes disponibles sont :

- La classe **CODE** : stockage des instructions accessibles par le registre PC.
- La classe **GLOBAL** : stockage de toutes les variables définies globalement dans le source C_Card.
- La classe **PARAMETER** : stockage de toutes les variables paramètres des fonctions du programme C_Card.
- La classe **LOCAL** : stockage de toutes les variables locales des fonctions du programme C_Card.
- La classe **WORK** : stockage de toutes les variables intermédiaires générées automatiquement par le compilateur pour l'évaluation des expressions.
- La classe **CONST** : stockage de toutes les constantes du programme C_Card.

Une variable dans le langage QUAD est entièrement définie par sa *classe mémoire* et par sa *taille*. Ceci est rendu possible par le fait que le compilateur C_Card ramène l'utilisation des variables à un niveau très simple dans le langage QUAD : celui-ci n'a plus qu'à gérer trois types distincts de variables : le type **char**, le type **int** et le type **pointer**.

Le code généré par le compilateur C_Card n'effectue aucune hypothèse sur l'organisation réelle de la mémoire : adressage par octet, par mot, par segment... Toutefois, le compilateur calcule les tailles des déclarations du source C en nombre d'octets. Pour cela, les trois informations élémentaires (taille en octets des trois types de variables) doivent être fournies au compilateur. Les tailles des variables de classes GLOBAL, PARAMETER, LOCAL et CONST sont évaluées à l'aide de ces trois informations. Les variables de la classe travail ont automatiquement un nombre d'octets égal à celui de POINTER.

Le langage QUAD généré est entièrement **symbolique**. Il n'existe donc pas à ce stade une codification quelconque de la machine QUAD. Ce langage possède deux types d'instructions :

1. **Les instructions exécutables**.

   Ces instructions à trois adresses se font uniquement sur des variables scalaires.

2. **Les pseudo-instructions**.

   Ces pseudo-instructions sont destinées à faciliter la conception du traducteur. Elles donnent des renseignements sur les variables (le type de variable, la classe de mémoire où elles se trouvent, leur durée de vie) et sur la structure de contrôle du programme.

Les instructions QUAD se répartissent en quatre catégories : les instructions d'échange, les instructions unaires, les instructions binaires et les instructions de schéma de contrôle. Il est important de souligner que nous retrouvons dans les instructions QUAD la richesse des opéra-

teurs du langage C. Cette caractéristique a pour principal avantage la génération aisée et surtout **compacte** du code intermédiaire par le compilateur C_Card.

TABLE 1 Instructions exécutables QUAD

Syntaxe	Sémantique
Instructions d'échange	
MOV res, variable, nboctets[a]	res = variable
LOAD variable, pointeur, nboctets	variable=*pointeur
STORE pointeur, variable, nboctets	*pointeur=variable
Instructions unaires	
LEA res,op	res = &op
NOT res,op	res = ~op
NEG res,op	res = - op
NOT_LOGICAL res,op	res = !op
Instructions binaires	
OP[b] res, op1, op2	res = op1 OP op2
Instructions de schémas de contrôle	
PARAM idvar, nom_fonction,idpar	passage de paramètres
CALL nom_fonction, nbparamètre	appel de fonction
INTR code_intr	appel d'interruption logicielle
LABEL étiquette	définition d'étiquettes
JMP_FALSE var, étiquette	if (var == FALSE) goto étiquette
JMP_TRUE var,étiquette	if (var == TRUE) goto étiquette
JMP étiquette	goto étiquette
RET nom_fonction	retour de fonction

a. nboctets : nombre d'octets à copier

b. OP = MUL, DIV, MOD, ADD, SUB, SHL, SHR, CMP_L, CMP_G, CMP_LE, CMP_GE, CMP_NE, CMP_E, AND, OR, AND_LOGICAL, OR_LOGICAL.

Les pseudo-instructions ont pour rôle de définir les déclarations des variables du source C, de fournir des informations facilitant l'implantation du traducteur, de permettre la réalisation d'un débogueur symbolique sur la machine cible. Ces pseudo-instructions se répartissent en trois catégories (Tableau 2 de la page 158) :

- les pseudo-instructions de déboguage
- les pseudo-instructions de déclarations de variables. Il y a actuellement cinq classes de variables : les classes GLOBAL, LOCAL, PARAMETER, WORK et CONST.

- les définitions de fonctions et de blocs. Ces pseudo-instructions permettent de déterminer la durée de vie des différentes variables. Ainsi, la durée de vie d'une variable locale est comprise entre les premières pseudo-instructions BEGIN_BLOCK et END_BLOCK qui suivent la pseudo-instruction de déclaration de cette variable.

TABLE 2 Les pseudo-instructions QUAD

Syntaxe	Sémantique
Directives de déboguage	
FILE nom_de_fichier	nom du fichier d'inclusion
LINE numéro	numéro de ligne du fichier en cours
Déclarations	
DCL nom_variable, classe, nboctets	déclaration de variables
CONST_STRING "xxx..."	constantes chaînes
CONST_CHAR val	constante caractères
CONST_INT val	constante entière
POINTER_SIZE val	taille du type pointeur en octets
INT_SIZE val	taille du type entier en octets
CHAR_SIZE val	taille du type caractère en octets
Définitions de blocs	
FUNCTION nom_fonction	début d'une fonction
END_FUNCTION	fin d'une fonction
BEGIN_BLOCK	début d'un bloc d'instructions
END_BLOCK	fin d'un bloc d'instructions
BEGIN_PARAMETER	début de déclaration de paramètres
END_PARAMETER	fin de déclaration de paramètres
BEGIN_LOCAL	début de déclaration de var. locales
END_LOCAL	fin de déclaration de var. locales

Nous illustrons le langage QUAD généré à l'aide de l'exemple suivant :

FIGURE 43 Exemple de code généré

```
/*----- FACT ----------*/
int fact(int n)
{
 int i,f;

 f=1;
 i=1;
 while(i<=n) {
 f=f*i;
 i++;
 }
 return f;
}
```

FUNCTION[1] fact
BEGIN_PARAMETER
DCL fact_n,Parameter,4
END_PARAMETER
BEGIN_LOCAL
DCL fact_i,Local,4
DCL fact_f,Local,4
END_LOCAL
BEGIN_BLOCK fact
MOV[2] fact_f,1,4
MOV fact_i,1,4
LABEL while1
DCL _fact_1,Work,4
CMP_LE _fact_1,fact_i,fact_n
JMP_FALSE _fact_1,endwhile1
BEGIN_BLOCK fact_1
MUL _fact_1,fact_f,fact_i
MOV fact_f,_fact_1,4
ADD fact_i,fact_i,1
END_BLOCK fact_1
JMP while1
LABEL endwhile1
MOV **_return_value**,fact_f,4
RET fact
END_BLOCK fact
END_FUNCTION

1. les pseudo-instructions sont en italiques
2. les instructions sont en gras

Toutes les instructions du langage QUAD utilisent les variables définies par la pseudo-instruction *DCL*. Il existe cependant une exception : la variable *_return_value* qui est l'unique variable standard prédéfinie du langage. Cette variable est utilisée pour retourner le résultat d'une fonction.

Table des Figures

Références bibliographiques

[Ale93] T. Alexandre, "La compression de données dans la carte à microprocesseur", Publication LIFL, décembre 1993.

[Ale95] T. Alexandre, «Manipulation De Données Multimédia Dans La Carte A Microprocesseur: Application A L'identification Biométrique Et Comportementale», Thèse de Doctorat en Informatique, LIFL, Université de Lille I, n° 1494, Février 1995

[BP91] R. Beuscart, P. Paradinas, «Smart Cards for Health Care», Telematics in Medicine, J.S. Duisterhout, A. Hasman, R. Salomon, Elsevier Science Publishers B.V. North Holland, 1991

[Bright88] R. Bright, «Smart Cards, Principles, Practice, Applications», Ellis Horwood Limited

[Bro94] G. Brouard, «Chaîne de compilation pour Carte Blanche» - Mémoire de fin d'étude de D.U.T. - I.U.T. de Calais, juin 1994

[BM84] JM. Busta, S. Miranda, «Introduction Aux Bases De Données», Ed. Eyrolles, , 1984

[Caron94] O. Caron, «Méthodologies De Conception Et D'évaluation D'architecture R.I.S.C. Adaptées Aux Futures Cartes A Microprocesseur», Thèse de Doctorat en Informatique du L.I.F.L., n°1256, ,Janvier 1994

[CG94] O. Caron, G. Grimonprez, "O.C.E.A.N. : A C compiler for new smart card applications", International Conference in Compiler Construction, CC'94, Edinburgh, avril 1994.

[CGB91] V. Cordonnier, G. Grimonprez, R. Beuscart, "Smart Cards and Portable Data Files: a glance at the future.", Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Vol 13, n°3, 1991, page 1387.

[CH94] P.C. Clark, L.J. Hoffman, «BITS : A Smartcard Protected Operating System», in communications of the ACM, Vol. 37, n° 11, p. 66, november 1994

[Cor91] V. Cordonnier, "Smart Cards: present and future applications and techniques", Electronics & communication engineering journal, Octobre 1991.

[Cor92] V.Cordonnier, "Assessing the future of smart cards", Proc. of the CardTech'92, Washington, DC, avril 92.

[CP93] V. Cordonnier, T. Peltier - A taxonomie of microprocessor cards - Conference "A la carte 93" - Hamburg, Juin 93

[Crindle90] J. McCrindle, «Smart Cards», IFS Publications/Springer-Verlag, 1990

[CS91] R. Carr, D. Shafer, «The power of PenPoint : The powerfull New Pen-Based Operating System from GO Corporation» Ed. Addison Wesley, 1991

[Den82] D. Denning, «Cryptography And Data Security», Addison-Wesley Publishing Company, 1982

[ETSI] Norme ETSI TE9 : «Terminal Equipment (TE); Requirements for IC cards and terminals for telecommunication use, Part 2 - Application independent card requirements» - European Telecommunications Standards Institute (ETSI) 1990

[Geo93] P. George, Manuel de référence de C-Card, Document Gemplus Card International, 1993.

[GG92] E. Gordons, G. Grimonprez, «A card as element of a distributed database», IFIP WG8.4 Workshop, Ottawa 1992

[Gor90] E. Gordons, "Un processeur pour cartes à micro-circuit : CAVIMA", Mémoire CNAM, Décembre 1990.

[GQU92] L. Guillou, J.J. Quisquater, M. Ugon, «The smart card : A Standardised Security Device Dedicated to Public Cryptology», Ed G. Simmons : Comtempory Cryptology, IEEE-Press, 1992

[Gri92] G. Grimonprez, «Etude et réalisation d'une carte à micro-processeur intégrée aux systèmes de gestion de bases de données», Mémoire d'habilitation, février 1992

[GRL88] F. Guez, C. Robert, A. Lauret, «Les cartes à microcircuit : Techniques et Applications», Masson 1988

[GV90] M. Griffiths, M. Vayssade, «Architecture des systèmes d'exploitation», Traité des Nouvelles Technologies, série informatique, Ed. Hermés, Janvier 1990

[ISO1] Normes internationales ISO, 7816-1 : «Identification cards - Integrated circuit(s) cards with contacts - Part 1 : Physical characteristics» - International Standard Organisation 1987

[ISO2] Normes internationales ISO, 7816-2 : «Identification cards - Integrated circuit(s) cards with contacts - Part 2 : Dimensions and location of the contacts» - International Standard Organisation 1988

[ISO3] Normes internationales ISO, 7816-3 : «Identification cards - Integrated circuit(s) cards with contacts - Part 3 : Electronic signals and transmission protocols» - International Standard Organisation 1989

[ISO4] Normes internationales ISO, 7816-4 : «Identification cards - Integrated circuit(s) cards with contacts - Part 4 : Interindustry commands for international interchange» - International Standard Organisation

[ISO5] Normes internationales ISO, 7816-4 : «Identification cards - Integrated circuit(s) cards with contacts - Part 5: Registration system for application in IC Cards» - Working Draft, International Standard Organisation

[LM90] J. Levine, T. Mason, «Lex & Yacc», O'Reilly & Associates, Inc, 1990

[Mas89] G. Masini, A. Napoli, D. Colnet, D. Léonard et K. Tombre, «Les langages à objets», - InterEditions, Paris, 1989

[MCOS] «MCOS 16K EEPROM, Manuel de Référence», Document Gemplus Card International, 1990

[Mey90] B. Meyer, «Conception et programmation par objets, pour du logiciel de qualité» - Inter-Editions, Paris, 1990

[Mou91] M. Mouly, M.B. Pautet, «The GSM Systel Mobile for Mobile Computing, 1991

[MR94] S. Miranda, A. Ruols, «CLIENT-SERVEUR : Concepts, moteurs SQL et architectures parallèles», Ed. Eyrolles, 1994

[NBS80] National Bureau of Standard, DES modes of operation, Federal Information Processing Standard. Ed : US Department of Commerce, FIPBS Pub 46, 1980 (Washington).

[Par88] P. Paradinas, «La BioCarte, Intégration d'une carte à microprocesseur dans un réseau professionnel santé», Thèse de Doctorat de 3ème cycle, 1988

[Pel93a] T.Peltier, «Operating System for Portable Device», in proceeding of TELEPRESENCE'93 : workshop Smart Cards and Cooperative Activities - Lille, March 24, 1993

[Pel93b] T. Peltier - «Système d'Exploitation pour Objet Nomade» - LIFL Publication AS-132, Juin 1993

[Pel95] T. Peltier , «Operating System for Blank Card», proceedings of CardTech/SecurTech '95 conference, Washington DC, April 1995

[Pey91] P. Peyret, «The Security of Executable Code in an ADF», Chip Card News n° 35, mars 1991

[Pey94] P. Peyret, «RISC-Based, Next-Generation Smart Card Microcontroller Chips», in Proceedings of the Cardtech/Securtech '94 International Conference, Arlington, Virginia, USA, April 10-13, pp 9-36, 1994

[PPT95] T. Peltier, J.-M. Place, P. Trane, «Secured Cooperation Of Partners and Applications In the Blank Card», in proceedings of the GMD-SmartCard WorkShop, Ed. Struif, Darmstadt, 31 january -01 february 1995

[PV94] P. Paradinas, J.-J. Vandevalle, «New Operating System for IC-Cards», 6th ACM SIGOPS European Workshop about "Operating Systems to Application Needs», Warden, Germany, September 1994

[Ros88] J.-P. Rosen, «Méthode de conception orientée objets», Génie logiciel, EC2, Nanterre, 1988

[RSA78] R.L. Rivest, A. Shamir, A. Adleman, A method for obtening digital signature and public key crypto-system. Ed : Communication of the ACM, février 1978

[SIM94] «European digital cellular telecommunications system (Phase 2); Specification of the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface (GSM 11.11) » - European Telecommunications Standards Institute (ETSI) 1994

[Svigals87] J. Svigals, «Smart Cards, the new bank cards», Macmillan, 1987

[Tan91] A. Tanenbaum, «Les systèmes d'exploitation. Conception et mise en oeuvre», InterEditions 1989

[Tas92] J. Tasket, «Communications protocoles», proceedings of CardTech '92 conference, Washington DC, 1992

[TB100] «TB100 Smart Card Reference Manual», Philips Telecommunicatie en Data Systemen Nederland B.V., Systems Support - Training & Documentation, 1990

[Trane95] P. Trane, «Conception et réalisation d'un système de contrôle d'accès pour la carte à micro-processeur», Thèse de Doctorat en Informatique, LIFL, Université de Lille I, n° 1562, septembre 1995

[Vand 91] J.-J. Vandewalle, «Une approche système d'exploitation pour équipement portable», Mémoire de DEA Informatique de l'Université de Lille I, juin 1991

[Vast92] P.A. Vast, «Communications d'un Système d'Exploitation pour Dossier Portable», Rapport de Stage de DESS Génie Informatique (TIR), Université de Lille I, Juin 1992

[Ugon93] M. Ugon,"Le futur de la carte à puce", Cartes'93, Octobre 1993.

[UNIX] C. Pélissier, «Utilisation et administration du système UNIX», Traité des Nouvelles Technologies, série informatique, Ed. Hermés, Janvier 1992


UNIVERSITÉ DES SC. ET TECHN.
BIBLIOTHÈQUE UNIVERSITAIRE
LILLE I