TABLE DES MATIÈRES

Page

INTRODUCTION 1

CHAPITRE 1 L'INFORMATIQUE DIFFUSE 6
1.1 Présentation de l'informatique diffuse 6
1.2 Exemple de scénario 10
1.2.1 Scénario 10
1.2.2 Analyse du scénario 11
1.3 Avantages et inconvénients de l'informatique diffuse 12

CHAPITRE 2 CONTEXTE ET SENSIBILITÉ AU CONTEXTE 14
2.1 Définitions générales 14
2.2 Définitions du contexte dans un système informatique 16
2.2.1 Définitions antérieures 16
2.2.2 Notre définition orientée service du contexte 18
2.3 Définitions antérieures de la sensibilité au contexte 23
2.3.1 Notre définition de la sensibilité au contexte 24
2.4 Utilisation du contexte 25
2.5 Sources d'informations 27
2.6 Catégorisation du contexte 28
2.6.1 Notre catégorisation du contexte 30
2.7 Qualité de l'information contextuelle 30

CHAPITRE 3 ARCHITECTURE POUR LES SYSTÈMES SENSIBLES AU CONTEXTE 33
3.1 Travaux connexes 33
3.1.1 ActiveBadge (1992) 34
3.1.2 ParcTab (1995) 36
3.1.3 Stick-e-notes (1997) 37
3.1.4 Cyberguide (1997) 38
3.1.5 CASS (2004) 39
3.1.6 CORTEX (2004) 40
3.1.7 Context management framework (2003) 42
3.1.8 JCAF (2005) 43
3.1.9 Context Toolkit (2001) 45
3.1.10 Hydrogen (2003) 46
3.1.11 SOCAM (2004) 47
3.1.12 CoBrA (2004) 49
3.2 Analyse des architectures antérieures 49
3.3 Architecture multiagents orientée service 52
3.4 Discussion 55
3.5 Modélisation et simulation 57

3.5.1 Les réseaux de Petri colorés et le CPN-Tools 58
3.5.2 Modélisation et simulation de l'architecture 58

CHAPITRE 4 MODÉLISATION DU CONTEXTE 80
4.1 Travaux connexes de modélisation du contexte 80
4.1.1 Approches non ontologiques 80
4.1.2 Approches ontologiques 92
4.1.3 Analyse des modèles ontologiques antérieurs 99
4.2 Proposition d'une ontologie de service 101
4.3 Scénario d'application 108

CHAPITRE 5 ADAPTATION DE SERVICES SELON LE CONTEXTE 113
5.1 La notion d'adaptation 113
5.2 Approche d'adaptation fondée sur un apprentissage automatique 114
5.2.1 Méthodologie 114
5.2.2 Scénario d'application 118
5.3 Approche d'adaptation sensible aux ressources limitées 122
5.3.1 Méthodologie 122
5.3.2 Modélisation et simulation 131

CONCLUSION 147

ANNEXE I CODE OWL DE SÉNARIO D'APPLICATION 150

ANNEXE II RÉSULTATS DE L'ALGORITHME D'APPRENTISSAGE POUR LE SÉNARIO D'APPLICATION 155

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES 159

LISTE DES TABLEAUX

Page

Tableau 2.1 Informations contextuelles du scénario 1 20

Tableau 2.2 Informations contextuelles du Scénario 2.....21

Tableau 2.3 Informations contextuelles du scénario 3 22

Tableau 3.1 Caractéristiques des architectures étudiées. 51

Tableau 3.2 Les entrées possibles d'un agent de service.....60

Tableau 3.3 Informations contextuelles et agent central capteur.....62

Tableau 3.4 Les configurations possibles et les résultats de simulation pour l'agent de service 65

Tableau 4.1 Table décrivant des documents. 81

Tableau 4.2 Le résultat des trois étapes de modélisation.....111

Tableau 5.1 Configurations du contexte et formes de services 120

Tableau 5.2 Architecture et caractéristiques du réseau de neurone utilisé 121

Tableau 5.3 Types d'informations contextuelles 124

Tableau 5.4 Exemple d'un MRL 128

LISTE DES EXTRAITS

Page

Extrait 3.1 Ensemble de déclarations utilisé pour la modélisation du SID…….61

Extrait 3.2 Ensemble de déclarations utilisé pour la modélisation de l'agent de service……64

Extrait 3.3 Ensemble de déclarations utilisé pour la modélisation de l'agent de service……68

Extrait 4.1 Exemple un message ConteXtML…….82

Extrait 4.2 Exemple d'un profil CC/PP d'un ordinateur portable……...83

Extrait 4.3 Code OWL de notre ontologie…….105

Extrait 5.1 Ensemble des déclarations utilisé pour la construction du modèle de la situation 1.....….....132

Extrait 5.2 Ensemble des déclarations utilisé pour la construction du modèle de la situation 2……139

LISTE DES FIGURES

Page

Figure 1.1 Évolution du marché mondial des ordinateurs....6

Figure 1.2 Ubiquité des équipements informatiques....7

Figure 1.3 Évolution vers l'informatique diffuse....8

Figure 1.4 Les composants d'un système informatique diffus....10

Figure 2.1 Les composants d'un système diffus. 19

Figure 2.2 Diagramme d'états du scénario 1....20

Figure 2.3 Diagramme d'états du Scénario 2. 21

Figure 2.4 Diagramme d'états du scénario 3....22

Figure 3.1 Architecture de l'ActiveBadge. 35

Figure 3.2 Architecture du ParcTab. 37

Figure 3.3 Architecture de CASS....40

Figure 3.4 Architecture d'un « sentient object ». 41

Figure 3.5 Architecture de CMF. 43

Figure 3.6 Architecture de JCAF. 44

Figure 3.7 Éléments de l'architecture du Contexte Toolkit. 46

Figure 3.8 Architecture de Hydrogen....47

Figure 3.9 Architecture de SOCAM....48

Figure 3.10 Architecture de CoBrA. 50

Figure 3.11 Diagramme d'états d'un service ayant trois formes. 53

Figure 3.12 Une partie de l'automate d'états pour l'indication d'appels d'un téléphone cellulaire....54

Figure 3.13 Architecture multiagents d'un système composé de quatre équipements matériels....55
Figure 3.14 L'architecture d'un agent central. 56
Figure 3.15 Le fonctionnement global de l'architecture. 60
Figure 3.16 Le modèle du SID. 63
Figure 3.17 Le modèle de l'agent de service....66
Figure 3.18 Le modèle global de l'agent central. 72
Figure 3.19 Le modèle de mise à jour et de sélection de service. 73
Figure 3.20 Le modèle de vérification d'appartenance au contexte de l'agent central....74
Figure 3.21 Le modèle de vérification d'appartenance au contexte d'un service (1) ...74
Figure 3.22 Le modèle de vérification d'appartenance au contexte d'un service (2) ...75
Figure 3.23 Le modèle de sélection et évaluation d'expression de déclenchement d'un service....76
Figure 3.24 Le modèle de sélection d'une forme de service. 77
Figure 3.25 Le modèle d'évaluation de l'expression de changement de forme. 78
Figure 3.26 Le modèle d'expédition de résultats à l'agent de service. 79
Figure 4.1 Exemple de modélisation du contexte par CML....86
Figure 4.2 Exemple de modélisation du contexte par GPM....87
Figure 4.3 Le modèle UML de l'approche Hydrogen. 89
Figure 4.4 Les éléments de l'ontologie SOUPA. 95
Figure 4.5 L'ontologie à deux niveaux de l'approche CONON. 97
Figure 4.6 Concepts centraux du modèle ASC. 98
Figure 4.7 Les éléments de l'ontologie de service. 104
Figure 4.8 Diagramme de transition du scénario d'application. 109
Figure 4.9 Les composants de l'ontologie de service du scénario d'application. 112

Figure 5.1 Éléments du contexte contenant différentes valeurs et configurations possibles du contexte 115
Figure 5.2 Architecture du système d'adaptation. 123
Figure 5.3 Types d'informations contextuelles. 124
Figure 5.4 Structure de DSC. 125
Figure 5.5 Modèle de la situation 1 : services déclenchés automatiquement....133
Figure 5.6 Modèle de la situation 1 (étape 1)....134
Figure 5.7 Modèle de la situation 1 (étape 2)....135
Figure 5.8 Modèle de la situation 1 (étape 3)....136
Figure 5.9 Modèle de la situation 2 : services déclenchés automatiquement et manuellement....140
Figure 5.10 Modèle de la situation 2 (étape 1)....141
Figure 5.11 Modèle de la situation 2 (étape 2)....142
Figure 5.12 Modèle de la situation 2 (étape 3)....143
Figure 5.13 Modèle de la situation 2 (étape 4)....144

LISTE DES ABRÉVIATIONS ET SIGLES

API :	Application programming interface (interface de programmation d'applications)
ASC:	Aspect-Scale-Context (aspect-échelle-contexte)
CASS:	Context-awareness sub-structure (sous-structure sensible au contexte)
CCML:	Centaurus Capability Markup Language (langage de balisage de capacités centaures)
CC/PP:	Composite Capabilities / Preferences Profiles (capacités composées/profiles de préférences)
CMF:	Context management framework (cadre de travail de gestion de contexte)
CML:	Context Modeling Language (langage de modélisation de contexte)
CMM:	Montext metamodel (metamodel de contexte)
CoBrA:	Context Broker Architecture (Architecture de courtier de contexte)
CONON:	CONtext ONtology (ontologie de contexte)
ConteXtML:	Contexte Markup Language (langage de balisage de contexte)
CoOL:	Context Ontology Language (langage d'ontologie de contexte)
CORTEX:	CO-operating Real-time senTient objects: architecture and EXperimental evaluation (Objets sensibles de coopération en temps réel : architecture et évaluation expérimentale)
CSCP:	Comprehensive Structured Context Profiles (profils de contexte structurés compréhensives)
DSC:	Déclencheur sensible au contexte
ELCF :	Expression logique de changement de forme
ELDS :	Expression logique de déclenchement du service
GPRS :	General Packet Radio Service (Service général de transmission radioélectrique par paquet).

GSM :	Global System for Mobile communications (Système global pour les communications mobiles).
GUI:	Graphical user interface (interface graphique utilisateur)
JCAF:	Java context awareness framework (cadre de travail Java sensible au contexte)
MRL:	Moniteur de ressources limitées
ORM:	Object Role Modeling (Modélisation de rôle d'objet)
OWL:	Ontology Web Language (langage web d'ontologie)
PDA:	Personal Digital Assistant (assistant numérique personnel)
PPDL:	Pervasive Profile Description Language (langage de description diffus de profil)
RDF:	Resource Description Framework (cadre de description de ressources)
RdPC:	Réseaux de Petri colorés
RFID :	Radio Frequency Identification (Identification par Radio Fréquence)
RMI:	Remote method invocation (invocation à distance de procedure)
RPC:	Remote Procedure Call (appel de procedure distante)
SGML:	Standard Generic Markup Language (Langage de balisage générique standard)
SID:	Système informatique diffus.
SMS :	Short Message Service (service de message court)
SOCAM:	Service-Oriented Context-Aware Middleware (intergiciel sensible au contexte orienté service)
SOUPA:	Standard Ontology for Ubiquitous and Pervasive Applications (ontologie standard pour les applications ubiquitaires et diffuses)
UAProf:	User Agent Profile (profil d'agent d'utilisateur)
UML:	Unified Modeling Language (langage de modélisation unifié)

UMTS:	Universal Mobile Telecommunications System (système de télécommunications mobiles universelles).
WiFi:	Norme IEEE 802.11 pour la communication sans fil.
XML:	eXtended Markup Language (langage de balisage étendu)

INTRODUCTION

Avant propos

L'évolution technologique réalisée dans le domaine de l'informatique ne cesse de croître et marque profondément notre société actuelle. Dans une société où l'accès au savoir et à l'information est primordial, l'intégration et le passage rapide à l'exploitation des dernières technologies de communication et d'accès à l'information ont contribué à l'émergence de l'informatique dans différents domaines. Le récent développement des terminaux, et plus généralement des équipements informatiques[1], et les technologies des systèmes embarqués rendent aujourd'hui réalisables la vision du 21ème siècle de Mark Weiser (Weiser, 1991) où les systèmes informatiques sont omniprésents dans la vie de tous les jours. Cette vision où la technologie devient invisible à l'utilisateur est connue sous le nom de l'informatique diffuse (en anglais « pervasive computing »). Ce domaine, combinant les aspects de l'informatique distribuée et de l'informatique mobile, adopte une vision nouvelle des équipements et des applications. Ceux-ci sont assimilés à des "tâches" plutôt qu'à une forme d'exploitation des ressources d'un terminal. L'un des aspects contribuant à la réalisation de cette notion d'informatique diffuse est de permettre à ces systèmes informatiques de s'adapter aux changements du contexte des utilisateurs et des applications.

Une caractéristique importante des systèmes informatiques diffus (SID), qui leur permet d'interagir de manière transparente avec les usagers et de les assister dans leurs tâches de tous les jours, est la sensibilité au contexte (en anglais « context-awareness »). Le contexte dans un système diffus regroupe des informations et des événements qui ont lieu dans l'environnement des entités du système informatique. Ces entités sont : les usagers, le matériel, les logiciels, etc. Les informations du contexte sont utilisées par le système informatique pour fournir du soutien et pour réagir proactivement aux demandes des utilisateurs. Tel que, augmenter automatiquement le volume de sonnerie d'un téléphone

[1] Le terme 'équipement informatique' sera employé dans le reste de ce document comme équivalent du terme anglais 'computer device'

cellulaire lorsque l'utilisateur se trouve dans un environnement bruité ou passer au mode silencieux lorsqu'il dort.

Définition de la problématique

Les caractéristiques principales d'un environnement diffus sont : l'ubiquité (plusieurs matériels informatiques), l'hétérogénéité et la dynamicité (à cause de la mobilité). Cela rend la tâche difficile, c'est-à-dire gérer le matériel informatique pour l'utilisateur et engendre un effort supplémentaire de sa part. De ce fait, la proactivité du matériel informatique s'impose. Par contre cela ne se réalise que lorsque le matériel est sensible à son contexte. D'où l'importance du contexte. Cela nous oblige à bien comprendre le contexte et à le définir d'une façon précise et formelle pour une meilleure adaptation selon sa variation. Malgré le grand nombre de définitions existantes et les similarités (la plupart font référence à la localisation et à l'environnement) le mot contexte reste toujours général et vague. La liste exhaustive des éléments du contexte requise pour le système qui effectue l'adaptation doit être fournie dans les délais et mise à jour d'une manière continue.

La catégorisation du contexte sera exigée pour l'éventail d'informations contextuelles hétérogènes dans les prochaines générations d'applications sensibles au contexte. La catégorisation du contexte aide les concepteurs et les développeurs d'applications à découvrir et à manipuler les contextes possibles. La classification des informations contextuelles peut alors aider à fournir des informations contextuelles de qualité.

Parce qu'elle constitue un nouveau domaine, l'informatique diffuse exige, depuis son avènement, la mise en place de nouveaux outils et de nouvelles mesures (architectures, cadres de travail, intergiciels) qui ne faisaient pas partie de l'informatique traditionnelle. La plupart des outils actuels ne tenaient pas compte des particularités des systèmes informatiques diffus telles que la mobilité et la sensibilité au contexte. Ces outils seront d'une grande importance pour appuyer le développement des systèmes sensibles au contexte.

La modélisation du contexte est une étape essentielle pour le développement des applications sensibles aux contextes. La plupart des approches proposées dans la littérature sont ou bien spécifiques à une application ou à un domaine particulier ou bien il leur manque le formalisme nécessaire pour la modélisation. Ces approches ne sont pas suffisamment génériques pour être réutilisables dans d'autres applications ou autres domaines et ont un niveau de reconfiguration limité. De plus, les plateformes proposées pour développer ou rendre des applications sensibles aux contextes sont souvent spécifiques à un domaine.

L'objectif principal d'un SID est de fournir proactivement des services bien adaptés aux utilisateurs et aux applications. Cette tâche d'adaptation doit se faire selon le contexte actuel et en tenant compte des ressources limitées des équipements. Les méthodes existantes ne prennent pas en considération la nature dynamique d'un SID causée par la mobilité et n'abordent pas en profondeur l'aspect de sensibilité au contexte.

Objectifs de recherche

Le cadre général de nos recherches est le domaine de l'informatique diffuse. Notre intérêt porte en particulier sur la proposition d'une nouvelle architecture logicielle pour appuyer le développement des systèmes sensibles au contexte. Ceci comprend a) la redéfinition du contexte et de la sensibilité au contexte ainsi que la catégorisation des informations contextuelles, b) l'architecture générale d'un système sensible au contexte, c) la gestion dynamique du contexte dans l'informatique diffuse en particulier la modélisation du contexte et d) l'adaptation dynamique des systèmes diffus aux changements dynamiques de l'environnement et aux contextes des utilisateurs.

Pour atteindre nos objectifs décrits précédemment, il est nécessaire d'avoir une bonne compréhension du terme contexte dans un environnement diffus et d'avoir une définition suffisamment abstraite et permettant de limiter l'ensemble des informations contextuelles pertinentes pour la tâche d'adaptation. Ces informations sont mises à la disposition des applications après avoir été collectées par un réseau hétérogène de capteurs disséminés dans l'environnement diffus. Pour tenir compte de ces informations (environnement, utilisateurs et

autres) dans les applications, il faut arriver à les représenter, à les stocker dans l'éventualité d'une utilisation ultérieure, à donner le moyen (des interfaces) aux applications d'accéder aux informations contextuelles et à les partager avec souplesse.

Une architecture extensible et réutilisable doit offrir une bonne abstraction du contexte par interprétation des informations brutes recueillies par les capteurs physiques de l'environnement diffus. Elle doit tenir compte de la distribution des équipements informatiques qui composent le SID et des informations contextuelles tout en favorisant l'autonomie de ces équipements. Elle doit être dotée d'un module de raisonnement sur le contexte pour permettre de déduire de nouvelles informations contextuelles qui ne peuvent pas être explicitement fournies par les capteurs du système diffus.

La nature particulièrement dynamique des environnements diffus, tant au niveau physique que logiciel, a pour conséquence que les informations contextuelles disponibles et demandées changent constamment. La même information peut être fournie par plusieurs sources selon des précisions différentes. Un type d'information fournie par un capteur dans une première configuration pourrait provenir d'un autre capteur dans une autre configuration selon chaque cas accompagné d'incertitude différente. Au cours de leur fonctionnement, les systèmes diffus doivent prendre en compte des nouveaux types d'information contextuelle qui n'étaient pas disponibles lors du chargement de l'application (loading time) ou au pire des cas au moment de sa conception (design time). Notre objectif principal est de développer un modèle abstrait et indépendant du domaine d'application pour représenter les informations contextuelles. C'est ce modèle qui sera présenté aux applications et servira à manipuler les données contextuelles. Notre but est de faire en sorte que les informations contextuelles soient disponibles à tout moment pour être utilisées au besoin, sur demande ou dans un entrepôt de données contextuelles. Leur représentation doit être indépendante de la source qui les a recueillies. Ainsi, notre modèle doit être persistant et suffisamment abstrait. À la différence des modèles de contexte existants, développés selon les exigences d'une application ou d'un domaine particulier ; nous envisageons de développer un modèle

suffisamment « générique », pour qu'il puisse être instancié avec le moindre effort dans n'importe quelle application.

Enfin, la tâche d'adaptation de services dans un système diffus doit se faire d'une manière transparente à l'utilisateur et doit être sensible au contexte global tout en prenant en considération les ressources limitées des équipements informatiques.

Structure de ce document

Le chapitre 1 fera l'objet d'une présentation générale de l'informatique diffuse qui constitue le cadre général de notre recherche, ses différents aspects et sa relation avec les autres domaines de l'informatique. Nous présenterons, dans le chapitre 2, les concepts de contexte, la sensibilité au contexte des systèmes informatiques en général, et des systèmes informatiques diffus en particulier. Nous présenterons notamment les catégorisations existante du contexte disponibles dans la littérature et les limitations des travaux antérieurs. Nous proposerons ensuite nos propres définitions du contexte, de la sensibilité au contexte et notre catégorisation fondée sur le concept de service. Nous utiliserons des scénarios simples pour mettre en évidence la validité de notre approche. Le chapitre 3 exposera une revue de la littérature sur les architectures des systèmes sensibles aux contextes, leurs points forts et leurs points faibles suivis de notre proposition d'une nouvelle architecture multiagents orientée service. Dans cette même partie, la modélisation et la validation de cette architecture seront aussi présentées. Dans le chapitre 4, nous présenterons les différents modèles de représentation des informations contextuelles disponibles dans la littérature ainsi que notre modèle de contexte à base d'ontologies. Le chapitre 5 fera l'objet d'une étude des méthodes d'adaptation de services existantes suivie de notre approche d'adaptation selon le contexte. Enfin, nous conclurons ce document par un résumé de nos principales contributions et d'une description des perspectives que nous envisageons pour l'avenir.

CHAPITRE 1

L'INFORMATIQUE DIFFUSE

1.1 Présentation de l'informatique diffuse

La notion de l'informatique diffuse a été proposée au départ comme une vision future de l'informatique par Mark Weiser (Weiser, 1991) suite à son analyse du marché mondial des ordinateurs et des équipements informatiques (Figure 1.1). Il a remarqué une croissance énorme du nombre d'ordinateurs : un ordinateur pour plusieurs utilisateurs, puis un ordinateur par utilisateur ensuite, plusieurs ordinateurs par utilisateur en utilisant l'intégration des processeurs dans des objets de la vie quotidienne (systèmes embarqués). Cette nouvelle forme de l'informatique qu'il a nommée informatique ambiante (en anglais "ubiquitous computing") et dont l'objet viserait à assister implicitement et discrètement un utilisateur dans les tâches qu'il accomplit au quotidien, devenant ainsi la base des systèmes informatique diffus (Cliquet, 2007).

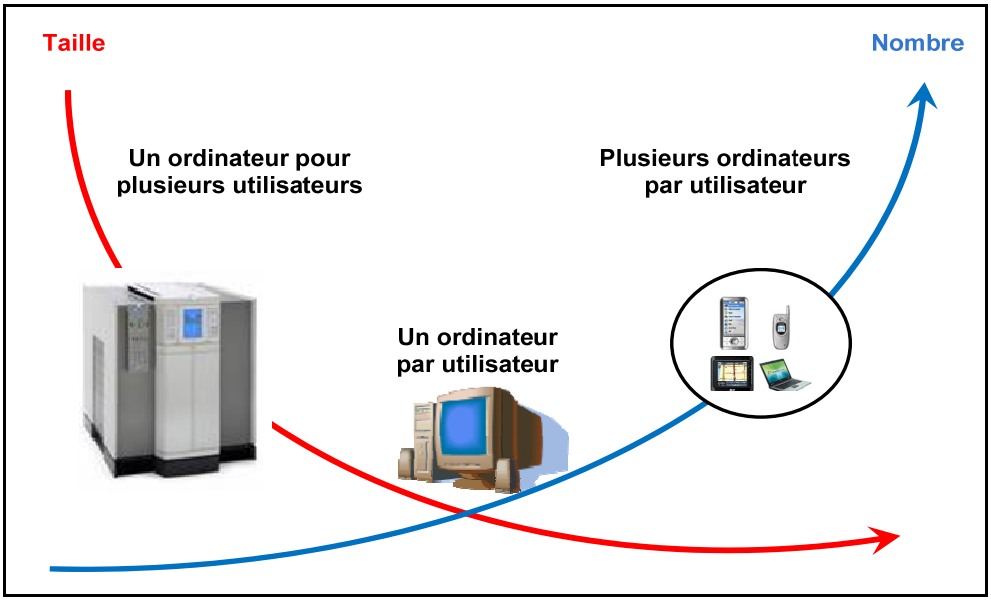

Figure 1.1 Évolution du marché mondial des ordinateurs.
Adaptée de Riveill (2002, p. 5)

L'évolution technologique actuelle rend aujourd'hui la vision de Mark Weiser réaliste à la suite de l'apparition des nouveaux systèmes embarqués ayant des tailles de plus en plus petites et enfouis dans différents objets de la vie quotidienne. L'avènement des réseaux de communications sans fil avec les standards de télécommunication, tels que GSM, GPRS, UMTS, particulièrement utilisés pour la téléphonie mobile, mais également avec l'apparition de WiFi, RFID et Bluetooth pour les équipements informatiques et particulièrement pour les terminaux informatiques mobiles tels que les assistants personnels (PDA : personal digital assistant) et les téléphones cellulaires,(resp. iPhone), a permis à ces équipements de communiquer (profil matériel, contexte, etc.) pour coopérer. Ceci se fait d'une manière transparente pour l'utilisateur sans son intervention explicite et lui offre la possibilité de se concentrer sur sa tâche principale au lieu de configurer et de gérer l'ensemble des équipements informatiques mis à sa disposition (Figure 1.2). L'informatique diffuse favorise par exemple la création d'environnements intelligents tels que la maison intelligente capable de gérer automatiquement les différents équipements présents au domicile de l'utilisateur.

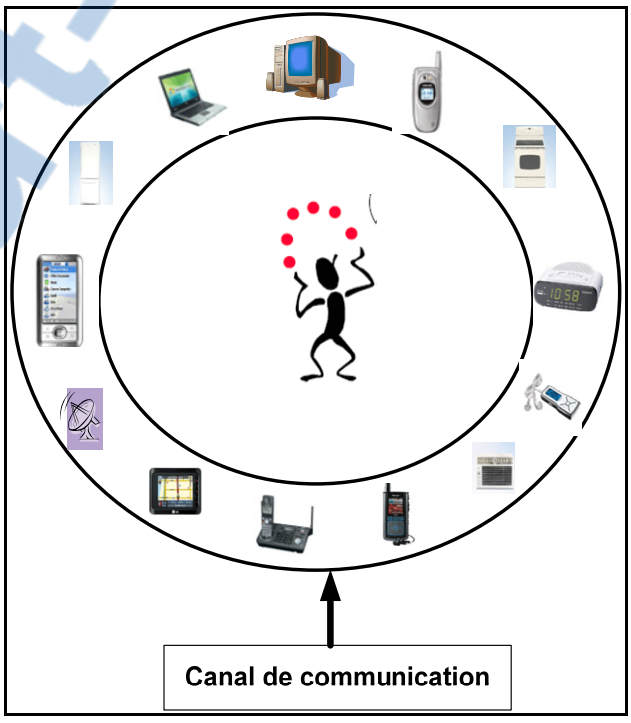

Figure 1.2 Ubiquité des équipements informatiques.

L'informatique diffuse est un domaine particulier de l'informatique résultant de la convergence des travaux existant dans les domaines de l'informatique mobile et des systèmes distribués. La Figure 1.3, donne un aperçu des apports des systèmes distribués, de l'informatique mobile ainsi que les avancées nécessaires à la réalisation d'un SID. Les systèmes distribués constituent la rencontre entre les ordinateurs personnels et les réseaux locaux. Ils permettent le partage des capacités et des ressources à travers un réseau et une infrastructure de communication. Cela constitue la première étape de l'informatique diffuse par l'introduction de l'omniprésence de l'information (exemple le Web). L'informatique mobile est le résultat de l'intégration de la technologie cellulaire et du web pour donner la possibilité aux utilisateurs d'accéder à l'information à n'importe quel endroit et en utilisant différents équipements de petites tailles et de faibles coûts. C'est en quelque sorte la deuxième étape vers l'informatique diffuse « le n'importe où, le n'importe quand » (Satyanarayanan, 2001).

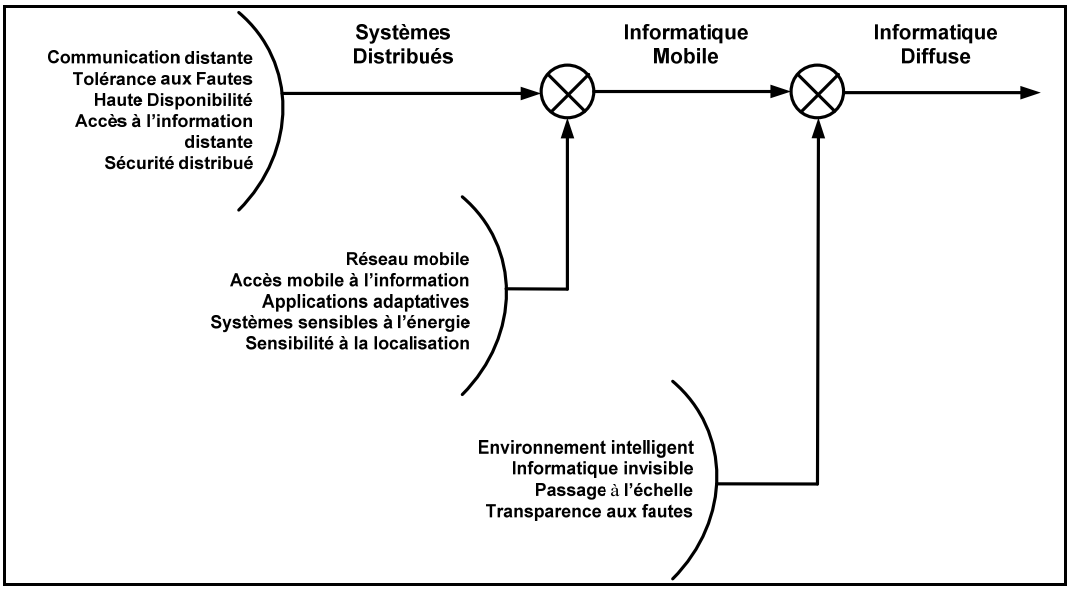

Figure 1.3 Évolution vers l'informatique diffuse.
Adaptée de Satyanarayanan (2001, p. 11)

Contrairement à l'informatique traditionnelle qui suppose qu'un utilisateur effectue une tâche définie dans un environnement déterminé, l'informatique diffuse repose sur la connaissance du contexte de l'usager pour lui délivrer le service approprié au moment opportun. Ainsi, le

badge actif (ActiveBadge) (Want et al., 1992b), inaugure l'une des premières applications de l'informatique diffuse : elle permet l'accès par identification à certains bureaux du PARC (Palo Alto Research Center / Centre de recherches de Xerox) et localise un usager afin de lui transférer ses appels téléphoniques dans le bureau où il se trouve. Depuis, les applications basées sur la géolocalisation se sont multipliées, et le guidage routier par GPS (Global Positioning System / Système de guidage par satellites) en est certainement l'illustration la plus visible. Le contexte ne peut cependant être circonscrit à la seule localisation d'un usager dans l'espace ; d'une manière plus générale, il doit être envisagé (Dey et Abowd, 2000) comme " l'ensemble des informations qui peuvent être utilisées pour caractériser la situation courante d'une entité, tel qu'un individu, un lieu ou un objet considéré comme ayant un rapport avec le service délivré par le système". " MyCampus " (Sadeh, Gandon et Kwon, 2005) exploite ainsi différentes données issues de l'environnement pour définir le contexte courant de l'usager. Ce système en vigueur sur le campus de Carnegie Mellon se compose de plusieurs "agents" qui remplissent des fonctions spécifiques. L'agent "concierge" indique dans quel restaurant universitaire l'étudiant peut prendre ses repas en fonction de ses préférences gastronomiques, de l'heure, de sa localisation sur le campus et des conditions météorologiques. L'agent " réunion " illustre quant à lui, un autre aspect de l'informatique ambiante : celui de l'échange direct de machines à machines ; il assiste les usagers dans le choix d'une date commune de réunion, en parcourant leurs agendas respectifs. Plus généralement, les applications ambiantes relèvent de l'assistance discrète et permanente d'un usager dans l'ensemble de ses activités courantes (Pascoe, 1998) ; en ce sens, elles dépassent le cadre actuel de l'informatique en mobilité. Pour éviter tout risque de confusion entre les termes utilisés dans le domaine de l'informatique diffuse, nous présentons les définitions suivantes qui le caractérisent (Laforest et Mouël, 2005/2006; Saha et Mukherjee, 2003; Weiser, 1993) :

- **ubiquitaire** : Accessible de n' importe où ;
- **mobile** : Qui intègre les terminaux mobiles ;
- **sensible au contexte** : Qui prend en compte le contexte d'exécution ;
- **pervasif** : Qui associe ubiquité, mobilité et sensibilité au contexte ;
- **ambiante** : Qui est intégré dans les objets quotidiens.

La Figure 1.4 illustre les composants d'un SID:

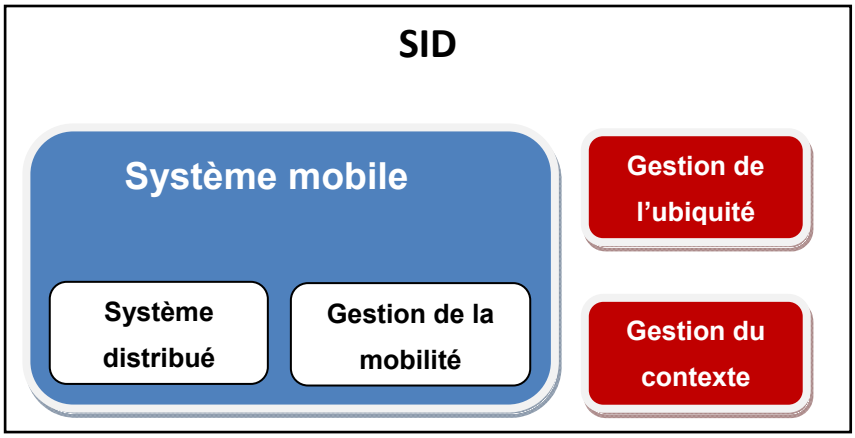

Figure 1.4 Les composants d'un système informatique diffus.
Adaptée de Saha et Mukherjee (2003, p. 26)

1.2 Exemple de scénario

Pour rendre clairs les concepts fondamentaux des systèmes diffus, nous présentons dans la section suivante un simple scénario d'un système diffus que nous supposons être installé dans le pavillon A de l'École de technologie supérieure. Ensuite, nous procédons à une analyse de ce scénario pour expliquer et mettre en évidence le fonctionnement de ces systèmes ainsi que leurs composants.

1.2.1 Scénario

En entrant au pavillon A de l'École de technologie supérieure pour assister à un cours qui démarrera dans dix minutes et qui durera trois heures. TOTO reçoit un appel sur son téléphone cellulaire de la part de son directeur de thèse pour lui rappeler qu'aujourd'hui (et avant deux heures) est la date limite pour envoyer trois rapports (documents volumineux sauvegardés dans son ordinateur portatif) à son codirecteur de thèse qui se trouve en France pour finaliser la procédure d'accueil pour l'été prochain. Tout calcul fait, TOTO se rend

compte qu'il est dans l'obligation de faire cette tâche avant d'assister au cours et sans s'absenter. Il se dirige alors vers la cafétéria et s'authentifie pour l'accès au réseau sans fil en utilisant son ordinateur portable. Il ouvre sa boîte de courriel et tape le message qui accompagne les rapports. Lors de la jointure de son premier document, un message apparaît sur son écran lui signalant que d'après la vitesse actuelle du réseau, l'expédition sera trop lente parce que plusieurs étudiants à la cafétéria naviguent sur internet. Le système remarque que le débit à la bibliothèque est très bon et qu'il y a peu d'étudiants qui utilisent internet, alors il propose à TOTO de se déplacer vers la bibliothèque qui se trouve à deux minutes de la cafétéria. TOTO accepte la proposition et se dirige vers la bibliothèque. En entrant dans celle-ci, son téléphone cellulaire se met automatiquement en mode vibreur pour respecter les règles du lieu. TOTO procède vite à l'attache des trois documents à son message. Au cours de cette opération, il reçoit une alerte SMS qui s'affiche automatiquement cette fois sur son téléphone cellulaire indiquant qu'il lui reste trois minutes pour le prochain cours. TOTO termine sa tâche et quitte le lieu en route vers la salle de cours.

1.2.2 Analyse du scénario

D'après le scénario, on peut facilement dégager les caractéristiques fondamentales d'un SID. Il offre :

- un service transparent sur un réseau, en particulier sur internet pour différentes tâches : naviguer ou expédier un courriel (mais dans le cas général ce n'est pas nécessaire que ce soit internet) ;
- des équipements connectés par différents moyens (WiFi, Bluetooth, …) ;
- système qui surveille le contexte et les paramètres du service qu'il offre aux utilisateurs.

Le système doit s'adapter aux différents contextes pour aider l'utilisateur dans ses tâches et lui permettre de se concentrer sur ces tâches principales sans se préoccuper des moyens pour les accomplir ; c'est le cas dans notre scénario :

1. Pour offrir un meilleur service à TOTO, le système détecte que la vitesse de la connexion internet est trop faible et propose une solution. La qualité de la communication ici est un élément important pour le choix de service, ainsi que l'activité (expédition d'un courriel contenant des documents volumineux) ;
2. Le type de service dépend dans notre scénario de la localisation de l'utilisateur (signalisation des appels pour le téléphone cellulaire avec mode sonnerie ou mode vibreur) ainsi que le service lui-même (alerte par bip sonore ou par SMS) ;
3. La communication entre les équipements est aussi présente dans notre scénario. À l'entrée dans la bibliothèque, le cellulaire est mis automatiquement en mode vibreur (ou le cellulaire est équipé d'un système GPS et il est programmé pour changer de mode dans la région de la bibliothèque ou le système de surveillance de la bibliothèque envoie des requêtes aux cellulaires à l'entrée dans la bibliothèque pour changer de mode et ces derniers sont programmés pour accepter ces requêtes) ;
4. De même, la communication entre le cellulaire et l'ordinateur portatif (le cellulaire en mode vibreur à l'intérieur de la bibliothèque est susceptible d'alerter TOTO du temps restant pour le prochain cours. Il communique alors avec son ordinateur portatif pour voir s'il est en train de l'utiliser. Si c'est le cas il envoie une alerte SMS) ;
5. La stratégie générale de découverte d'activité utilisée par les systèmes informatiques diffus consiste à la lecture d'une base de données (le cas de l'alerte du cellulaire pour indiquer le temps restant pour le prochain cours. L'emploi du temps de TOTO est inscrit dans l'agenda interne du cellulaire qui constitue une source d'informations).

1.3 Avantages et inconvénients de l'informatique diffuse

En ce qui a trait à l'invasion du domaine récent de l'informatique diffuse dans notre vie, plusieurs chercheurs ont commencé à évaluer les impacts, les avantages et les inconvénients de cette dernière (*Bellucci*, *Hilty* et *Bütschi*, 2002). Hormis des applications plus pratiques, l'informatique diffuse annonce une optimisation de la communication sur le plan économique, et promet des possibilités de réduction de la charge environnementale concernant la consommation de matières et d'énergie, ainsi que des améliorations sur le plan

médical (comme par exemple la mise en place des systèmes de soins médicaux diffus). À ces avantages espérés s'opposent des inconvénients éventuels : l'emploi répandu d'appareils connectés sans fil très diffusés sur le plan territorial et un accroissement des champs d'interférences électromagnétiques ; dans le domaine de la santé, multiplication possible des allergies ; dans le milieu environnemental, augmentation éventuelle de la consommation d'énergie ; difficulté d'élimination de la ferraille électronique qui n'est pas encore résolue.

CHAPITRE 2

CONTEXTE ET SENSIBILITÉ AU CONTEXTE

La notion du contexte désigne en général l'ensemble des informations qui entourent une activité et des informations supplémentaires sur son environnement. Dans leurs interactions, les humains utilisent et tiennent compte de leur contexte d'une manière implicite pour mieux se comprendre et changer leurs comportements. Par exemple, dans une salle de conférence regroupant un grand nombre d'étudiants, un enseignant doit parler à haute voix ou utiliser un microphone pour se faire entendre de tout le monde. Malheureusement dans une interaction entre un humain et un système informatique (classique), le système informatique n'a pas généralement cette aptitude à tirer le plein avantage du contexte qui entoure l'interaction. Pour tirer profit des énormes possibilités qu'offrent les informations environnantes sur leurs fonctionnements, les systèmes diffus ont cette caractéristique d'être sensible et de s'adapter au contexte. Ceci nécessite alors une bonne compréhension de ce terme pour son utilisation efficace.

2.1 Définitions générales

Le mot contexte est un ancien mot dans le domaine de l'informatique, en particulier dans les systèmes d'exploitation. Il caractérise l'ensemble minimal d'informations sur une tâche en exécution (processus) et permet de retourner à l'exécution de cette tâche après l'occurrence d'une interruption et l'exécution du programme de traitement de cette dernière. La définition de ce terme reste cependant vague du fait de son utilisation dans plusieurs domaines.

Avant de présenter les définitions du contexte proposées dans le domaine de l'informatique diffuse, nous allons donner les définitions fournies par quelques dictionnaires pour le mettre dans un cadre général. Parmi les dictionnaires de référence, citons :

- (Robert, 1991) : « ensemble du texte qui entoure un élément de la langue (mot, phrase, fragment d'énoncé) et dont dépend son sens, sa valeur » ou encore « ensemble des circonstances dans lesquelles s'insère un fait ».
- (Encyclopédie Larousse) « ensemble des conditions naturelles, sociales, culturelles dans lesquelles se situe un énoncé, un discours ». Ou encore : « ensemble des circonstances dans lesquelles se produit un événement, se situe une action ».
- (Hachette Multimédia) « ensemble des éléments qui entourent un fait et permettent de le comprendre ».
- (Grand Dictionnaire : Office québécois de la langue française) « Énoncé dans lequel figure le terme étudié » ou encore « ensemble d'un texte précédant ou suivant un mot, une phrase, un passage qui éclaire particulièrement la pensée d'un auteur ». Et, si l'on parle d'informatique : « Ensemble d'informations concernant l'action du stylet, en rapport principalement avec sa localisation à l'écran, qui permet au système d'exploitation de l'ordinateur à stylet de différencier les commandes et l'entrée des données, et de fonctionner en conséquence ».

D'après ces définitions, un contexte est un ensemble d'éléments ou de circonstances qu'on peut assimiler à des informations qui entourent, dépend ou se situe une action et qui permettent de comprendre une chose dont dépend une valeur. C'est donc l'ensemble d'informations utiles pour interpréter quelque chose.

Une étude sur la sensibilité au contexte dans l'informatique mobile réalisée par (Chen et Kotz, 2000) a permis de montrer que les définitions existantes du contexte sont des définitions générales, vagues et n'aident pas beaucoup à comprendre ce concept dans un environnement informatique. Dans la suite de ce rapport, nous nous intéresserons au contexte dans le domaine de l'informatique diffuse.

2.2 Définitions du contexte dans un système informatique

2.2.1 Définitions antérieures

Le contexte considéré comme une information sur l'environnement d'un système informatique, ou bien comme conditions qui déterminent un événement nous semble sans limite apparente. En effet, qu'obtiendrons-nous si nous envisageons de décrire dans le moindre détail les composants d'un système informatique ? De plus, il peut sembler ambitieux de vouloir décrire l'ensemble de conditions qui régissent le déclenchement d'un événement donné. C'est pour ces raisons et bien d'autres que des chercheurs, non satisfaits des définitions générales, ont essayé de définir ce terme pour permettre son utilisation dans leurs recherches. Pour faire suite à notre recherche, nous allons présenter dans l'ordre chronologique d'apparition une liste non exhaustive des définitions du contexte dans le domaine de l'informatique.

(Schilit, Adams et Want, 1994) ont considéré que le contexte possède trois aspects importants qui consistent en des réponses aux questions suivantes : Où es-tu ? Avec qui es-tu ? De quelles ressources disposes-tu à proximité ?

(Schilit et Theimer, 1994) ont défini le contexte comme la localisation, la description de personnes et d'objets dans l'entourage et les changements à ces objets.

(Brown, 1995) a défini le contexte comme : « les éléments de l'environnement d'un utilisateur dont l'ordinateur à connaissance ».

(Brown, Bovey et Chen, 1997) ont proposé un ensemble d'éléments extensibles pour caractériser le contexte dont les éléments de base sont : la localisation, l'ensemble des objets dont l'utilisateur a besoin, le temps et l'orientation spatiale (direction).

(Ryan, Pascoe et Morse, 1997) : « les éléments du contexte sont : la localisation de l'utilisateur, l'environnement, l'identité et le temps ».

(Ward et Hopper, 1997) : « les états des environnements possibles de l'application ».

(Pascoe, 1998) : « ensemble d'états physiques et conceptuels ayant un intérêt pour une entité particulière ».

(Schmidt et al., 1999) : « connaissances à propos de l'utilisateur et les états des équipements, l'entourage, la situation et la localisation ».

(Brézillon et Pomerol, 1999) : « tout ce que n'intervient pas explicitement dans la résolution d'un problème mais le contraint ».

(Chen et Kotz, 2000) : « ensemble des états environnementaux et paramètres qui déterminent le comportement d'une application ou dans lequel un événement de l'application se déroule et ayant un intérêt pour l'utilisateur ».

(Dey, 2001) : « toute information qui peut être utilisée pour caractériser la situation d'une entité. Toute entité est une personne, ou un objet qui est considéré significatif à l'interaction entre l'utilisateur et l'application, incluant l'utilisateur et l'application eux-mêmes ».

(Henricksen, Indulska et Rakotonirainy, 2002) : « la circonstance ou la situation dans laquelle une tâche informatique se déroule ».

Malgré le grand nombre de définitions existantes et les similarités (la plupart font références à la localisation et l'environnement), le mot contexte reste toujours général. Deux techniques sont utilisées par les chercheurs pour la définition du contexte : l'une est basée sur l'énumération des exemples du contexte et l'autre fait plutôt des tentatives en vue de formaliser le terme. L'importance du contexte dans le domaine de l'interaction personne-

machine et les systèmes mobiles a généré des définitions centrées sur l'utilisateur et d'autres sur l'application. Une analyse faite par (Brezillon et al., 2004) concernant les définitions du terme contexte les a conduits à conclure que la plupart des définitions sont des réponses aux questions suivantes:

- **qui ?** identité de l'utilisateur courant et d'autres personnes présentes dans l'environnement ;
- **quoi ?** percevoir et interpréter l'activité de l'usager ;
- **où ?** localisation de l'usager, ou d'un événement du système ;
- **quand ?** repère temporel d'une activité, indexation temporelle d'un événement, temps écoulé de la présence d'un sujet à un point donné ;
- **pourquoi ?** il s'agit de comprendre la raison d'être de l'activité ;
- **comment ?** la manière de déroulement de l'activité.

Les réponses aux questions citées ci-dessus peuvent engendrer un grand ensemble d'informations dont une grande partie est inutile. Cela requiert également un plus grand effort pour la gestion de ces informations.

2.2.2 Notre définition orientée service du contexte

L'objectif général d'un SID est de fournir proactivement des services bien adaptés aux utilisateurs et aux applications selon le contexte global de ces derniers et sans intervention explicite des utilisateurs. Cette adaptation des services peut être réalisée de deux façons : déclenchement automatique d'un service selon le contexte ou changement de la qualité d'un service fournis selon le contexte (le service sera fourni sous une autre forme) à cause du changement de la valeur d'un ensemble d'informations (Figure 2.1). Cela nous conduit à faire l'abstraction de la notion du contexte et de le voir du point de vue des services offerts par un SID, ce qui nous permet de le définir comme suit :

« Toute information dont le changement de sa valeur déclenche un service ou change la qualité (forme) d'un service » (Miraoui et Tadj, 2007b).

Cette définition est plus abstraite que les autres définitions dans le sens ou elle n'énumère pas des exemples du contexte et limite l'ensemble des informations contextuelles à celles qui sont reliées aux services (objectif d'un système diffus). Elle ne prend donc pas en compte les autres informations qui peuvent caractériser le contexte, mais ne jouent pas un rôle crucial pour l'adaptation des services.

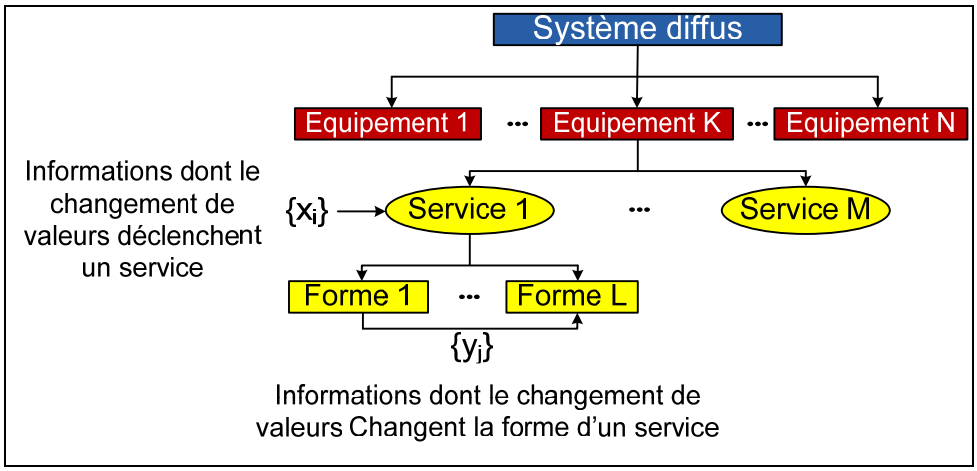

Figure 2.1 Les composants d'un système diffus.

Pour mettre en évidence la consistance de cette définition, nous allons présenter trois scénarios et nous allons montrer comment il est plus commode de caractériser les informations contextuelles.

Scénario 1

L'afficheur d'un téléphone cellulaire augmente sa luminosité lorsqu'il se trouve dans un milieu sombre et le réduit lorsqu'il se trouve dans un milieu éclairé (Figure 2.2). Dans ce scénario on trouve :

- **service** : affichage ;
- **la forme (ou qualité)** : luminosité augmentée ou réduite ;
- **information du contexte** : éclairage dans le milieu où se trouve le cellulaire (le changement de cette information change la qualité de l'affichage (Tableau 2.1)).

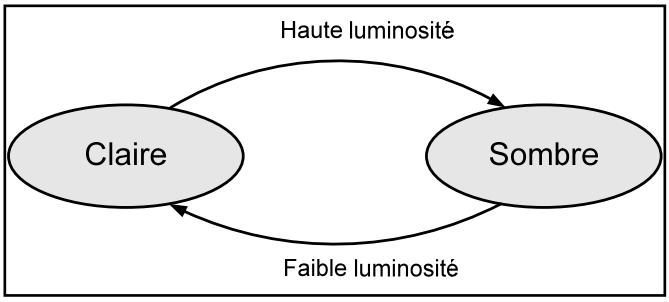

Figure 2.2 Diagramme d'états du scénario 1.

Tableau 2.1 Informations contextuelles du scénario 1

Équipement	**Service**	**Forme**	**Informations contextuelles**	
			Déclenchement	**Changement de forme**
	Affichage	Sombre	___	*Luminosité de l'environnement*
		Claire		

Scénario 2

Un ordinateur fixe est équipé d'une connexion internet par câble et une autre sans fil (WiFi). Normalement, on utilise pour internet la connexion par câble mais lorsque le débit devient faible le système bascule automatiquement vers la connexion sans fil (
Figure 2.3). Dans ce scénario il y a :

- **service** : connexion à internet ;
- **la forme (ou qualité)** : connexion avec câble ou sans fil ;

- **information du contexte** : débit d'une connexion internet (le changement de cette valeur provoque le changement de la qualité du service (Tableau 2.2)).

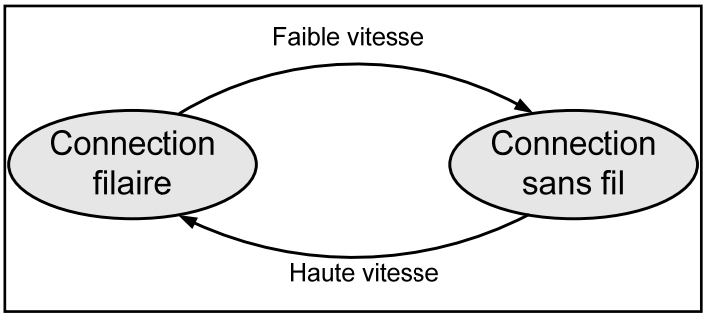

Figure 2.3 Diagramme d'états du Scénario 2.

Tableau 2.2 Informations contextuelles du Scénario 2

Matériel	**Service**	**Forme**	**Informations contextuelles**	
			Déclenchement	**Changement de forme**
	Connexion internet	Câblée	—	*Vitesse de la connexion câblée*
		Sans fil		

Scénario 3

Un utilisateur a inscrit dans son calendrier d'activités de son téléphone cellulaire une tâche à faire à la date X et à l'heure Y. À ces coordonnés temporelles, le cellulaire a été mis par l'utilisateur en mode silencieux (volume de sonnerie = 0) parce que ce dernier se trouve dans une réunion. Le téléphone cellulaire communique alors avec l'ordinateur portable de l'utilisateur pour voir si l'utilisateur est en train de l'utiliser, si c'est le cas il lui envoie un message (SMS) qui contient l'activité à faire ; ce message s'affiche directement sur l'écran de l'ordinateur portable (Figure 2.4). Dans ce scénario il y a :

- **service** : rappel électronique ;

- **la forme (ou qualité)** : rappel par sonnerie d'un téléphone cellulaire ou par affichage sur écran d'un ordinateur portable ;
- **déclenchement d'un service** : envoi d'un SMS par le cellulaire à l'ordinateur portable ;
- **informations du contexte** : date et heure (le changement de ces valeurs déclenche le service de rappel) et le volume de sonnerie du téléphone cellulaire (si volume = 0 alors le service sera fourni sous une autre forme (Tableau 2.3)).

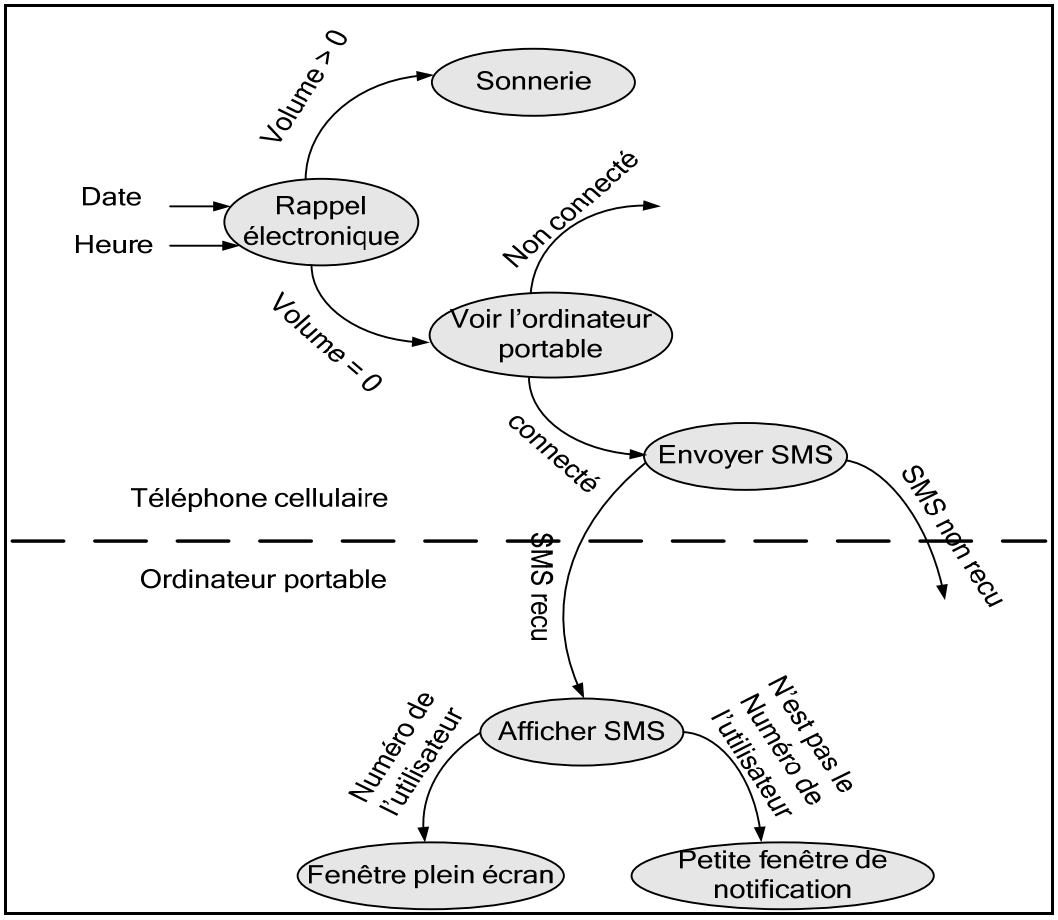

Figure 2.4 Diagramme d'états du scénario 3.

D'après ces scénarios, il est évident que cette approche facilite énormément la caractérisation des informations contextuelles en se limitant seulement aux informations nécessaires pour l'adaptation des services.

Tableau 2.3 Informations contextuelles du scénario 3

Matériel	Service	Forme	Informations contextuelles	
			Déclenchement	**Changement de forme**
	Rappel électronique	Sonnerie	*Date, Heure*	*Volume de sonnerie*
		Autres		
	Connecter à l'ordinateur portable	Sans fil	*Volume de sonnerie*	___
	Envoyer SMS à l'ordinateur portable	Sans fil	*Ordinateur portable connecté*	___
	Afficher SMS	Fenêtre plein écran	*SMS reçu*	*Numéro expéditeur*
		Petite fenêtre de notification		

2.3 Définitions antérieures de la sensibilité au contexte

La caractéristique principale d'un environnement informatique diffus est le changement dynamique de leurs environnements ou bien plus précisément leurs contextes. Pour mieux aider l'utilisateur dans ces tâches quotidiennes, les systèmes informatiques diffus doivent tenir compte du contexte global et adapter leurs services aux utilisateurs selon ce dernier. Cette aptitude est connue sous le nom de sensibilité au contexte. Dans la partie suivante, nous allons faire un bref survol des définitions existantes de ce terme comme dans le cas du contexte.

Le terme sensibilité au contexte a été évoqué pour la première fois par (Schilit, Adams et Want, 1994) dans leurs travaux sur un système de localisation. Ils ont défini la sensibilité au

contexte comme l'aptitude d'une application à s'adapter au contexte de son exécution selon : la localisation, l'ensemble des personnes à proximité, les machines, les équipements accessibles, de même que les changements de ces objets dans le temps. (Brown, 1995) a défini la sensibilité au contexte dans son travail relatifs à un guide touristique comme toute application qui prend en compte le contexte de l'utilisateur. (Dey, 2001) l'a défini comme un système qui utilise le contexte pour fournir des informations et/ou des services pertinents à l'utilisateur. Selon lui la pertinence dépend de la tâche de l'utilisateur. (Chen et Kotz, 2000) ont donné deux définitions. Il y a selon eux :

1. Sensibilité active au contexte : une application qui s'adapte automatiquement au contexte découvert par le changement du comportement de l'application ;
2. Sensibilité passive au contexte : une application qui présente le nouveau contexte ou celui mis-à-jour à un utilisateur intéressé ou rendre le contexte persistant pour une utilisation ultérieure.

Ces définitions et d'autres sont en effet très proches et s'articulent autour de l'aptitude d'un équipement informatique à changer ou à adapter son comportement en se basant sur le contexte de l'utilisateur en premier lieu, en deuxième lieu de l'environnement et de l'application. De plus, ces définitions ne sont pas suffisamment générales pour caractériser un système sensible au contexte et ne sont pas suffisamment abstraites et restent liées à un domaine particulier d'application.

2.3.1 Notre définition de la sensibilité au contexte

En se basant aussi sur l'approche orientée service, nous donnons cette définition d'un système sensible au contexte :

«Un système est dit sensible au contexte s'il peut changer automatiquement ces formes des services ou déclencher un service comme réponse au changement de la valeur d'une

information ou d'un ensemble d'informations qui caractérisent le service. » (Miraoui et Tadj, 2007b).

Cette définition décrit mieux un système sensible au contexte puisqu'elle explique la sensibilité en termes de réaction du système aux changements du contexte par le déclenchement d'un service ou le changement des formes de services.

2.4 Utilisation du contexte

Le contexte est pris en compte dans différent domaines informatiques incluant le traitement du langage naturel, l'apprentissage automatique, la vision par ordinateur, l'extraction de l'information, l'informatique diffuse et même la sécurité informatique. Tout comme dans l'interaction entre humains, le but de la prise en compte du contexte est de renforcer l'adaptabilité et le support à la décision du système. L'informatique diffuse est très intimement liée au contexte du fait de l'hétérogénéité et de l'ubiquité des entités communicantes de ce type d'environnement. Ces deux aspects de l'infrastructure informatique diffuse requièrent l'adaptabilité des services fournis et des médias des utilisateurs en fonction du lieu, de l'activité, c'est à dire du contexte. On parle ainsi des systèmes informatiques diffus en disant qu'ils sont sensibles au contexte (en anglais context-aware) parce qu'ils sont capables d'utiliser le contexte d'une entité afin de modifier leurs fonctionnements pour offrir des meilleurs services aux utilisateurs. (Chalmers, 2004) identifie cinq principales utilisations du contexte dans les systèmes informatiques comme suit :

1. Senseur du contexte : où le contexte est capté et les informations décrivant le contexte courant (température, localisation, …) sont présentés à l'utilisateur ;
2. À associer le contexte aux données, encore appelé augmentation contextuelle. Par exemple : les enregistrements sur les objets inspectés peuvent être associés à leur localisation ; les notes d'une réunion peuvent être associées aux personnes assistant à la réunion et le lieu où elle s'est déroulée ;
3. Permettre la découverte de ressources contextuelles, par exemple, faire en sorte que l'impression d'un document ait lieu sur l'imprimante la plus proche ;

4. Dans le cas des évènements déclenchés par le contexte pour déclencher les actions telles que le chargement de données cartographiques à l'entrée dans une région ;
5. Médiation contextuelle : elle consiste à utiliser le contexte pour modifier un service. Par exemple pour décrire les limites et les préférences dans une large variété de données offertes, et ainsi pouvoir afficher les plus appropriées.

Selon (Dey et Abowd, 2000), les chercheurs s'accordent pour dire que les premiers travaux de recherche dans le domaine des systèmes informatiques sensibles au contexte concernent le système de badges actifs d'Olivetti (Want et al., 1992a). Le badge actif a été développé entre 1989 et 1992. Son but est de fournir un moyen de localisation direct du personnel dans un immeuble pour acheminer automatiquement les appels téléphoniques. Pour ce faire, chaque personne porte un badge qui transmet périodiquement des signaux vers un système centralisé de localisation. Le plus grand système de badges actifs actuellement utilisé se trouve au laboratoire d'informatique de l'université Cambridge, comptant plus de 200 badges et 300 capteurs utilisés quotidiennement.

Ces derniers systèmes ne sont qu'un exemple des multiples systèmes informatiques qui font un usage intensif des informations contextuelles pour améliorer le quotidien des usagers pour lesquels ces systèmes sont destinés. (Dey, 2001), puis (Chen et Kotz, 2000) ont recensés des travaux de recherche relatifs au contexte, en mettant l'accent sur les applications, les informations contextuelles utilisées, et la manière dont ils s'en servent. Il ressort de ces deux revues de littérature que la plupart des systèmes informatiques diffus font appel surtout à la localisation de l'utilisateur comme information contextuelle. Dans de rares cas, un index temporel est utilisé ainsi que la localisation des objets avoisinants. Cela pourrait s'expliquer par les difficultés associées au prélèvement des informations contextuelles ainsi qu'à son traitement.

2.5 Sources d'informations

Pour pouvoir assurer sa fonction principale qui consiste à l'adaptation selon le contexte, un SID doit faire la collection des informations contextuelles de différentes sources d'informations ayant des caractéristiques et des propriétés différentes. Ces sources d'informations sont assez nombreuses à cause de l'ubiquité et dépendent du domaine d'application. Les méthodes d'acquisition de l'information contextuelle peuvent être classées en catégories. Selon (Henricksen, Indulska et Rakotonirainy, 2002), il y a trois méthodes pour l'acquisition d'information pour les systèmes de localisation. Ces trois méthodes sont :

1. Capteurs physiques qui peuvent être intégrés ou non dans d'autres outils (cellulaire, PDA, …) et permettant de capturer plusieurs types d'information physique telles que la température, la localisation géographique, niveau de bruit, lumière, etc. ;
2. Les capteurs virtuels qui permettent d'extraire les informations contextuelles à partir des espaces virtuels tels que les programmes, systèmes d'exploitation, réseau, etc. (exemple : l'information sur la localisation peut être extraite à partir d'un calendrier électronique et d'un journal de connexion au réseau) ;
3. Les capteurs logiques qui utilisent les informations des capteurs physiques et virtuels pour déduire d'autres informations.

De même (Mostefaoui, Pasquier-Rocha et Brezillon, 2004) ont classifié trois méthodes pour l'acquisition du contexte. Ces méthodes sont :

- le contexte capturé par les capteurs physiques (température, pression, lumière, …) ;
- contexte dérivé : acquisition ah hoc (temps, date) ;
- contexte fourni explicitement tel que le profil utilisateur, informations sur une machine dans un réseau, etc.

2.6 Catégorisation du contexte

La sensibilité au contexte exige que les informations contextuelles soient collectées et présentées à l'application d'adaptation. Vu l'hétérogénéité, la diversité et la qualité de ces informations, il est désirable de faire une classification ou bien une catégorisation pour faciliter l'opération d'adaptation. Dans ce domaine précis, plusieurs chercheurs ont proposé des catégorisations selon différentes approches. (Schilit, Adams et Want, 1994) et (Dey, 2001) ont catégorisé le contexte en deux classes : le contexte primaire qui contient les informations sur la localisation, l'identité, le temps et l'activité (statut) ; le contexte secondaire qui peut être déduit de ce dernier (exemple : de la localisation, on peut déduire les personnes à proximité). (Chen et Kotz, 2000) ont proposé deux catégories : le contexte actif qui influence le comportement d'une application et le contexte passif qui est nécessaire mais pas critique pour l'application. (Petrelli et al., 2000) ont fait connaître le contexte matériel (localisation, machine, plateforme existante) et le contexte social (les aspects sociaux comme la relation entre les individus). (Gwizdka, 2000) a présenté deux catégories : le contexte interne contenant l'état de l'utilisateur et le contexte externe englobant l'état de l'environnement. (Hofer et al., 2002) ont élaboré une catégorisation en deux classes : le contexte physique qui peut être mesuré par les capteurs physiques et le contexte logique qui contient les informations sur l'interaction (l'état émotionnel de l'utilisateur, ses buts, etc.) (Razzaque, Dobson et Nixon, 2005) ont proposé comme suit une catégorisation en six classes :

1. **Contexte utilisateur** : il permet d'obtenir les informations sur les utilisateurs du système informatique. Le profil de l'utilisateur par exemple en fait partie. Il peut contenir des informations sur son identification, ses relations avec les autres usagers, la liste de ses tâches, et d'autres ;
2. **Contexte physique** : il offre la possibilité d'intégrer des informations relatives à l'environnement physique, telles que la localisation, l'humidité, la température, le niveau de bruit, etc. ;

3. **Contexte du réseau** : il fournit aussi des informations de l'environnement, mais ces dernières se rapportent essentiellement au réseau informatique. Exemple : connectivité, bande passante, protocole, etc. ;
4. **Contexte d'activité** : répertorie les évènements qui se sont déroulés dans l'environnement ainsi que leur estampille temporelle. Exemple d'évènements : entrée d'une personne, tempête de neige, etc. ;
5. **Contexte matériel** : il permet d'identifier les appareils de l'environnement qui peuvent être utilisés. Il inclut par exemple le profil et les activités des dispositifs de l'environnement (identification, localisation, niveau de la batterie, etc.) ;
6. **Contexte de service** : il informe sur ce qui peut être obtenu. Par exemple : les informations relatives aux fonctionnalités que le système peut offrir.

Cette dernière catégorisation a l'avantage de couvrir les éléments considérés par les premières catégorisations qui ont été faites. Toutefois, si nous nous référons aux six questions proposées par (Brezillon et al., 2004) et qui apporteraient les informations minimales relative au contexte. La question « Pourquoi ? » est clairement ignorée dans cette catégorisation.

Une autre catégorisation proposée par les mêmes auteurs mais fondée sur les valeurs que peut prendre une information contextuelle. Il y a :

1. **Le contexte continu** : dans cette catégorie, les valeurs varient continuellement. Un élément d'un contexte continu est fonction de différents paramètres et sa valeur est calculée en se servant d'une formule. Exemple : les informations GPS ;
2. **Le contexte énumératif** : les valeurs d'un composant du contexte sont un ensemble discret de valeurs ;
3. **Information contextuelle d'état** : les éléments de cette catégorie peuvent prendre deux valeurs opposées. Par exemple : la lumière dans une pièce peut être allumée ou éteinte. Les valeurs de ces éléments sont obtenues à partir d'un calcul de prédicat ;
4. **Le contexte descriptif** : il est basé sur les descriptions des éléments du contexte.

Il existe bien d'autres catégorisations qu'ont été proposées que celles qui ont été présentées ici, mais aucune d'elles ne se veut exhaustive. De nouveaux regroupements seront effectués à mesure que de nouvelles caractéristiques des informations contextuelles seront découvertes. Il n'en demeure pas moins que ces efforts de classification sont louables et permettent aux développeurs de l'informatique diffuse de manipuler plus efficacement les informations contextuelles.

2.6.1 Notre catégorisation du contexte

En adoptant la même approche que précédemment (fondée sur le service), nous pouvons proposer une catégorisation des informations contextuelles en deux classes (Miraoui et Tadj, 2007b). Elles comprennent :

1. Les informations de déclenchement d'un service qui regroupent les informations dont le changement de valeurs provoque le déclenchement automatique des services fournis par un équipement ;
2. Les informations de changement de forme d'un service qui regroupent les informations dont le changement de valeurs provoque le changement de forme d'un service (qualité).

Cette catégorisation comporte deux avantages principaux : elle est simple parce qu'elle a seulement deux classes ; elle est complète parce qu'elle couvre tous les aspects du contexte, en particulier les six questions de (Brezillon et al., 2004).

2.7 Qualité de l'information contextuelle

Les particularités d'un SID, c'est qu'il possède un environnement très dynamique en plus de l'existence de plusieurs équipements mobiles hétérogènes. Cela conduit les informations contextuelles qui aident à appuyer proactivement l'interaction personne-machine ou machine-machine à contenir des erreurs au niveau de la collection, de l'interprétation et de la

présentation de ces informations. Ainsi, pour assurer une adaptation fiable selon le contexte, une évaluation de la qualité de l'information contextuelle est requise.

Selon (Krause et Hochstatter, 2005), les informations contextuelles peuvent être erronées pour les raisons suivantes :

- l'information nécessaire (ou la source) n'est pas valable ;
- l'information n'est pas applicable à la situation actuelle ;
- les contraintes physiques et temporelles limitent la précision des capteurs ;
- l'information sur le profil par défaut n'est pas applicable à la situation actuelle ;
- les règles de raisonnement ne sont pas applicables dans toutes les situations (exemple : la détection d'une lumière dans une chambre ne veut pas dire toujours qu'il y a quelqu'un est présent) ;
- la possibilité d'acquisition d'informations de sources malicieuses.

(Dey et al., 2002) ainsi que d'autres chercheurs ont remarqué que le contexte capturé et celui qui a été interprété sont souvent ambigus. Le défi majeur pour le développement des systèmes sensibles au contexte robuste sera l'habilité à détecter cette ambiguïté. (Henricksen, Indulska et Rakotonirainy, 2002) ont classé l'imperfection de l'information contextuelle comme une des caractéristiques du contexte dans un SID. Cette imperfection a pour effet que l'information peut être :

- **incorrecte** : si elle échoue à refléter l'état réel du monde qu'elle modélise ;
- **inconsistante** : si elle contient des informations contradictoires ;
- **incomplète** : si des aspects du contexte ne sont pas connus.

(Razzaque, Dobson et Nixon, 2005) ont proposé une approche pour la modélisation de la qualité de l'information contextuelle. Selon ces auteurs, cette approche est suffisamment générique et consiste en un processus par étapes. (Krause et Hochstatter, 2005) ont présenté une méthode pour la représentation des aspects de la qualité du contexte intégré dans leur

modèle de représentation du contexte CMM (context metamodel). La plupart des travaux sur la qualité de l'information contextuelle ne définissent pas des méthodes formelles pour la modélisation de la qualité, mais plutôt des méthodes incomplètes et limitées à une application ou à un domaine spécifique.

CHAPITRE 3

ARCHITECTURE POUR LES SYSTÈMES SENSIBLES AU CONTEXTE

L'informatique diffuse constitue un nouveau domaine de l'informatique qui depuis son avènement, exigeait la mise en place de nouveaux outils et de nouvelles mesures (architectures, cadres de travail, intergiciels). Jusqu'alors, ces outils ne faisaient pas partie de l'informatique traditionnelle afin de supporter les particularités de tels systèmes, à savoir : la mobilité, la sensibilité et la variabilité du contexte. L'architecture d'un logiciel décrit l'organisation globale d'un système en termes de composants et de leurs interactions. Elle aide à gérer et à contrôler la complexité du développement de logiciels (Albin, 2003). C'est dans ce cadre que nous allons proposer à la suite d'une étude détaillée des architectures existantes (Miraoui, Tadj et Amar, 2008a), une architecture fondée sur le concept de service utilisant le paradigme agent.

3.1 Travaux connexes

Dans la littérature de l'informatique, nous trouvons plusieurs études (« surveys ») des architectures des systèmes sensibles au contexte, mais les plus significatives selon nous sont les suivantes : (Baldauf, Dustdar et Rosenberg, 2007) ont présenté un bon nombre d'architectures et ils se sont focalisés en particulier sur les architectures en couches. Leur étude est fondée sur la comparaison des couches de différents architectures et les mécanismes utilisés par chaque couche. Malgré la diversité des architectures étudiées, l'étude n'est pas faite selon des attributs spécifiques au domaine de l'informatique diffuse. (Kjær, 2007) a fait une étude plutôt orientée intergiciel, mais qui touche aussi quelques aspects architecturaux des systèmes étudiés. Elle consiste en la classification selon une taxonomie jugée importante par l'auteur, mais qui nous semble une description plus détaillée des couches d'une architecture classique d'un système sensible au contexte (capteurs, interprétation, raisonnement, gestion du contexte et adaptation). De même, cette étude, comme la précédente, n'est pas élaborée selon des critères spécifiques à l'informatique diffuse. (Singh

et Conway, 2006) ont décrit dans leur étude seulement quatre architectures, mais ils n'ont pas fait des comparaisons entre elles. Il ne s'agit donc que d'une simple description. (Winograd, 2000) a fait une étude spécifique aux modèles organisationnels des architectures des systèmes sensibles au contexte pour le domaine de l'interaction personne-machine. Elle ne comprend pas tous les aspects architecturaux. L'étude se limite à un seul domaine d'application. Enfin, (Henricksen et al., 2005) ont élaboré une étude limitée à cinq architectures, mais qui nous semble la seule à réaliser la comparaison des architectures selon des attributs qui touchent la spécificité de l'informatique diffuse. Cependant, cette étude ne couvre pas d'autres architectures qui nous semblent intéressantes dans l'évolution des architectures des systèmes sensibles au contexte. Les attributs de qualité utilisés sont plutôt orientés vers les systèmes distribués, bien que la plupart d'entre eux soient fondamentaux pour l'informatique diffuse.

Les premiers systèmes sensibles au contexte étaient en réalité des systèmes sensibles à la localisation ; dans ce cas cette dernière consiste en l'élément principal du contexte. Dans le prochain paragraphe, nous allons décrire brièvement les architectures d'un ensemble de systèmes qui ont marqué l'évolution des systèmes sensibles au contexte.

3.1.1 ActiveBadge (1992)

Le projet ActiveBadge (Want et al., 1992a) développé par la firme Olivetti a permis la réalisation d'une application permettant l'acheminement des appels téléphoniques à une personne selon sa localisation au téléphone le plus près de lui. Le système utilise des badges émettant des signaux infrarouges (à une fréquence bien déterminée). Ces badges sont portés par le personnel d'une entreprise et chaque badge contient l'identification du porteur. Les signaux émis par ces badges sont interceptés par des capteurs distribués dans les locaux de l'entreprise. Les signaux perçus par les capteurs sont ensuite envoyés à un serveur. Ce dernier présente à un agent (réceptionniste) les informations de localisation de personnes porteuses de badges ainsi que les locaux où elles se trouvent. Celui-ci achemine les appels reçus pour une personne au téléphone qui se trouve dans le local le plus près de la personne

appelée (cette tâche peut être automatisée). L'ActiveBadge est basé sur une architecture distribuée des capteurs (Figure 3.1) et sur une architecture en quatre couches au niveau du serveur ; ces quatre couches sont les suivantes :

- contrôle du réseau qui surveille le fonctionnement du réseau de capteurs ;
- présentation des données pour le stockage et le contrôle des informations de localisation ;
- traitement des données pour la sélection des informations intéressante lors du changement de localisation ;
- interface utilisateur pour l'affichage des informations textuelles sur le changement de position des badges.

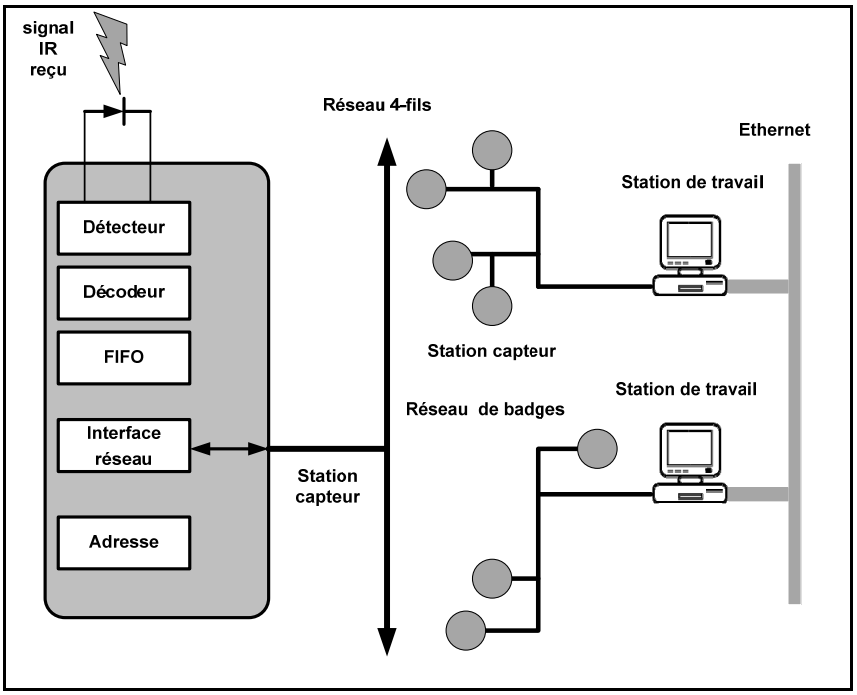

Figure 3.1 Architecture de l'ActiveBadge.
Adaptée de Want et al. (1992, p. 4)

ActiveBadge est plutôt une infrastructure matérielle pour des systèmes sensibles à la localisation et ne présente pas en réalité une architecture logicielle. Cette dernière est spécifique aux applications de localisation.

3.1.2 ParcTab (1995)

Le projet ParcTab de Xerox (Want et al., 1995) est une infrastructure qui favorise le développement des applications sensibles au contexte de localisation (localisation d'une personne, matériel qui l'entoure, personnes avoisinants, etc.). Le ParcTab est un assistant personnel digital (PDA) porté par l'usager et fonctionne comme un terminal graphique. Il est en communication infrarouge avec des émetteurs-récepteurs d'un édifice local. Ces émetteurs-récepteurs communiquent à leur tour (en infrarouge) avec des passerelles (« gateway »). Ces passerelles sont connectées à un réseau local de postes de travail au moyen d'une connexion RS-232. À chaque ParcTab correspond un agent logiciel qui contrôle la communication avec des applications qui s'exécutent sur les postes de travail du réseau local. Cela décharge les ParcTabs des opérations de traitement qui épuisent leurs ressources limitées (Figure 3.2). Le système de communication est basé sur le mécanisme d'appel de procédure à distance (RPC) entre les ParcTabs et les applications sur les postes de travail du réseau local. Cette infrastructure favorise le développement d'applications sensibles au contexte, en particulier la localisation telle que la notification des courriels pour un utilisateur selon sa localisation et les personnes avoisinants par un texte qui s'affiche par le ParcTab ou une simple notification par un signal sonore (le cas échéant de filtrer les courriels urgents lorsque l'utilisateur se trouve en conférence ou réunion). Ses auteurs envisagent aussi plusieurs autres applications sensibles au contexte se basant sur cette infrastructure telles qu'un contrôleur de programme à distance, une collaboration assistée, un accès aux informations et aux ressources selon le contexte et la possibilité d'autres applications.

Le ParcTab aussi est un système primitif sensible à la localisation fondé également sur une architecture matérielle. L'architecture logicielle est très dépendante du matériel et n'offre pas une bonne abstraction des informations du contexte.

3.1.3 Stick-e-notes (1997)

Le projet stick-e-notes (Pascoe, 1997) consiste en un cadre de travail (en anglais : Framework) pour le développement des applications sensibles au contexte principalement de localisation. Il est basé sur l'utilisation d'un ensemble de PDAs connectés à un capteur de localisation (GPS ou ActiveBadge). Ces PDAs peuvent être en communication ou non selon l'application. L'idée de base est inspirée des bouts de papiers collant (stick notes) qu'on met généralement sur les portes, tables, équipements, etc. pour se rappeler de quelque chose ou attirer l'attention de quelqu'un. Ces notes seront électroniques et écrites par l'utilisateur et attachées à des contextes bien précis (exemple : localisation) et sauvegardés dans son PDA. Ces notes sont déclenchées (affichées ou signalées) plus tard lorsque le même contexte apparaît de nouveau (exemple : l'utilisateur attache une note descriptive d'un musée lorsqu'il se trouve dans ce dernier. Chaque fois que l'utilisateur entre dans cette zone géographique, la note descriptive du musé est affichée). Les notes peuvent être de différents formats : texte, pages HTML, son, vidéo, programme à exécuter, etc.

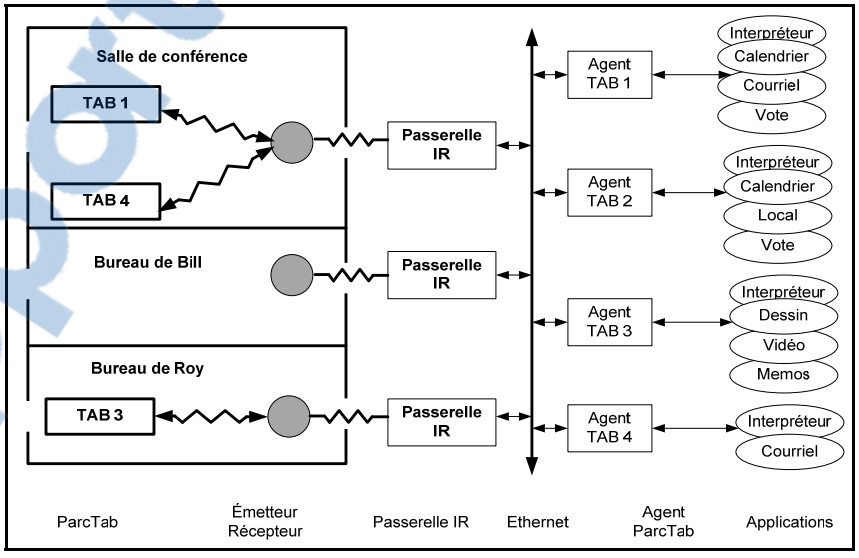

Figure 3.2 Architecture du ParcTab.
Adaptée de want et al. (1995, p. 15)

Les auteurs ont défini quatre composantes logicielles pour construire le support de stick-e-note. Les composants sont les suivantes :

- **seprepare** : permettant à l'utilisateur de préparer les notes ;
- **semanage** : offrant la possibilité de la gestion des notes ;
- **setrigger** : donnant la possibilité de déclencher des notes si leurs contextes sont satisfaisants ;
- **seshow** : permet de stocker des notes déclenchées et de les présenter à l'utilisateur.

Le contenu d'une note est décrit en SGML pour faciliter l'échange et l'extensibilité de l'application.

Le stick-e-note utilise des informations limitées pour décrire le contexte (localisation), très couplé au matériel (utilise un matériel dédié) et ne présente pas une amélioration significative au niveau de l'abstraction du contexte par rapport aux systèmes précédents.

3.1.4 Cyberguide (1997)

L'idée derrière le projet cyberguide (Abowd et al., 1997) est d'équiper l'utilisateur d'un guide touristique électronique sensible à son contexte (localisation, orientation). L'infrastructure matérielle est composée d'un ensemble d'assistants personnels digitaux (PDAs) connectés à des GPS pour détecter la position du touriste. Ces assistants personnels peuvent communiquer entre eux ou avec un réseau local par infrarouge. L'objectif est de guider un touriste dans sa visite touristique en lui offrant les sites intéressants à visiter selon la localisation et les chemins à suivre ainsi que des informations utiles selon sa position. Les composants de l'architecture logicielle sont les suivants :

1. Carte géographique de l'environnement physique que le touriste est en train de visiter avec des représentations des objets intéressants (édifices, tours, etc.) ;
2. Information (librairies) contenant l'ensemble des informations et descriptions des sites que le touriste peut visiter et ayant une symbolisation spéciale sur la carte géographique ;
3. Navigateur permettant de détecter la localisation actuelle du touriste dont le but et de lui offrir des informations sur l'environnement qui l'entoure ;

4. Messager permettant de fournir un service de livraison de messages pour le touriste. Il lui est également permis d'envoyer une question, de recevoir des messages diffusés ou des suggestions. Il peut aussi entrer en communication avec d'autres touristes.

Un autre projet GUIDE à été proposé par (Cheverest et al., 2000) ayant le même objectif que le cyberguide, mais il nous semble très similaire à ce dernier avec des différences mineures sur le plan matériel utilisé et l'accès au web.
Les deux projets cyberguide et GUIDE sont deux systèmes spécifiques aux applications de localisation. Ces deux projets n'effectuent pas des transformations de contexte brut (pas d'interprétation). Ils dépendent du matériel utilisé et ne présentent pas une architecture logicielle extensible et réutilisable.

3.1.5 CASS (2004)

L'outil CASS (Context-awareness sub-structure) (Fahy et Clarke, 2004) est un intergiciel (middleware) pour appuyer le développement des applications sensibles au contexte. Il offre une bonne abstraction des informations contextuelles et utilise un modèle orienté objet pour la représentation du contexte. Cette architecture (Figure 3.3) est basée sur un serveur contenant une base de données des informations contextuelles ainsi qu'une base de connaissances. Celle-ci permet au moteur d'inférence d'inférer d'autres informations de contexte en utilisant la technique de chaînage arrière. Les équipements mobiles sont munis de différents types de capteurs. Ceux-ci perçoivent les changements du contexte et les envoient au serveur sans aucun traitement local. La communication entre les équipements mobiles et le serveur est assurée par des connexions sans fils. Le serveur contient aussi un module pour l'interprétation du contexte perçu. Cette architecture offre une bonne modularité permettant à un utilisateur de modifier facilement les composants du serveur, en particulier le mécanisme d'inférence.

Les équipements mobiles dans CASS ne font aucun traitement local (tout se fait au niveau du serveur). Cela limite leurs autonomies, en revanche elle favorise l'extensibilité du système.

CASS offre une bonne interprétation du contexte, donc plus d'abstraction. Le module de raisonnement qui lui permet d'être plus proactif. La structure centralisée quand à elle présente une faiblesse (en cas de panne du serveur).

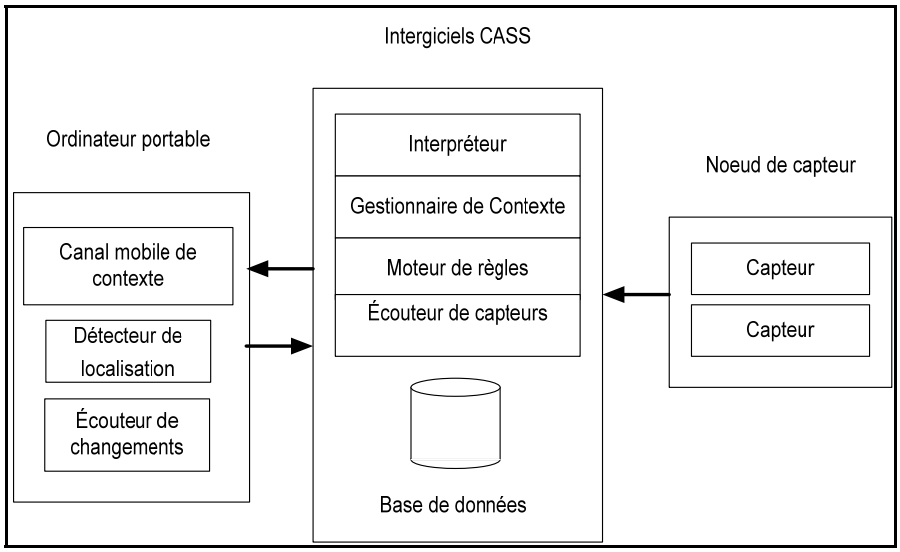

Figure 3.3 Architecture de CASS.
Adaptée de Fahy et Clarke (2004, p. 3)

3.1.6 CORTEX (2004)

(Biegel et Cahill, 2004) ont proposé un cadre de travail (Framework) pour faciliter le développement des applications mobiles sensibles au contexte qui s'appelle CORTEX (CO-operating Real-time sentTient objects: architecture and EXperimental evaluation). L'architecture est basée sur les « sentient objects » qui ont des caractéristiques avantageuses pour un environnement informatique diffus. Ces caractéristiques sont les suivantes :

1. Elles sont douées de sensation en ayant la capacité de percevoir l'état de l'environnement par des capteurs ;
2. Elles sont autonomes en ayant la capacité de fonctionner indépendamment du contrôle humain d'une manière décentralisé ;
3. Elles sont proactives en prenant des initiatives pour accomplir un but proposé.

L'architecture d'un « sentient object » (Figure 3.4) est composée de deux interfaces : capture des événements perçus par les capteurs, émission des événements pour s'adapter au contexte actuel. Cette architecture contient un module pour la fusion du contexte et son interprétation dans un haut niveau d'abstraction. Elle contient également, un module pour la représentation hiérarchique du contexte dont le but est de limiter le contexte de la situation actuelle et par la suite de limiter les actions possibles. Un moteur d'inférence pour spécifier le comportement de l'application à un contexte donné qui utilise le modèle d'exécution « événement-condition-action ». La communication entre les différents « sentient object », capteurs et actuateurs qui composent le système utilise le mécanisme basé sur l'événement qui s'établit d'une manière dynamique pendant le fonctionnement du système.

Cette architecture présente maints avantages, comme nous l'avons décrit plus haut, mais demeure une solution ad hoc pour un réseau mobile. Le système d'inférence écrit en CLIPS demande des développeurs qualifiés pour le mettre en œuvre dans une autre application, ce qui limite son utilisation.

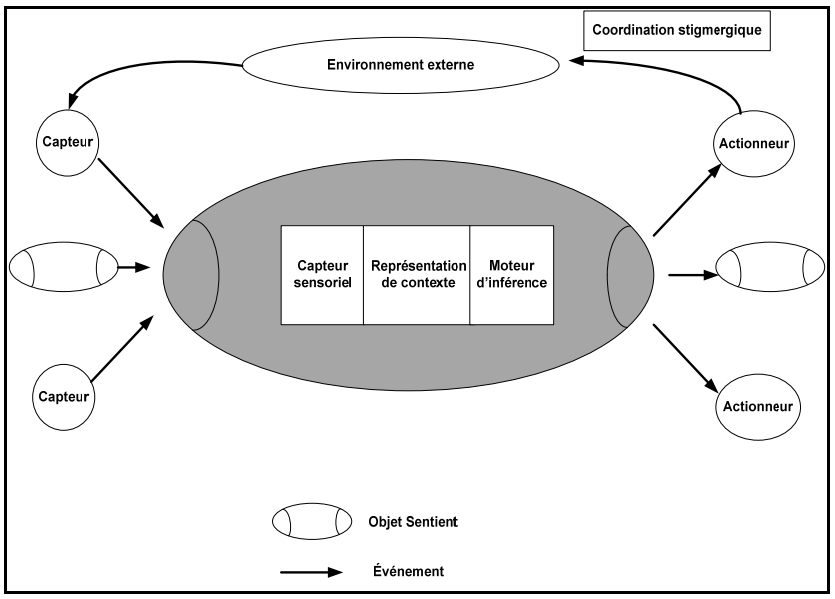

Figure 3.4 Architecture d'un « sentient object ».
Adaptée de Biegel et Cahill (2004, p. 2)

3.1.7 Context management framework (2003)

Le CMF (context management framework) (Korpipää et al., 2003) permet de faire le raisonnement sur le contexte sémantique en temps réel en présence d'incertitude de bruit et le changement rapide des informations ; délivrer le contexte aux applications selon un mécanisme de communication basé sur les événements. Le CMF propose une architecture client/serveur comme modèle de communication entre les entités qui le composent. L'architecture (Figure 3.5) contient cinq composants de base comme suit :

1. **Le Gestionnaire du contexte** : stockage des informations contextuelles (serveur) et livraison aux clients selon différents mécanismes (requête/réponse, inscription/notification etc.) ;
2. **Le Serveur de ressources** : acquisition des informations de contexte de capteurs physiques et leur interprétation selon un schéma spécifique avant de les envoyer au gestionnaire de contexte ;
3. **Le Service de reconnaissance du contexte** : conversion d'un flux de données à une représentation définie dans l'ontologie du contexte ;
4. **Le Service de détection des changements** : détecter les changements d'un service suite à un changement de contexte ;
5. **La Sécurité** : vérifie et contrôle les informations de contexte.

Le CMF utilise les ontologies pour la représentation du contexte, mais n'offre pas un mécanisme de raisonnement sur le contexte. Il possède un bon mécanisme d'interprétation, ce qui favorise bien l'abstraction du contexte et la réutilisation en plus du module de sécurité et de contrôle du contexte. Il utilise également un serveur pour la gestion du contexte (système centralisé) qui présente un problème lors de défaillance du serveur.

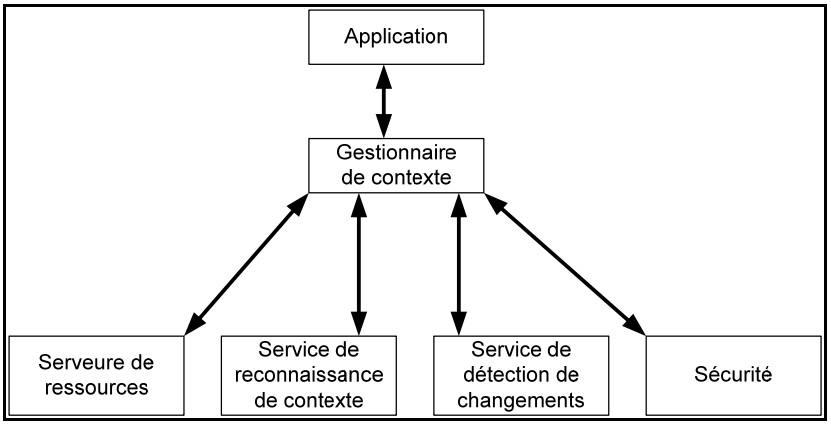

Figure 3.5 Architecture de CMF.
Adaptée de Korpipää et al. (2003, p. 2)

3.1.8 JCAF (2005)

(Bardram, 2005) a proposé un cadre de travail (framework) JCAF (Java context awareness framework) basé sur le langage Java dont le but est d'aider et d'appuyer les développeurs des applications sensibles au contexte. L'architecture de JCAF (Figure 3.6) est composée d'un ensemble de « ContextService » communicant selon le modèle d'égale-à-égale (peer-to-peer) et responsable de la collecte du contexte dans un environnement spécifique (chambre, hôpital, laboratoire, etc.). Un « ContextService » est composé de quatre composants :

1. **Le Récipient d'entité** : responsable d'échange d'informations de contexte avec les clients de contexte en utilisant un modèle de communication basé sur l'événement (inscription/notification). Il Contient en plus une ou plusieurs entités décrivant le contexte d'un objet de l'environnement (personne, ordinateur, médecin, … etc.) ;
2. **Le Transformateur** : effectue principalement deux opérations : a) l'agrégation des informations de contexte et b) la traduction entre les types du contexte ;
3. **Les Entités d'environnement** : permettent la communication entre les entités et gèrent l'accès aux ressources partageables ;
4. **Le Contrôleur d'accès** : contrôle l'accès au « ContextService » par une authentification correcte des requêtes de clients pour accéder aux contextes des entités.

Les clients du contexte peuvent être de trois types :

1. **L'écouteur d'entité** : qui peut être une entité d'un autre « ContextService » et qui peut accéder au contexte des entités d'un « ContextService », soit par le schéma requête/réponse, soit par le mécanisme d'événement inscription/notification. Il est possible d'utiliser le mécanisme d'inscription selon le type du contexte ;
2. **Le moniteur du contexte** : permet l'acquisition du contexte par des capteurs et assure une transformation du contexte brut ;
3. **L'actionneur du contexte** : offre la possibilité de commander des actuateurs de l'environnement physique.

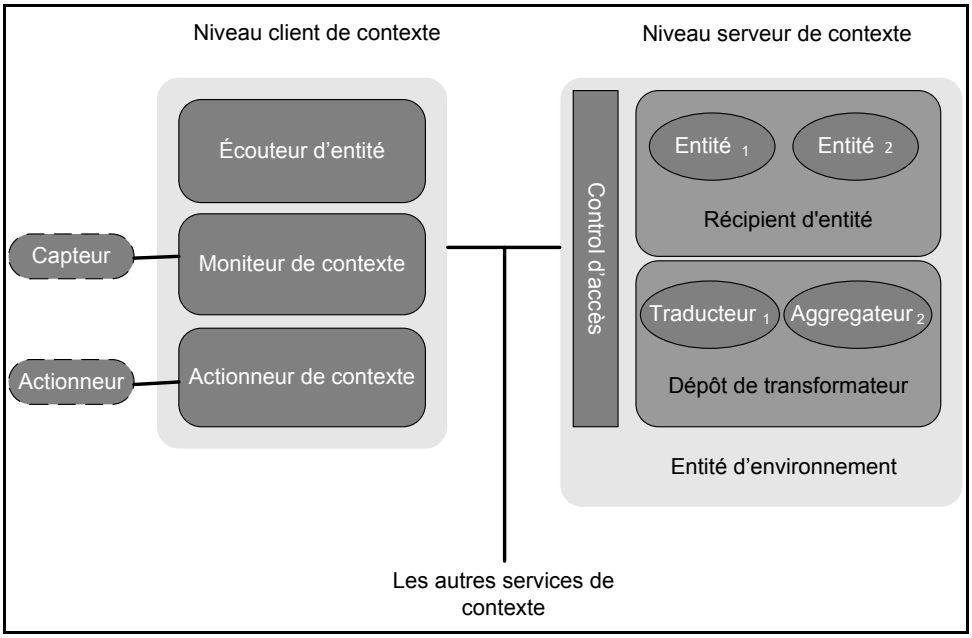

Figure 3.6 Architecture de JCAF.
Adaptée de Bardram (2005, p. 4)

Le JCAF assure un contrôle des informations contextuelles (degré de confiance d'une information provenant d'un capteur, probabilité d'erreur d'une information d'un capteur particulier, etc.). La communication à distance entre les composantes de l'architecture est assurée par le mécanisme RMI (remote method invocation) de Java. Les « ContextService »

ne possèdent pas un mécanisme de découverte automatique mais un fichier de configuration qui contient tous les autres « ContextService» actifs.
Le JCAF ne contient pas une couche pour raisonner sur le contexte et pour déduire des informations à partir du contexte courant. De plus, il n'offre pas une bonne abstraction du contexte à cause de l'absence d'une composante qui fait l'interprétation du contexte d'une manière explicite. L'inexistence d'un moyen de découverte automatique des éléments du système limite son extensibilité. Le JCAF offre cependant des modules réutilisables et portables à cause de l'utilisation de Java.

3.1.9 Context Toolkit (2001)

Le Context Toolkit (Dey, Abowd et Salber, 2001) a été proposé pour appuyer le développement des systèmes sensibles au contexte. Il a une architecture en couche pour permettre la séparation de processus d'acquisition et de représentation de contexte de l'adaptation. Il est basé sur les gadgets du contexte (context widgets). Ces gadgets fonctionnent de la même manière que ceux d'une interface utilisateur graphique (GUI) pour cacher la complexité des capteurs physiques utilisés par les applications. Cela donne au contexte plus d'abstraction et fournit des blocs réutilisables et personnalisés pour la capture du contexte. Les composants de l'architecture (Figure 3.7) sont :

- **les capteurs** : capture de contexte physique ;
- **les gadgets** : encapsulent les informations contextuelles et fournissent des méthodes pour accéder à ces informations de la même manière que pour les gadgets graphiques ;
- **les interpréteurs** : transforment l'information du contexte dans le but de lever le niveau d'abstraction ;
- **le groupeur** : groupement des informations du contexte relatif à un sujet ou à une situation ;
- **le découvreur** : maintient un registre des capacités existantes dans le cadre de travail (les composantes actuellement disponibles pour utilisation par des applications) ;
- **le service** : exécute des actions au profit des applications.

Cette architecture est simple à implémenter. Elle offre une communication distribuée entre les équipements du système et des gadgets réutilisables. Le mécanisme de découverte étant centralisé, ce qui l'empêche d'avoir un modèle de communication égale-à-égale parfait. L'architecture possède une extensibilité limitée quand le nombre de composantes augmente. Le mécanisme de prise en charge des événements (pour notifier le changement du contexte) consiste à utiliser un processus léger (thread) pour chaque événement, ce qui implique une charge importante pour le système. L'architecture ne contient pas un mécanisme de raisonnement sur le contexte à cause du modèle utilisé pour le représenter (clé/valeur).

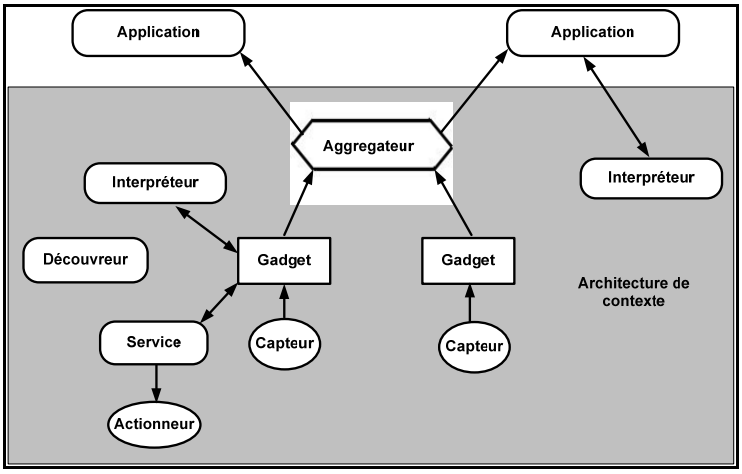

Figure 3.7 Éléments de l'architecture du Contexte Toolkit.
Adaptée de Dey et al. (2001, p. 24)

3.1.10 Hydrogen (2003)

Hydrogen (Hofer et al., 2003) est une architecture et un cadre de travail (framework) développés pour appuyer la sensibilité au contexte. C'est une architecture en trois couches pour convenir aux besoins particuliers des équipements mobiles composés des couches (Figure 3.8) : adaptateur, gestion (serveur de contexte) et application. Le serveur de contexte contient toutes les informations captées par la couche adaptateur et fournit à la couche application ou à d'autres équipements le contexte requis en utilisant le modèle de communication égale-à-égale. L'approche Hydrogen considère le contexte comme toute information pertinente sur l'environnement d'une application et décrit le contexte par un

modèle orienté objet. L'architecture est facile à implémenter. Elle prend en compte des ressources limitées des équipements mobiles en les déchargeant de tâches de traitement des informations du contexte. Elle utilise un modèle de communication non centralisé. La couche adaptateur effectue les deux tâches de capture et d'interprétation du contexte, ce qui n'offre pas le niveau d'abstraction requis pour de telles applications et rend le contexte dépendant des capteurs. Cela affecte la réutilisation des composants de l'architecture. Cette dernière ne comprend pas un module pour le raisonnement sur le contexte et sur la prise d'initiative d'adaptation.

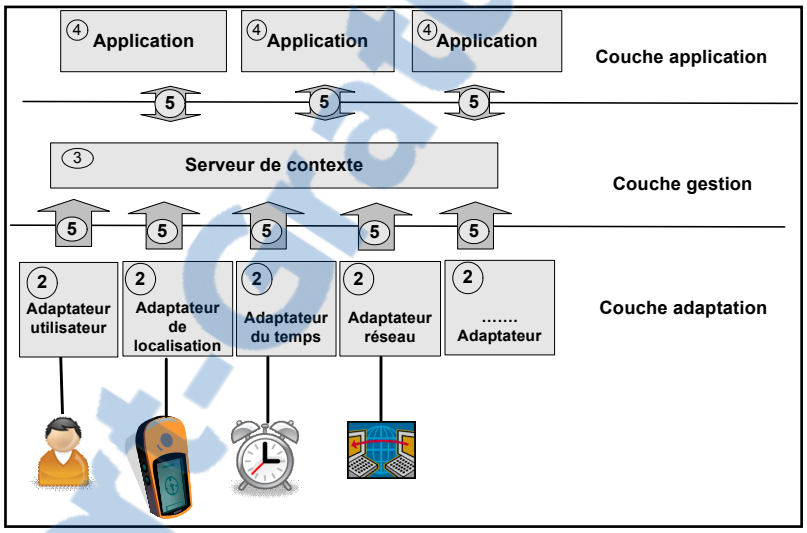

Figure 3.8 Architecture de Hydrogen.
Adaptée de Hofer et al. (2003, p. 4)

3.1.11 SOCAM (2004)

SOCAM (A Service-Oriented Context-Aware Middleware) (Gu et al., 2004a) est une architecture d'un intergiciel (middleware) sensible au contexte orienté service pour la construction et le prototypage rapide des services mobiles et sensibles au contexte dans un environnement intelligent d'une automobile. L'architecture comprend les composantes suivantes (Figure 3.9) : fournisseur de contexte, interpréteur de contexte (raisonnement sur le contexte et connaissance sur le contexte), service de localisation de services, service mobile sensible au contexte et la base de données du contexte. L'architecture utilise le modèle client/serveur. L'interpréteur du contexte collecte les informations du contexte des

fournisseurs de contexte (internes et/ou externes) et de la base de données du contexte. Il les livre ensuite au service mobile sensible au contexte et au service de localisation de services. La force principale de l'architecture de SOCAM est son système de raisonnement robuste sur le contexte. Celui-ci utilise les ontologies pour la description du contexte. L'architecture utilise deux classes d'ontologies : spécifiques au domaine et généralisées. Plusieurs systèmes de raisonnement peuvent être incorporés dans l'interprétateur du contexte pour appuyer des tâches variées de raisonnement. L'architecture est utilisée pour le développement d'une petite application (voiture intelligente). Évidement cela limite son utilisation dans des domaines variés de l'informatique diffuse. L'interpréteur du contexte est chargé par une quantité importante sous forme d'ontologies de différents domaines. Ce choix affecte la performance globale du système et présente les inconvénients de l'architecture centralisée, mais favorise la réutilisation.

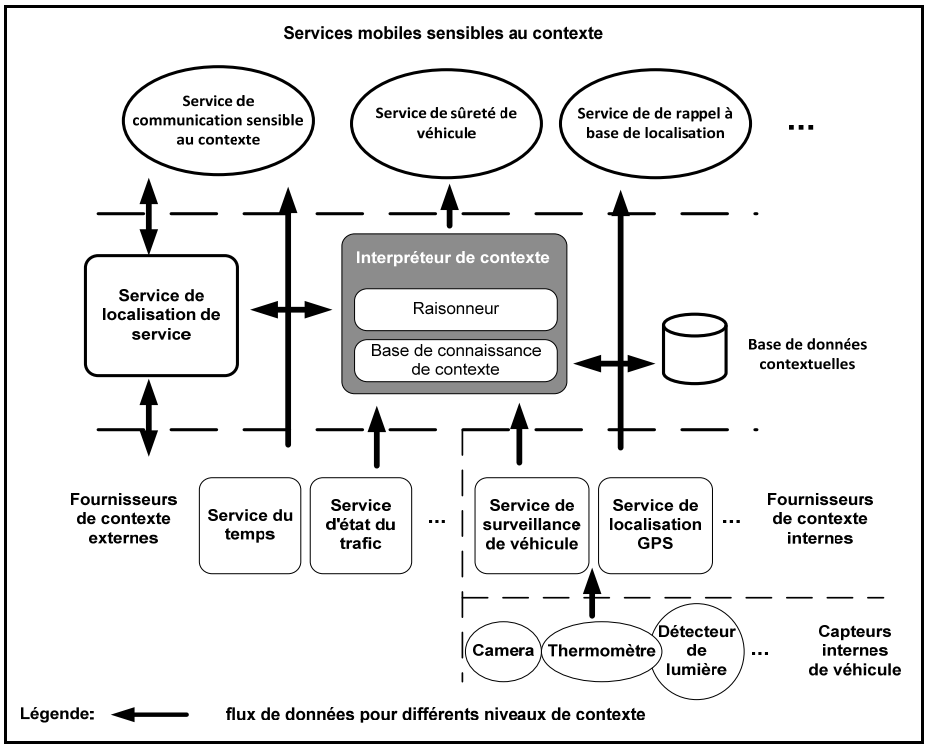

Figure 3.9 Architecture de SOCAM.
Adaptée de Gu et al. (2004, p. 2)

3.1.12 CoBrA (2004)

CoBrA (Context Broker Architecture) (Chen, 2004) est une architecture basée sur un agent courtier (broker) pour appuyer le développement des applications sensibles au contexte dans un espace intelligent. Le courtier est un agent autonome qui gère et contrôle le modèle de contexte d'un domaine spécifique. Il est implanté sur une machine dédiée (serveur) et possède des ressources puissantes. L'agent courtier possède une architecture en couche comprenant les éléments suivants (Figure 3.10) : base de connaissance du contexte, moteur de raisonnement sur le contexte, module d'acquisition du contexte et module de gestion de confidentialité. L'agent courtier fait l'acquisition du contexte des équipements, agents (applications) et les capteurs de son environnement. Il les fusionne par la suite dans un modèle cohérent qui sera ensuite partagé par les équipements et leurs agents. CoBrA utilise les ontologies pour la description du contexte, ce qui permet un raisonnement robuste et un partage efficace des informations du contexte. Elle utilise un modèle centralisé pour le stockage et le traitement du contexte puisque les équipements mobiles dans un système diffus possèdent des ressources limitées et une politique de confidentialité pour l'utilisateur. L'architecture nécessite un serveur dédié pour l'agent courtier, ce qui surcharge le système et augmente les risques de congestion au niveau du serveur. Le système présente également, les inconvénients de l'architecture centralisée.

3.2 Analyse des architectures antérieures

Hormis les architectures primitives des systèmes sensibles à la localisation, la plupart des architectures proposées font une séparation du processus de capture de contexte et son utilisation. Cela permet une abstraction des détails de capture de bas niveau et un accroissement de l'extensibilité et de la réutilisation des systèmes. Parmi les architectures proposées, il y a deux approches selon que les informations contextuelles soient centralisées ou distribuées.

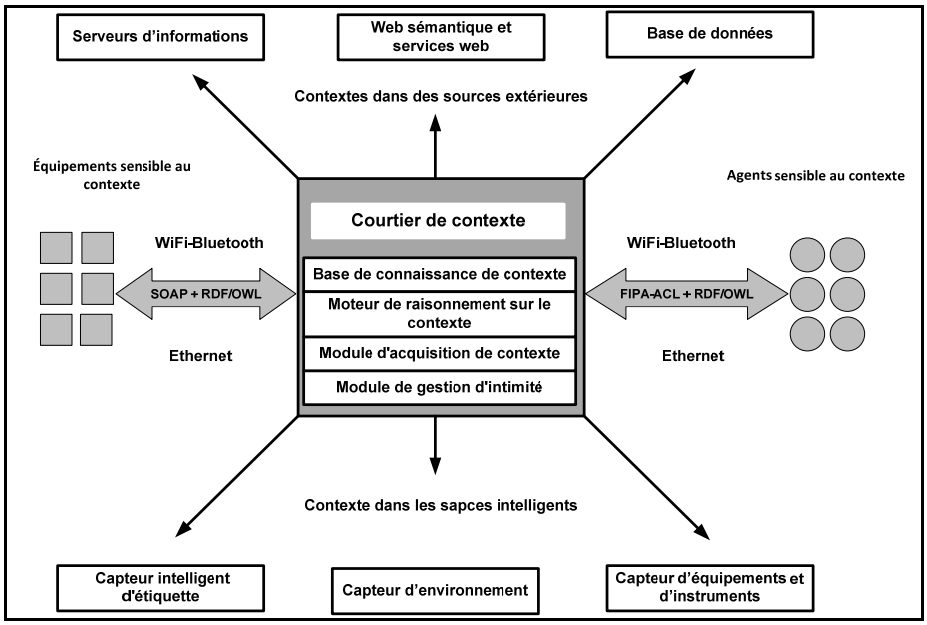

Figure 3.10 Architecture de CoBrA.
Adaptée de Chen (2004, p. 22)

La plupart de ces architectures sont en couches et contiennent principalement les éléments suivants :

1. **La capture** : capture physique du contexte par utilisation de différent type de capteurs ;
2. **L'interprétation** : transformation des informations brutes capturées en d'autres plus significatives et plus utilisables ;
3. **Le raisonnement** : (qui n'est pas présent dans toute l'architecture) déduction et prédiction des nouvelles informations du contexte ;
4. **La gestion et le stockage** : opérations de base pour la gestion des informations contextuelles (ajout, suppression, recherche, mise à jour, etc.) ;
5. **L'adaptation** : adaptation de services offerts selon le contexte.

Les architectures proposées sont la plupart spécifiques à un domaine d'application (systèmes de localisation, interaction personne-machine, etc.) et demandent des efforts supplémentaires pour l'adaptation. Les architectures fondées sur un serveur de contexte souffrent du problème principal d'un système centralisé : lorsque le serveur tombe en panne, tous les autres

composants seront affectés. Cela contredit la nature de l'information contextuelle dans un environnement diffus qui est en général distribuée. Rares sont les architectures qui contiennent tous les composants requis cités ci-dessus ou ne reposent pas sur un modèle de contexte solide et fiable pour permettre un raisonnement efficace sur le contexte. Le Tableau 3.1 résume les caractéristiques des architectures étudiées.

Tableau 3.1 Caractéristiques des architectures étudiées.

	Abstraction du contexte	**Modèle de communication**	**Composante logicielle de base**	**Raisonnement sur le contexte**	**Extensibilité**	**Réutilisation**
Activebadge	--	Client/Serveur	-	--	-	-
ParcTab	--	Client/Serveur	-	--	-	-
Stick-e-note	-	Égale-à-égale	-	--	-	-
Cyberguide	-	Hybride	-	--	-	-
Context toolkit	++	Égale-à-égale	Gadget	-	-	++
CASS	++	Client/Serveur	Objet	++	+	++
SOCAM	+	Client/Serveur	-	++	-	+
CORTEX	++	Égale-à-égale	Objet sentient	+	+	+
CoBrA	+	Client/Serveur	Agent	++	+	+
HYDROGEN	-	Égale-à-égale	Objet	--	+	+
JCAF	++	Égale-à-égale	« ContextService »	--	+	+
CMF	++	Client/Serveur	-	-	++	+

3.3 Architecture multiagents orientée service

Dans un système diffus, les équipements qui le composent collaborent et communiquent ensemble d'une manière transparente afin d'offrir à l'utilisateur ainsi qu'aux applications des services adaptés au contexte global. La notion de service joue un rôle crucial pour assurer le fonctionnement d'un tel système. Comme nous avons défini le contexte et la sensibilité au contexte en se basant sur la notion de service, nous allons étendre ces définitions pour mieux concevoir une architecture fondée sur les services (Miraoui et Tadj, 2007a) tout en utilisant une architecture répartie (égale-à-égale), multiagents (l'utilisation de l'approche multiagents sera argumentée au fur et au mesure).

Un système diffus est composé d'un ensemble d'équipements qui communiquent ensemble et fournissent des services sous différentes formes (différentes qualités de service) en utilisant différents médias et modalités. Chaque équipement fournit un ensemble de services à l'utilisateur et aux applications. Un service peut être fournit selon différentes formes. Un service sera alors modélisé en employant un automate d'états finis déterministe dont les états sont les formes d'un service (parmi plusieurs services) offert par un équipement et les transitions sont dues au changement de la valeur (respectivement les valeurs) d'une information (respectivement des informations) du contexte (cf. définition du contexte au paragraphe 2.2.2). Pour chaque service, nous pouvons donc limiter l'ensemble des informations contextuelles qui le concerne et lui permet de s'adapter selon le contexte. Pour faire suivre notre approche, nous allons présenter l'architecture au niveau d'un équipement, ensuite au niveau de l'ensemble des équipements, enfin au niveau de l'architecture globale. La Figure 3.11 montre un exemple de diagramme d'états d'un service ayant trois formes.

Par exemple, l'indication des appels pour un téléphone cellulaire a plusieurs formes : sonnerie, vibreur, sonnerie avec vibreur et silencieux. Nous allons nous restreindre aux deux formes : sonnerie et sonnerie avec vibreur. Initialement le téléphone cellulaire indique les appels entrants avec une sonnerie. Il perçoit le niveau de bruit dans son environnement, si ce niveau est haut (supérieure à un seuil), il change automatiquement sa forme d'indication

d'appels à la sonnerie avec vibreur pour attirer l'attention de l'utilisateur. La Figure 3.12 montre une partie de l'automate d'états pour cet exemple.

Un service perçoit les changements de l'environnement (contexte global) et réagit, soit changement de forme, soit par son déclenchement : entité réactive. Il peut être fourni différentes formes indépendamment des autres services : autonome. Il peut être doté de mécanismes lui permettant de prendre des initiatives pour réagir (prédire la forme service) : proactive. Finalement, il peut échanger des informations de contexte avec services pour faire l'agrégation (ou fusion) des informations contextuelles ou pour des informations utiles qui ne sont pas fournies par les capteurs d'attache : sociabilité. sont les principales caractéristiques d'un agent logiciel. Ceci nous amène à modéliser service par un agent avec des états internes. Ces agents seront contrôlés et administrés agent central (agent matériel) qui joue le rôle d'un serveur de contexte pour les autres Chaque agent doit être inscrit pour un ensemble d'informations du contexte qui le Il sera notifié par l'agent central chaque fois que la valeur de l'une des informations qui concerne change de valeur (pour limiter le flux d'informations). La communication autres agents de service du même équipement sera faite seulement à travers l'agent Dans le cas d'un simple capteur, il sera modélisé par un agent central sans les agents puisqu'il ne fournit qu'un seul type de service (capter des informations). La

Figure 3.13 présente l'architecture globale d'un système composé de quatre équipements matériels.

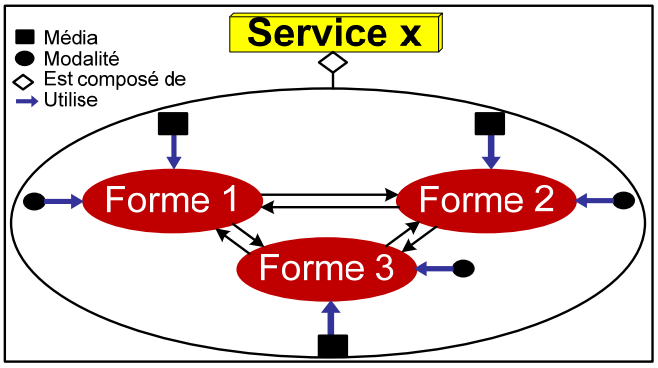

Figure 3.11 Diagramme d'états d'un service ayant trois formes.

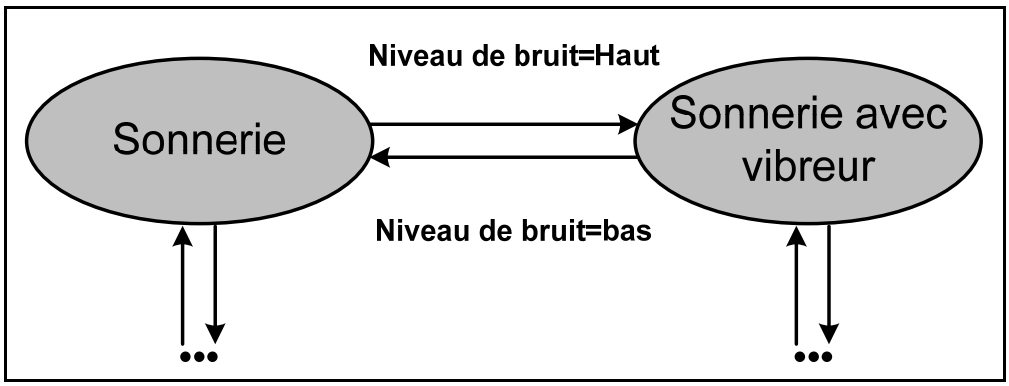

Figure 3.12 Une partie de l'automate d'états pour l'indication d'appels d'un téléphone cellulaire.

L'agent central possède une architecture en couche contenant en plus des couches classiques (capteur, interprétation, raisonnement et gestion/stockage) deux modules de communication : le module de communication interne pour l'échange d'informations avec les agents, service de l'équipement ; le module de communication externe pour l'échange d'informations avec les autres agents centraux du système diffus. Si un agent central perçoit ou capte le changement de la valeur d'une information contextuelle (à travers les capteurs propres du matériel d'attache) et qui concerne ou non un de ses agents, il diffusera la nouvelle valeur à l'ensemble des agents centraux pour permettre aux autres agents de réagir et sera notifié par les agents centraux concernés. Cela offre à l'agent central diffuseur la possibilité de se rappeler la prochaine fois des agents centraux concernés par cette même information et de mettre à jour la liste des agents centraux actifs dans le système diffus. Pour cela, nous envisageons de doter chaque agent central d'une mémoire cache pour ce type de transaction avec un système de vidage FIFO (first in first out) lorsque la mémoire est pleine. Pour la même raison, nous envisageons d'utiliser une cache interne pour les transactions internes avec les agents services. La même information contextuelle peut être diffusée par plusieurs agents centraux. Cela leur permet d'appliquer le processus de fusion afin de mieux contrôler les erreurs. La Figure 3.14 montre l'architecture d'un agent central.

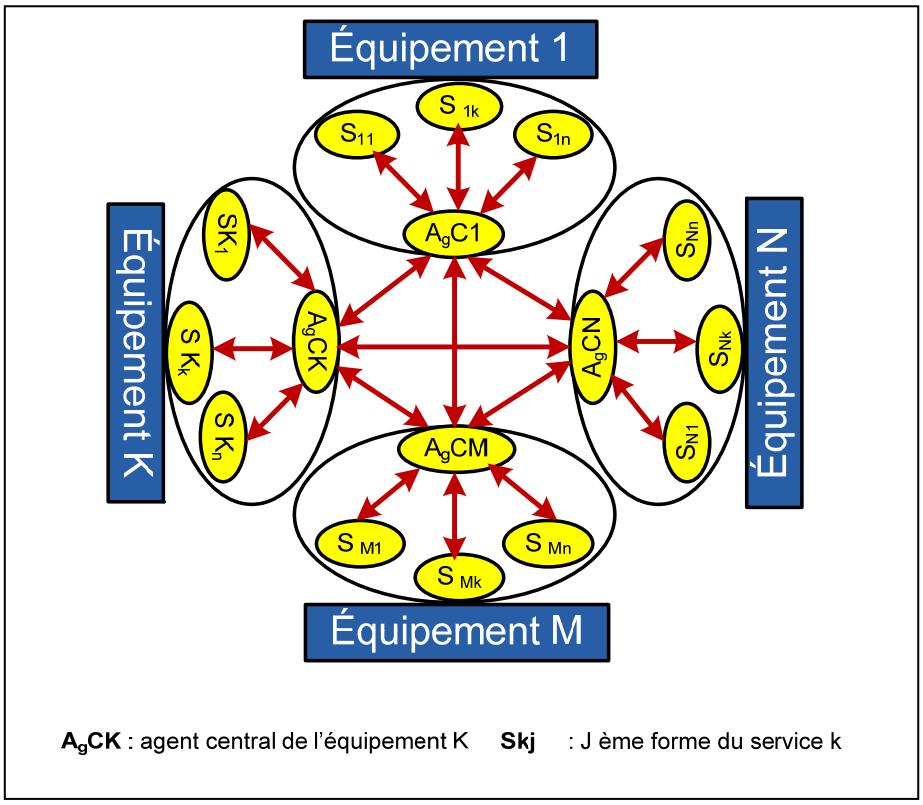

Figure 3.13 Architecture multiagents d'un système composé de quatre équipements matériels.

3.4 Discussion

L'architecture que nous avons exposée est une architecture hybride client/serveur au niveau du système multiagents (agent central et agents service) d'un équipement et d'égale-à-égale au niveau du système multiagents du système diffus (agents centraux). Cette approche permet de tirer profit des deux architectures de communication. L'architecture égale-à-égale permet de distribuer l'information contextuelle et le traitement relatif à ces informations entre les entités qui composent le système. Cette façon réduit la charge de traitement et diminue le flux d'informations entre les entités du système. L'architecture client/serveur favorise un contrôle efficace d'accès multiple concurrent aux informations contextuelles et décharge les clients (agents services) des opérations de stockage et de gestion de l'information.

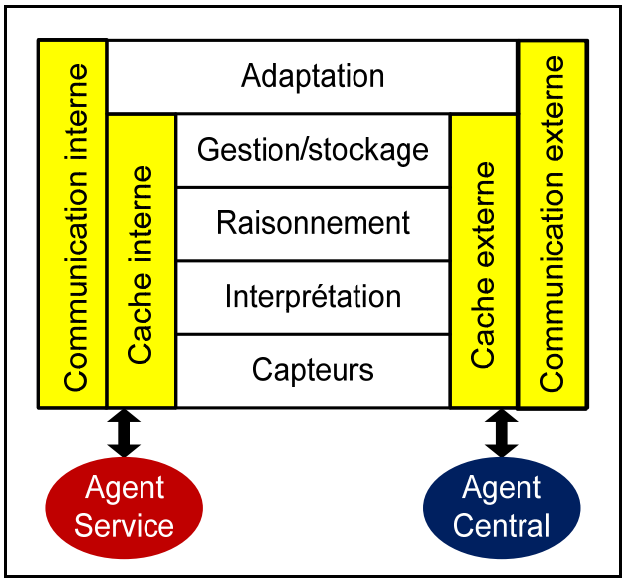

Figure 3.14 L'architecture d'un agent central.

Dans l'architecture multiagents d'égal à égal au niveau des agents centraux, si un équipement du système diffus tombe en panne, par suite son agent central sera inaccessible (inactif). Cela n'affectera pas le fonctionnement du système. Les services fournis par cet équipement peuvent être remplacés par des services d'un autre équipement. Dans le même contexte, il est facile d'ajouter de nouveaux équipements au système sans apporter des modifications aux éléments de ce dernier. Le système détectera le nouvel équipement automatiquement s'il échange des informations avec un des équipements du système (architecture dynamique extensible).

Cette architecture est réutilisable pour la raison suivante : les systèmes multiagents (agent central et agents services) d'un équipement peuvent être utilisés dans d'autres systèmes diffus avec des modifications mineures car les services offerts par un équipement sont les mêmes quel que soit le système qui l'englobe (exemple : un GPS fournit toujours les coordonnées géographiques quelle que soit son utilisation).

L'encapsulation des détails des services d'un équipement ainsi que le modèle de gestion de contexte au sein du dispositif favorise la protection de ces équipement contre les accès

indésirables qui peuvent abîmer la qualité de ses services (l'accès se fait à travers une interface : agent central).

Dans cette architecture, nous n'avons pas considéré le contexte utilisateur par ce qu'il est inclus implicitement dans le contexte des équipements (exemple : la localisation de l'utilisateur est détectée par un GPS qui est en possession de l'utilisateur).

3.5 Modélisation et simulation

Afin de franchir une étape importante vers la validation de notre architecture, nous avons réaliser sa modélisation et sa simulation (Miraoui, Tadj et Amar, 2009) afin d'évaluer un certain nombre de ses aspects en décrivant le comportement de ses composants et de leurs propriétés. Nous avons utilisé les réseaux de Petri colorés (RdPC) comme formalisme de modélisation et le CPN-Tools comme outil de simulation et d'évaluation.

Plusieurs méthodes d'évaluation des architectures logicielles selon des qualités désirées ont été proposées (Babar et Gorton, 2004; Kazman et al., 1996; Mattsson, Grahn et Mårtensson, 2006). Parmi ces méthodes, nous pouvons citer celles qui sont basées sur les expériences, les simulations, les scénarios et la modélisation mathématique. Ces méthodes peuvent être utilisées séparément ou combinées pour obtenir de meilleurs résultats. Il est fortement recommandé de faire l'évaluation des performances tôt au niveau de la conception architecturale, en développant et en évaluant un modèle de d'architecture logicielle qui aide à comprendre la fonctionnalité du système, ses possibilités et sa complexité. L'identification des problèmes de performance au niveau de la conception architecturale est très utile et plus rentable (économique) parce qu'elle évite de changer la conception plus tard (Balsamo et Marzolla, 2003). L'utilisation des méthodes formelles pour modéliser et décrire le comportement du système à un niveau élevé d'abstraction peut contribuer à la consistance et à la fiabilité de la conception architecturale, facilitant ainsi le procédé de validation. La simulation permet d'étudier le comportement de l'architecture logicielle pour vérifier ses

propriétés au moyen de banc d'essai (en anglais model checking), pour analyser le comportement dynamique et les performances du système dans certains cas d'utilisation.

3.5.1 Les réseaux de Petri colorés et le CPN-Tools

Les réseaux de Petri (Petri nets) ont été inventés en 1962 par Carl Adam Petri. Ils sont utilisés comme méthode formelle pour la modélisation d'une classe importante de systèmes (Reisig, 1985). Ils permettent d'exprimer les propriétés comportementales et structurales des systèmes. Un aspect important des réseaux de Petri est qu'ils peuvent être représentés graphiquement ou mathématiquement, ce qui améliore leur clarté et leur lecture. Cependant, le problème d'explosion d'états reste l'obstacle majeur pour la modélisation et l'analyse des systèmes complexes. Les RdPC ont été proposés comme extension des réseaux de Petri ordinaires pour surmonter le problème d'explosion d'états en offrant un modèle plus compact. Les RdPC utilisent une modélisation à événement discret, qui combine les réseaux de Petri ordinaires et le langage de programmation fonctionnel ML fondé sur le standard ML. Un modèle de RdPC d'un système décrit ses états et les événements (transitions) qui offre la possibilité au système de changer d'état comme les réseaux de Petri ordinaires mais en utilisant des jetons de couleurs différentes (une couleur est une information additionnelle sur le jeton) (Jensen, 1998). Notre utilisation des RdPC est motivée aussi par l'existence des outils logiciels qui appuient la construction graphique et la visualisation des modèles. Parmi ces outils, nous avons choisi le CPN-Tools (CPN Tools: Computer Tool for Coloured Petri Nets) pour explorer le comportement du modèle de l'architecture. Nous avons utilisé des simulations pour vérifier ses propriétés par la technique de banc d'essai (model checking) et pour faire une analyse de performance.

3.5.2 Modélisation et simulation de l'architecture

Globalement, l'architecture se compose de deux systèmes multiagents : le système multi-agent du SID composé d'agent central de chaque équipement informatique, le système multiagent de l'équipement informatique composé d'agent central et d'agents de service de

l'équipement. Le système multiagents du SID assure la communication entre les équipements en envoyant les informations contextuelles captées aux autres agents centraux du SID et en recevant des accusés de réception. Il fonctionne comme une base de données distribuée : chaque base de données contextuelle (ou gestionnaire de contexte) d'un agent central d'un équipement informatique est mis à jour chaque fois qu'il y a capture d'une information contextuelle. Cela permet le partage d'informations contextuelles entre les équipements informatique du SID et augmente leur niveau de sensibilité au contexte. L'agent central d'un équipement informatique a deux tâches essentielles: 1) mettre à jour sa base de données contextuelle (gestionnaire de contexte) en recevant un message d'autres agents centraux du SID (qui doit être accusé) ou en recevant de nouvelles valeurs d'information contextuelle des ses propres capteurs de l'équipement informatique (qui doivent être diffusées aux autres agents centraux du SID). 2) l'autre tâche se compose de quatre sous tâches comme suit :

1. Sélectionner le service concerné par l'information contextuelle capté par l'équipement informatique lui-même ou reçu des autres agents centraux du SID ;
2. Apporter du gestionnaire de contexte de l'agent central l'ensemble de valeurs des informations contextuelles du service concerné ;
3. Évaluer l'expression logique de déclenchement du service (ELDS) avec la nouvelle information contextuelle et envoyer le résultat d'évaluation à l'agent de service concerné (ELDS est une combinaison logique d'un sous-ensemble des valeurs d'éléments du contexte de service (par exemple *si ((a=x) et (b=y)) alors* le service sera déclenché sachant que a et b appartiennent à l'ensemble d'information contextuelle du service) ;
4. Évaluer pour chaque forme du service, l'expression logique de changement de forme (ELCF) et envoyer le résultat à l'agent de service concerné. ELCF a une structure semblable à ELDS

Les sous tâches ci-dessus constituent l'adaptation de services selon le contexte.

Sur la réception des résultats d'évaluation de l'ELDS et de l'ELCF de l'agent central, l'agent de service réagit de différentes façons comme il est indiqué dans le Tableau 3.2

Tableau 3.2 Les entrées possibles d'un agent de service

ELDS	ELCF	Résultats
vrai	faux	déclencher le service sous sa forme par défaut
faux	vrai	arrêter le service
vrai	vrai	déclencher le service sous une autre forme
Faux	faux	arrêter le service

Le fonctionnement global de l'architecture est donné par la Figure 3.15. Il consiste en l'échange des informations contextuelles entre le système multiagents du SID et un agent central d'un équipement informatique. Cet échange a lieu chaque fois qu'une nouvelle valeur d'information contextuelle est captée par les équipements du SID ou l'agent central de l'équipement.

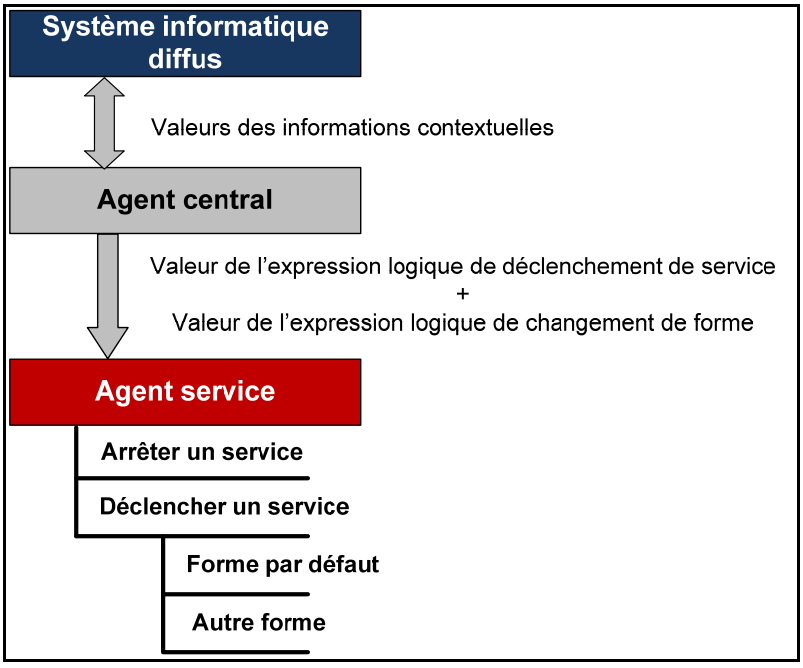

Figure 3.15 Le fonctionnement global de l'architecture.

Après avoir effectué le traitement nécessaire sur la nouvelle valeur de l'information contextuelle, l'agent central de l'équipent informatique envoie les résultats d'évaluation de

l'ELDS et de l'ELCF à l'agent de service. Ce dernier réagit comme il a été indiqué dans le Tableau 3.2. Cette décomposition logique de l'architecture nous mène à réaliser la modélisation et la simulation des trois composants de base de l'architecture, à savoir : le SID, l'agent central d'un équipement informatique et l'agent de service d'un équipement informatique.

■ **Le SID :**

Modèle

L'ensemble des déclarations utilisées pour la construction du modèle de SID (Figure 3.16) est donné par l'extrait 3.1 suivant :

```
colset UNIT = unit;
colset INT = int;
colset BOOL = bool;
colset STRING = string;
val n =4;
colset E = with e;
colset Context=product STRING*INT;
colset CentralAgent=index CA with 1..n;
colset AA=product Context*CentralAgent;
colset Couple=product CentralAgent*CentralAgent;
fun diff(x,y) = (x<>y);
colset Message= subset Couple by diff;
colset MessageContext=product Context*Message;
fun Mes(s) = Couple.mult(1`s,CentralAgent.all() --1`s);
colset ContextVal=product Context*Couple;
var s,r : CentralAgent;
var x:Context;
```

Extrait 3.1 Ensemble de déclarations utilisé pour la modélisation du SID.

Scénario de simulation et résultats

Pour valider notre modélisation du SID de l'architecture, nous avons employé un scénario typique, détaillé comme suit :

- Quatre agents centraux : CA (1), CA (2), CA (3) et CA (4).
- Quatre informations contextuelles avec nouvelles valeurs captées sous cette forme : *(information contextuelle, nouvelle valeur)*.
  L'information contextuelle est représentée sous forme d'une chaîne de caractères et peut être par exemple un débit d'une connexion internet, un niveau de charge d'une batterie, un niveau de bruit, une heure précise, une date, etc. Nous posons hypothèse suivante, à savoir que la valeur d'une information contextuelle est numérique, soit l'exemple suivant :
  ("a",50), ("b",0), ("c",1) et ("d",20)

Chaque information contextuelle est captée par un agent central selon le Tableau **3**.**3**.

Tableau 3.3 Informations contextuelles et agent central capteur

Information contextuelle	**Agent Central capteur**
"a"	CA(1)
"b"	CA(2)
"c"	CA(3)
"d"	CA(4)

Sur la détection d'une nouvelle valeur d'information contextuelle, un agent central met à jour sa base de données de l'information contextuelle (ou gestionnaire de contexte). Il construit ensuite une liste des messages pour diffuser la nouvelle valeur de l'information contextuelle captée aux autres agents centraux du SID et ayant la forme suivante :
message (expéditeur, récepteur)
Soit l'exemple suivant :

- CA(1): (CA(1),CA(2)), (CA(1),CA(3)), (CA(1),CA(4));

- CA(2): (CA(2),CA(1)), (CA(2),CA(3)), (CA(2),CA(4));
- CA(3): (CA(3),CA(1)), (CA(3),CA(2)), (CA(3),CA(4));
- CA(4): (CA(4),CA(1)), (CA(4),CA(2)), (CA(4),CA(3)).

La nouvelle valeur de l'information contextuelle captée sera alors envoyée à cette liste. Chaque récepteur (agent central) doit accuser la réception d'une nouvelle valeur de l'information contextuelle captée. Après exécution de la simulation du scénario, nous avons obtenu en tout quatre informations contextuelles mis à jour et douze accusés de réception (trois accusés pour chaque agent central).

Nous avons apporté plusieurs modifications aux paramètres du scénario (nombre d'agents centraux et nombre d'informations contextuelles captées) et chaque fois nous avons obtenu les résultats prévus que le système doit fournir.

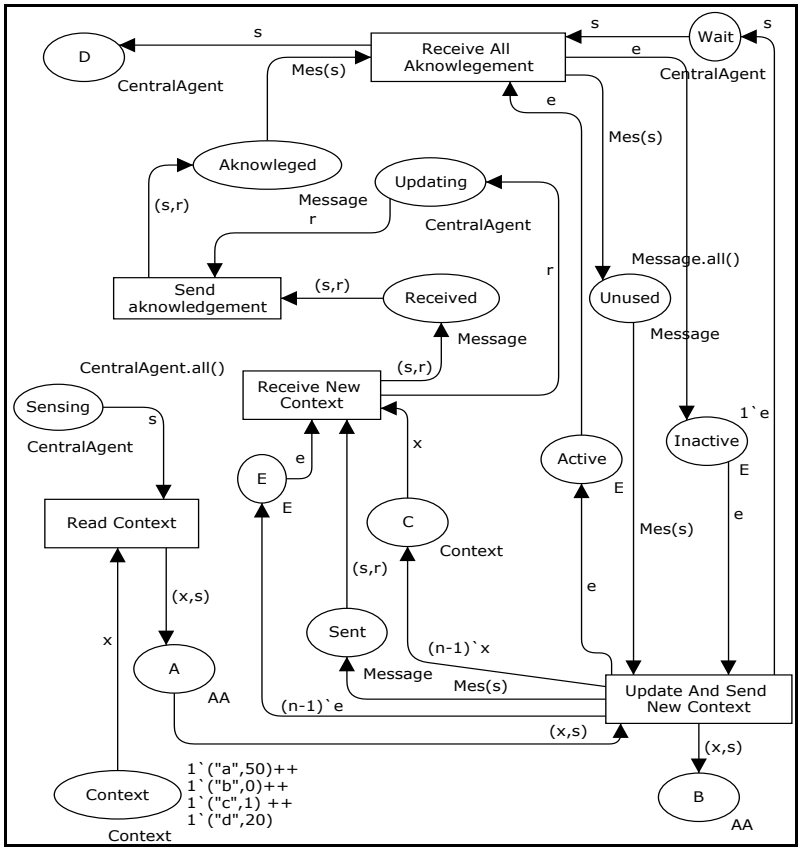

Figure 3.16 Le modèle du SID.

■ **Agent de service :**

À la réception d'une nouvelle valeur d'information contextuelle et après l'évaluation des ELDS et d'ELCF par l'agent central, un service peut être déclenché sous sa forme par défaut ou sous une autre forme. Il peut également être arrêté.

Modèle

L'ensemble des déclarations utilisées pour la construction du modèle de l'agent de service (Figure 3.17) est donné par l'extrait 3.2 suivant :

```
colset E=with e;
colset BOOL=bool;
colset INT=int;
colset STRING=string;
colset FormChange=product STRING*BOOL;
colset ServiceTrig=product INT*BOOL;
colsetServiceAndForm=product ServiceTrig*FormChange;
colset TrigAndChange=product BOOL*BOOL;
var x1,x1,a,b:BOOL;
var s:INT;
 var f:STRING;
```

Extrait 3.2 Ensemble de déclarations utilisé pour la modélisation de l'agent de service.

Scénario de simulation et résultats

Pour valider notre modélisation de la partie agent de service de l'architecture, nous avons employé un scénario typique, détaillé comme suit :

- Numéro de l'agent de service =1
- Formes= {forme par défaut, F11, F12, F13}
- Les configurations possibles reçues de l'agent central sous la forme :
  ((Numéro de service, valeur de l'ELDS), (Numéro de la forme, valeur de l'ELCF))

Les configurations possibles et les résultats de la simulation sont montrés dans le Tableau 3.4.

Tableau 3.4 Les configurations possibles et les résultats de simulation pour l'agent de service

No. de service	**Valeur de l'ELDS**	**Numéro de la forme**	**Valeur de l'ELCF**	**Résultat**
1	Vrai	F11	Faux	déclenché sous la forme par défaut
1	Vrai	F12	Faux	déclenché sous la forme par défaut
1	Vrai	F13	Faux	déclenché sous la forme par défaut
1	Faux	F11	Vrai	arrêté
1	Faux	F12	Vrai	arrêté
1	Faux	F13	Vrai	arrêté
1	Faux	F11	Faux	arrêté
1	Faux	F12	Faux	arrêté
1	Faux	F13	Faux	arrêté
1	Vrai	F11	Vrai	déclenché sous la forme F11
1	Vrai	F12	Vrai	déclenché sous la forme F12
1	vrai	F13	vrai	déclenché sous la forme F13

Après chaque exécution d'une simulation, nous avons obtenu les résultats prévus.

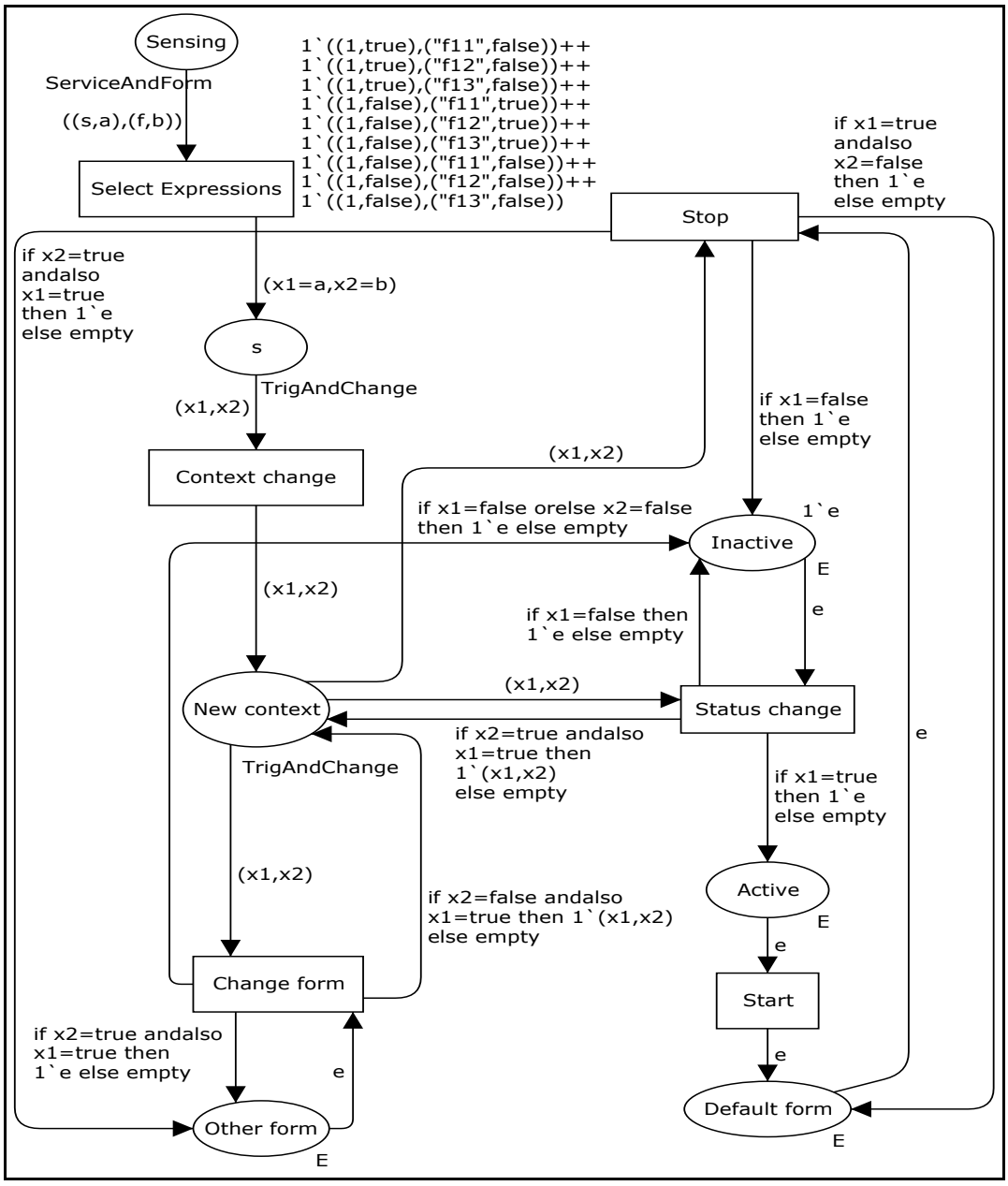

Figure 3.17 Le modèle de l'agent de service.

■ Agent central d'un équipement :

Modèle

Soit la notation suivante :

- Ensemble des informations contextuelles

  $C = \{C_1,\ C_2, \dots, C_n\}$
  C_i peut etre le niveau de charge d'une batterie, vitesse d'une connexion internet, date, heure, etc.
- Ensemble des valeurs des informations contextuelles

  $V = \{(C_i, D_i)\}\ \ i = 1, \dots, n$
  D_i **:** étant une valeur associée à un élément du contexte.
  Exemple : (localisation, université)
- Ensemble des services fournis par un équipement

  $S = \{S_1,\ S_2, \dots, S_k\}$
- Ensemble des formes d'un service i

  $SF_i = \left\{F_{i_1}, F_{i_2}, \dots, F_{i_{x_i}}\right\}$
  x_i est le nombre de formes du service i
  F_{ij} : la forme j du service i, $j = 1, \dots, x_i$ et $i = 1, \dots, k$
- Chaque service utilise un ensemble d'informations contextuelles

  $SC_i = \{C_y\}, y = 1, \dots, m,\ \ m \leq n$
- Chaque service S_i a une expression logique de déclenchement $SE_i, i = 1, \dots, k$

  SE_i est une combinaison logique (ET, OU, NON) de $V_i = \{(C_i\ , D_i)\}\ avec\ C_i \in\ SC_i$
- Un service sera caractérisé par trois composants : nom du service, ensemble des informations contextuelles et expression logique de déclenchement. Nous le représentons par un triplet :

  $\langle S_i,\ SC_i,\ SE_i\rangle\ \ i = 1, \dots, k$
- Chaque forme j d'un service i a une expression logique de changement de forme FE_{ij}

  $j = 1, \dots, x_i$ et $i = 1, \dots, k$

  FE_{ij} est une combinaison logique (ET, OU, NON) de $V_i = \{(C_i\ , D_i)\}$ avec $C_i \in\ SC_i$
- Finalement un service est représenté par :

  $\langle S_i, \{F_{ij}, FE_{ij}\}, SC_i, SE_i\rangle\ avec\ j = 1, \dots, x_i$ et $i = 1, \dots, k$

L'ensemble des déclarations utilisées pour la construction du modèle de l'agent central est défini comme le montre l'extrait 3.3 suivant :

```
colset UNIT = unit;
colset INT = int;
colset BOOL = bool;
colset STRING = string;
colset Context=STRING;
colset E=with e;
colset Form=product STRING*BOOL;
val u=4;
colset FormList=list Form;
colset ContextList=list Context;
colset ContextVal=product Context*INT;
colset Service=int with 1..u;
colset Trigger=product Service*BOOL;
colset ServiceContext=product Service*ContextList;
colset ServiceForm=product Service*FormList;
colset TT=product INT*INT;
colset ServiceAndForm=product Trigger*Form;
var n,n1:Context;
var i,b,p,j,k,z,z1:INT;
var w,m:ContextList;
var s,s1:Service;
var x,exp:BOOL;
var y:ContextVal;
var t: Trigger;
var f:FormList;
var ft:Form;
```

Extrait 3.3 Ensemble de déclarations utilisé pour la modélisation de l'agent de service.

Scénario de simulation et résultats

Pour valider notre modélisation de la partie agent central de l'architecture, nous avons employé un scénario typique détaillé comme suit :

- Liste d'informations contextuelles= {"a","b","c","d","e","f","g","h","i","j"}.
- Valeurs actuelles des informations contextuelles représentées sous cette forme : *(Information contextuelle, valeur).*

Soit l'exemple suivant :

("a",10), ("b",5), ("c",20), ("d",1), ("e",30), ("f",3), ("g",50), ("h",0), ("i",2), ("j",15).

- Services fournis par l'équipement (Numéro de service) = {1, 2, 3, 4}.
- Liste d'informations contextuelles pour chaque équipement représentées sous cette forme :
  (Numéro de service, [Liste d'informations contextuelles])

Soit l'exemple suivant :

(1, ["a","b","c"]), (2, ["d","a","e","f"]), (3, ["g","h","b"]), (4, ["i","j"]).

- Nouvelle valeur captée de l'information contextuelle : ("i", 50).

L'agent central effectue les étapes suivantes (Figure 3.18) :

1. Vérifier si l'information contextuelle captée (respectivement reçu) concerne (appartient à l'ensemble des informations du contexte de) l'agent central ou non (Figure 3.20) ;
2. Mettre à jour sa valeur si cette information de contexte concerne l'agent central, (Figure 3.19) ;
3. Transmettre cette nouvelle valeur aux autres agents centraux du SID si la nouvelle valeur de l'information contextuelle est différente de l'ancienne valeur (Figure 3.20) ;
4. Sélectionner un des services fournis par l'équipement informatique (Figure 3.19) ;
5. Vérifier si l'information du contexte appartient à l'ensemble des éléments du contexte du service. Si c'est le cas, alors passer à 6, sinon retourner à 4 (Figure 3.21) ;

6. Apporter les valeurs de l'ensemble des informations du contexte du service (Figure 3.22) ;
7. Évaluer l'ELDS avec les valeurs du contexte apportées (Figure 3.23) ;
8. Choisir une des formes du service (qui a été sélectionnée dans 4) (Figure 3.24) ;
9. Évaluer l'ELCF avec les valeurs du contexte apportées (dans 6) (Figure 3.25) ;
10. Transmettre les résultats d'évaluations de l'ELDS et l'ELCF à l'agent de service du service déjà sélectionné dans 4 et retourner à 8 (Figure 3.26) ;
11. Retourner à l'étape 4 pour sélectionner un autre service si l'évaluation de toutes les ELCF de toutes les formes est terminée ;
12. Répéter les étapes de 4 à 12 jusqu'à la sélection de tous les services concernés par l'information contextuelle captée.

- Les formes de chaque service : *(Numéro de service, [liste de formes])*
  Soit la situation suivante :
  (1, [F11, F12, F13]), (2, [F21, F22]), (3, [F31]), (4, [F41, F42, F43, F44]).
- Résultats d'évaluations de l'ELDS représentés sous cette forme :
  (Numéro de service, valeur booléenne).
  Soit l'exemple suivant :
  (1, vrai), (2, faux), (3, vrai), (4, vrai).
- Résultats d'évaluations des expressions logiques de changement de forme représentés sous cette forme :
  (Numéro de service, [liste de (forme, valeur booléenne)]).
  Soit la situation suivante :
  (1, [(F11, vrai), (F12, faux), (F13, faux)]), (2, [(F21, faux), (F22, faux)]), (3, [(F31, vrai)]), (4, [(F41, faux), (F42, faux), (F43, vrai), (F44, faux)]).

Après exécution de la simulation de ce scénario, nous avons obtenu quatre messages qui ont été envoyé au service 4 de l'équipement parce que c'est le service concerné par la nouvelle valeur captée de l'information contextuelle.

- Les messages ont la forme suivante :

((Numéro de service, valeur de l'ELDS), (forme, valeur de l'ELCF))

((4, vrai), (F41, faux))
((4, vrai), (F42, faux))
((4, vrai), (F43, vrai))
((4, vrai), (F44, faux))

Nous avons apporté plusieurs modifications aux paramètres du scénario (nombre de services, nombre d'informations contextuelles captées, liste d'informations contextuelles pour chaque service, nombre de formes de chaque service, valeur de l'ELDS et valeur de l'ELCF) et à chaque fois nous avons obtenu les résultats prévus que doit fournir le système.

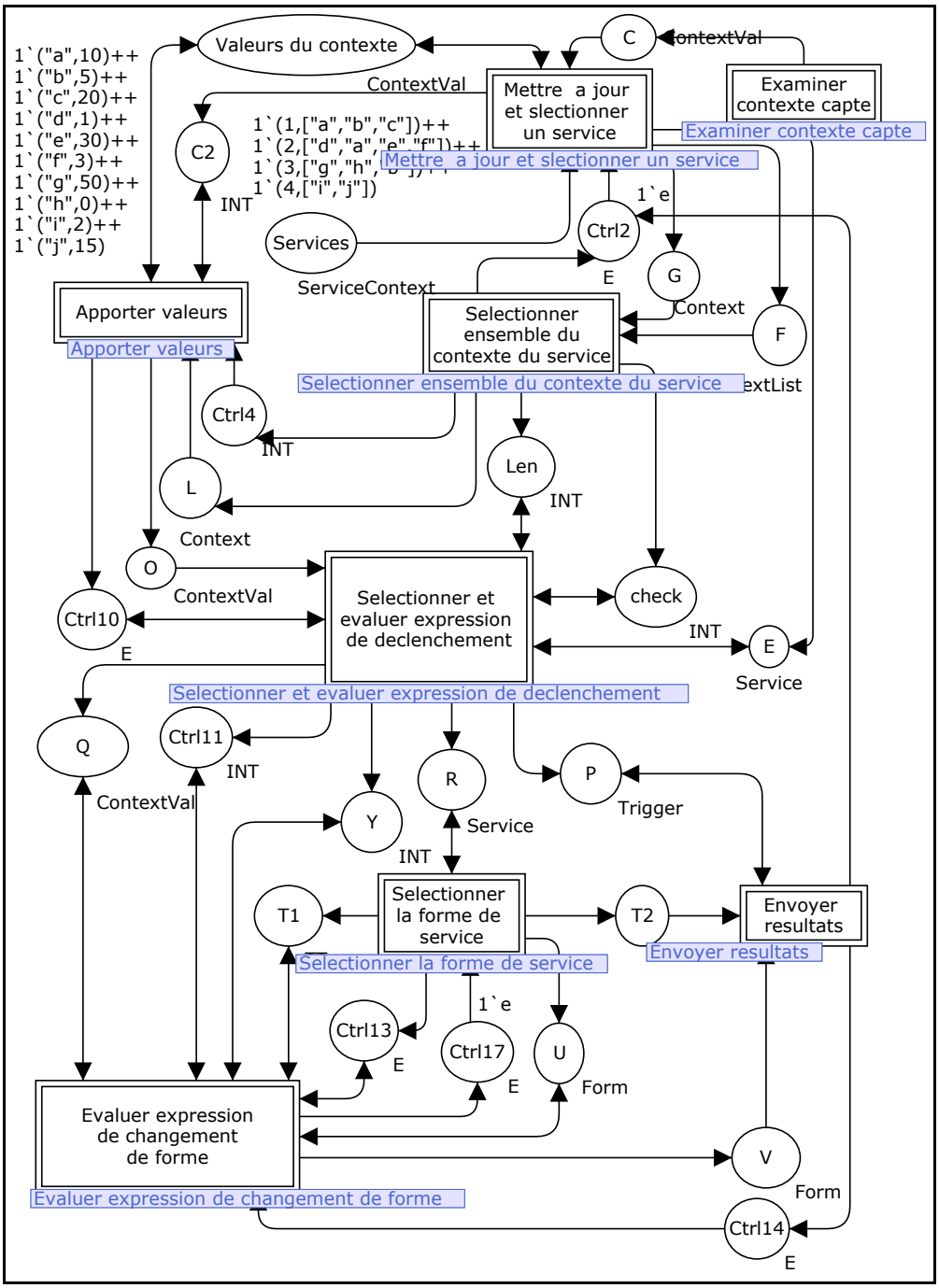

Figure 3.18 Le modèle global de l'agent central.

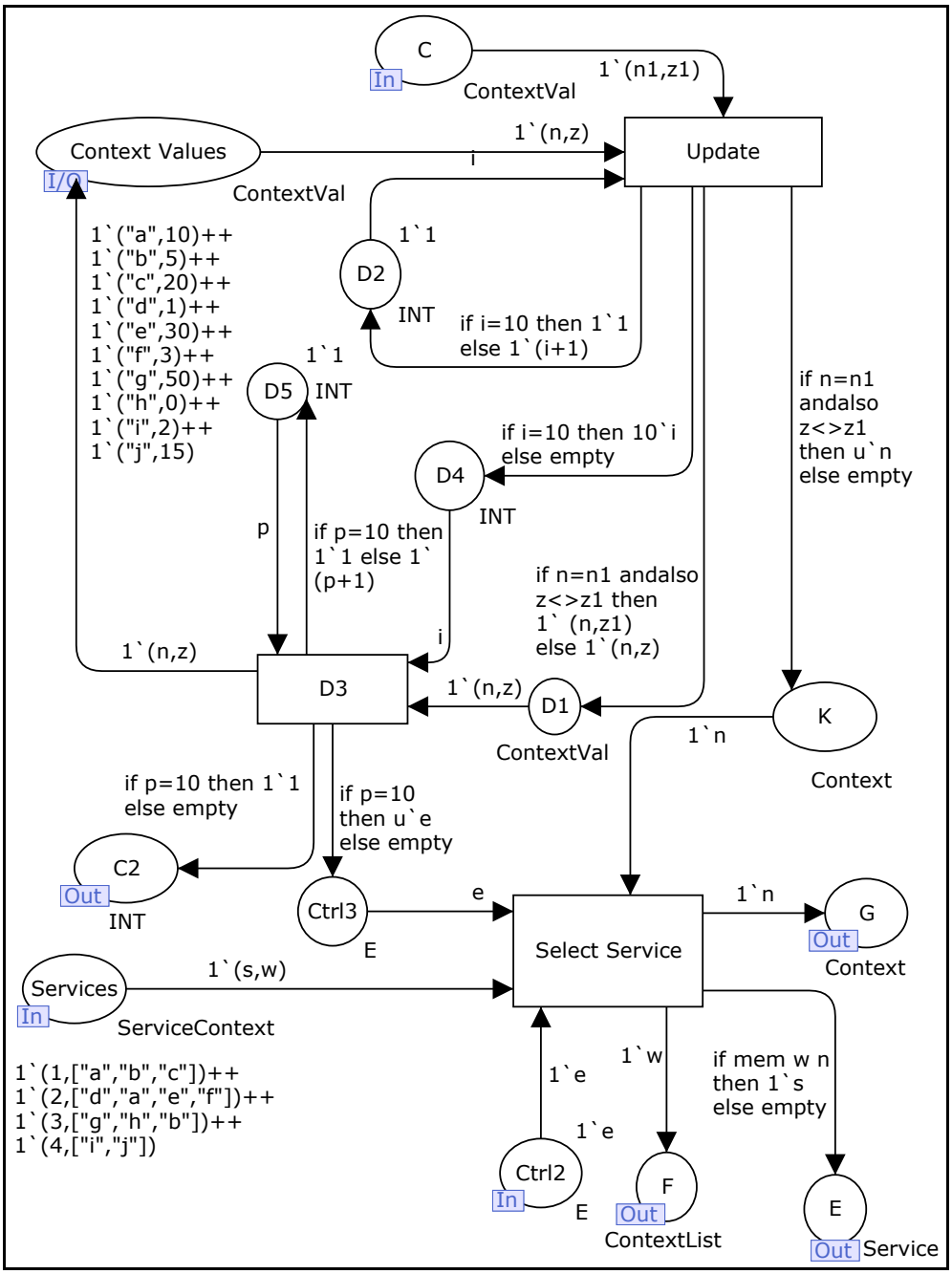

Figure 3.19 Le modèle de mise à jour et de sélection de service.

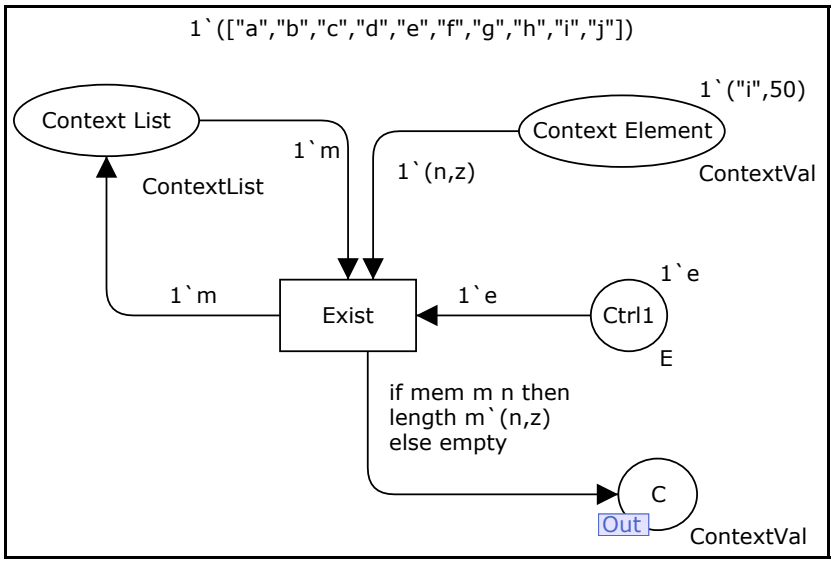

Figure 3.20 Le modèle de vérification d'appartenance au contexte de l'agent central.

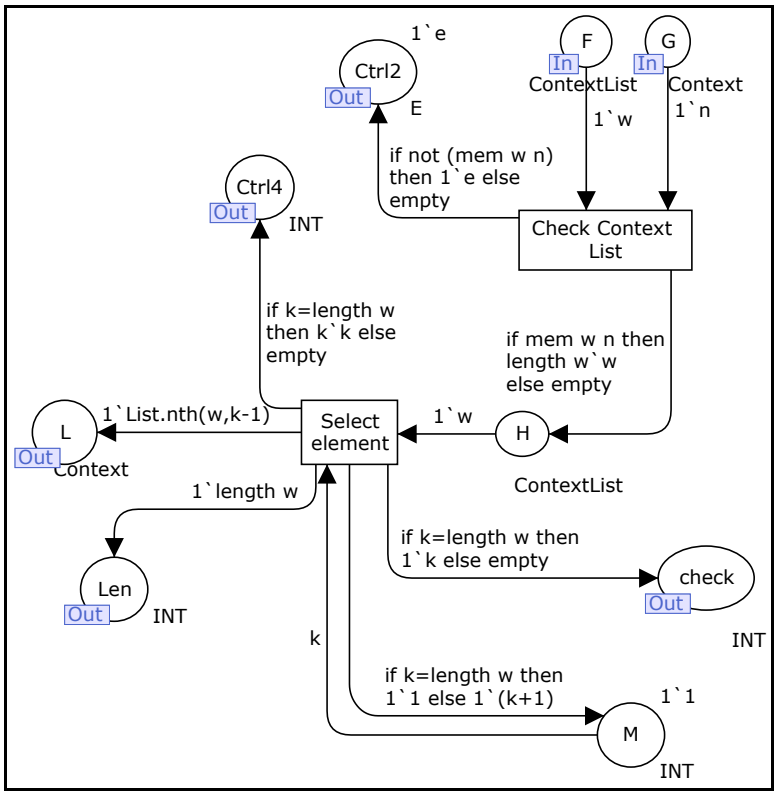

Figure 3.21 Le modèle de vérification d'appartenance au contexte d'un service (1).

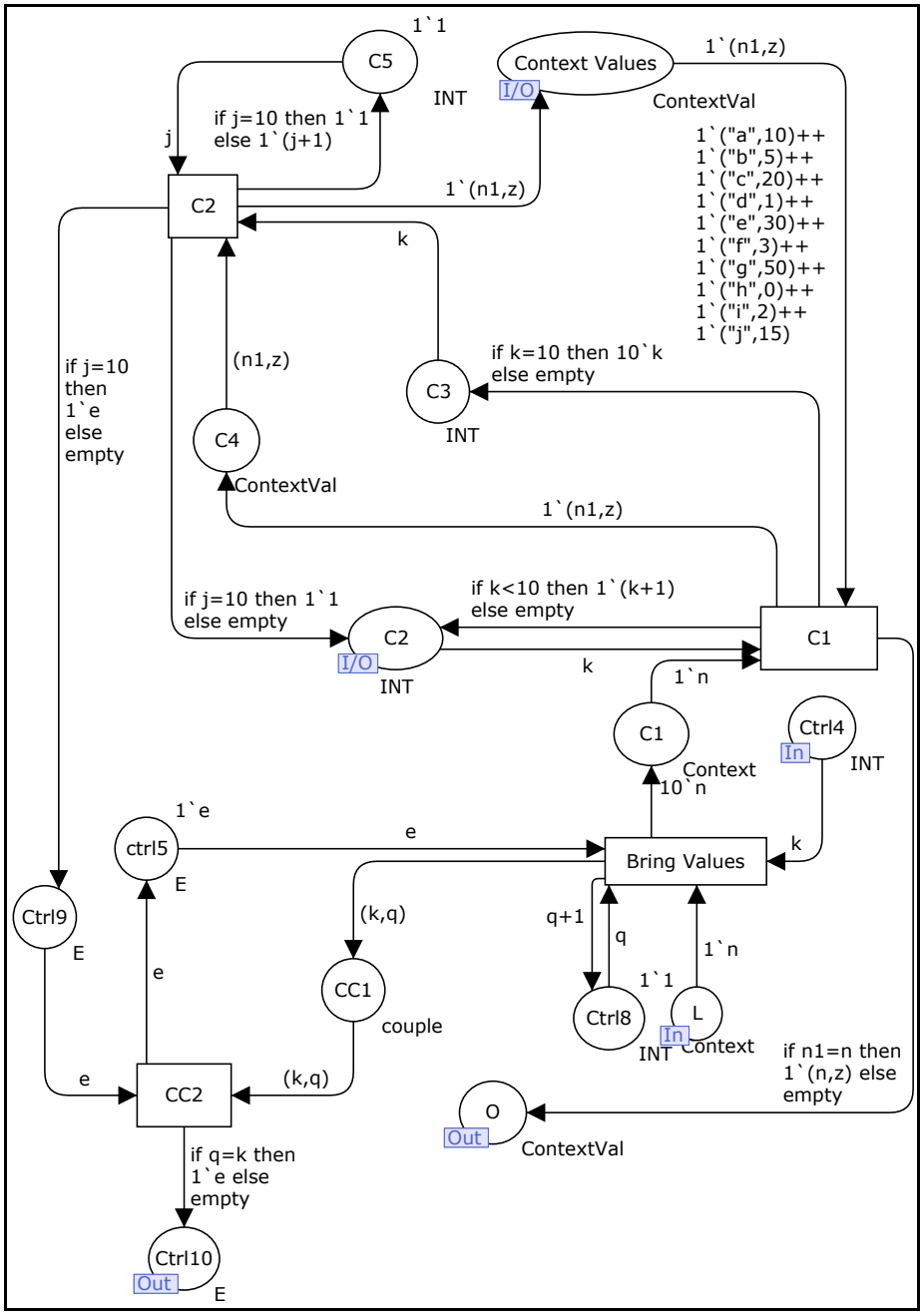

Figure 3.22 Le modèle de vérification d'appartenance au contexte d'un service (2).

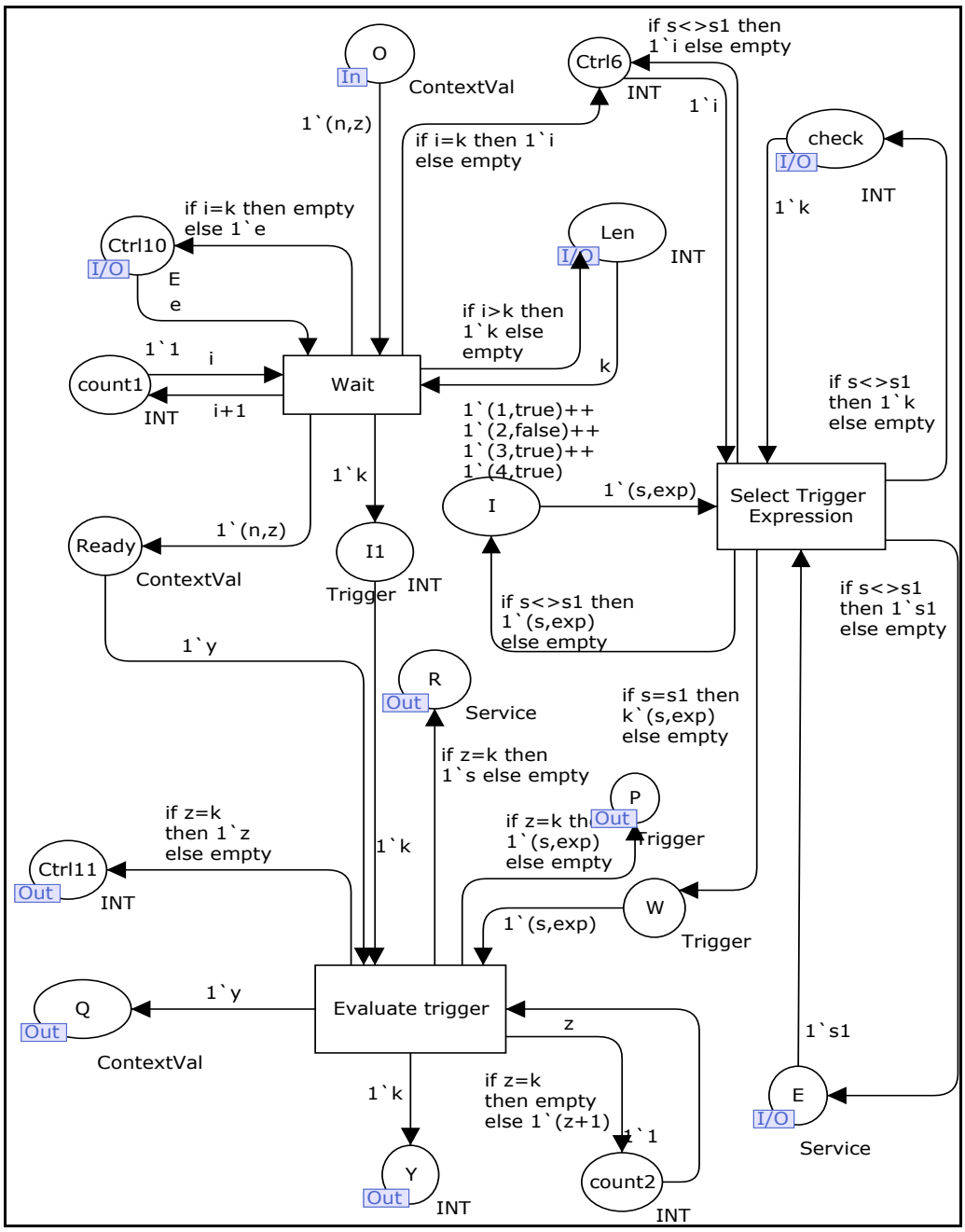

Figure 3.23 Le modèle de sélection et évaluation d'expression de déclenchement d'un service.

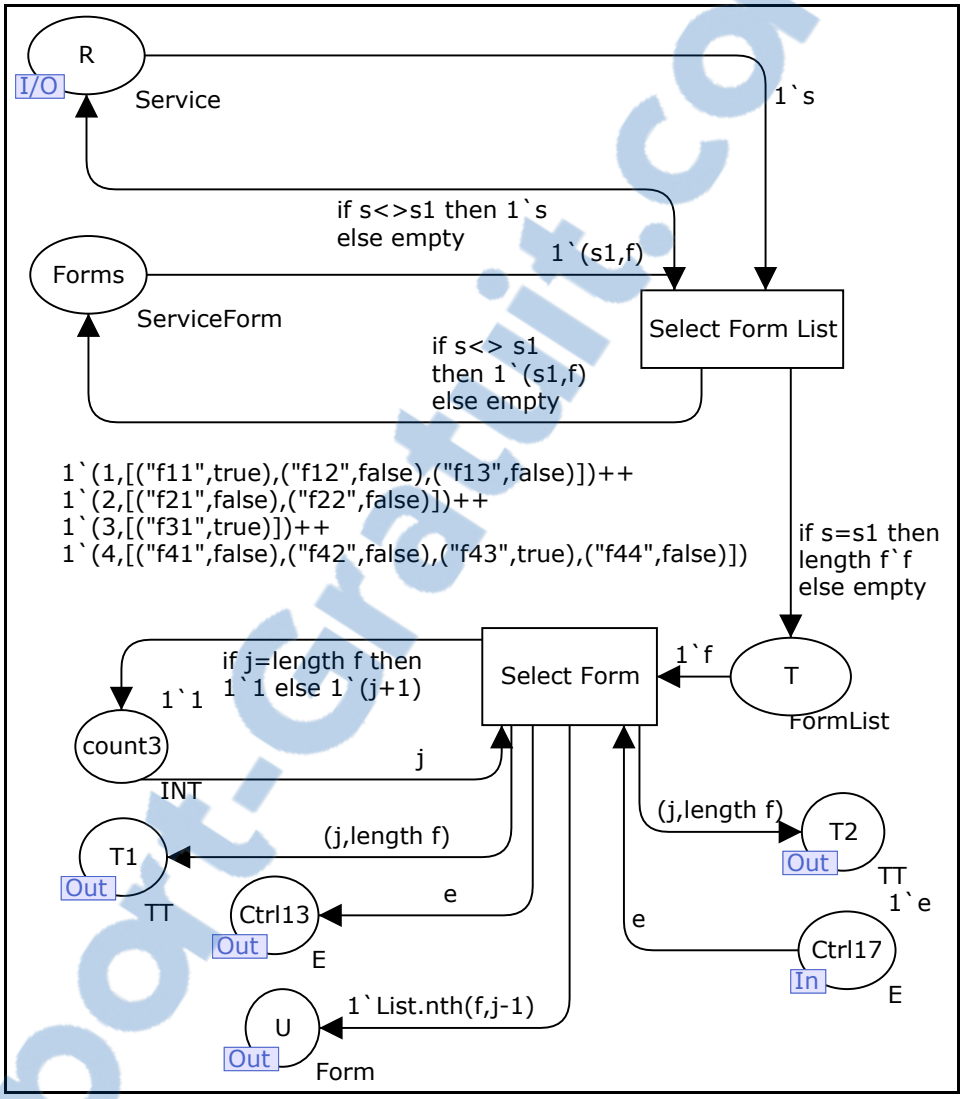

Figure 3.24 Le modèle de sélection d'une forme de service.

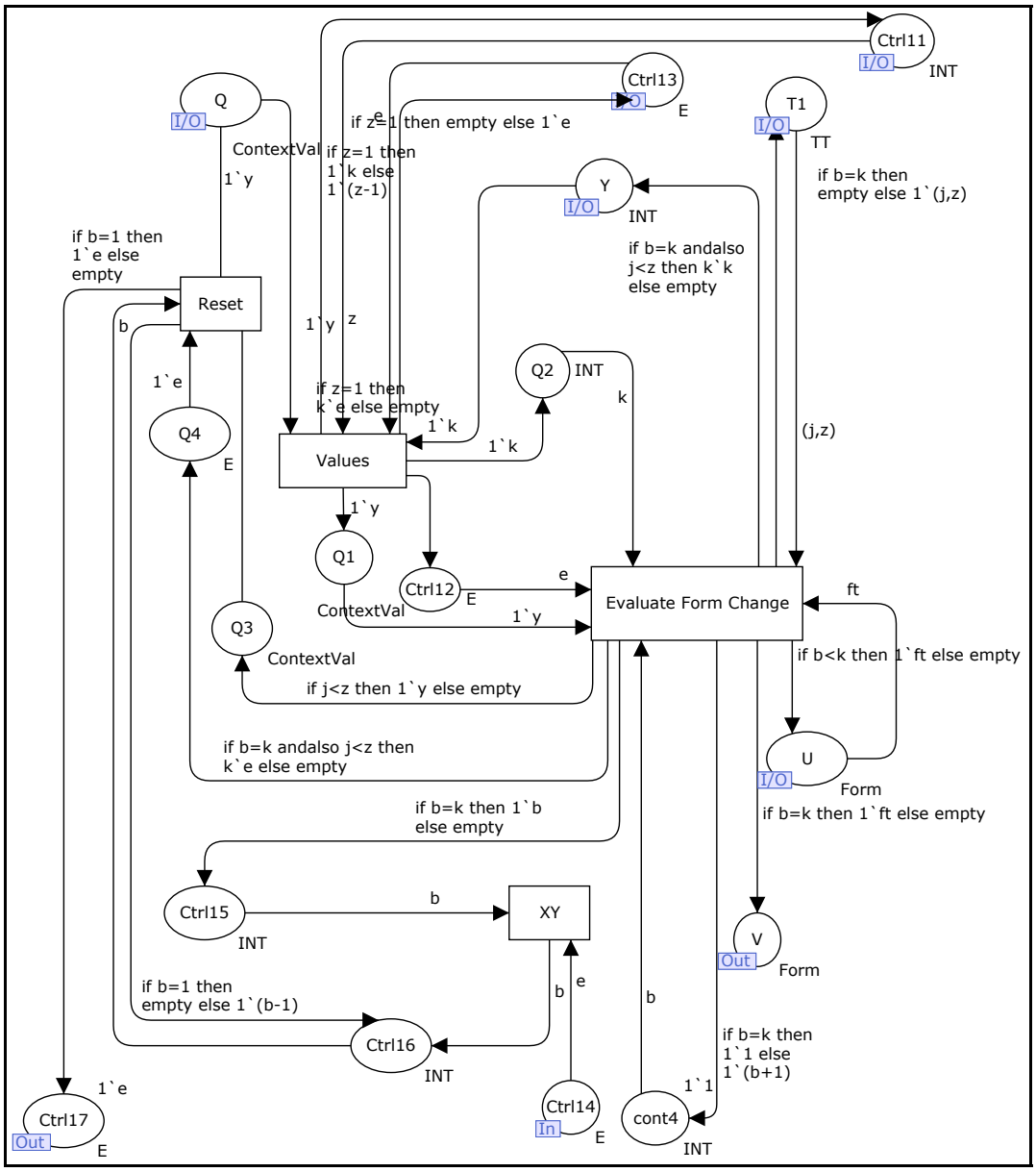

Figure 3.25 Le modèle d'évaluation de l'expression de changement de forme.

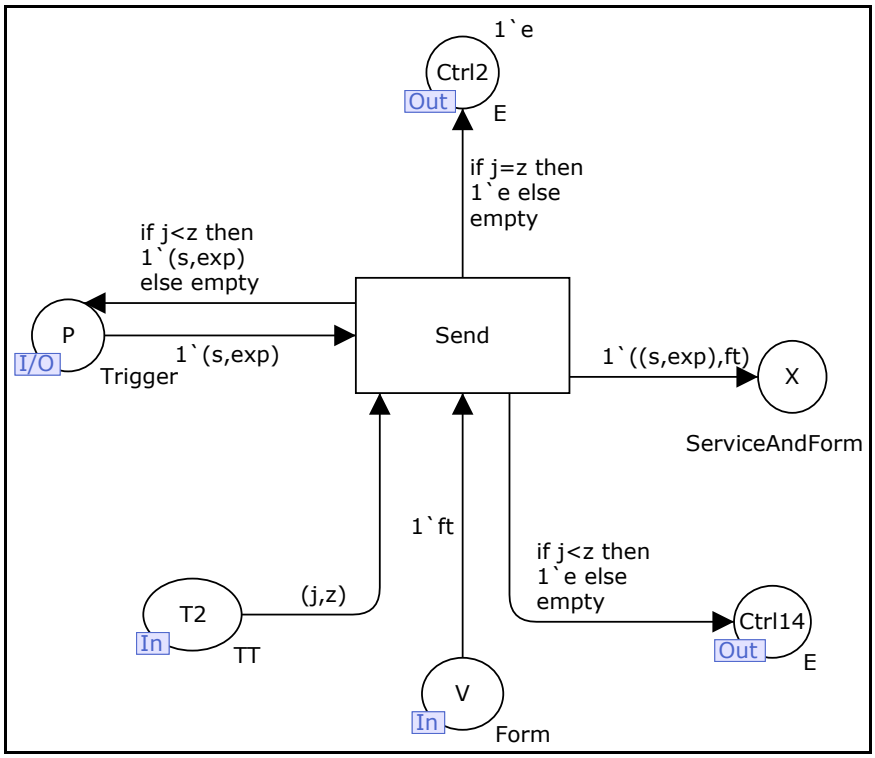

Figure 3.26 Le modèle d'expédition de résultats à l'agent de service.

CHAPITRE 4

MODÉLISATION DU CONTEXTE

Une caractéristique importante des équipements informatiques dans un SID est leurs sensibilités au contexte. Celle-ci leur permet d'offrir proactivement des services adaptés au contexte global (de l'utilisateur, des applications, …). Le contexte doit être bien compris et modélisé sous une forme appropriée pour favoriser son partage entre les différents équipements informatiques et offrir un haut niveau d'abstraction pour faciliter la tâche d'adaptation. Cette modélisation consiste à faire l'analyse et la conception de l'information contextuelle contenue dans le système sous forme d'une représentation abstraite au niveau de la structure de données et au niveau de la sémantique. Plusieurs approches de modélisation ayant des particularités liées à la technique utilisée ont été proposées dans la littérature. Les plus intéressantes sont celles basées sur les ontologies parce qu'elles permettent le partage du contexte et un raisonnement efficace. Dans ce chapitre, nous allons présenter, analyser et classifier ces approches en mettant l'accent sur les approches ontologiques. Nous proposons ensuite notre modèle de contexte à base d'ontologie.

4.1 Travaux connexes de modélisation du contexte

4.1.1 Approches non ontologiques

Attribut/Valeur

La représentation attribut/valeur est la structure des données la plus simple pour la modélisation des informations contextuelles. Elle a été proposée au départ par (Schilit, Adams et Want, 1994) pour la gestion des informations contextuelle d'un environnement. L'attribut est une métadonnée qui décrit la nature (type) de la valeur à laquelle il est associé. Par exemple, pour une application de localisation, une information sur l'occupation d'un espace physiquement délimité (pièce, bâtiment) est représentée par l'égalité suivante :

OCCUPANTS = adams:schilit:theimer:weiser:Welch

Cette approche est très utilisée dans les services distribués (les services sont décrits en général avec une liste d'attributs sous forme d'attribut/valeur et la découverte de service est ensuite appliquée par utilisation d'un algorithme qui utilise ces paires attribut/valeur) (Held, Buchholz et Schill, 2002). Les informations peuvent être sauvegardées dans une base de données sous forme de tables dont les colonnes correspondent à une information représentée par une même métadonnée. chaque ligne (tuple) représente une suite d'informations pour caractériser les différents documents suivant le terminal de l'utilisateur (Tableau 4.1) (Kammanahalli et al., 2004).

Tableau 4.1 Table décrivant des documents.

Activité	**Équipement**	**Type de document présenté**
Courriel	Ordinateur	Entier
Courriel	Ordinateur	Entier
Courriel	Ordinateur	Résumé
Courriel	Ordinateur	Résumé
Lecture	Ordinateur	Entier
Lecture	Ordinateur	Entier
Réunion	PDA	Résumé
Réunion	Mobile	Résumé vocal
Réunion	PDA	Résumé

Ce modèle est facile à gérer mais ne convient pas pour les structures complexes et ne permet pas de faire un raisonnement efficace sur le contexte ; en plus cette représentation est liée à une application particulière et ne peut être réutilisée.

Modèle de représentation par balises

Cette représentation est formée d'une structure de donnée hiérarchique constituée de balises avec des attributs et des contenus. Ces derniers peuvent être définis récursivement par d'autres balises. Comme elle est formée de balises, elle utilise des langages dérivés du SGML (Standard Generic Markup Language) en particulier, le XML (eXtended Markup Language). Les entités du contexte sont exprimées sous forme de pages web qui peuvent être

extraites par un URL (adresse web). Les profils qui décrivent les données et les ensembles de données constituent des exemples typiques de cette approche. Pour faire suivre notre idée, nous allons présenter trois langages qui utilisent cette approche.

- **ConteXtML**

Le ConteXtML (Contexte Markup Language) est un protocole simple basé sur XML pour échanger des informations contextuelles entre un client mobile et un serveur. Les messages ConteXtML sont regroupés dans des balises <context> ou des éléments. Ci-dessous un exemple (extrait 4.1) proposé par Ryan (Ryan, 2007) qui présente un message ConteXtML qui peut être envoyé par un client pour indiquer sa localisation courante (l'élément <spatial>) et qui nécessite une note contenant un élément de donnée nommé « landuse » et ayant une valeur « pasture » (l'élément <require>).

```
<context session="123" action="update">
<spatial proj="UTM" zone="33" datum="Euro 1950 (mean)">
<point x="281993" y="4686790" z="205" />>
</spatial>
<require>
<note>
<data name="landuse" value="pasture" />
</note>
</require>
</context>
```

Extrait 4.1 Exemple un message ConteXtML.

- **CC/PP**

Le langage CC/PP (Composite Capabilities / Preferences Profiles), est une recommandation de W3C (World Wide Web Consortium) dans le cadre des travaux sur « device dependance » pour supporter la négociation de contenu entre un navigateur web et un serveur. Il est basé

sur RDF (Resource Description Framework) qui est utilisé afin de créer des profils décrivant les capacités des terminaux et les préférences d'un agent utilisateur. CC/PP est utilisé pour personnaliser le contenu sur la base de ses capacités et ses préférences. Nous donnons ci-dessous (extrait 4.2) l'exemple d'un profil CC/PP d'un ordinateur portable.

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.etsmtl.ca/2007/04/01-rdf-syntax-ns#"
xmlns:prf="http://www.etsmtl.ca/TR/WD-profile-vocabulary#">
<rdf:Description about="HardwarePlatform">
<prf:Defaults
Vendor="Acer"
Model="TravelMate2428AWXMi"
Type="LapTop"
ScreenSize="14.1"
CPU="IntelPentiumM"
Speed="1.7GHz"
Memory="512mB"
HardDisk="60Gb"
Dvd="Yes"
Wireless="Yes"
Bluetooth="YES"
Speaker="Yes" />
<prf:Modifications
Memory="1gB" />
```

Extrait 4.2 Exemple d'un profil CC/PP d'un ordinateur portable.

```
</rdf:Description>
 <rdf:Description about="SoftwarePlatform">
  <prf:Defaults
   OS="WindowsXP"
   InternetExplorerVersion="7.0"
   Anti-virus="NortonInternetsecurity/>
  <prf:Modifications
   OS="WindowsVista" />
 </rdf:Description>
  <rdf:Description about="UserPreferences">
 <prf:Defaults Language="English"/>
 </rdf:Description>
</rdf:RDF>
```

Extrait 4.2 Exemple d'un profil CC/PP d'un ordinateur portable (suite).

(Held, Buchholz et Schill, 2002) ont proposé une extension de ce langage, le CSCP (Comprehensive Structured Context Profiles) qui ne définit aucune hiérarchie fixe pour résoudre les problèmes de CC/PP. Le CSCP appuie la pleine flexibilité de RDF pour exprimer les structures naturelles d'informations d'un profil requis pour l'information contextuelle. Le CSCP est un métalangage basé aussi sur RDF et hérite de ce dernier, l'interchangeabilité, la décomposabilité et l'extensibilité. Il résout les lacunes de CC/PP concernant la structuration et étend le mécanisme pour exprimer les préférences de l'utilisateur.

(Indulska et al., 2003) ont présenté une approche similaire : le CC/PP Context Extension qui étend le vocabulaire de base de CC/PP et UAProf (User Agent Profile) avec un certain nombre d'arbres composant/attribut relatif à certains aspects du contexte dans l'informatique diffuse. Cependant, selon les auteurs, il reste incapable de représenter des contextes complexes, comme c'est souvent le cas des systèmes diffus.

D'autres approches de modélisation par le langage de balises existent, mais elle sont la plupart de temps spécifique à un domaine et limitées à un ensemble d'aspects du contexte (localisation, environnement, etc.) telles que le PPDL (Pervasive Profile Description Language) (Pascoe, 1997) ou le CCML (Centaurus Capability Markup Language) (Kagal et al., 2001).

Les modèles Graphiques

Cette approche consiste à modéliser les informations contextuelles selon un graphe conceptuel. (Bauer, 2003) a utilisé une représentation graphique basé sur UML (Unified Modeling Language) pour la modélisation des informations contextuelles pour le susteme de contrôle du trafique aérien.

(Henricksen, Indulska et Rakotonirainy, 2002) ont proposé un modèle graphique qui se base sur le formalisme « entité/association ». Les éléments du contexte (utilisateur, machine, réseau, etc.) sont représentés par des entités avec des attributs et en associations entre eux (une association peut aussi avoir des attributs). Ils ont fourni aussi une notation graphique pour leur concept de modélisation.

(Henricksen et Indulska, 2004; Henricksen, Indulska et McFadden, 2005) ont développé une approche de modélisation graphique basée sur la méthode ORM (Object Role Modeling). C'est une méthode orientée « fait » pour l'analyse de l'information au niveau conceptuel, en particulier pour les bases de données relationnelles. La modélisation dans ORM consiste à identifier les types des faits appropriés et les rôles des types d'entités. Selon les auteurs, cette extension est plus formelle et plus expressive pour capter différents types d'informations contextuelles ; elle appuie le raisonnement sur le contexte, décrit bien les informations imparfaites et résout l'ambiguïté dans l'information contextuelle. Cette méthode qui a subi des améliorations est devenue CML (Context Modeling Language) (Henricksen et Indulska, 2006) et apparaît comme une extension vers une représentation basée sur XML, XCML

(Robinson, Henricksen et Indulska, 2007). Un exemple de modélisation par CML est donné par la Figure 4.1.

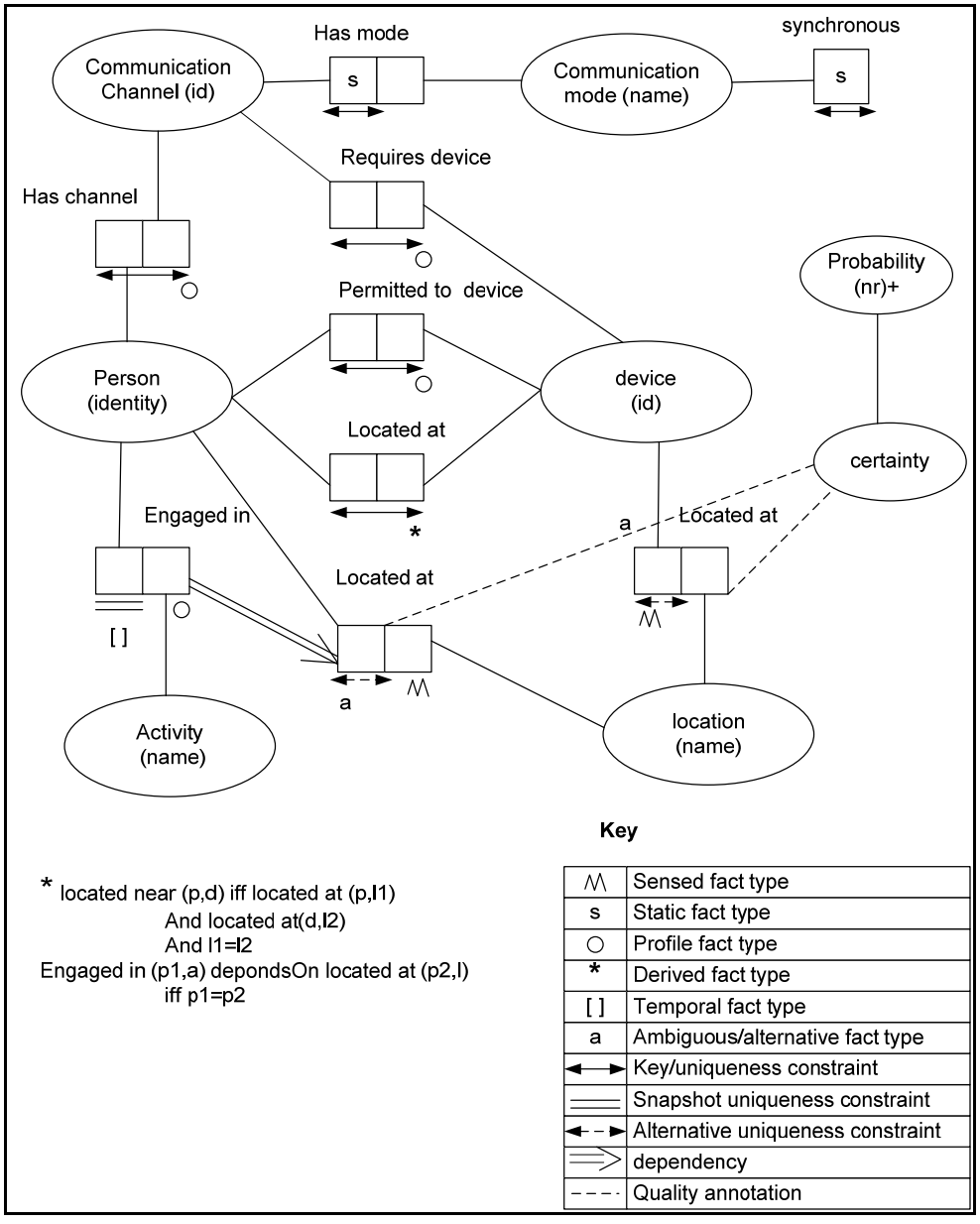

Figure 4.1 Exemple de modélisation du contexte par CML.
Adaptée de Henricksen et al (2005, p. 4)

(Virgilio et Torlone, 2006) ont présenté le GPM (General profile Model) pour la description de représentations hétérogènes de données web d'une manière uniforme. Dans GPM, un

profil est une description d'un aspect autonome du contexte dans lequel le site web est visité et il doit influencer la livraison de son contenu. Des exemples de profils peuvent être l'utilisateur, la machine, la localisation, etc. GPM présente un ensemble limité de primitives de base pour la description graphique d'une représentation conceptuelle du contexte. Les primitives de base sont : profil, dimension, attribut, type de base, choix, séquence ordonnée, séquence non ordonnée et cardinalité. Selon les auteurs, GPM peut être utilisé pour décrire plusieurs contextes d'une manière uniforme et fournit un outil puissant pour la conception et l'analyse des applications sensibles au contexte. De plus, ces auteurs ont défini un processus permettant de traduire les profils d'un modèle à un autre en utilisant GPM comme niveau de représentation intermédiaire. La Figure 4.2 montre les primitives de base de GPM ainsi qu'un exemple de modélisation du contexte par GPM.

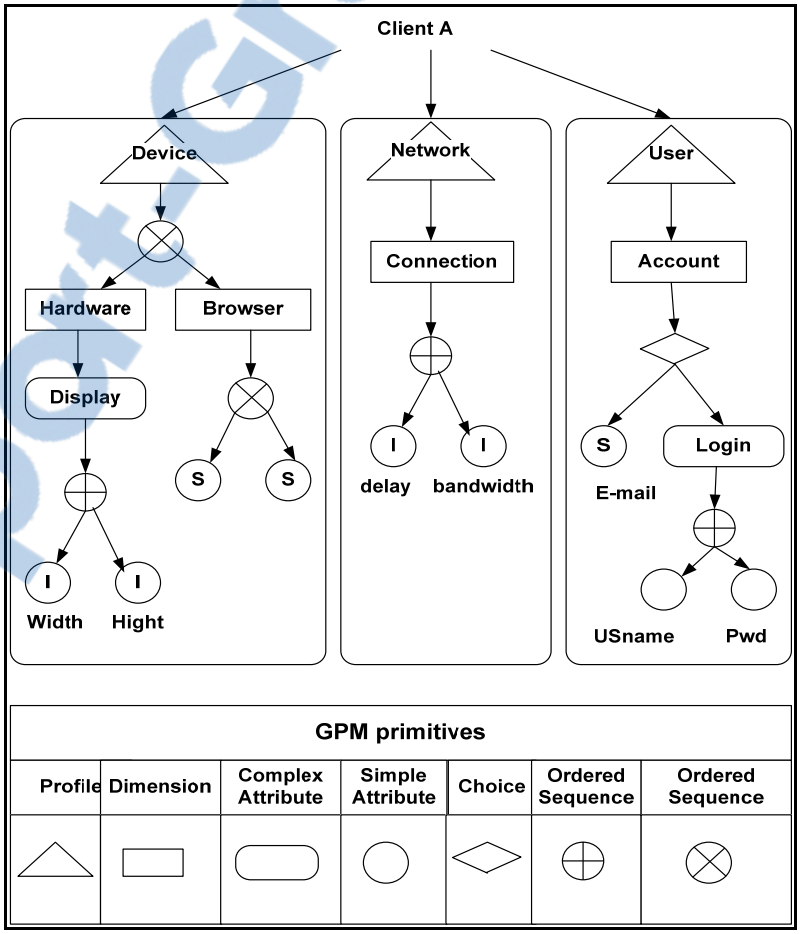

Figure 4.2 Exemple de modélisation du contexte par GPM.
Adaptée de Virgilio et Torlone (2006, p. 3)

Malgré la simplicité de ces modèles au niveau de la représentation et des moyens d'expression du contexte, ils restent moins formels que les autres méthodes et n'offre aucune approche empirique.

Les modèles orientés objets

Le but de la modélisation du contexte par l'approche orientée objet est de profiter de la puissance offerte par les mécanismes de cette technique (encapsulation, réutilisation, héritage,…). (Bouzy et Cazenave, 1997) ont proposé un mécanisme orienté objet pour la modélisation du contexte pour simplifier la représentation des connaissances dans des systèmes complexes comme le cas du jeu GO (un jeu très connu depuis 4000 années au Japon, en Chine et en Corée). Ils ont montré que le contexte à plusieurs types et ont donné des exemples : temporel, but, spécial et global. Ils ont justifié leur approche de modélisation orientée objet par sa capacité d'héritage et de réutilisation. Celles-ci permettent de définir un petit nombre de propriétés, fonctions et règles dans le but de simplifier la représentation de connaissances dans des systèmes complexes.

(Cheverest et al., 2000) ont proposé dans le cadre du projet GUIDE (un guide touristique sensible au contexte) une approche fondée sur l'intégration d'un modèle orienté objet et un modèle d'information hypertexte. Leur modèle est constitué de deux objets distincts : l'objet point de navigation et l'objet localisation. Ils ont limité l'information significative du contexte à la localisation. Ils ne couvrent pas l'aspect général du contexte.

(Hofer et al., 2003) ont introduit l'approche « HYDROGEN ». Ils ont proposé une architecture à 3 niveaux pour la modélisation du contexte dans le domaine de l'informatique mobile. Le contexte est modélisé sous forme des diagrammes de classes UML. Ils ont décomposé le contexte en contexte local et en contexte distant (les machines distantes en communication avec la machine locale). Chaque type de contexte est composé de plusieurs objets contexte qui constitue la superclasse de plusieurs éléments du contexte, notamment : le temps, le réseau, la localisation, l'utilisateur, la machine et autres qui peuvent être ajoutés par héritage de la super classe (Figure 4.3). Cette modélisation offre la possibilité de représenter

le contexte comme une hiérarchie et de décrire chaque élément indépendamment des autres en utilisant la technique d'encapsulation. Cela permet de favoriser la réutilisation de ces éléments dans d'autres applications mais ne donne pas la possibilité de représenter les relations entre les objets. Cette approche de modélisation est bien efficace en termes de distribution et d'abstraction. Cependant, elle reste propre à une application spécifique et ne permet pas de partager le contexte entre applications.

L'approche objet est utilisée pour pouvoir intégrer facilement la représentation du contexte au sein de l'application qui en dépend. Cette représentation du contexte s'appuie sur les propriétés de nommage, d'encapsulation, de réutilisation et d'héritage. Généralement, les termes sont représentés par les classes et les informations par les attributs de la classe. Le détail du contexte est invisible aux autres objets, grâce à la technique d'encapsulation. L'accès et la mise à disposition des informations sont contrôlés grâce aux interfaces des classes.

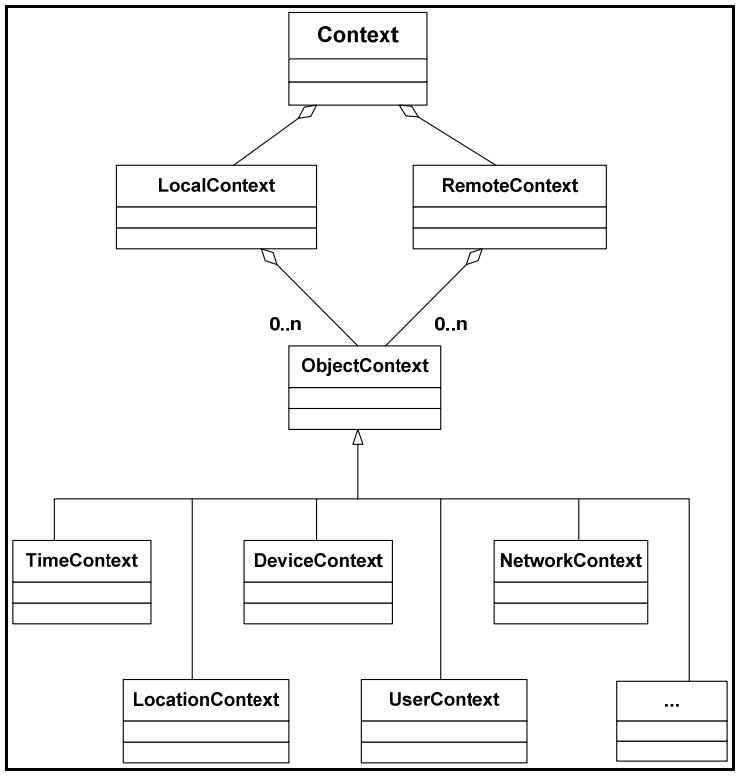

Figure 4.3 Le modèle UML de l'approche Hydrogen.
Adaptée de Hofer at al. (2003, p. 6)

4.1.1.1 Modèles logiques

Dans un modèle logique, les informations contextuelles doivent être représentées d'une façon formelle comme des faits. Un processus de raisonnement ou d'inférence est ensuite utilisé pour déduire des nouveaux faits en se basant sur des règles existant dans le système. Ces méthodes permettent une représentation formelle du contexte et offrent un mécanisme d'inférence, offrant la possibilité d'abstraire les données.

Parmi les premiers travaux de modélisation du contexte utilisant cette approche logique et celui de (Carthy et Buvac, 1994) qui ont introduit le contexte comme objet formel. Leur objectif a été de définir de simples axiomes pour les phénomènes ayant un sens commun et de traiter le contexte associé à une situation particulière. La relation de base dans cette approche est $ist(c, p)$ qui signifie que la proposition p est vraie dans le contexte c. Cela permet de définir des formules telles que :

$$C_0 : ist(context - of(\textit{Sherlock Holmes stories}), Holmes\ is\ detective).$$

Ce modèle utilise aussi la notion d'héritage.

(Ranganathan et al., 2002) ont proposé une modélisation du contexte basée sur la logique de prédicats dans le cadre du projet ConChat. Il s'agit d'un programme de discussion sensible au contexte qui enrichit la communication électronique en fournissant l'information contextuelle et en résolvant les conflits sémantiques entre les utilisateurs. Le modèle permet d'utiliser des opérations sur le contexte telles que la conjonction, la disjonction, la négation et la quantification (universelle et existentielle). Leur approche offre la possibilité de créer des expressions complexes en logique du premier ordre et de déduire du contexte de haut niveau à partir du contexte de base (capturé) en utilisant une approche basée sur des règles. Le modèle définit une structure de base pour caractériser de manière atomique chaque élément du contexte (concepts de base) en tenant compte de la règle suivante :

- ***Context (⟨ContextType⟩, ⟨Subject⟩, ⟨Relator⟩, ⟨Object⟩)***
- ***⟨ContextType⟩*** *: définit le type de contexte défini par le prédicat,*
- ***⟨Subject⟩*** *: faisant référence à un objet, un lieu ou une personne,*

- ***⟨Relator⟩*** *: correspondant à la relation entre objet et sujet,*
- ***⟨Object⟩*** *: la valeur associée avec le sujet.*

La quantification permet d'établir des relations d'ensemble. La combinaison de quantificateurs et d'opérateurs booléens favorise la construction des prédicats plus complexes, comme l'exemple qui suit où l'on exprime qu'une personne entre dans la salle 8 dans laquelle se déroule une réunion :

$$\exists_{personne}\, x,\ Context\ (lieu,\ x,\ entre,\ Salle08)\ \wedge\ (activité,\ réunion,\ dans,\ Salle08).$$

L'application ConChat est implémentée au-dessus de l'infrastructure Gaia qui fournit le maintien et le support de l'information contextuelle via des fournisseurs du contexte. Les valeurs maintenues par ces serveurs sont spécifiques à un domaine du contexte tel que la localisation, la température. La représentation du contexte de cette approche, décrite par la formule précédente, est similaire à la représentation des termes et des propriétés de notre contexte. Par exemple : Contexte (identité, nom, égale, John) regroupe sous le terme identité, l'information nom = John. Cependant, l'interprétation du contexte au-delà de la définition du terme n'est pas possible, car si nous essayons de préciser le domaine utilisateur (contexte utilisateur, utilisateur, possède, identité), la représentation du contexte devient ambiguë. Le paramètre ⟨ContextType⟩ n'a pas de réelle signification : il caractérise à la fois le type de contexte et son sujet.

Les mécanismes de déductions des solutions basées sur une modélisation logique sont les mécanismes les mieux adaptés pour réaliser l'abstraction des informations en concepts. Cependant, l'inférence sur un ensemble de prédicats, est une opération qui nécessite un grand nombre de ressources systèmes. Cela affecte la consommation d'énergie. Inversement, l'informatique diffuse considère généralement des environnements dans lesquels les terminaux sont limités en ressources (terminaux mobiles). Cette représentation non structurée est ambiguë et les solutions qui utilisent ce type de modèle s'appuient souvent sur une gestion du contexte centralisé, solution qui ne s'adapte pas au principe de distribution physique des services dans un environnement informatique diffus.

Une étude (« survey ») des méthodes de modélisation de contexte faite par (Strang et Linnhoff-Popien, 2004) contient une comparaison intéressante de ces méthodes. Elle conclut que les méthodes à base d'ontologies font une meilleure description du contexte par rapport aux autres méthodes du fait qu'elles offrent un bon partage d'informations avec une sémantique commune.

4.1.2 Approches ontologiques

Une ontologie est un ensemble structuré de concepts. Les concepts sont organisés dans un graphe dont les relations peuvent être des relations sémantiques ou des relations de composition et d'héritage. L'objectif d'une ontologie est de modéliser un ensemble de connaissances dans un domaine donné, c'est-à-dire un choix quant à la manière de décrire un domaine sous une forme utilisable par une machine. Elle fournit un vocabulaire représentatif pour un domaine donné, un ensemble de définitions et d'axiomes qui contraignent le sens des termes de ce vocabulaire de manière suffisante pour permettre une interprétation consistante des données représentées au moyen de ce vocabulaire.

Les ontologies constituent de puissantes ressources pour partager la connaissance. Elles ont généré un grand intérêt en intelligence artificielle et une grande variété de disciplines qui sont confrontés au problème d'intégration d'information et d'interprétation de données. Une ontologie Formelle est un ensemble de blocs à partir desquels des modèles du monde sont construits. Un agent ou un robot autonome utilisant un modèle particulier ne peut percevoir que la partie du monde que son ontologie est capable de représenter. Seuls les concepts de son ontologie existent pour cet agent. Une ontologie formelle est le niveau de base d'un schéma de représentation des connaissances. Des ontologies formelles ont été construites pour un nombre croissant de disciplines scientifiques, de l'ingénierie ou de la production (Noy et Hafner, 1997). Par analogie aux modèles orientés objets, les ontologies permettent de structurer les concepts et les propriétés de la même manière que les classes et les attributs.
Le langage OWL (Ontology Web Language) est une récente recommandation du W3C. Il permet de décrire des ontologies. Tout comme CC/PP, le langage OWL est basé sur un

schéma RDF. Les ontologies représentent actuellement une solution favorisée pour modéliser les informations du contexte.

Parmi les travaux s'intéressant à une modélisation par les ontologies, (Chen, Finin et Joshi, 2004; Chen et al., 2004) ont proposé une approche fondée sur l'idée d'un courtier de contexte (Context Broker Architecture ou CoBrA) ; le contexte est entièrement géré et maintenu par un serveur central (d'une bonne performance) qui s'occupe de maintenir l'état du contexte vis-à-vis d'un ensemble de terminaux et d'agents (localisation, activité, etc.). Il définit une collection d'ontologies appelée COBRA-ONT pour la modélisation de contexte dans un environnement d'une salle de rencontre intelligente. COBRA-ONT est exprimée dans le langage OWL qui définit des concepts typiques associés avec des places, des agents et des événements. Les ontologies jouent un rôle important dans CoBrA. Elles aident le courtier de contexte à partager les informations contextuelles avec d'autres agents et lui permet de raisonner sur le contexte. Le développement de COBRA-ONT se focalise sur la création d'ontologies convenables pour la construction de systèmes sensibles au contexte pragmatique. Dans le but d'appuyer le raisonnement sur les ontologies dans CoBrA, les auteurs ont prototypé un moteur d'inférence OWL appelé F-OWL. Ce moteur d'inférence est implémenté en utilisant Flora-2 qui est un langage de base de connaissance orientée objet et une plateforme de développement des applications qui traduit un langage unifié de F-logic, Hilog, et « transaction logic » au moteur de déduction XSB. Les principales caractéristiques de F-OWL comprennent : 1) la capacité de raisonner avec le modèle d'ontologie défini par la dernière recommandation du standard du langage OWL par le W3C, 2) la capacité pour le support de la vérification de consistance de connaissances en utilisant des règles axiomatiques définit dans Flora-2, 3) une interface de programmation d'applications (API) ouverte pour l'intégration des applications Java. La liste complète de classes et propriétés dans COBRA-ONT consiste en 41 classes (exemple : les ressources RDF qui sont de type owl:class) et 36 propriétés (exemple: les ressources RDF qui sont de types owl:ObjectProperty ou owl:DatatypeProperty). L'ontologie est catégorisée en quatre classes distinctes mais constituée de thèmes liés : (i) ontologies des places physiques, (ii) ontologies

pour les agents (humain ou logiciel), (iii) ontologies pour le contexte de locations des agents et (iv) ontologies pour le contexte d'activités des agents.

(Chen et al., 2004) ont proposé une ontologie partageable nommée SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications) pour favoriser le partage de connaissances, le raisonnement sur le contexte et l'interopérabilité dans un système ubiquitaire diffus. SOUPA peut aider les développeurs qui n'ont pas assez d'expériences dans la représentation de connaissances à démarrer rapidement la conception des applications basées sur les ontologies sans avoir besoin de définir l'ontologie à partir de zéro, ce qui leur permet de se concentrer sur les fonctionnalités de l'implémentation du système. Les parties du vocabulaire SOUPA sont adoptées à partir d'un certain nombre de différents consensus d'ontologies. La stratégie de développement de SOUPA est d'emprunter des termes de ces ontologies. Plusieurs termes empruntés par SOUPA sont convertis à des termes étrangers d'ontologies en utilisant le standard « OWL ontology mapping constructs » (owl:equivalentClass and owl:equivalentProperty). Les ontologies qui sont référencées par SOUPA comprennent les Friend-Of-A-Friend ontology (FOAF), DAMLTime, ontologies spatiales OpenCyc, Regional Connection Calculus (RCC), COBRA-ONT, MoGATU BDI ontology et le Rei policy ontology. SOUPA est constitué de deux ensembles liés des ontologies : le noyau SOUPA et l'extension SOUPA. Les ontologies du noyau SOUPA tentent de définir des vocabulaires génériques qui sont universels pour différentes applications informatiques diffuses. L'ensemble des ontologies de l'extension SOUPA, étendue du noyau SOUPA, définit des vocabulaires additionnels pour soutenir des types spécifiques d'applications et fournit des exemples pour l'extension de l'ontologie future. Les ontologies d'extension de SOUPA sont définies selon deux objectifs: (i) définir un ensemble étendu de vocabulaires pour contenir des types de domaines spécifiques d'applications diffuses et (ii) démontrer comment définir de nouvelles ontologies par extension de l'ontologie noyau de SOUPA (core ontology). Actuellement l'extension de SOUPA contient des ontologies expérimentales pour soutenir les applications diffuses sensibles au contexte dans un espace intelligent et la communication des données d'égale-à-égale dans un environnement diffus. La Figure 4.4 montre les éléments de l'ontologie SOUPA.

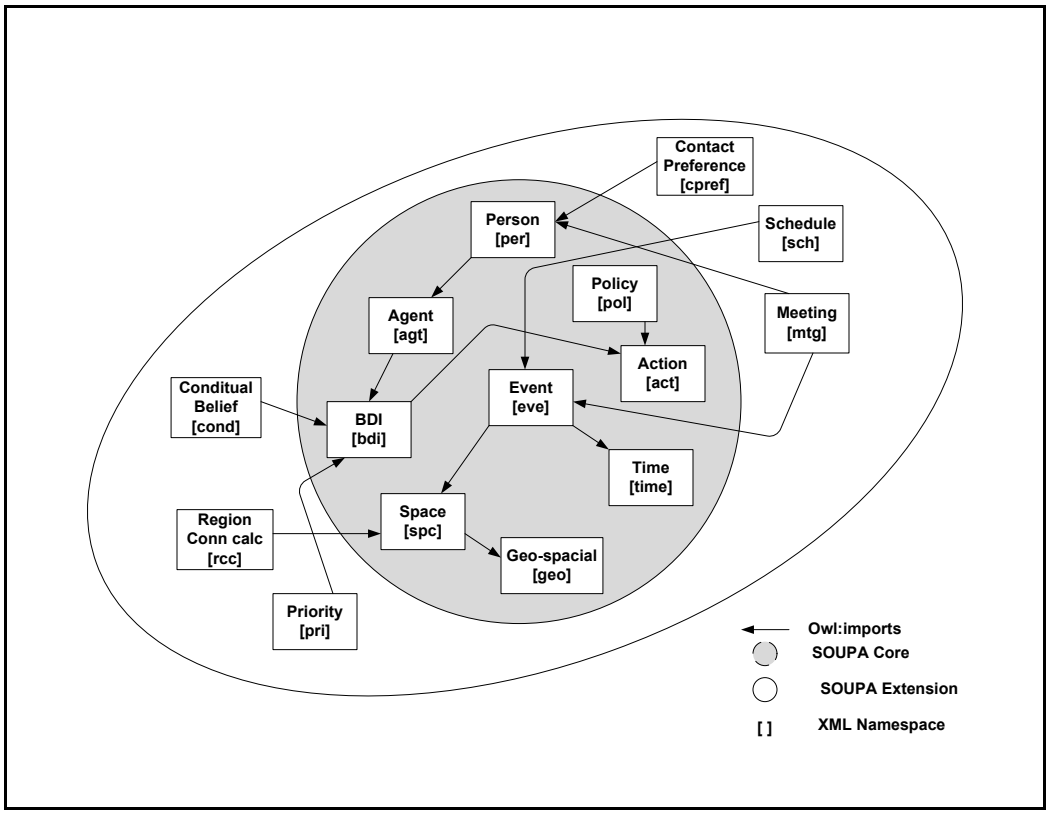

Figure 4.4 Les éléments de l'ontologie SOUPA.
Adaptée de Chen et al. (2004, p. 3)

(H.Wang et al., 2004) ont proposé une ontologie du contexte CONON (CONtext ONtology) codée en OWL pour la modélisation du contexte dans un environnement diffus et pour le support de raisonnement logique sur le contexte. CONON fournit une première ontologie du contexte supérieur qui capte les concepts généraux du contexte de base, et fournit aussi l'extensibilité pour ajouter des ontologies spécifique à un domaine particulier d'une manière hiérarchique. En raison de l'évolution naturelle des applications sensibles au contexte, la formalisation complète de toutes les informations contextuelles est susceptible d'être une tâche surmontable. Cependant les auteurs ont jugé que la localisation, l'utilisateur, l'activité et les entités informatiques forment le contexte fondamental pour capturer les informations sur la situation d'exécution. La deuxième ontologie est une spécialisation de la première pour un domaine du contexte. Elle constitue un ensemble de sous-classes qui viennent se greffer à l'ontologie de base afin de détailler la modélisation suivant un environnement tel que la maison ou le lieu de travail. Alors que l'ontologie de premier niveau définit les termes qui

caractérisent l'environnement d'une manière abstraite, la partie de l'ontologie spécifique au domaine est constituée d'un ensemble de termes concrets (TV, PDA, maison, bureau, cuisine, etc.). L'avantage des ontologies réside dans la possibilité d'effectuer un raisonnement logique grâce à l'apport de nombreux outils. Dans l'approche CONON (Figure 4.5), le moteur de raisonnement est sollicité pour expliciter la description du contexte (par exemple déterminer de manière explicite qu'un utilisateur se trouve dans une salle, implique que cette même personne se situe également dans le bâtiment où se trouve cette salle !). En plus des règles de déductions de bases utilisées pour les ontologies, l'approche CONON permet d'ajouter des règles personnalisées. Ces règles sont écrites grâce à des prédicats associés à la logique de premier ordre. Elles sont introduites pour permettre de caractériser la situation de l'utilisateur.

(Strang, Linnhoff-Popien et Frank, 2003) ont introduit l'approche CoOL (Context Ontology Language). Celle-ci utilise les ontologies et le modèle de contexte ASC (Aspect-Scale-Context). CoOL est dérivé d'un modèle qui peut être utilisé pour permettre la sensibilité au contexte, l'interopérabilité du contexte durant la découverte des services et l'exécution dans un système à architecture distribuée. L'élément principal de cette architecture et le raisonneur qui infère des conclusions sur le contexte basé sur une ontologie construit avec CoOL. Les auteurs n'ont pas défini CoOL comme un seule langage monolithique, mais comme une collection de plusieurs fragments regroupés dans deux sous ensembles :

- OWL et DAML+OIL sont deux langages basés sur XML et RDF/S. Ils font partie des ontologies du web sémantique.
- F-Logic est un langage logique combinant les caractéristiques de l'orienté objet et de la logique de prédicats.

L'utilisation de plusieurs langages d'ontologies permet du point de vue de représentation des connaissances à un développeur d'utiliser un de ces langages qui lui semble adéquat (utiliser OWL à cause de la diversité des outils d'aide à la création/validation des fragments d'ontologies ou utiliser F-logic pour sa syntaxe orientée objet et son extensibilité basées sur les règles. La Figure 4.6 illustre les concepts de CoOL.

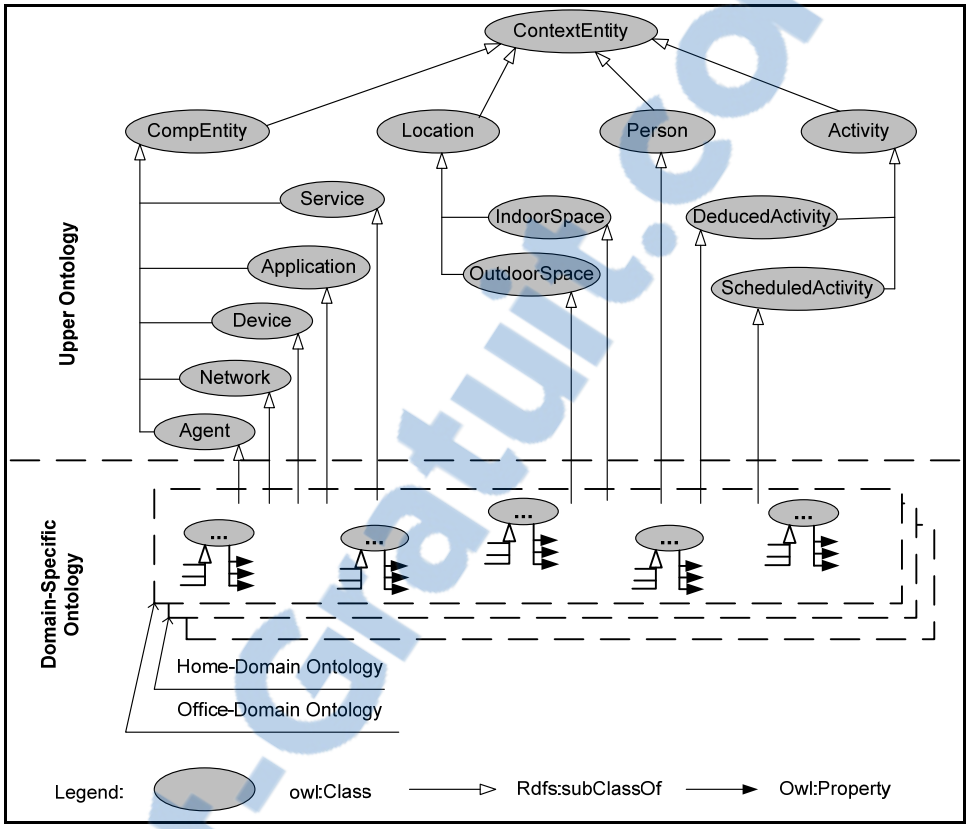

Figure 4.5 L'ontologie à deux niveaux de l'approche CONON.
Adaptée de Wang et al. (2004, p. 2)

Le modèle ASC (Aspect-Scale-Context) est nommé relativement aux concepts centraux du modèle : aspect, échelle et information de contexte. Chaque aspect agrège une ou plusieurs échelles. Chaque échelle agrège une ou plusieurs informations contextuelles. Ces concepts centraux sont reliés par les relations : hasAspect, hasScale et constructedBy. Un aspect est un ensemble d'une ou de plusieurs échelles reliées. De même, chaque aspect est la dimension de l'espace de situation utilisée comme un terme collectif pour information sur des objets ayant le même type sémantique. Une échelle est un ensemble non ordonné. Elle définit l'intervalle des informations contextuelles valides. Une information de contexte qui caractérise le contenu d'une autre information de contexte est une métainformation. C'est donc une information de contexte d'ordre supérieure qui exprime la qualité de l'information

de contexte d'ordre inférieure. Le model ASC peut être utilisé comme un modèle de transfert pour les connaissances exprimées dans un autre modèle.

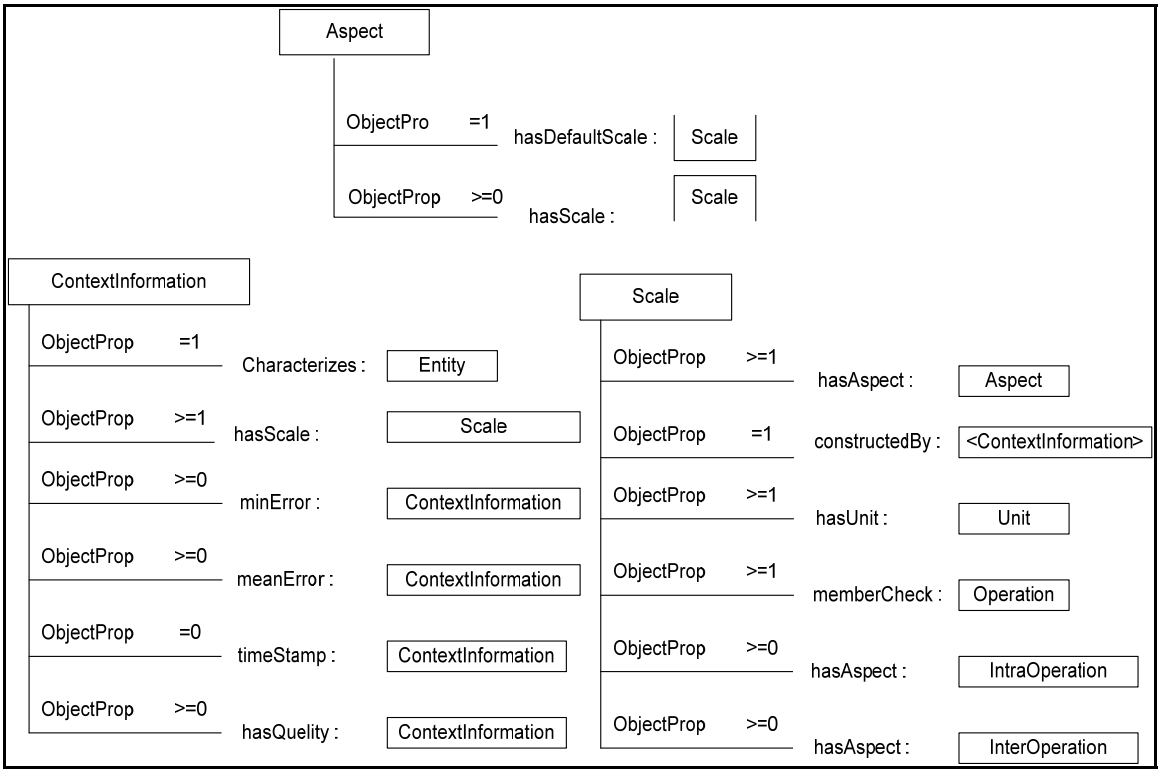

Figure 4.6 Concepts centraux du modèle ASC.
Adaptée de Strang, Linnhoff-Popien et Frank (2003, p. 4)

(Gu et al., 2004b) ont proposé une approche de modélisation du contexte à base d'ontologies dans leur architecture SOCAM (Service-Oriented Context-Aware Middleware). Elle est composée de deux parties : une ontologie supérieure qui capte les connaissances générales du contexte du monde physique d'un environnement diffus. Celle-ci comprend les équipements informatiques, la localisation, l'utilisateur et l'activité. L'ontologie spécifique à un domaine comprend une collection d'ontologies de bas niveau qui définissent les détails de concepts généraux et leurs propriétés. L'ontologie de bas niveau de chaque sous domaine peut être branchée ou débranchée dynamiquement de l'ontologie supérieure quand l'environnement change.

(Preuveneers et al., 2004) ont proposé une ontologie du contexte adaptable et extensible pour la création d'infrastructures sensibles au contexte qui varie des petits systèmes embarqués aux plateformes de services. Actuellement, l'ontologie est exprimée en OWL mais peut être exprimée dans un autre langage d'ontologies. Elle est constituée de quatre ontologies de base : (i) utilisateur : le concept central dans un système sensible au contexte, (ii) environnement : la description des aspects pertinent dans l'environnement de l'utilisateur, (iii) plateforme : matérielle ou logicielle à travers laquelle, un utilisateur interagit avec l'application ou le service, (iv) service : fonctionnalité offerte dans l'environnement utilisateur. La description de ces concepts reste abstraite, ce qui rend cette approche extensible. La description des services basée sur OWL-S, est introduite dans ce modèle afin d'intégrer les services comme éléments du contexte.

Nous avons présenté les approches de modélisation à base d'ontologies qui nous semble les plus intéressantes. Plusieurs autres approches ont été aussi proposées telles que celles qui ont été introduites par (Kown, 2003) et (Kang et al., 2006).

4.1.3 Analyse des modèles ontologiques antérieurs

L'utilisation d'ontologies est particulièrement appropriée pour représenter la connaissance. Nous venons de voir dans le paragraphe précédent que OWL est également intéressant pour modéliser les entités et les relations définissant un contexte. Au-delà de la simple hiérarchisation de données utilisées par les modèles balisés ou objets, les ontologies permettent, comme la modélisation orientée objet, de spécifier les paramètres et les liens qui existent entre les différentes entités du contexte. Les liens peuvent être annotés et possèdent des propriétés. Chaque lien existant entre deux entités peut être défini par une sémantique (dépendance, équivalence, sous-propriété, etc.). C'est grâce à la définition de la sémantique entre les termes qu'il est possible d'effectuer les opérations de raisonnement pour extraire les informations. Les ontologies possèdent quatre propriétés majeures qui expliquent pourquoi elles représentent une solution intéressante pour modéliser le contexte. Ces propriétés sont :

- le partage de la connaissance, grâce à un langage flexible et extensible ;
- le raisonnement logique, pour la déduction de faits implicites au contexte ;
- la description structurée de la connaissance ;
- la sémantique qui caractérise les relations existantes entre concepts.

Cependant, le manque de normalisation pour exprimer les éléments du contexte rend le partage de la connaissance complexe entre différents terminaux. Ainsi, bien qu'il soit possible de faire interopérer plusieurs services web de différentes plateformes, il est difficile de réaliser une exécution dépendante du contexte d'un service distant, lorsque la description du contexte entre plateformes est différente.

La plupart des modèles ontologiques proposés n'offrent pas une description complète des informations contextuelles. Dans le but d'offrir une ontologie extensible et réutilisable, ces approches suggèrent l'utilisation de deux catégories d'ontologies. Une première composée d'ontologies de concepts de base et une seconde adaptée selon le domaine d'application. Mais l'ontologie de base doit être utilisée même si elle ne présente pas d'utilité pour l'application. L'ontologie de base (ou ontologie noyau) diffère d'une méthode à une autre et ses classes de base dépendent de la définition du contexte adoptée par les auteurs. L'ontologie d'extension (ou ontologie spécifique à un domaine) exige un effort supplémentaire de la part des développeurs pour l'adapter à une application spécifique. Les modèles proposés ont échoué dans la présentation d'une ontologie qui couvre tous les aspects du contexte. Ils peuvent contenir des ontologies inutiles lors de leur utilisation dans une application particulière, ce qui limite leurs employabilités et leurs extensibilités (par exemple l'ontologie de localisation est inutile dans une application sensible au contexte qui opère dans un espace local). Nous pensons que ceci est un problème commun pour toutes les ontologies. Dans le cas de modélisation du contexte, la généricité de l'ontologie est fortement liée à la définition du terme contexte : une définition générique implique alors une ontologie générique.

Notre objectif est de proposer une ontologie de service qui couvre tous les aspects du contexte dans un SID, suffisamment générique et facile à utiliser dans diverses applications sans efforts considérables d'adaptation (Miraoui, Tadj et Amar, 2008b).

4.2 Proposition d'une ontologie de service

Dans un environnement diffus, le modèle de contexte doit fournir une compréhension commune d'informations contextuelles entre les différents équipements qui le composent. Cela a pour effet de rendre les équipements sensibles au contexte et permet un bon partage de leurs informations. Une ontologie générique ne doit pas être basée sur des aspects prédéfinis du contexte pour favoriser son employabilité et son extensibilité parce que ces aspects varient d'une application à une autre selon le domaine d'application. L'ontologie de contexte ne devrait saisir que les informations contextuelles requises pour la tâche d'adaptation de services ainsi qu'aucun autre type d'information inutile. Chaque équipement informatique dans un système diffus fournit un ensemble de services. L'objectif principal est d'adapter ces services selon le contexte global en les déclenchant automatiquement selon le contexte actuel ou en changeant automatiquement les formes des services fournies, sachant que chaque service a un ensemble de formes (par exemple le service d'indication des appels entrants pour un téléphone cellulaire peut être fourni sous différentes formes : sonneries, vibreur, silencieux, sonneries avec le vibreur, etc.). Nous avons proposé (Miraoui et Tadj, 2007b) une définition du contexte orientée service (voir chapitre 3) qui nous semble assez générique, ayant un bon niveau d'abstraction qui aide au plus haut point à limiter l'ensemble des informations contextuelles pour effectuer la tâche d'adaptation. En s'appuyant sur cette définition et sur le concept du service, qui joue un rôle crucial dans un SID, nous allons proposer une ontologie de service pour modéliser des informations contextuelles. L'ontologie du contexte ne devrait contenir que les éléments de base qui sont communs à n'importe quel SID. Notre approche de conception est fondée sur l'idée générale suivante :

« Dans un SID, un équipement informatique fournit un ensemble de services à l'utilisateur et aux applications. Chaque service peut être fourni sous plusieurs formes. Il renferme un

ensemble de capteurs intégrés de différents types qui lui permettent de recueillir, avec d'autres capteurs du SID, l'ensemble des informations contextuelles. L'ensemble de ces informations contextuelles déclenchera les services ou changera leurs formes si un sous-ensemble de ces informations change ses valeurs. »

En se basant sur cette description générale, nous allons proposer une ontologie de service. L'approche de construction de cette ontologie est fondée sur l'extraction des principaux concepts, des relations qui existent entre ces concepts (relations d'équivalence, hiérarchiques, etc.) et des instances de concepts. Cette ontologie est composée de cinq classes : équipement informatique, capteur avec deux sous classes (capteur intégré et capteur autonome), service, forme et contexte. La description de chaque classe de l'ontologie se présente comme suit :

- **Équipement informatique** : c'est l'élément de base d'un SID. Il fournit des services à l'utilisateur et aux applications. Il peut renfermer différents types de capteurs intégrés pour collecter des informations contextuelles. Chaque équipement a ses propres caractéristiques et capacités selon son utilisation (par exemple : ordinateur de bureau (fréquence d'unité centrale de traitement, mémoire, capacité de disque dur, etc.), téléphone cellulaire (niveau de batterie, résolution graphique, taille, mémoire, etc.)). Les équipements informatiques dans un système diffus devraient pouvoir communiquer les uns avec les autres afin d'échanger les informations contextuelles pour faire l'adaptation nécessaire des services.

- **Capteur** : peut être intégré à un équipement informatique ou autonome. C'est une source importante des informations contextuelles puisqu'il permet de recueillir beaucoup de types d'informations contextuelles (lumière, température, localisation, temps, etc.). Un équipement informatique peut contenir plusieurs types de capteurs.

- **Service** : il est fourni par un équipement informatique à l'utilisateur et aux applications. Chaque service a une certaine fonctionnalité et peut être fourni sous plusieurs formes. Un service possède un ensemble d'informations de déclenchement qui permettent de le

démarrer quand cet ensemble prend certaines valeurs spécifiques. Un service a également certaines caractéristiques comme la durée, les entrées, les sorties, etc.

- **Forme** : c'est la façon de fournir un service ou la qualité d'un service. Chaque forme d'un service spécifique a ses propres caractéristiques (médias utilisés, modalité utilisée etc.). Pour chaque forme d'un service, il y a un ensemble d'informations de changement qui permettent de changer d'une forme de service à une autre quand une information prend certaines valeurs spécifiques.

- **Contexte** : un ensemble d'informations recueillies par les différents capteurs du SID. Celui-ci spécifie la situation actuelle qui regroupe des informations et des événements qui se sont produits dans le système diffus. Il y a deux catégories d'informations contextuelles : a) informations de déclenchement de services dont le changement de leurs valeurs démarre les services et b) informations de changement de formes dont le changement de leurs valeurs change les formes des services selon les formes actuelles.

D’après ces descriptions, nous avons dégagé l’ensemble des relations qui peuvent exister entre les différentes classes (concepts) de notre ontologie de services. La Figure 4.7 montre les classes de notre ontologie et les relations entre elles.

Selon l’équipement informatique utilisé, nous pouvons énumérer l'ensemble des services qui peuvent être fournis par cet équipement aussi bien que les différentes formes de chaque service. De la même manière, nous pouvons énumérer pour chaque service, l'ensemble des informations qui le déclenchent si leurs valeurs changent et pour chaque forme de ce service l'ensemble des informations qui permettent de changer d'une forme à une autre si leurs valeurs changent. Ces informations feront partie du contexte. Seulement ce type d'informations sera contenu dans la classe contexte.

Les relations entre ces classes existeront quel que soit le type de l’équipement informatique utilisé. Cela nous permet d'affirmer que cette ontologie est plus au moins générique. Il n'y a

donc aucun besoin de définir une ontologie de base (ou ontologie noyau) et une ontologie d'extension (ou ontologie spécifique à un domaine) parce que cela varie selon l'application. En plus, il est très facile de déterminer et de limiter l'ensemble d'informations contextuelles en employant notre définition précédente du contexte (Miraoui et Tadj, 2007b).

Un avantage majeur de la modélisation du contexte à base d'ontologies réside dans la possibilité d'effectuer des raisonnements logiques. Ces derniers permettent de déduire de nouvelles informations contextuelles appropriées qui ne peuvent pas être explicitement fournies par des capteurs du SID. Le raisonnement logique sur les ontologies permet de faire trois tâches principales. Ces dernières consistent :

1. À trouver les contradictions et conflits du contexte provoqués par les captures imparfaites d'informations contextuelles ;
2. À vérifier si les concepts sont non contradictoires ;
3. À trouvez les rapports de subsomption entre les classes et les instances (par exemple un utilisateur situé dans sa salle de bain veut dire qu'il est localisé à la maison aussi, parce que sa salle de bain fait partie de sa maison).

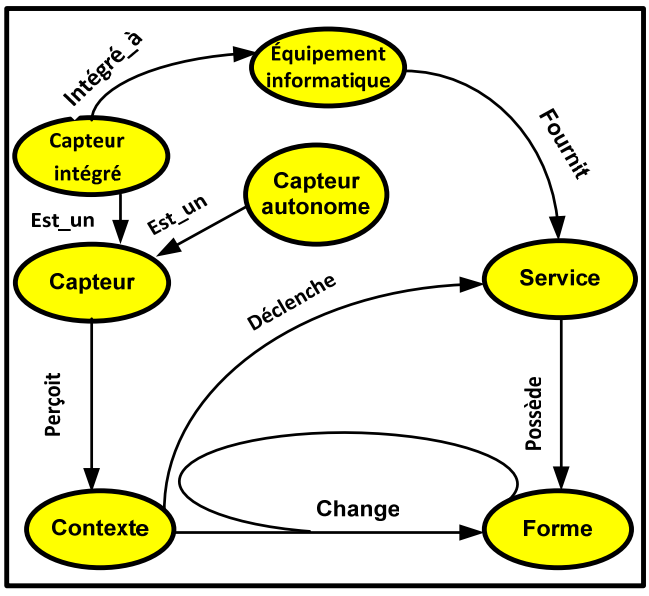

Figure 4.7 Les éléments de l'ontologie de service.

Il existe plusieurs outils automatisés pour faire un raisonnement logique sur les ontologies. Dans notre travail de recherche, nous avons employé le moteur de raisonnement « PELLET » (Parsia et Sirin, 2004) pour vérifier la consistance de notre ontologie et pour déduire des nouvelles informations implicites du contexte. Nous présentons ci-dessous le code OWL de notre modèle de l'ontologie de service.

```
<owl:Ontology rdf:about=""/>
<owl:ObjectProperty rdf:ID="Attaché_à">
<rdfs:domain
rdf:resource="#CAPTEUR_INTEGRÉ "/>
<rdfs:range
rdf:resource="#ÉQUIPEMENT"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="CAPTEUR">
<rdfs:subClassOf
rdf:resource="&owl;Thing"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#Perçoit  "/>
<owl:someValuesFrom
rdf:resource="#CONTEXTE"/>
</owl:Restriction>
</rdfs:subClassOf>
<owl:disjointWith
rdf:resource="#CONTEXTE"/>
<owl:disjointWith
rdf:resource="#ÉQUIPEMENT"/>
<owl:disjointWith
rdf:resource="#SERVICE"/>
<owl:disjointWith
rdf:resource="#FORME"/>
</owl:Class>
<owl:Class
rdf:ID="CAPTEUR_AUTONOME">
<rdfs:subClassOf
rdf:resource="#CAPTEUR"/>
</owl:Class>
<owl:Class rdf:ID="CAPTEUR_INTEGRÉ
">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty
rdf:resource="#Attaché_à"/>
<owl:someValuesFrom
rdf:resource="#ÉQUIPEMENT"/>
<rdfs:subClassOf
</rdfs:subClassOf>
</owl:Restriction>
rdf:resource="#CAPTEUR"/>
</owl:Class>
```

Extrait 4.3 Code OWL de notre ontologie.

```
<owl:ObjectProperty rdf:ID="Change">
<rdfs:domain
rdf:resource="#CONTEXTE"/>
<rdfs:range rdf:resource="#FORME"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="CONTEXTE">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#Change"/>
<owl:someValuesFrom
rdf:resource="#FORME"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty
rdf:resource="#Déclenche"/>
<owl:someValuesFrom
rdf:resource="#SERVICE"/>
</owl:Restriction>
<owl:disjointWith
</rdfs:subClassOf>
<rdfs:subClassOf
rdf:resource="&owl;Thing"/>
<owl:disjointWith
rdf:resource="#ÉQUIPEMENT"/>
rdf:resource="#FORME"/>
<owl:disjointWith
rdf:resource="#SERVICE"/>
<owl:disjointWith
rdf:resource="#CAPTEUR"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="Déclenche">
<rdfs:domain
rdf:resource="#CONTEXTE"/>
<rdfs:range rdf:resource="#SERVICE"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="ÉQUIPEMENT">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty
rdf:resource="#Fournit"/>
<owl:someValuesFrom
rdf:resource="#SERVICE"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf
rdf:resource="&owl;Thing"/>
<owl:disjointWith
rdf:resource="#CONTEXTE"/>
<owl:disjointWith
rdf:resource="#FORME"/>
<owl:disjointWith
rdf:resource="#SERVICE"/>
<owl:disjointWith
rdf:resource="#CAPTEUR"/>
</owl:Class>
```

Extrait 4.3 Code OWL de notre ontologie (suite).

```
<owl:Class rdf:ID="FORME">
<owl:disjointWith
rdf:resource="#ÉQUIPEMENT"/>
<owl:disjointWith
rdf:resource="#CONTEXTE"/>
<owl:disjointWith
rdf:resource="#SERVICE"/>
<owl:disjointWith
rdf:resource="#CAPTEUR"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="Fournit">
<rdfs:domain
rdf:resource="#ÉQUIPEMENT"/>
<rdfs:range rdf:resource="#SERVICE"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Perçoit  ">
<rdfs:domain>
<owl:Class>
<owl:unionOf
rdf:parseType="Collection">
<owl:Class
rdf:about="#CAPTEUR_AUTONOME"/>
<owl:Class
rdf:about="#CAPTEUR_INTEGRÉ "/>
</owl:unionOf>
</owl:Class>
</rdfs:domain>
<rdfs:range
rdf:resource="#CONTEXTE"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Possède">
<rdfs:domain rdf:resource="#SERVICE"/>
<rdfs:range rdf:resource="#FORME"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="SERVICE">
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty
rdf:resource="#Possède"/>
<owl:someValuesFrom
rdf:resource="#FORME"/>
</owl:Restriction>
</rdfs:subClassOf>
<rdfs:subClassOf
rdf:resource="&owl;Thing"/>
<owl:disjointWith
rdf:resource="#FORME"/>
<owl:disjointWith
rdf:resource="#ÉQUIPEMENT"/>
<owl:disjointWith
rdf:resource="#CONTEXTE"/>
<owl:disjointWith
rdf:resource="#CAPTEUR"/>
</owl:Class>
</rdf:RDF>
```

Extrait 4.3 Code OWL de notre ontologie (suite).

4.3 Scénario d'application

Pour mettre en évidence la clarté, la cohérence et l'employabilité de notre ontologie de service, nous allons présenter dans ce chapitre un scénario d'application. Dans ce scénario, l'équipement informatique est un téléphone cellulaire. Nous montrons dans ce qui suit les quatre principales étapes à suivre afin de modéliser le contexte pour un service particulier. Les mêmes étapes peuvent être suivies pour tout autre service.

Étape 1 : spécification des services

Dans le cas général, le processus de modélisation commence en spécifiant pour chaque équipement informatique du système diffus, l'ensemble des services qu'il peut fournir. Dans notre scénario, nous allons prendre, comme exemple de service fourni par un téléphone cellulaire, l'indication des appels entrants (modélisation d'informations contextuelles pour un équipement et un service particulier).

Étape 2 : spécification des formes de services

Cette étape du processus de modélisation consiste à spécifier pour chaque service, l'ensemble de formes sous lesquelles, le service peut être fourni. Dans notre cas et par défaut, le téléphone cellulaire indique des appels entrants en employant des sonneries (forme par défaut) mais pour une raison ou une autre, il peut indiquer les appels entrants sous d'autres formes comme suit :

- **Vibreur** : quand l'utilisateur (nous supposons qu'il est un étudiant) est dans la bibliothèque de l'université (pour éviter de déranger d'autres étudiants et de respecter les consignes) ou dans une salle de classe (les heures d'étude sont extraites de l'emploi de temps de l'étudiant. Nous supposons qu'il est sauvegardé dans la mémoire du téléphone cellulaire) ;

- **Sonneries avec vibreur** : quand l'utilisateur est dans un endroit bruyant, cette forme sera employée pour attirer l'attention de l'utilisateur ;
- **Silencieux** : quand l'utilisateur dort (nous supposons que l'utilisateur dort habituellement de minuit à 8 h et son environnement est obscure avec faible bruit).

Dans un SID, les équipements portables ont en général des ressources limitées, en particulier la charge de batterie. Pour cette raison, si celle-ci est au niveau bas, le service (indication d'appel) sera fourni sous la forme silencieuse quelle que soit la forme commode. Cela est fait pour éviter de vider la batterie (pile) et pour maintenir le téléphone cellulaire en fonction aussi longtemps que possible. La Figure 4.8 montre le diagramme de transition des formes pour le service d'indication d'appels entrants.

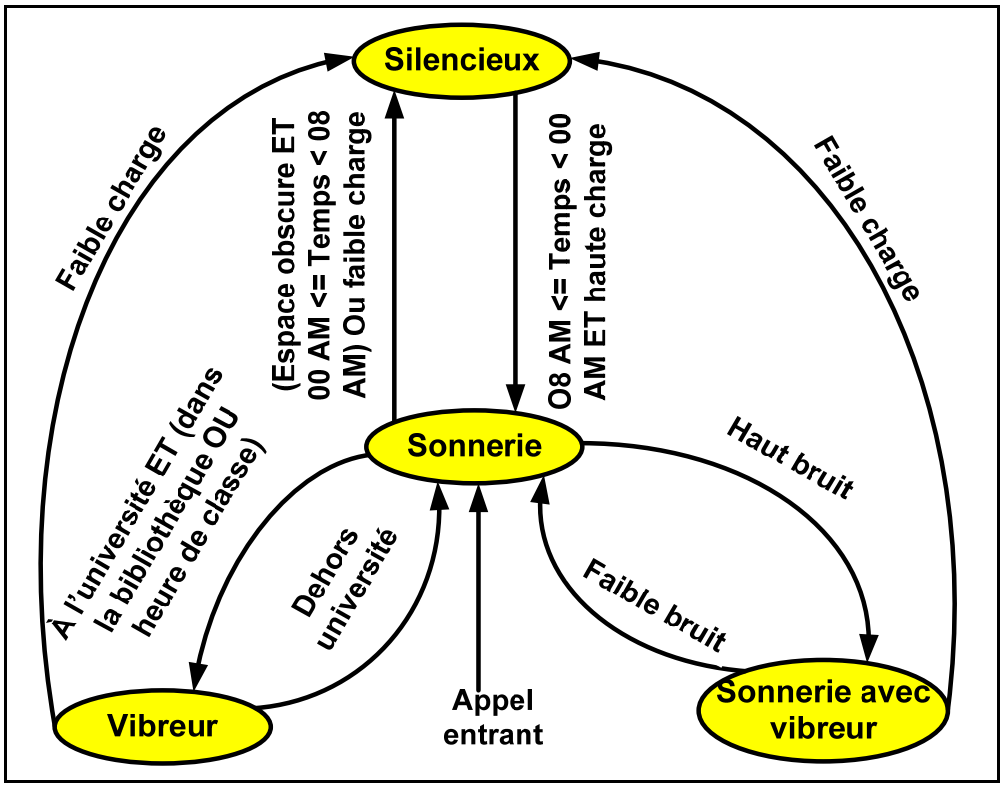

Figure 4.8 Diagramme de transition du scénario d'application.

Étape 3 : spécification des informations contextuelles

Cette étape du processus de modélisation consiste à spécifier pour chaque service, l'ensemble des informations dont le changement de valeurs déclenche le service ainsi que l'ensemble des informations dont le changement de valeurs change la forme du service. Ces informations constituent le contexte global de ce scénario. Il est clair que notre méthode aide à limiter l'information contextuelle seulement à celles qui sont liées à l'application et qui ne contiennent aucune information inutile. Nous avons fait cela explicitement lors de spécification des formes du service (indication des appels entrants) dans la deuxième étape. Le Tableau **4**.**2** donne le résultat des trois étapes.

Étape 4 : représentation graphique de l'ontologie de service

La dernière étape consiste à faire la représentation graphique de l'ontologie du service d'indication des appels entrants selon le modèle de l'ontologie de service présenté dans la section précédente (Figure 4.7). Ce graphique est ensuite traduit en un fichier OWL codé dans XML après vérification de sa consistance.

Il est clair que notre méthode permet un bon partage de la connaissance contextuelle entre les équipements informatiques du système diffus. La Figure 4.9 montre les composants de l'ontologie de service du scénario d'application.

Nous avons employé PROTÈGÈ 3.4 bêta pour l'implémentation de l'ontologie de service de ce scénario. Nous avons obtenu un fichier OWL codé en XML (voir annexe 1) qui présente la caractéristique forte de la portabilité et permet l'échange standardisé de données structurées. Cela favorise le partage des connaissances contextuelles entre les équipements qui composent le SID.

Tableau 4.2 Le résultat des trois étapes de modélisation

Équipement	Service	Informations de déclenchement	Informations de changement de formes	Forme
Téléphone mobile	**Indication des appels entrants**	**Appel entrant**	Niveau de charge = faible	sonnerie
			Forme actuelle = sonnerie	
			Forme actuelle = vibreur	
			Forme actuelle = sonnerie avec vibreur	
			Localisation = université	Vibreur
			Localisation = bibliothèque	
			Forme actuelle = sonnerie	
			Niveau de bruit = haut	Sonnerie avec vibreur
			Localisation = à l'extérieur de l'université	
			Forme actuelle = sonnerie	
			Niveau lumière = bas	Silencieux
			Temps = 00 AM et 08 AM	
			Localisation = à la maison	
			Forme actuelle = sonnerie	
			Localisation = université	Vibreur
			temps = heures de classe	
			Forme actuelle = silencieux	
			Niveau de charge = haut	sonnerie
			Temps = 08 AM et 00 Am	
			Forme actuelle = silencieux	
			Localisation = à l'extérieur de l'université	Sonnerie
			Forme actuelle = vibreur	
			Niveau de bruit = bas	sonnerie
			Forme actuelle = sonnerie avec vibreur	

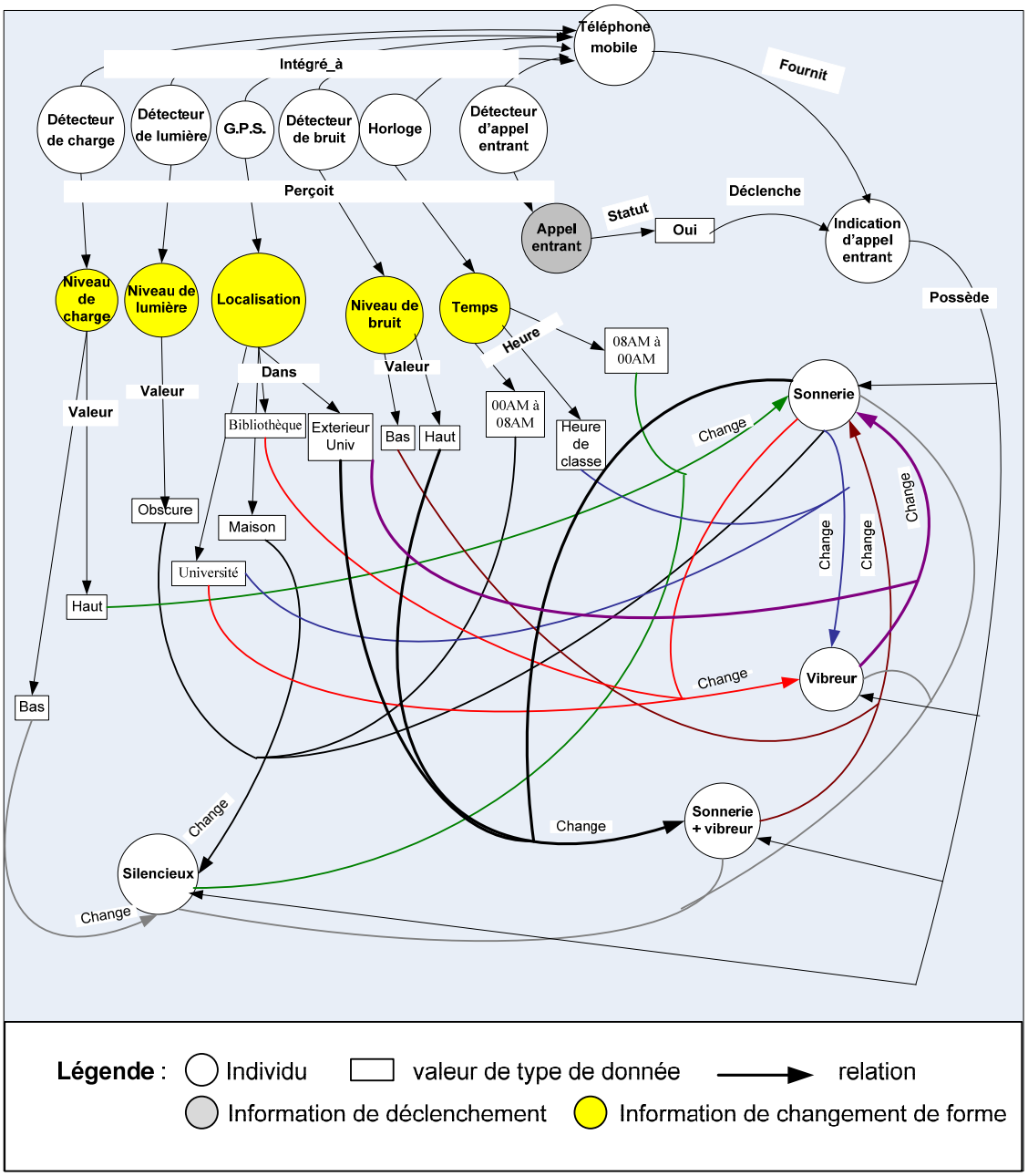

Figure 4.9 Les composants de l'ontologie de service du scénario d'application.

CHAPITRE 5

ADAPTATION DE SERVICES SELON LE CONTEXTE

5.1 La notion d'adaptation

Les équipements informatiques dans un SID doivent être sensibles au contexte. Ceci exige une bonne compréhension et utilisation du contexte. La tâche d'adaptation de services doit être faite selon ce contexte. Principalement, il y a quatre genres d'adaptation : adaptation de contenu, adaptation du comportement (service), adaptation de présentation (ou interface) et adaptation de composants logiciels. Nous limiterons nos approches à ceux qui sont en relation avec l'adaptation de services selon le contexte. Le besoin de l'adaptation de services a été longtemps identifié (Aksit et Choukair, 2003; Efstratiou et al., 2001; Fayad et Cline, 1996; Keeney et Cahill, 2003; Narayanan, Flinn et Satyanarayanan, 2000; Segara et André, 2000; South, Lenaghan et Malyan, 2000; Yarvis, Reiher et Popek, 1999). Plusieurs approches manuelles et automatiques ont été proposées pour accomplir l'adaptation. Généralement, la tâche d'adaptation de service consiste à déclencher un service ou le changement de la forme d'un service si quelques événements se produisent dans l'environnement de l'équipement informatique. Les méthodes conventionnelles d'adaptation de services emploient essentiellement les structures conditionnelles. Une condition se compose de plusieurs variables représentant les entrées de système combinées avec les opérateurs logiques. Une fois qu'une condition devient vraie, le système devrait se comporter d'une façon bien définie. En général, toutes les alternatives doivent être préparées avant que le système ne soit opérationnel, ce qui signifie que la tâche principale du développeur est de prévoir toutes les conditions possibles, ce qui n'est pas toujours évident ! Un autre inconvénient de ces méthodes est leur aspect statique, qui ne prend pas en considération la nature fortement dynamique d'un SID causée par la mobilité des utilisateurs et des équipements. En effet, plusieurs paramètres changent soudainement lors de fonctionnement (exécution) de système. En plus, les méthodes existantes d'adaptation de services ne sont pas vraiment sensibles au contexte, alors que cette caractéristique est un facteur clé des équipements informatiques

dans un système diffus. Les approches existantes d'adaptation qui sont sensibles au contexte demeurent superficielles et ne traitent pas le problème en profondeur. Elles ne parviennent pas à établir un couplage fort entre l'adaptation de service et la sensibilité au contexte.

Les travaux connexes ne considèrent pas les ressources très limitées des équipements informatiques dans un système diffus (qui sont en général portatifs), comme le niveau de charge de batterie (pile), la mémoire, la capacité de traitement, etc., alors que l'adaptation de services dans un SID doit prendre en considération ces contraintes, c'est-à-dire les ressources limitées des équipements informatiques et la sensibilité au contexte. Dans le but de surmonter les problèmes ainsi que les limitations des méthodes précédentes, nous allons proposer deux approches d'adaptation dynamique de services selon le contexte dans un SID.

5.2 Approche d'adaptation fondée sur un apprentissage automatique

5.2.1 Méthodologie

Un SID est composé d'un ensemble d'équipements informatiques communiquant ensemble afin de fournir proactivement des services adaptés à l'utilisateur et aux applications. Chaque équipement du SID fournit plusieurs services et chacun a plusieurs formes sous lesquelles il peut être fourni. Un service ou une forme de service est fourni quand une configuration de contexte se produit. L'adaptation selon le contexte consiste à associer une configuration du contexte à un service ou à une forme d'un service d'une manière automatique. Pour des raisons de simplification, nous allons limiter notre approche à un équipement qui comprend plusieurs services et plusieurs formes pour chaque service. La même approche peut être appliquée pour tous les équipements du SID. Les étapes de l'approche seront détaillées dans les paragraphes qui suivent.

Détermination des composants du contexte

Pour chaque équipement informatique du système diffus et en employant notre définition orientée service (Miraoui et Tadj, 2007b) proposée dans le chapitre 3, nous allons spécifier pour chaque service, l'ensemble des informations qui, si leurs valeurs changent, déclenche le service. De la même manière, nous allons spécifier l'ensemble des informations qui, si leurs valeurs changent, change la forme de service. Le contexte global sera composé de ces deux types d'information.

Détermination des configurations du contexte

Pour chaque élément du contexte, nous spécifierons l'ensemble des valeurs possibles (par exemple niveau de lumière (haut, bas), localisation (à la maison, à l'université, etc.). Selon le nombre d'éléments du contexte (qu'on suppose égale à N), nous obtiendrons N vecteurs de différentes tailles. La tâche suivante consiste à énumérer toutes les configurations possibles du contexte. Ce processus fournira des vecteurs de taille variables contenant toutes les valeurs possibles des éléments du contexte. Cette tâche peut être automatisée en employant un algorithme simple qui produirait toutes les configurations possibles des vecteurs d'éléments du contexte. La Figure 5.1 montre des éléments du contexte contenant différentes valeurs et configurations possibles selon la notation suivante :

E_i **:** élément de contexte i $(i = 1 \ldots N)$.

Max_i**:** le nombre maximal d'éléments dans E_i**.**

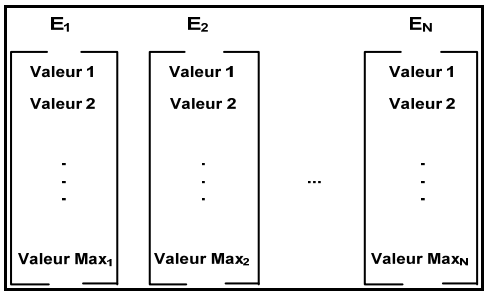

Figure 5.1 Éléments du contexte contenant différentes valeurs et configurations possibles du contexte.

Le nombre total de configurations possibles pour les éléments du contexte de la Figure 5.1 peut être calculé selon cette formule :

$$Max = \prod_{i=1\ldots N} Max_i \tag{6.1}$$

Alors la configuration du contexte est définie par :

$$CC_i\left(Valeur_{i_1}(E_1), Valeur_{i_2}(E_2), \ldots, Valeur_{i_N}(E_N)\right),$$
$$i = 1, \ldots, Max, i_1 = 1, \ldots, Max_1, i_2 = 1, \ldots, Max_2 \ \ldots \ i_N = 1, \ldots, Max_N$$

Chaque configuration du contexte peut être interprétée comme conjonction de différentes valeurs des éléments de contexte.

L'ensemble de vecteurs des configurations de contexte peut inclure quelques configurations de contexte sans signification. Par exemple supposons que nous avons une configuration de contexte composée de trois éléments : connexion réseau ("active/inactive"), localisation (place 1, place 2, place 3, place 4) et vitesse de connexion (haut, bas), la configuration du contexte peut contenir les configurations suivantes :

CC_1 (active, place 1, haut).
CC_2 (active, place 2, bas).
CC_3 (inactive, place 3, haut).
CC_4 (inactive, place 4, bas).

Les configurations CC_3 et CC_4 n'ont aucune signification parce que si la connexion réseau est inactive, nous ne pouvons parler de la vitesse de la connexion. Nous avons besoin de certains procédures pour éliminer les configurations non significatives.

Simplification des configurations du contexte

Une configuration du contexte est une conjonction de valeurs d'éléments qui la compose :

$Valeur_{i_1}(E_1), Valeur_{i_2}(E_2), \ldots, Valeur_{i_N}(E_N)$.

Pour éliminer une configuration du contexte, nous allons utiliser certaines règles contradictoires composées d'au moins deux valeurs contradictoires de deux éléments du contexte. La procédure prendra ces règles une par une et passera en revue toutes les configurations du contexte et éliminera celles qui contiennent les éléments de la règle contradictoire. Dans l'exemple ci-dessus du contexte composé de la connexion réseau, de la localisation et de la vitesse de connexion (Connexion réseau, localisation, vitesse de connexion), nous pouvons choisir comme règles contradictoires :

1. (inactive,?, haut).
2. (inactive,?, bas).

Le symbole **?** Signifie quelle que soit la valeur.

Nous éliminerons ainsi les configurations du contexte CC_3 et CC_4 de l'ensemble des configurations possibles du contexte. Cette étape comprend deux tâches : la première doit être faite par le développeur pour déterminer l'ensemble des règles contradictoires ; la seconde peut être exécutée automatiquement en utilisant un algorithme de recherche simple.

Apprentissage et association

Cette étape consiste à associer chaque configuration du contexte à la forme appropriée de service, c'est-à-dire selon les valeurs des éléments du contexte le service sera fournie sous une forme spécifique. Cette tâche sera réalisée pour chaque service et pour chaque équipement informatique du système diffus. Pour chaque service d'un équipement

informatique du système diffus, nous allons construire l'ensemble des configurations du contexte qui permet de changer la forme du service quand cette configuration de contexte se produit. Cet ensemble de configurations du contexte servira d'ensemble d'apprentissage à un algorithme d'apprentissage supervisé. Le modèle entraîné associera alors pour chaque configuration de contexte la forme sous laquelle le service sera fourni.
En raison des ressources limitées d'un équipement informatique (portatif) dans un système diffus (charge de batterie (pile), traitement, mémoire…), le processus d'apprentissage sera effectué hors ligne (en différé) et le résultat de l'apprentissage sera sauvegardé dans la mémoire de l'équipement informatique avant son utilisation dans le système diffus.

5.2.2 Scénario d'application

Pour mettre en évidence cette approche d'adaptation de services selon le contexte, nous allons utiliser le même scénario d'application déjà utilisé dans le chapitre précédent (cf. définition du contexte au paragraphe 2.2.2) afin de prouver la complémentarité des processus de modélisation et celui d'adaptation. Dans ce scénario, l'équipement informatique est un téléphone cellulaire. Nous montrerons les étapes à suivre pour l'adaptation dynamique de service d'indication d'appels entrants. Les mêmes étapes peuvent être suivies pour tout autre service.

Dans une première étape, le processus d'adaptation consiste à collecter l'ensemble des informations qui changent les formes du service (indication d'appels entrants) si leurs valeurs changent selon les éléments suivants :

- Localisation ;
- Temps ;
- niveau de bruit ;
- niveau lumineux ;
- niveau de charge de batterie.

Ces informations composeront le contexte global de ce scénario.

La deuxième étape consiste à spécifier pour chaque élément du contexte, l'ensemble de ses valeurs possibles. Elle se présentera comme suit :

- localisation (salle de classe, bibliothèque, maison, à l'extérieur de l'université) ;
- temps (heures de sommeil, heures de classe, heures libres) ;
- niveau de bruit (haut, faible) ;
- niveau de lumière (haut, faible) ;
- niveau de charge de batterie (haut, faible).

Évidement, certaines valeurs de ces éléments de contexte exigent de l'interprétation préliminaire. Nous supposons que les heures de sommeil sont comprises entre minuit et 8h. Des seuils prédéfinis sont utilisés pour mesurer les différents niveaux (bruit, lumière, charge de batterie, etc.).
La troisième étape consiste à générer toutes les configurations possibles du contexte en se basant sur différentes valeurs des éléments de contexte. Dans notre cas, il y a quatre-vingt-seize (96) configurations possibles du contexte dont chacune se compose de cinq éléments.

La prochaine étape consiste à spécifier les règles contradictoires pour éliminer des configurations du contexte non significatif. Dans notre scénario, nous pouvons définir les règles suivantes :

- Aux heures de sommeil, la localisation ne peut pas être la salle de classe ou la bibliothèque. Ainsi, nous devons éliminer toutes les configurations du contexte contenant les couples (salle de classe, heures de sommeil) et (bibliothèque, heures de sommeil) ;
- Aux heures libres, la localisation ne peut être la salle de classe. Ainsi, nous devons éliminer toutes les configurations du contexte contenant les couples (salle de classe, heures libres).

Après l'application du procédé de simplification, il est resté soixante-douze (72) configurations possibles du contexte.

La prochaine étape consiste à déterminer l'ensemble des configurations du contexte qui permet de changer la forme du service (quand cette configuration de contexte se produit) et qui servira ensuite comme ensemble d'apprentissage à un algorithme d'apprentissage supervisé. Dans notre scénario, cet ensemble se compose de configurations du contexte comme l'indique le Tableau 5.1.

Tableau 5.1 Configurations du contexte et formes de services

Configuration du contexte	**Forme de service**
?,?,?,?, faible charge	silencieux
en dehors,?, haut bruit,?,haute charge	sonnerie avec vibreur
salle de classe,?,?,?, haute charge	vibreur
bibliothèque,?,?,?, haute charge	vibreur
à la maison, heures de sommeil,bas bruit, bas lumière,?	silencieux
en dehors,?, bas bruit,?, haute charge	sonnerie

Le Symbole **?** Signifie quelle que soit la valeur.

La dernière étape consiste à employer le modèle d'apprentissage pour prévoir les formes de toutes les configurations possibles du contexte.

Pour l'implémentation du scénario d'application, nous avons utilisé l'outil Weka (Weka 3: Data Mining Software in Java). Il contient une collection d'algorithmes d'apprentissage automatique. Nous avons choisi parmi cette collection, l'algorithme d'apprentissage le plus célèbre et le plus intéressant : le réseau de neurones multicouches ayant l'architecture et les caractéristiques présentées par le Tableau 5.2.

Après exécution de notre programme, nous avons fait une vérification manuelle des sorties prévues. Nous avons remarqué que parmi les soixante-douze (72) configurations possibles, il y avait seulement une erreur de la forme prévue de service (voir annexe 2). La configuration du contexte (à la maison, heures de sommeil, faible lumière, faible bruit et charge élevée) produit la forme de service sonnerie au lieu de silencieux (la forme correcte). Ce bas taux d'erreurs nous permet d'affirmer que notre approche présente de bons résultats pour l'adaptation des services selon le contexte.

Tableau 5.2 Architecture et caractéristiques du réseau de neurone utilisé

Légende	**Caractéristiques**
Entrées	Cinq neurones, un pour chaque élément du contexte {localisation, temps, niveau de bruit, niveau de lumière et niveau de charge de batterie}
Sorties	Quatre neurones, un pour chaque forme de service (classe) {silencieux, sonnerie, vibreur, sonnerie avec vibreur}
Nombre de couches cachées	Une seule couche cachée
Fonction d'activation	La fonction sigmoïde
Algorithme d'apprentissage	Apprentissage supervisé avec rétropropagation du gradient
Taux d'apprentissage	0,7
Terme inertiel (momentum)	0,3
Ensemble d'apprentissage	10
Ensemble de test et validation	72
Temps d'apprentissage	1000 ms
Temps pour construire le modèle	0,61 seconds
Erreur moyenne absolue	0,0129

5.3 Approche d'adaptation sensible aux ressources limitées

5.3.1 Méthodologie

L'adaptation dynamique de services consiste à fournir des services déclenchés manuellement et des services déclenchés automatiquement (selon le contexte actuel) en prenant en considération les ressources limitées de l'équipement. Cette adaptation doit se faire d'une manière transparente à l'utilisateur et au cours du fonctionnement du système (dynamique). L'idée principale est de fournir des services sans épuiser les ressources limitées de l'équipement informatique. Dès qu'un service récemment déclenché (manuellement ou automatiquement) épuise une ressource limitée, le système cherche une autre configuration des services fournis qui évite l'épuisement des ressources limitées. Cela permet d'augmenter la durée de fonctionnement de l'équipement informatique. L'architecture du système d'adaptation est fondée sur trois composants : le déclencheur sensible au contexte, le moniteur de ressources limitées et l'adaptateur comme le montre la Figure 5.2.

Pour faire suite à notre approche, nous allons détailler chacun des composants de l'architecture du système d'adaptation et le flux de communication entre les composants.

Déclencheur sensible au contexte

Un de plus importants composants de l'architecture du système d'adaptation est le déclencheur sensible au contexte (DSC). Celui-ci recueille dynamiquement des informations contextuelles en utilisant les capteurs de l'équipement informatique et en se basant sur notre définition précédente du contexte (cf. définition du contexte au paragraphe 2.2.2). La tâche du développeur consiste à déterminer pour chaque équipement, l'ensemble des services qu'il peut fournir et pour chaque service l'ensemble des formes sous lesquelles le service peut être fourni. Par exemple, l'ordinateur portable fournit le service de la connexion internet sous deux formes : câblée ou sans fil. Les informations contextuelles sont fixées en utilisant notre

définition précédente du contexte. Le changement des valeurs des informations déclenchent un service ou changent la forme d'un service (Figure 5.3 et
Tableau 5.3).

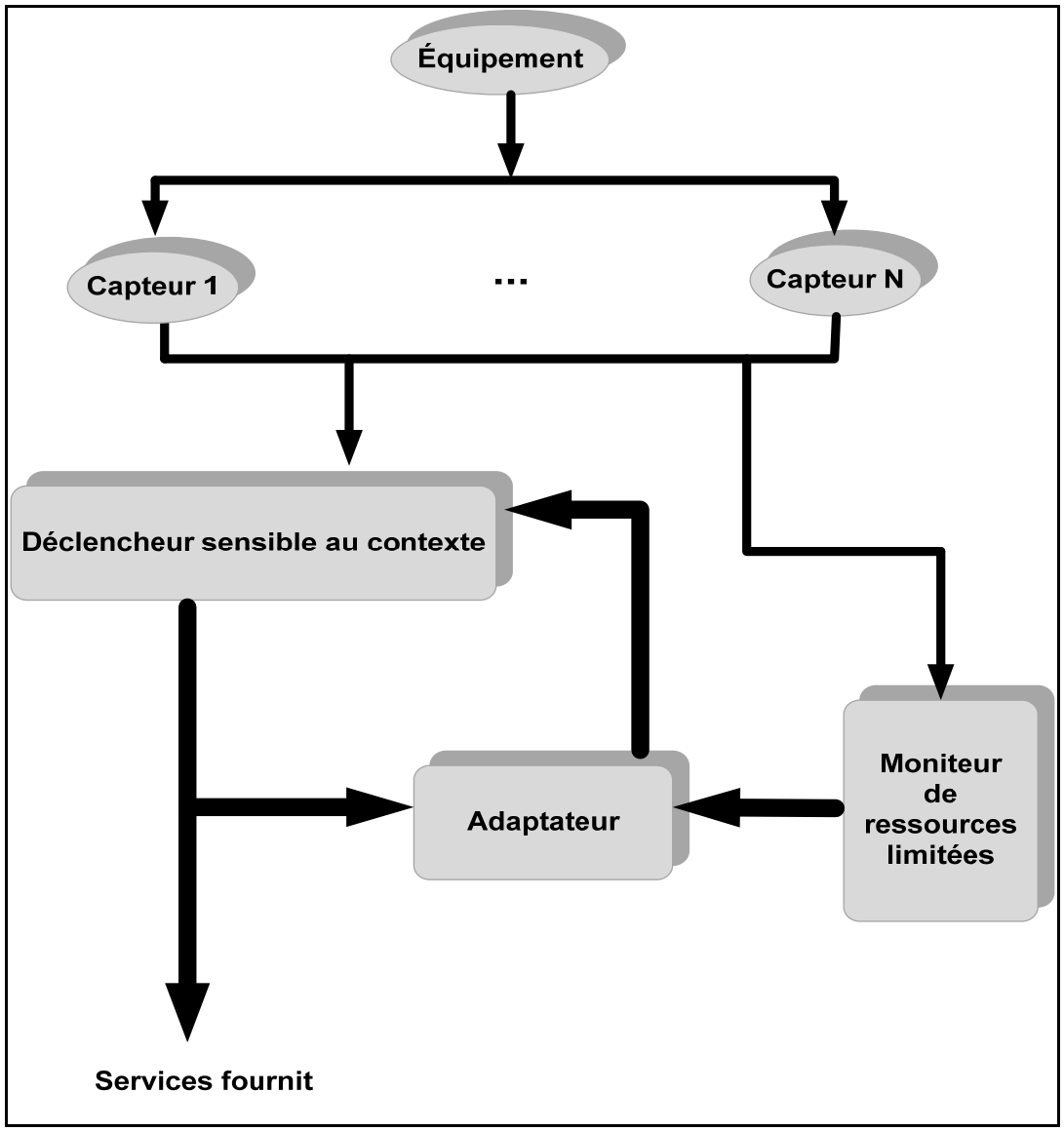

Figure 5.2 Architecture du système d'adaptation.

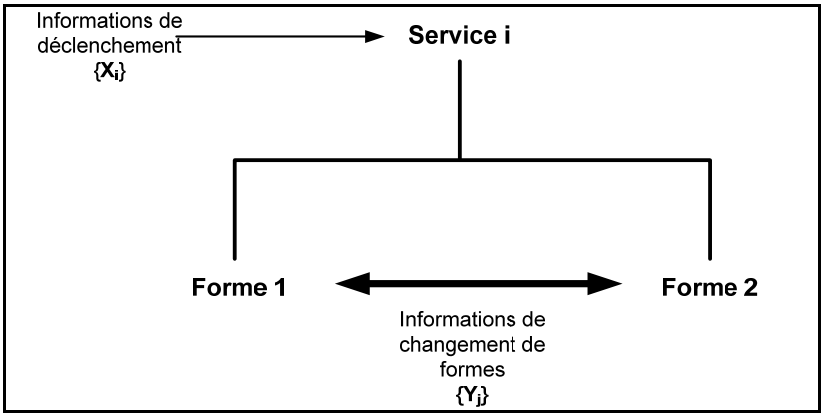

Figure 5.3 Types d'informations contextuelles.

Tableau 5.3Types d'informations contextuelles

Services	**Informations de déclenchement**	**Formes**	**Informations de changement de formes**
Service i	$\{X_i\}$	Forme 1	$\{Y_j\}$
		Forme 2	

Il y a deux types de services : services déclenchés manuellement par l'utilisateur ; services déclenchés automatiquement selon le contexte actuel. Pour différencier entre ces deux types de services, le DSC utilise un bit de contrôle noté **K** (K=0 pour le service déclenché automatiquement et K=1 pour le service déclenché manuellement). Le DSC utilise un autre bit de contrôle noté **S** pour sauvegarder l'état actuel de chaque service (S=0 pour le service actuellement inactif et S=1 pour le service actuellement actif). le DSC utilise aussi un bit noté **A** pour chaque forme d'un service pour indiquer si cette forme est actuellement active ou non (A=1 la forme est actuellement active et A=0 la forme est actuellement désactive). Ce bit est placé automatiquement à 0 pour toutes les formes d'un service si ce dernier est actuellement inactif. En outre, le DSC, contient pour chaque forme d'un service, l'ensemble des ressources limitées de l'équipement informatique affectées (consommées), c'est.-à-dire quelles sont les ressources limitées consommées si un service est fourni sous une forme particulière. La Figure 5.4 montre la structure du DSC.

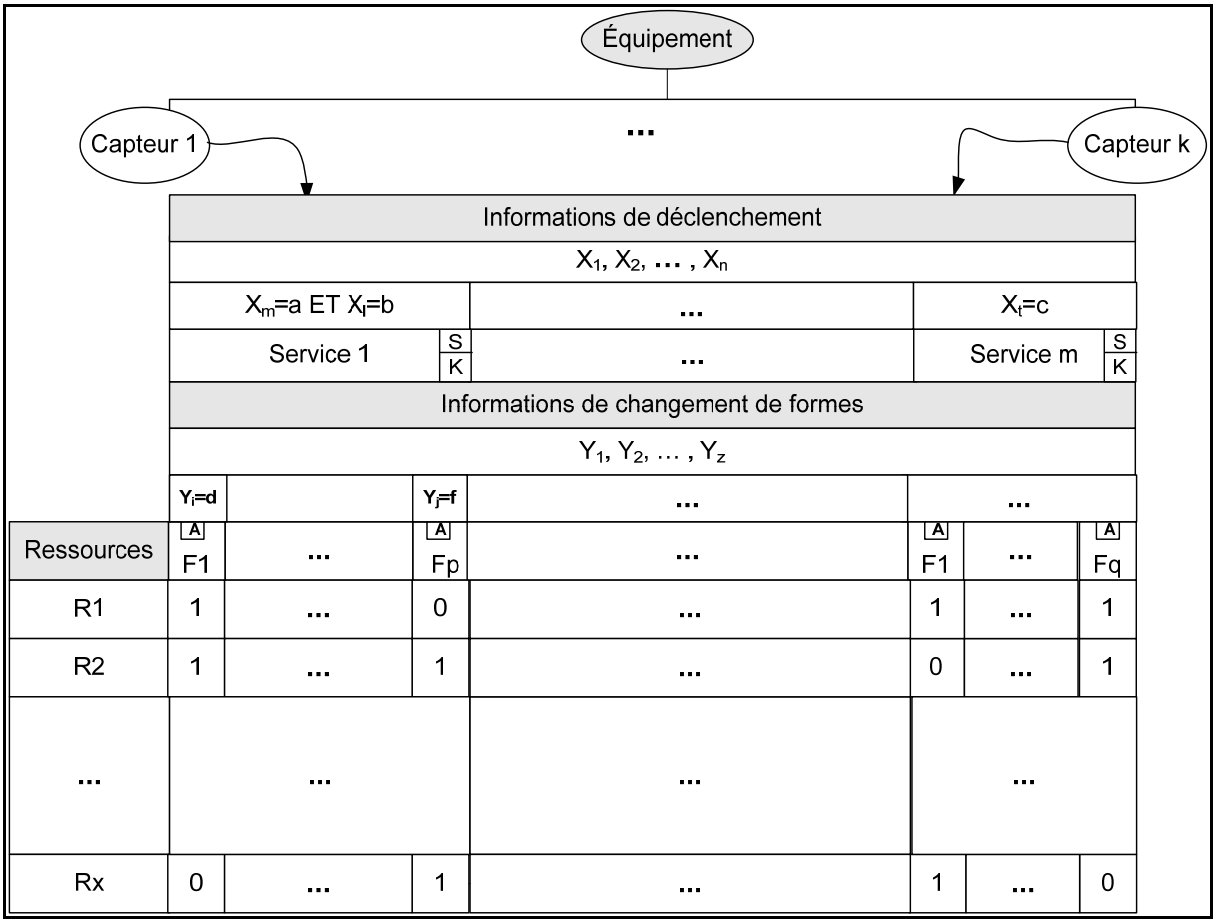

Figure 5.4 Structure de DSC.

La structure du DCS peut être formulée selon la notation suivante :

- Ensemble des informations contextuelles

  $C = \{C_1, C_2, \dots, C_n\}$

  C_i peut être la localisation d'une personne, vitesse d'une connexion internet, date, heure, etc.
- Ensemble des ressources limitées d'un équipement

  $R = \{R_1, R_2, \dots, R_p\}$

  $R_q,\ q = 1, \dots, p$ peut être la charge d'une batterie, connexion internet, capacité d'une mémoire intégrée, etc.
- Ensemble des valeurs des informations contextuelles

$V = \{(C_i, D_i)\}\ i = 1, \dots, n$

D_i étant une valeur associée à un élément du contexte.

Exemple : (localisation, université)

- Ensemble des taux d'utilisation (%) des ressources limitées

  $T = \{(R_q, P_q)\}\ q = 1, \dots, p$

  P_i étant un pourcentage d'utilisation d'une ressource limitée

  Exemple : (connexion internet, 80)

- Ensemble des services fournis par un équipement

  $S = \{S_1,\ S_2, \dots, S_k\}$

- Ensemble des formes d'un service i

  $SF_i = \{F_{i_1}, F_{i_2}, \dots, F_{i_{x_i}}\}$

  x_i est le nombre de formes du service i

  F_{ij} : la forme j du service i, $j = 1, \dots, x_i$ et $i = 1, \dots, k$

- Chaque service utilise un ensemble d'informations contextuelles

  $SC_i = \{C_y\}, y = 1, \dots, m,\ \ m \leq n$

- Chaque service $\boldsymbol{S_i}$ utilise un ensemble de ressources limitées

  $SR_i = \{R_z\}, z = 1, \dots, v,\ \ v \leq p$

- Chaque service S_i a une expression logique de déclenchement $SE_i, i = 1, \dots, k$

  SE_i est une combinaison logique (ET, OU, NON) de $V_i = \{(C_i\ , D_i)\}\ avec\ C_i \in\ SC_i$

- Chaque forme j d'un service i a une expression logique de changement de forme FE_{ij}

  $j = 1, \dots, x_i$ et $i = 1, \dots, k$

  FE_{ij} est une combinaison logique (ET, OU, NON) de $V_i = \{(C_i\ , D_i)\}$ avec $C_i \in\ SC_i$

- Chaque forme j d'un service i utilise un ensemble de ressources limitées $FR_{ij} \subseteq\ SR_i$

  $j = 1, \dots, x_i$ et $i = 1, \dots, k$

- Chaque forme j d'un service i a un état selon l'équation 6.1

  $j = 1, \dots, x_i$ et $i = 1, \dots, k$

- Chaque service S_i, $i = 1, \dots, k$ a un état selon l'équation 6.1

$$état \begin{cases} 0 & si\ service\ inactif \\ 1 & si\ service\ actif\ déclenché\ automatiquement \\ 2 & si\ service\ actif\ déclenché\ manuellement \end{cases} \quad (6.2)$$

Si un service est actif alors l'état d'une et une seule de ses formes sera marqué par 1 ou 2.

- Un service est représenté par :
  $\langle S_i, \{F_{ij}, FE_{ij}, FR_{ij}, état\}, SC_i, SE_i \rangle$ $avec\ j = 1, \dots, x_i$ et i $= 1, \dots,$ k

L'état standard du DSC est en mode veille afin de réduire la consommation de ressources. Une fois qu'un élément de contexte (information de déclenchement ou information de changement de forme) change sa valeur, Le DSC devient éveillé et réagit en déclenchant un service ou en changeant la forme d'un service.

Moniteur de ressources limitées

Un SID se compose d'un ensemble d'équipements informatiques. Chaque équipement fournit des services. Chaque service a plusieurs formes et l'une d'entre elles est la forme par défaut du service. La tâche d'adaptation consiste à déclencher automatiquement un service ou changer sa forme selon le contexte actuel et en tenant compte des ressources disponibles. La plupart des équipements dans un SID sont portatifs et mobiles (exemple : téléphone cellulaire, ordinateur portable, etc.). Nous pouvons classer les ressources de tels équipements en deux catégories : illimitées et limités. Une fois utilisées, nous pouvons définir les ressources illimitées en tant que catégorie qui à le moins de chances d'avoir un problème de disponibilité (exemple le disque dur d'un ordinateur de bureau). Par contre, les ressources limitées peuvent être définies en tant que celles qui devraient être employées soigneusement parce que leur disponibilité diminue rapidement (exemple : charge de batterie d'un téléphone cellulaire, la largeur de bande d'une connexion internet limitée, etc.).

Il appartient au développeur du système d'adaptation de considérer une ressource comme étant limitée ou illimitée pour chaque équipement et selon le domaine d'application. L'allocation des ressources illimitées ne pose pas un grand problème comparé à l'allocation de ressources limitées. Par conséquent, dans ce qui suit, nous nous limiterons à l'allocation

de ressources limitées. Nous supposerons que chaque équipement a ses propres capteurs qui fournissent la quantité de disponibilité de ses ressources limitées (l'exemple de capteur de niveau de charge de batterie d'un téléphone cellulaire). La tâche d'adaptation consiste entre autres à fournir un service sous une forme qui affecte (consomme) le moins possible les ressources limitées d'un équipement pour éviter leurs épuisement. C'est pourquoi, nous emploierons dans notre architecture du système d'adaptation un moniteur de ressources limitées (MRL) qui fournit d'une manière continue le taux de disponibilité de chaque ressource limitée pour chaque équipement. Par exemple, un MRL d'un téléphone mobile peut être représenté comme le montre le Tableau 5.4. Dans cet exemple, nous considérons comme ressources limitées : la charge de batterie et la mémoire intégrée.

Tableau 5.4 Exemple d'un MRL

Équipement	**Ressources limitées**	**Pourcentage de disponibilité**
Téléphone cellulaire	charge de batterie	70%
	mémoire	50%
	…	….

Adaptateur

L'adaptateur est le troisième composant de l'architecture : sa tâche principale est d'adapter dynamiquement des services fournis selon le contexte actuel et selon le taux de disponibilité des ressources limitées. L'adaptateur a accès au contenu du DSC pour identifier l'ensemble de services actuellement actifs, sous quelles formes et leurs types (service automatiquement déclenché ou service manuellement déclenché). Il est également en communication avec le MRL en utilisant le modèle inscription/notification afin d'être notifié chaque fois que la quantité disponible d'une ressource limitée devient nulle (la ressource est entièrement employée). L'état par défaut de cet adaptateur est le mode veille pour éviter la consommation de ressources. Une fois notifié par le MRL, il devient actif. Les services manuellement déclenchés sont prioritaires sur ceux qui sont déclenchés automatiquement parce qu'ils sont

déclenchés par l'utilisateur, ce qui veut dire qu'ils constituent un besoin immédiat pour l'utilisateur.

L'objectif principal de la sensibilité au contexte dans un SID est de satisfaire en premier lieu les besoins de l'utilisateur. Le système doit mettre toutes les ressources limitées nécessaires à la disposition des services déclenchés manuellement par l'utilisateur : c'est la tâche principale de l'adaptateur.

Nous pouvons distinguer deux situations de fonctionnement selon les types de services actuellement actifs :

- **situation 1 :** tous les services actuellement actifs sont déclenchés automatiquement (selon le contexte actuel) ;
- **situation 2 :** un (ou plusieurs) des services actuellement actifs est déclenché (sont déclenchés) manuellement par l'utilisateur.

Une fois que l'adaptateur est notifié par le MRL par un message contenant quelle ressource limitée devient entièrement utilisée (quantité de disponibilité devient nulle), il devient éveillé et réagit selon deux stratégies et selon la situation actuelle.

Stratégie 1

Tous les services actuellement actifs sont déclenchés automatiquement selon le contexte courant. Dans ce cas, tous les services ont le même degré de priorité. L'adaptateur exécute alors les étapes suivantes :

Pour chaque ressource entièrement utilisée R_i :

Étape 1 : rechercher les formes de services actuellement actifs qui emploient la ressource R_i en accédant au contenu du DSC et en employant les bits de contrôle K, S et A ;

Étape 2 : limiter l'ensemble des services actuellement actifs à ceux qui peuvent être fournis sous une autre forme qui n'emploie pas la ressource Ri. Si cela n'est pas possible, choisir alors aléatoirement un service actif, le désactiver et passer à l'étape 3 ;

Étape 3 : choisir aléatoirement un de ces services et choisir aléatoirement une de ses autres formes qui n'utilisent pas la ressource Ri. Envoyer une commande au DSC afin de mettre à jour son contenu selon les actions de l'adaptateur.

Stratégie 2

Dans ce cas, il y a un ou plusieurs services actuellement actifs déclenchés manuellement par l'utilisateur. Le système doit alors mettre sous sa disposition toutes les ressources limitées nécessaires pour satisfaire les besoins de l'utilisateur. L'adaptateur exécute alors les étapes suivantes :

Pour chaque ressource entièrement utilisée R_i :

Étape 1 : rechercher les formes de services actuellement actifs qui emploient la ressource Ri en accédant au contenu du DSC et en employant les bits de contrôle K, S et A ;

Étape 2 : limiter l'ensemble des services actuellement actifs (seulement déclenchés automatiquement) à ceux qui peuvent être fournis sous une autre forme qui n'emploie pas la ressource R_i. Si cela n'est pas possible, rechercher alors les formes de services actuellement actives (déclenchées manuellement) qui emploient la ressource R_i ;

Étape 3 : choisir les services qui peuvent être fournis sous une autre forme qui n'utilise pas la ressource R_i, proposer à l'utilisateur un service que le système devrait changer de forme et passer à l'étape 4 ;

Étape 4 : choisir aléatoirement un de ces services et choisir aléatoirement une de ses autres formes qui n'utilisent pas la ressource R_i. Envoyer une commande au DSC afin de mettre à jour son contenu selon les actions de l'adaptateur.

L'adaptateur fonctionne comme un régulateur en boucle fermée pour le DSC où les informations du MRL sont utilisées pour ajuster ses sorties.

5.3.2 Modélisation et simulation

Dans cette partie, nous allons faire la modélisation et la simulation de deux situations possibles de l'approche d'adaptation sensible aux ressources limitées. Les étapes de chaque stratégie seront modélisées en utilisant le formalisme de RdPC et simulées par l'outil CPN-Tools.

Stratégie 1 : services déclenchés automatiquement

Modélisation

La Figure 5.5 montre le modèle hiérarchique simplifié d'adaptation qui comprend trois étapes. La Figure 5.6, la Figure 5.7 et la Figure 5.8 montrent les modèles détaillés des étapes 1, 2 et 3 respectivement. L'ensemble des déclarations utilisé pour la construction du modèle de la situation 1 est donné par l'extrait 5.1 suivant :

```
colset UNIT = unit;
colset INT = int;
colset BOOL = bool;
colset STRING = string;
val NS=5;
colset Resource=STRING;
colset E =with e;
colset ResList=list Resource;
colset Form=STRING;
colset ResAmt=product Resource*INT;
colset Service=STRING;
colset SerStat=product Service*ResList*INT;
colset FormRes=product Form*ResList*INT;
colset FormList=list FormRes;
colset SerFormList=product Service*FormList;
colset SerForm=product Service*FormRes;
var r: Resource;
var q,p,i,n:INT;
var s,sr:Service;
var y:ResList;
var fl:FormList;
var fr:Form;
var rl:ResList;
```

Extrait 5.1 Ensemble des déclarations utilisé pour la construction du modèle de la situation 1.

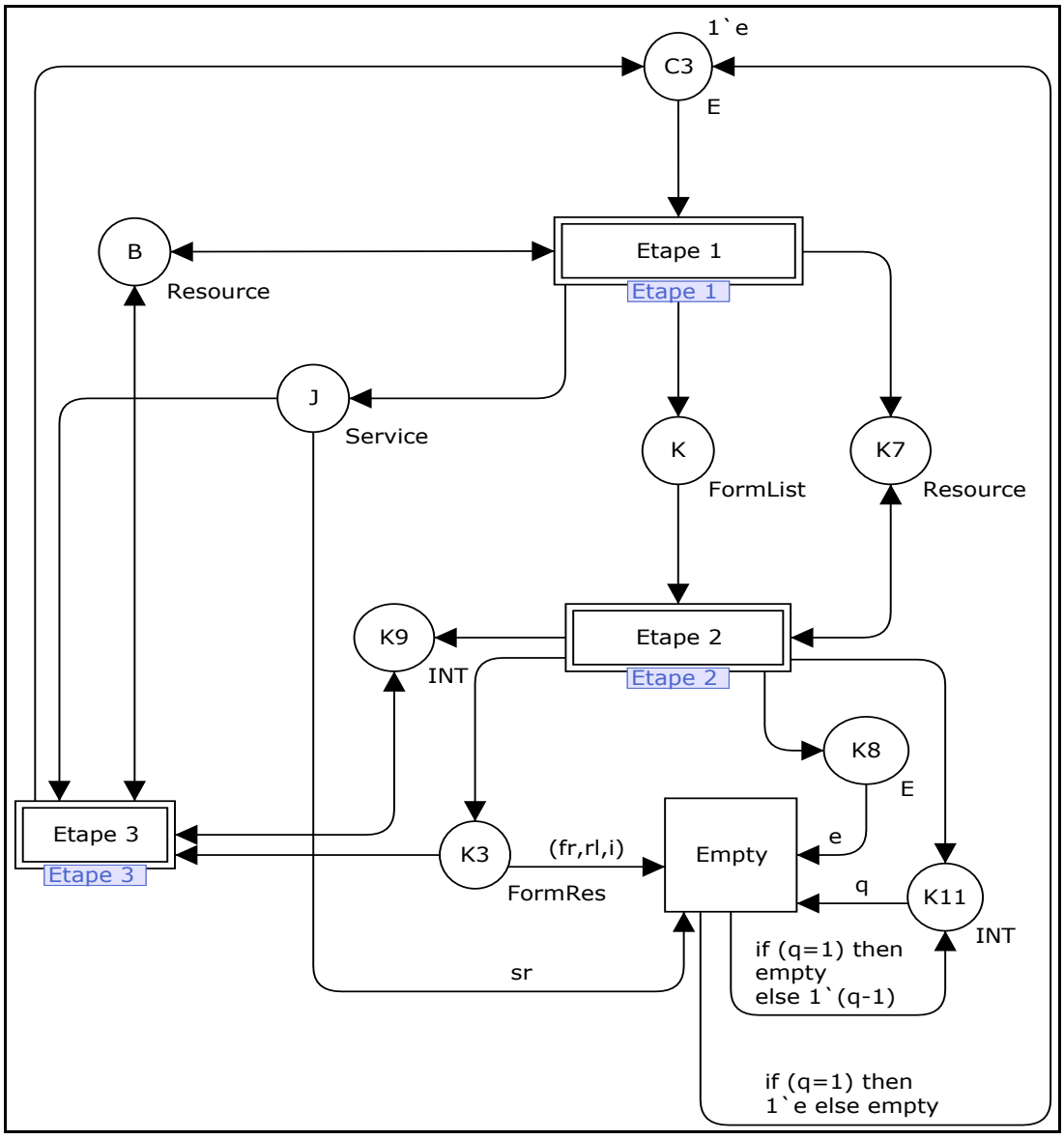

Figure 5.5 Modèle de la situation 1 : services déclenchés automatiquement.

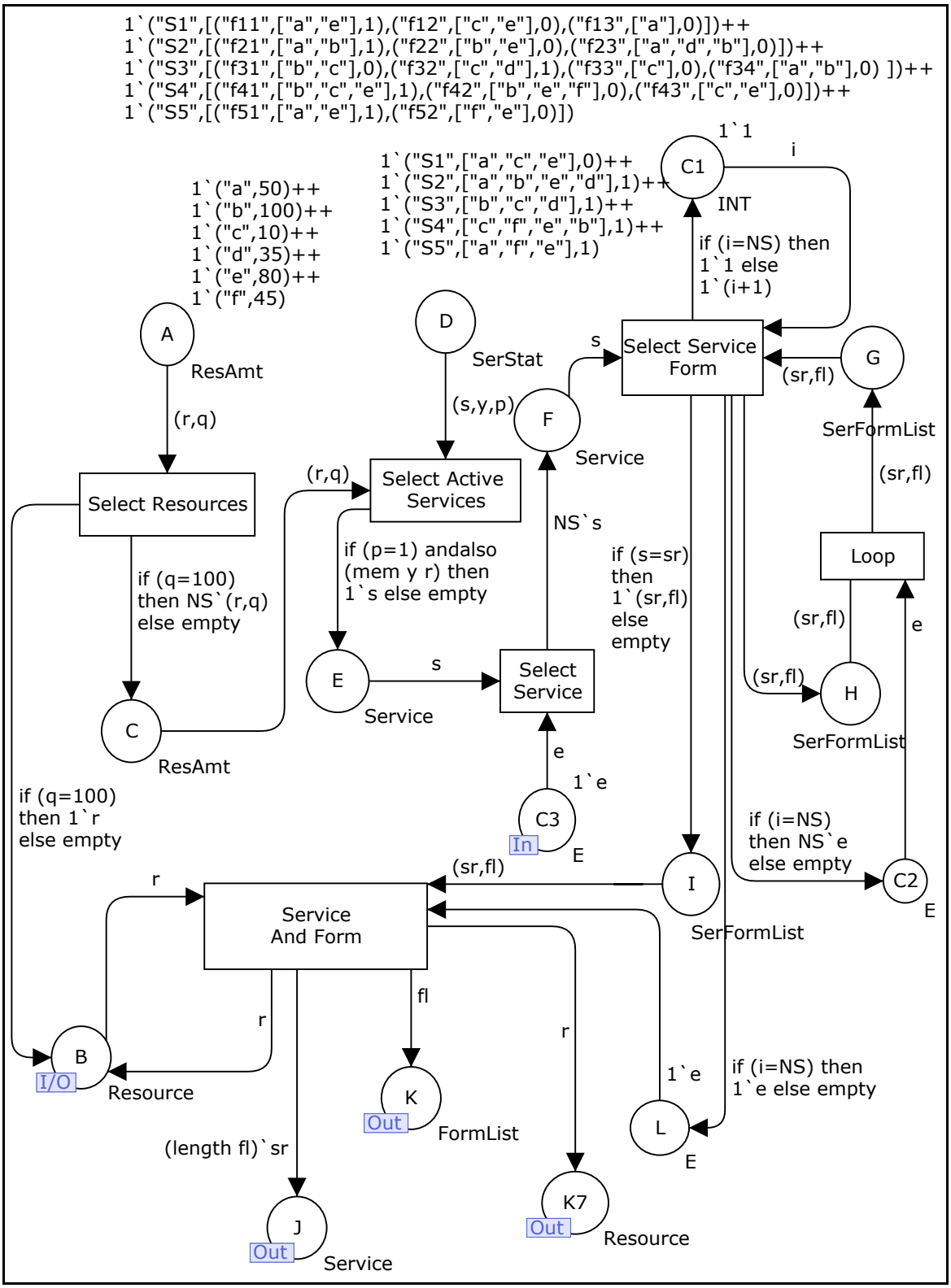

Figure 5.6 Modèle de la situation 1 (étape 1).

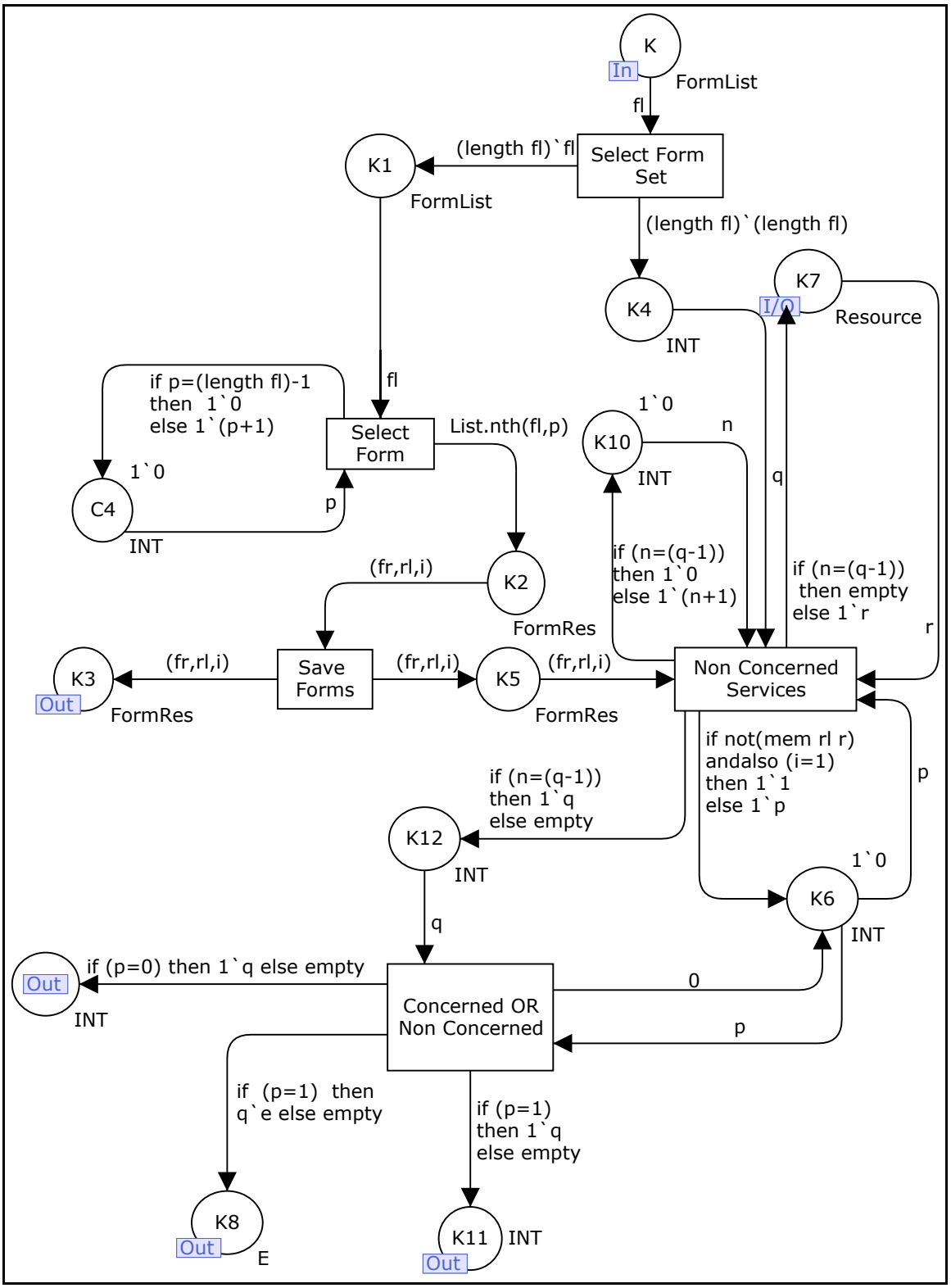

Figure 5.7 Modèle de la situation 1 (étape 2).

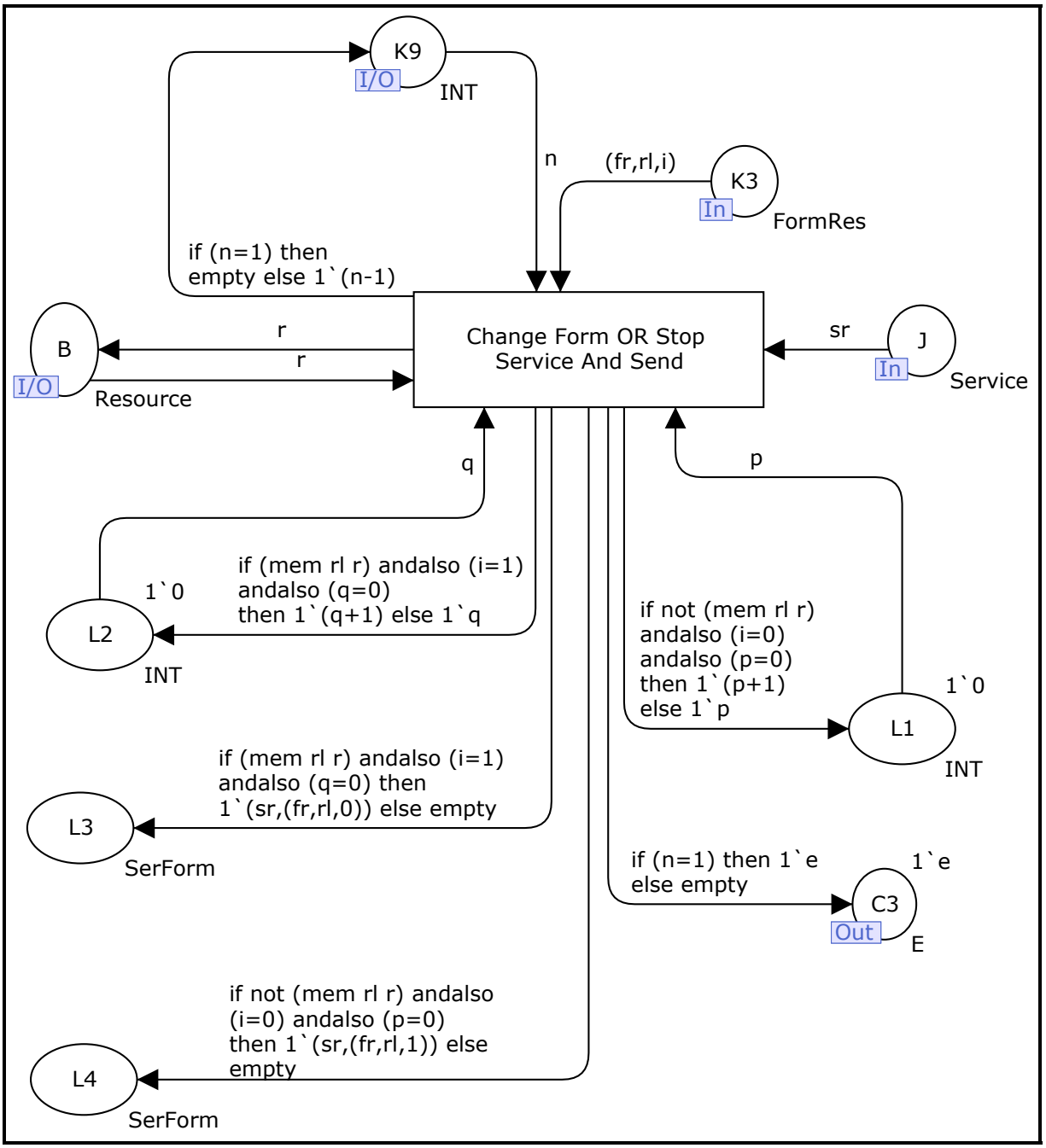

Figure 5.8 Modèle de la situation 1 (étape 3).

Scénario de simulation et résultats

Pour valider notre modélisation de la situation 1, nous avons employé un scénario typique détaillé comme suit :

- Un équipement informatique a six ressources limitées dont l'une d'entre elles ayant un taux d'utilisation à 100% (ressource nommée "b"). Nous représentons cette information par le couple :

  (Nom de la ressource, taux d'utilisation (%))

  Considérons l'exemple suivant :

  ("a",50), ("b",100), ("c",10), ("d",35), ("e",80), ("f",45)

- Cinq services fournis par l'équipement informatique dont quatre d'entre eux sont actifs (déclenchés automatiquement selon le contexte). Chaque service utilise un ensemble de ressources limitées. Nous représentons cette information par le triplet :

  (Nom service, [liste de ressources limitées utilisées], état)
  Où :

$$état = \begin{cases} 1 & si\ service\ actif \\ 0 & sinon \end{cases} \quad (6.3)$$

Considérons la situation suivante :

("S1",["a","c","e"],0)
("S2",["a","b","e","d"],1)
("S3",["b","c","d"],1)
("S4",["c","f","e","b"],1)
("S5",["a","f","e"],1)

- Chaque service a un ensemble de formes sous lesquelles il est fourni et chaque forme utilise un ensemble de ressources limitées. Si un service est actuellement actif, alors l'état de l'une de ses formes est marqué selon l'équation 6.3. Nous définissons le triplet de cette manière :

(Nom de service, [[(nom de la forme, [liste de ressources utilisées], état)]])

Considérons la situation suivante :

("S1",[("F11",["a","e"],0), ("F12",["c","e"],0), ("F13",["a"],0)])
("S2",[("F21",["a","b"],1), ("F22",["b","e"],0), ("F23",["a","d","b"],0)])
("S3",[("F31",["b","c"],0), ("F32",["c","d"],1), ("F33",["c"],0), ("F34",["a","b"],0)])
("S4",[("F41",["b","c","e"],1), ("F42",["b","e","f"],0), ("F43",["c","e"],0)])
("S5",[("F51",["a","e"],1), ("F52",["f","e"],0)])

Dans ce scénario, les deux services (S2 et S4) ont une forme active qui utilise la ressource épuisée (b). Après exécution de la simulation, nous avons obtenu une nouvelle forme active (f43 : forme 3 du service 4) qui n'utilise pas la ressource épuisée. Le service S2 ne peut être fourni sous une autre forme puisque toutes ses formes utilisent la ressource épuisée.
Nous avons apporté plusieurs modifications des paramètres de scénario (nombre de services, nombre de ressources limitées épuisées, liste de ressources limitées utilisées par chaque service, nombre de formes de chaque service, nombre de services actifs, etc.) et chaque fois, nous avons obtenu les résultats prévus.

Stratégie 2 : services déclenchés automatiquement et manuellement

Modélisation

La Figure 5.9 montre le modèle hiérarchique simplifié d'adaptation qui comprend quatre étapes. Les Figure 5.10, 5.11, 5.12 et 5.13 montrent les modèles détaillés de l'étape 1, 2, 3 et 4 respectivement. L'ensemble des déclarations utilisées pour la construction du modèle de la situation 2 est donné par l'extrait 5.2 suivant :

```
colset UNIT = unit;
colset INT = int;
colset BOOL = bool;
colset STRING = string;
val NS=5;
colset Resource=STRING;
colset E =with e;
colset ResList=list Resource;
colset Form=STRING;
colset ResAmt=product Resource*INT;
colset Service=STRING;
colset SerStat=product Service*ResList*INT;
colset FormRes=product Form*ResList*INT;
colset FormList=list FormRes;
colset SerFormList=product Service*FormList;
colset SerForm=product Service*FormRes;
colset SerNum=product Service*INT;
colset SerFormRes=product Service*FormRes*Resource;
var r: Resource;
var k,q,p,i,n,m:INT;
var s,sr:Service;
var y:ResList;
var fl:FormList;
var fr:Form;
var rl:ResList;
```

Extrait 5.2 Ensemble des déclarations utilisé pour la construction du modèle de la situation 2.

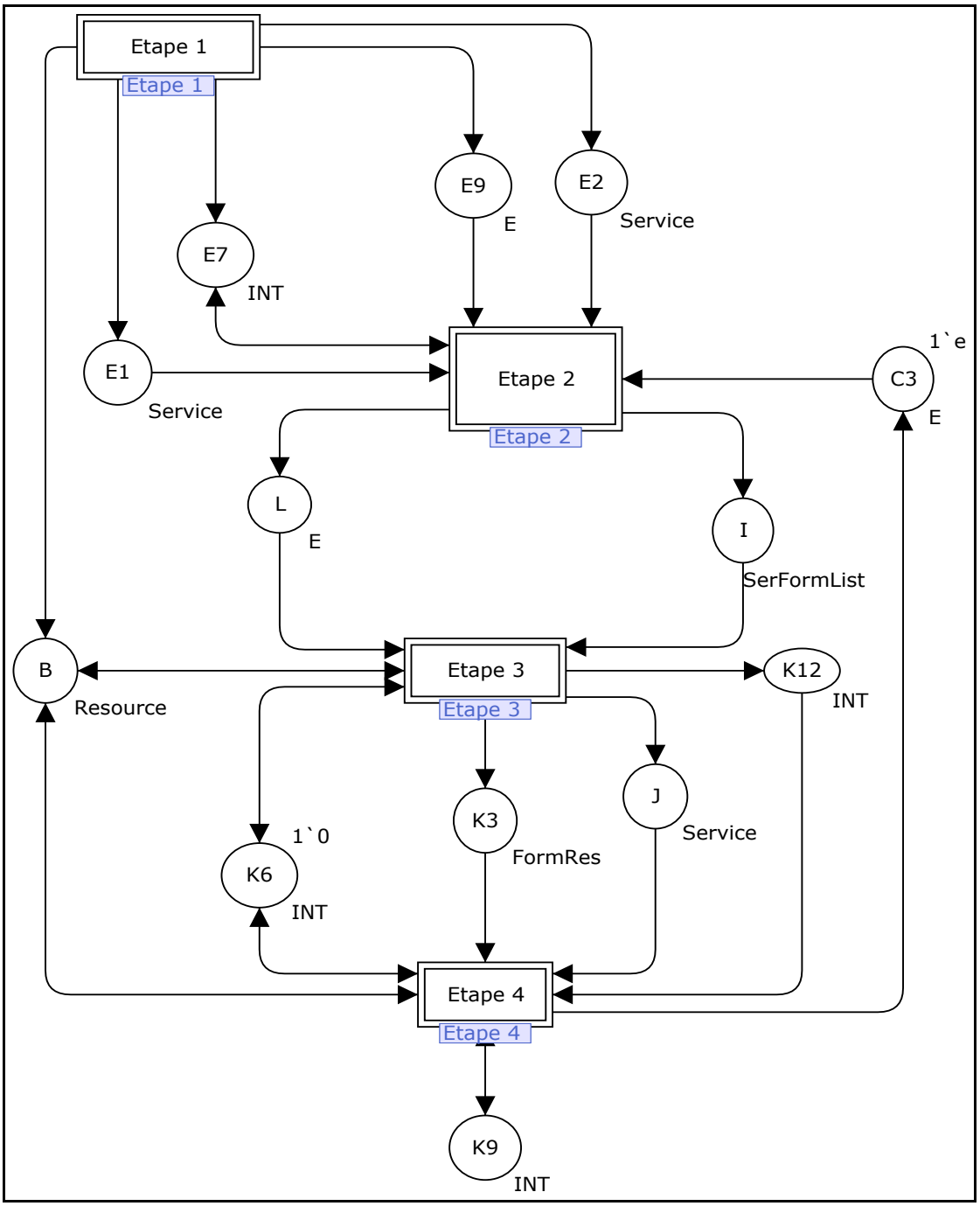

Figure 5.9 Modèle de la situation 2 : services déclenchés automatiquement et manuellement.

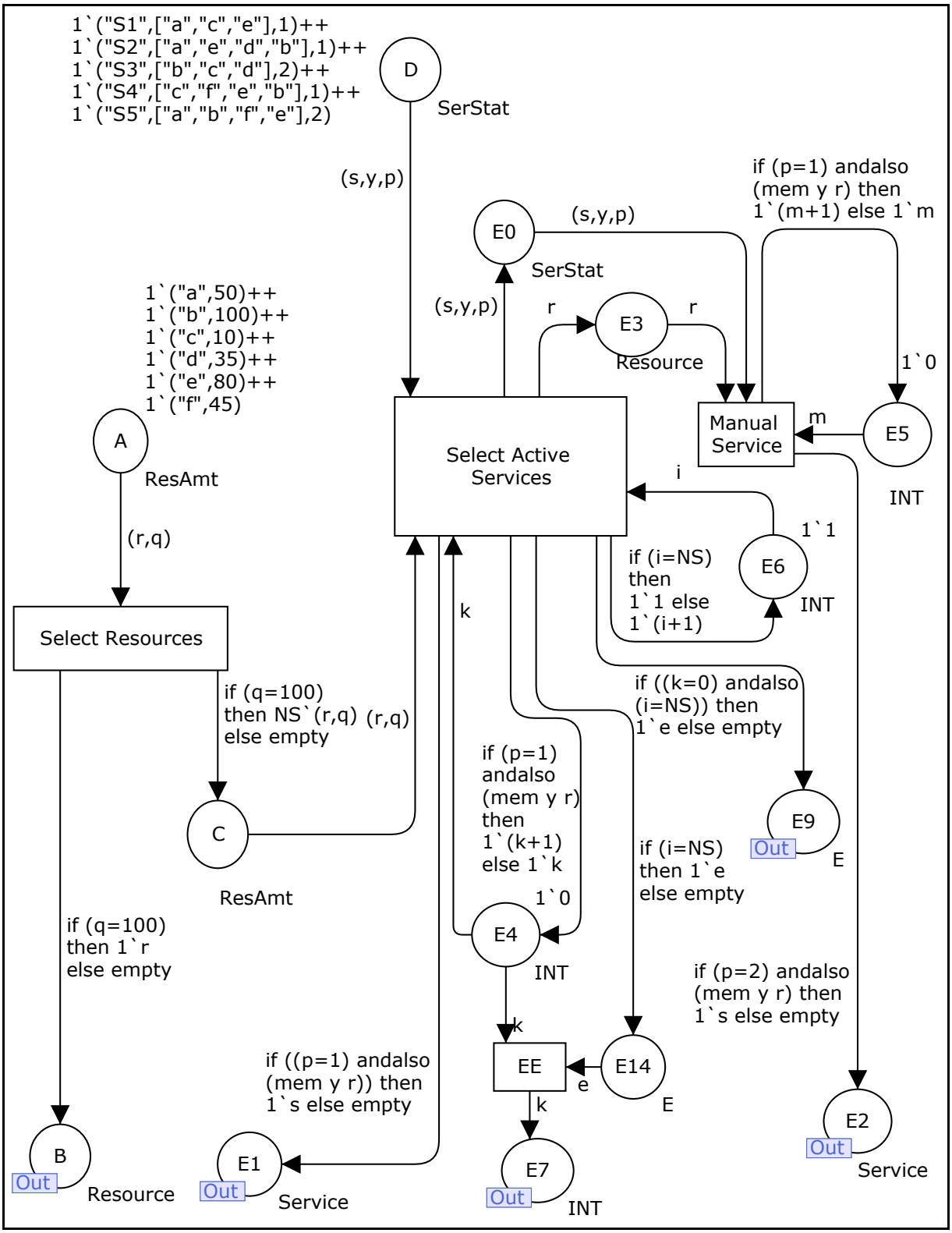

Figure 5.10 Modèle de la situation 2 (étape 1).

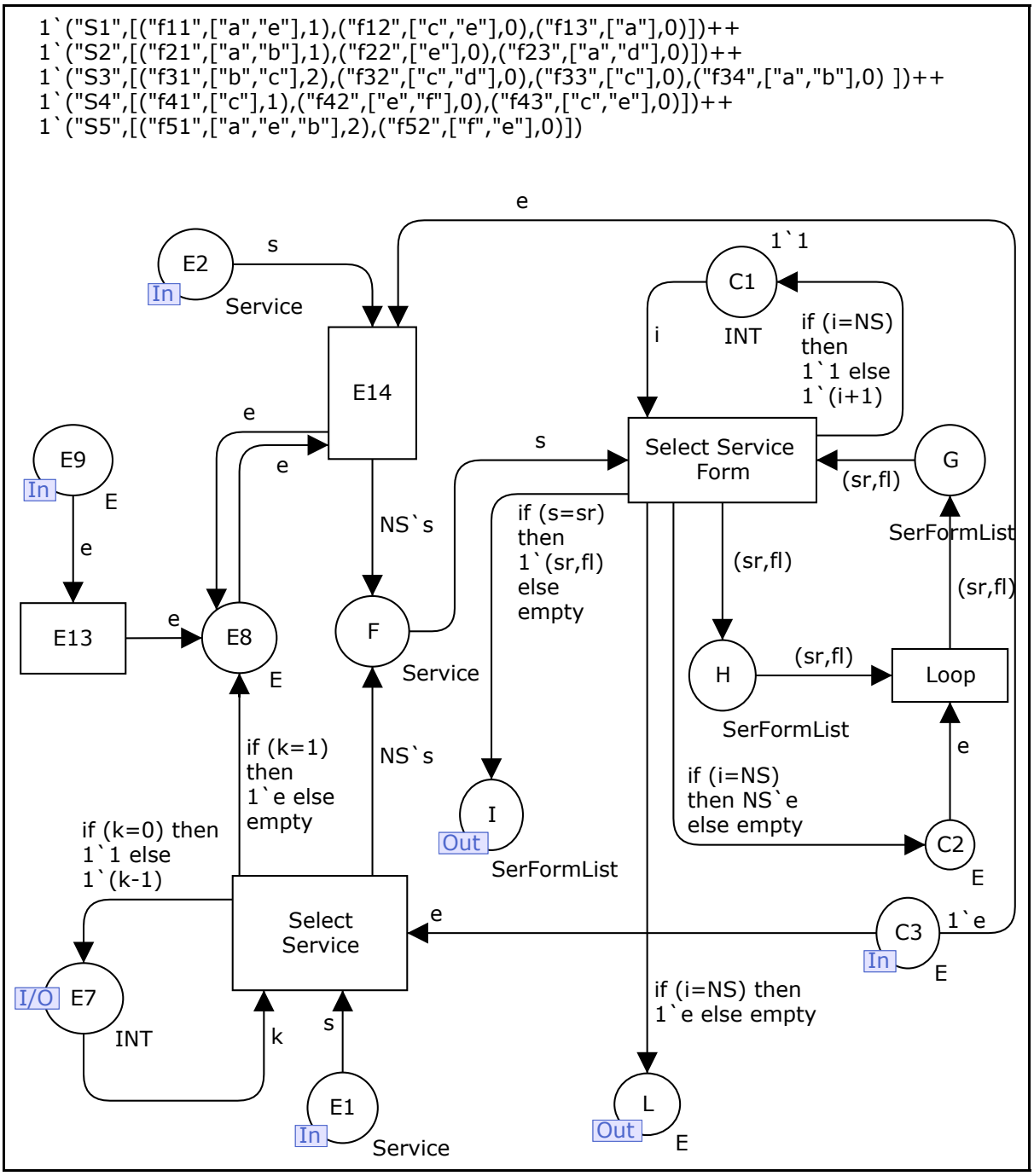

Figure 5.11 Modèle de la situation 2 (étape 2).

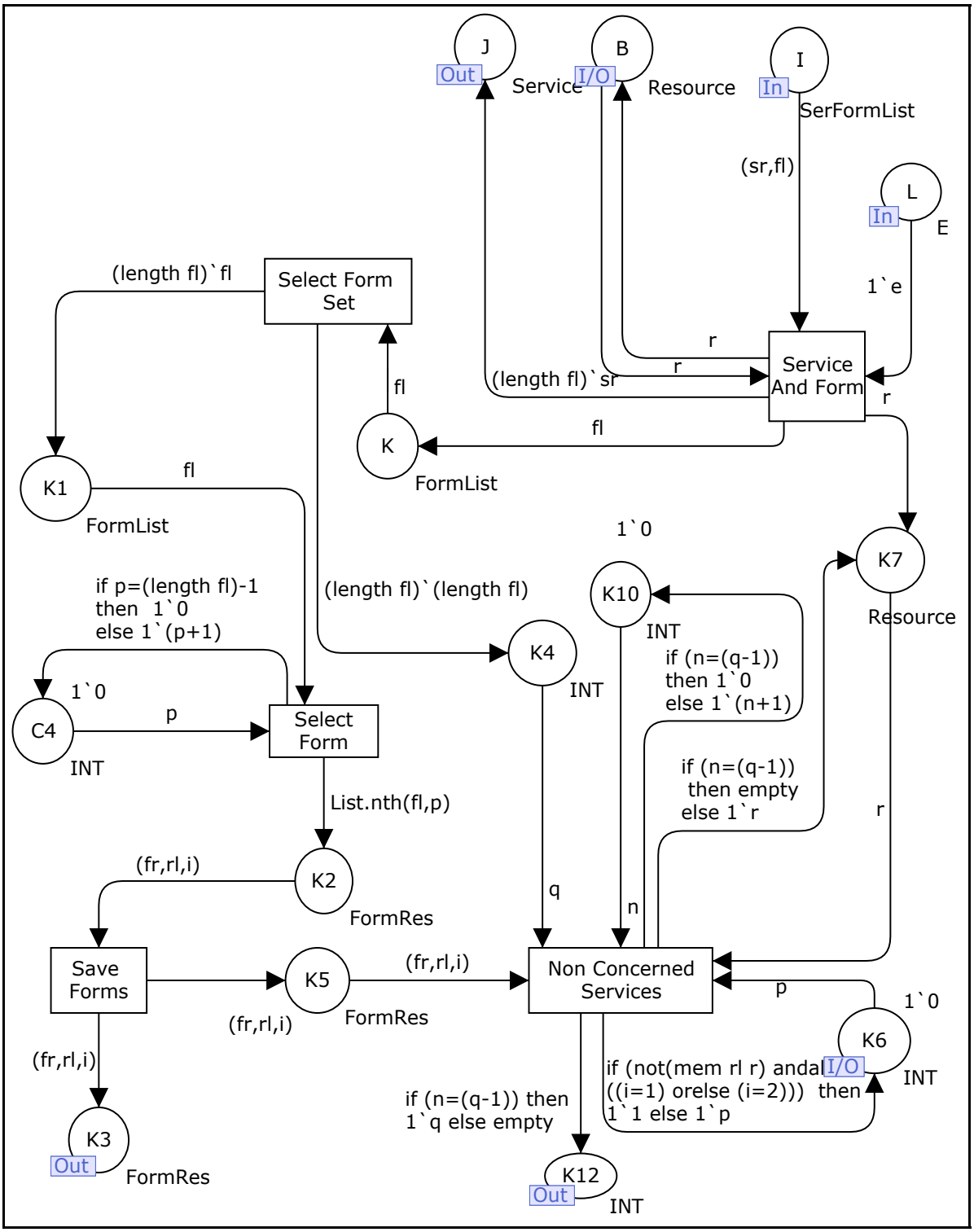

Figure 5.12 Modèle de la situation 2 (étape 3).

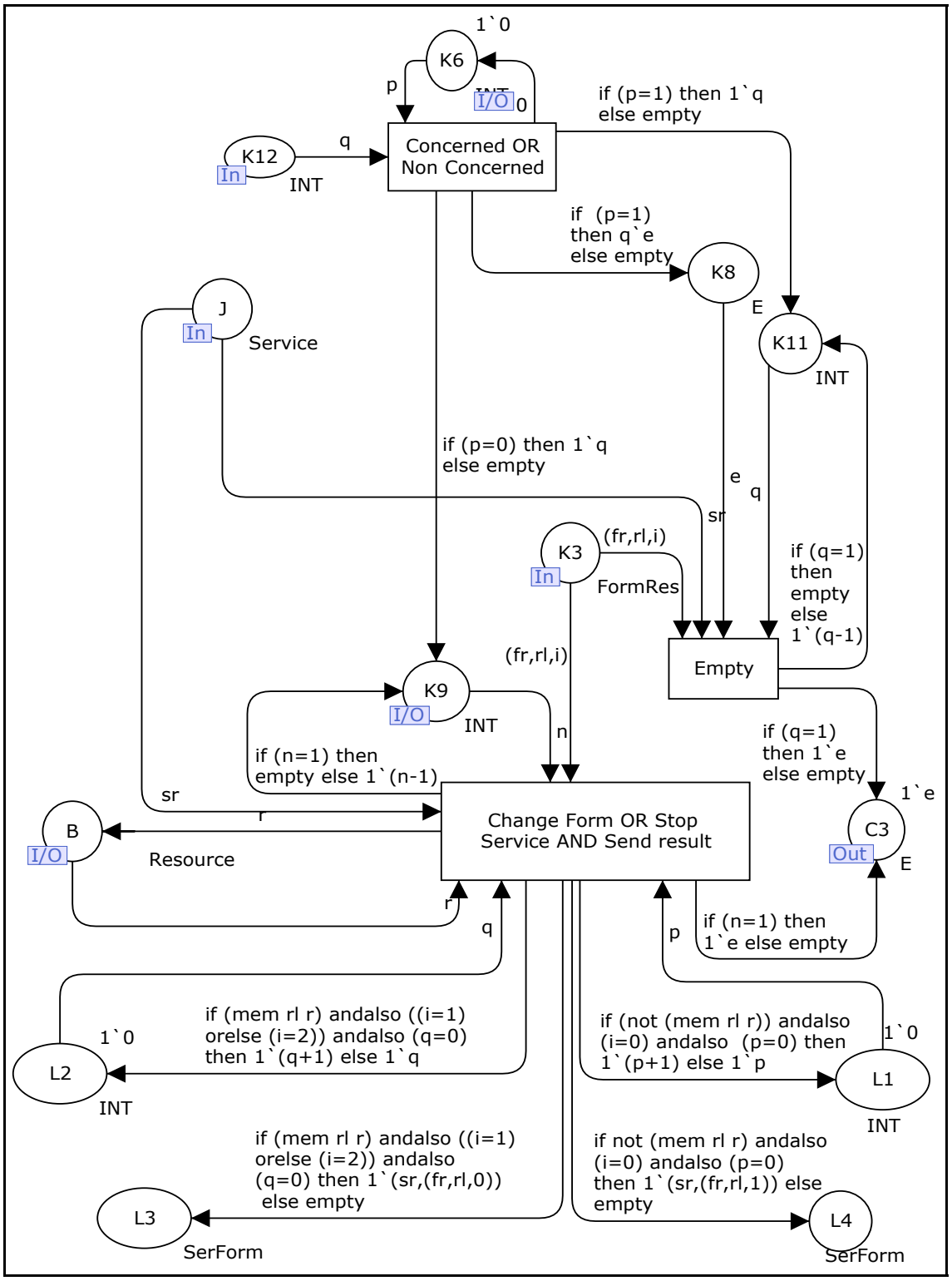

Figure 5.13 Modèle de la situation 2 (étape 4).

Scénario de simulation et résultats

Pour valider notre modélisation de la situation 2, nous avons employé un scénario typique détaillé comme suit :

- un équipement informatique a six ressources limitées dont l'une d'entre elles ayant un taux d'utilisation à 100% (ressource nommée "b"). Nous représentons cette information par le couple :
  (Nom de la ressource, taux d'utilisation (%))

  Considérons l'exemple suivant :
  ("a",50), ("b",100), ("c",10), ("d",35), ("e",80), ("f",45)

- cinq services fournis par l'équipement informatique dont quatre d'entre eux sont actifs (déclenchés automatiquement selon le contexte). Chaque service utilise un ensemble de ressources limitées. Nous représentons cette information par le triplet suivant :

  (Nom service, [liste de ressources limitées utilisées], état)

  Où l'état est marqué selon l'équation 6.3.
  Considérons la situation suivante :

  ("S1",["a","c","e"],1)
  ("S2",["a","e","d","b"],1)
  ("S3",["b","c","d"],2)
  ("S4",["c","f","e","b"],1)
  ("S5",["a","b","f","e"],2)

- Chaque service a un ensemble de formes sous lesquelles il est fourni et chaque forme utilise un ensemble de ressources limitées. Si un service est actuellement actif, alors l'état

de l'une de ses formes est marqué selon l'équation 6.1. Nous représentons cette information par le couple suivant :

(Nom de service, [[(nom de la forme, [liste de ressources utilisées], état)]])

Considérons la situation qui va suivre :

("S1",[("F11",["a","e"],1), ("F12",["c","e"],0), ("F13",["a"],0)])
("S2",[("F21",["a","b"],1), ("F22",["e"],0), ("F23",["a","d"],0)])
("S3",[("F31",["b","c"],2), ("F32",["c","d"],0),("F33",["c"],0), ("F34",["a","b"],0)])
("S4",[("F41",["c"],1), ("F42",["e","f"],0), ("F43",["c","e"],0)])
("S5",[("F51",["a","e","b"],2), ("F52",["f","e"],0)])

Dans ce scénario, il y a trois services (S2, S3 et S5) dont la forme active utilise la ressource épuisée (b). Le service S2 est déclenché automatiquement tandis que les services S3 et S5 sont déclenchés manuellement par l'utilisateur. Après exécution de la simulation, nous avons obtenu une nouvelle forme active (f23 : forme 3 du service 2) qui n'utilise pas la ressource épuisée. Comme il a déjà été mentionné dans la stratégie 2, le système doit chercher en premier lieu une forme qui n'utilise pas la ressource épuisée parmi les services déclenchés automatiquement. S'il ne trouve pas la forme convenable parmi ces services, alors il désactive la forme active qui utilise la ressource épuisée du service déclenché automatiquement. Sinon, il passe aux services déclenchés manuellement pour faire le même traitement.

Nous avons apporté plusieurs modifications des paramètres de scénario (nombre de services, nombre de ressources limitées épuisées, liste de ressources limitées utilisées par chaque service, par nombre de formes de chaque service, par nombre de services actifs déclenchés automatiquement, par nombre de services actifs déclenchés manuellement, etc.) et chaque fois nous avons obtenu les résultats prévus que doit fournir le système.

CONCLUSION

Le cadre général de notre recherche est l'informatique diffuse qui constitue la pierre angulaire de l'informatique future. Ce domaine qui combine les aspects de l'informatique distribuée et de l'informatique mobile, adopte une nouvelle vision des équipements et des applications. L'un des aspects contribuant à la réalisation de cette notion d'informatique diffuse est de permettre à ces systèmes informatiques de s'adapter proactivement aux changements du contexte des applications et de celui de l'utilisateur.

La compréhension du contexte par un système informatique impose plusieurs défis dans le domaine de l'informatique diffuse ; permettre au système de comprendre l'environnement dans lequel il évolue ; autoriser l'acquisition d'une connaissance qui ne se limite pas aux limites physiques du terminal ; rendre ce système réactif et adaptable aux informations qu'il perçoit. Un tel système est capable de réagir de manière "intelligente" en ce qui a trait à son environnement. Nous avons remarqué à quel point le contexte pouvait être une notion difficilement assimilable, et la difficulté de représenter un contexte qui se caractérise par une information naturellement subjective et sans qu'on parvienne à une certaine précision. Dans ce document, nous avons proposé une nouvelle définition du contexte pour l'informatique diffuse fondée sur la notion de service. Nous avons démontré comment une telle définition aide efficacement à limiter l'ensemble des informations contextuelles et offre une abstraction raisonnable du concept contexte.

À la lumière de notre étude des architectures existantes, des systèmes sensibles aux contextes, en se basant sur la notion de service ainsi que notre nouvelle définition du contexte, nous avons proposé une architecture multiagents pour appuyer le développement des systèmes sensibles au contexte dans un environnement diffus. Nous avons discuté des caractéristiques et des points déterminants d'une telle architecture. Afin de faire un pas important pour valider cette architecture, nous avons procédé à sa modélisation et sa simulation en utilisant les RdPC comme formalisme de modélisation et le CPN-Tools comme outil de simulation.

Un des éléments principaux de l'architecture est la modélisation du contexte. Nous avons procédé à une étude de l'état de l'art pour explorer les différentes méthodes de représentation du contexte. Ceci nous a amené à privilégier la modélisation du contexte par ontologie. Nous avons proposé une nouvelle méthode de modélisation du contexte à base d'ontologies en se servant de notre nouvelle définition du contexte. Pour prouver l'employabilité de l'ontologie de service proposée, nous avons utilisé un Scénario réel de l'informatique diffuse et nous avons montré les étapes à suivre pour réaliser la modélisation du contexte.

La tâche d'adaptation dynamique des services dans un système diffus doit être faite selon le contexte et en prenant en considération les ressources limitées des équipements dans un système diffus. Nous avons proposé deux approches d'adaptation dynamique de services : la première est basée sur l'apprentissage automatique et a été validée par un scénario d'application typique ; la seconde est une nouvelle architecture pour l'adaptation de service qui est sensible au contexte et aux ressources limitées d'un équipement. L'architecture a été modélisée en utilisant le formalisme des réseaux de Pétri colorés. La simulation a été faite en utilisant l'outil CPN-Tools sur des scénarios réels de l'informatique diffuse.

Les perspectives offertes par notre recherche seraient l'intégration des différents composants de l'architecture multiagents proposée selon un processus incrémental. Il est recommandé de commencer en premier lieu par l'intégration du module de représentation du contexte et le module d'adaptation dynamique de services pour un seul équipement informatique. En deuxième lieu, on doit faire l'implémentation d'un SID composé de plusieurs équipements informatiques.

Un SID est de nature temps réel. Un équipement ne doit pas simplement fournir proactivement des services adaptés, il doit aussi les fournir dans des délais acceptables. Une autre perspective de nos travaux serait d'introduire des contraintes temporelles dans notre architecture multiagents. Procéder ensuite à la modélisation et simulation de l'architecture dont le but est d'évaluer les temps de réponses.

Dans les SID, des erreurs peuvent apparaître dans les informations contextuelles à la suite des défauts de capture, d'interprétation ou de présentation. Les applications doivent donc disposer de moyens de décider de la fiabilité des informations manipulées. Malheureusement, la plupart des modèles de représentation du contexte n'abordent pas les problèmes de la qualité de l'information contextuelle. Dans la mesure où une information de mauvaise qualité peut avoir des graves répercussions sur les actions du système, une autre perspective de nos travaux serait d'explorer les critères de qualité de notre modèle du contexte. L'évaluation de la qualité de l'information contextuelle peut se faire suivant plusieurs critères tels que la complétude (l'information reçu décrit-elle complètement l'état de la caractéristique contextuelle recherchée ?), la précision (lorsque c'est possible, quelle incertitude est associée à une mesure effectuée ?), et d'autres…

ANNEXE I

CODE OWL DE SÉNARIO D'APPLICATION

```
<owl:Ontology rdf:about=""/>
<Contexte rdf:ID="à,_la_bibliothèque
<Change rdf:resource="#Vibreur"/>
</Contexte>
<Contexte rdf:ID="à_l_université">
<Change rdf:resource="#Vibreur"/>
</Contexte>
<owl:ObjectProperty rdf:ID="Attaché_à">
<rdfs:domain rdf:resource="#Capteur"/>
<rdfs:range rdf:resource="#Équipement"/>
</owl:ObjectProperty>
<Sensor rdf:ID="Calendrier">
< Attaché_à rdf:resource="#Téléphone_cellulaire"/>
<Perceives rdf:resource="#Date"/>
</Capteur>
<Device rdf:ID=" Téléphone_cellulaire ">
<Fournit rdf:resource="#Indication_appels_entrants"/>
</ Équipement >
<owl:ObjectProperty rdf:ID="Change">
<rdfs:domain rdf:resource="#Contexte"/>
<rdfs:range rdf:resource="#Forme"/>
</owl:ObjectProperty>
<Capteur rdf:ID="Niveau_Charge">
< Attaché_à rdf:resource="# Téléphone_cellulaire "/>
<Percoit rdf:resource="#Charge_bas"/>
<Percoit rdf:resource="#Charge_haut"/>
```

</Capteur>

<Contexte rdf:ID="Heures_Classe">

<Change rdf:resource="#Vibreur"/>

</Contexte>

<Capteur rdf:ID="Horloge">

< Attaché_à rdf:resource="# Téléphone_cellulaire "/>

<Percoit rdf:resource="#Minuit_00AM"/>

< Percoit rdf:resource="#Matin_08AM"/>

< Percoit rdf:resource="# Heures_Classe "/>

</capteur>

<owl:Class rdf:ID="Contexte">

<rdfs:subClassOf rdf:resource="&owl;Thing"/>

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#Déclenche"/>

<owl:someValuesFrom rdf:resource="#Service"/>

</owl:Restriction>

</rdfs:subClassOf>

</owl:Class>

<Contexte rdf:ID="Noir">

<Change rdf:resource="#Silencieux"/>

</Contexte>

<Contexte rdf:ID="Date">

<Change rdf:resource="#Vibreur"/>

</Contexte>

<owl:Class rdf:ID=" Équipement ">

<rdfs:subClassOf rdf:resource="&owl;Thing"/>

<rdfs:subClassOf>

<owl:Restriction>

<owl:onProperty rdf:resource="#Fournit"/>

<owl:someValuesFrom rdf:resource="#Service"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Forme"/>
<owl:ObjectProperty rdf:ID="Possède">
<rdfs:domain rdf:resource="#Service"/>
<rdfs:range rdf:resource="#Forme"/>
</owl:ObjectProperty>
<Contexte rdf:ID="Charge_haut">
<Change rdf:resource="#Sonnerie"/>
</Contexte>
<Contexte rdf:ID="Haut_bruit">
<Change rdf:resource="#Sonnerie_avec_vibreur"/>
</Contexte>
<Contexte rdf:ID="Appel_entrant">
<Déclenche rdf:resource="#Indication_appel_entrant"/>
</Contexte>
<Service rdf:ID=" Indication_appel_entrant ">
<Possède rdf:resource="#Silencieux"/>
< Possède rdf:resource="#Sonnerie"/>
< Possède rdf:resource="# Sonnerie_avec_vibreur "/>
< Possède rdf:resource="#Vibreur"/>
</Service>
<Capteur rdf:ID="Lumière">
< Attaché_à rdf:resource="# Téléphone_cellulaire "/>
<Percoit rdf:resource="#Noir"/>
</Capteur>
<Capteur rdf:ID="Localisation">
< Attaché_à rdf:resource="# Téléphone_cellulaire "/>

<Percoit rdf:resource="# à,_la_bibliothèque "/>
< Percoit rdf:resource="# à_l_université "/>
< Percoit rdf:resource="# Dehors_l_université "/>
</Capteur>
<Contexte rdf:ID=" Charge_bas">
<Change rdf:resource="#Silencieux"/>
</Contexte>
<Contexte rdf:ID="Bas_bruit">
<Change rdf:resource="#Sonnerie"/>
</Contexte>
<Contexte rdf:ID="Minuit_00AM">
<Change rdf:resource="#Silencieux"/>
</Contexte>
<Contexte rdf:ID="Matin_08AM">
<Change rdf:resource="#Sonnerie"/>
</Contexte>
<Capteur rdf:ID="Bas_bruit">
< Attaché_à rdf:resource="# Téléphone_cellulaire "/>
<Percoit rdf:resource="# Bas_bruit "/>
<Percoit rdf:resource="# Haut_bruit "/>
</Capteur>
<Contexte rdf:ID=" Dehors_l_université ">
<Change rdf:resource="#Sonnerie"/>
</Contexte>
<owl:ObjectProperty rdf:ID="Percoit">
<rdfs:domain rdf:resource="#Capteur"/>
<rdfs:range rdf:resource="#Contexte"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Fournit">
<rdfs:domain rdf:resource="# Équipement "/>

<rdfs:range rdf:resource="#Service"/>
</owl:ObjectProperty>
<Forme rdf:ID="Sonnerie"/>
<Form rdf:ID=" Sonnerie_avec_vibreur "/>
<owl:Class rdf:ID="Capteur">
<rdfs:subClassOf rdf:resource="&owl;Thing"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="# Attaché_à "/>
<owl:someValuesFrom rdf:resource="# Équipement "/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Service">
<owl:Class rdf:ID="Service">
<rdfs:subClassOf rdf:resource="&owl;Thing"/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#Possède"/>
<owl:someValuesFrom rdf:resource="#Forme"/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
<Form rdf:ID="Silencieux"/>
<owl:ObjectProperty rdf:ID="Déclenche">
<rdfs:domain rdf:resource="#Contexte"/>
<rdfs:range rdf:resource="#Service"/>
</owl:ObjectProperty>
<Form rdf:ID="Vibreur"/>
</rdf:RDF>

ANNEXE II

RÉSULTATS DE L'ALGORITHME D'APPRENTISSAGE POUR LE SÉNARIO D'APPLICATION

Localisation	Temps	Bruit	Lumière	Batterie	Forme
SalleClasse	HeuresClasse	HautBruit	HautLumière	HautCharge	Vibreur
SalleClasse	HeuresClasse	HautBruit	HautLumière	BasCharge	Silencieux
SalleClasse	HeuresClasse	HautBruit	BasLumière	HautCharge	Vibreur
SalleClasse	HeuresClasse	HautBruit	BasLumière	BasCharge	Silencieux
SalleClasse	HeuresClasse	BasBruit	HautLumière	HautCharge	Vibreur
SalleClasse	HeuresClasse	BasBruit	HautLumière	BasCharge	Silencieux
SalleClasse	HeuresClasse	BasBruit	BasLumière	HautCharge	Vibreur
SalleClasse	HeuresClasse	BasBruit	BasLumière	BasCharge	Silencieux
Bibliothèque	HeuresClasse	HautBruit	HautLumière	HautCharge	Vibreur
Bibliothèque	HeuresClasse	HautBruit	HautLumière	BasCharge	Silencieux
Bibliothèque	HeuresClasse	HautBruit	BasLumière	HautCharge	Vibreur
Bibliothèque	HeuresClasse	HautBruit	BasLumière	BasCharge	Silencieux
Bibliothèque	HeuresClasse	BasBruit	HautLumière	HautCharge	Vibreur
Bibliothèque	HeuresClasse	BasBruit	HautLumière	BasCharge	Silencieux
Bibliothèque	HeuresClasse	BasBruit	BasLumière	HautCharge	Vibreur
Bibliothèque	HeuresClasse	BasBruit	BasLumière	BasCharge	Silencieux
Bibliothèque	HeuresLibre	HautBruit	HautLumière	HautCharge	Vibreur
Bibliothèque	HeuresLibre	HautBruit	HautLumière	BasCharge	Silencieux
Bibliothèque	HeuresLibre	HautBruit	BasLumière	HautCharge	Vibreur
Bibliothèque	HeuresLibre	HautBruit	BasLumière	BasCharge	Silencieux
Bibliothèque	HeuresLibre	BasBruit	HautLumière	HautCharge	Vibreur
Bibliothèque	HeuresLibre	BasBruit	HautLumière	BasCharge	Silencieux
Bibliothèque	HeuresLibre	BasBruit	BasLumière	HautCharge	Vibreur
Bibliothèque	HeuresLibre	BasBruit	BasLumière	BasCharge	Silencieux

RÉSULTATS DE L'ALGORITHME D'APPRENTISAAGE POUR LE SÉNARIO D'APPLICATION (suite)

Localisation	Temps	Bruit	Lumière	Batterie	Forme
Maison	HeuresSommeil	HautBruit	HautLumière	HautCharge	Sonnerie avec vibreur
Maison	HeuresSommeil	HautBruit	HautLumière	BasCharge	Silencieux
Maison	HeuresSommeil	HautBruit	BasLumière	HautCharge	Sonnerie avec vibreur
Maison	HeuresSommeil	HautBruit	BasLumière	BasCharge	Silencieux
Maison	HeuresSommeil	BasBruit	HautLumière	HautCharge	Sonnerie
Maison	HeuresSommeil	BasBruit	HautLumière	BasCharge	Silencieux
Maison	HeuresSommeil	BasBruit	BasLumière	HautCharge	**Sonnerie**
Maison	HeuresSommeil	BasBruit	BasLumière	BasCharge	Silencieux
Maison	HeuresClasse	HautBruit	HautLumière	HautCharge	Sonnerie avec vibreur
Maison	HeuresClasse	HautBruit	HautLumière	BasCharge	Silencieux
Maison	HeuresClasse	HautBruit	BasLumière	HautCharge	Sonnerie avec vibreur
Maison	HeuresClasse	HautBruit	BasLumière	BasCharge	Silencieux
Maison	HeuresClasse	BasBruit	HautLumière	HautCharge	Sonnerie
Maison	HeuresClasse	BasBruit	HautLumière	BasCharge	Silencieux
Maison	HeuresClasse	BasBruit	BasLumière	HautCharge	Sonnerie
Maison	HeuresClasse	BasBruit	BasLumière	BasCharge	Silencieux
Maison	HeuresLibre	HautBruit	HautLumière	HautCharge	Sonnerie avec vibreur
Maison	HeuresLibre	HautBruit	HautLumière	BasCharge	Silencieux
Maison	HeuresLibre	HautBruit	BasLumière	HautCharge	Sonnerie avec vibreur
Maison	HeuresLibre	HautBruit	BasLumière	BasCharge	Silencieux
Maison	HeuresLibre	BasBruit	HautLumière	HautCharge	Sonnerie

RÉSULTATS DE L'ALGORITHME D'APPRENTISAAGE POUR LE SÉNARIO D'APPLICATION (suite)

Localisation	Temps	Bruit	Lumière	Batterie	Forme
Maison	HeuresLibre	BasBruit	HautLumière	BasCharge	Silencieux
Maison	HeuresLibre	BasBruit	BasLumière	HautCharge	Sonnerie
Maison	HeuresLibre	BasBruit	BasLumière	BasCharge	Silencieux
En dehors	HeuresSommeil	HautBruit	HautLumière	HautCharge	Sonnerie avec vibreur
En dehors	HeuresSommeil	HautBruit	HautLumière	BasCharge	Silencieux
En dehors	HeuresSommeil	HautBruit	BasLumière	HautCharge	Sonnerie avec vibreur
En dehors	HeuresSommeil	HautBruit	BasLumière	BasCharge	Silencieux
En dehors	HeuresSommeil	BasBruit	HautLumière	HautCharge	Sonnerie
En dehors	HeuresSommeil	BasBruit	HautLumière	BasCharge	Silencieux
En dehors	HeuresSommeil	BasBruit	BasLumière	HautCharge	Sonnerie
En dehors	HeuresSommeil	BasBruit	BasLumière	BasCharge	Silencieux
En dehors	HeuresClasse	HautBruit	HautLumière	HautCharge	Sonnerie avec vibreur
En dehors	HeuresClasse	HautBruit	HautLumière	BasCharge	Silencieux
En dehors	HeuresClasse	HautBruit	BasLumière	HautCharge	Sonnerie avec vibreur
En dehors	HeuresClasse	HautBruit	BasLumière	BasCharge	Silencieux
En dehors	HeuresClasse	BasBruit	HautLumière	HautCharge	Sonnerie
En dehors	HeuresClasse	BasBruit	HautLumière	BasCharge	Silencieux
En dehors	HeuresClasse	BasBruit	BasLumière	HautCharge	Sonnerie
En dehors	HeuresClasse	BasBruit	BasLumière	BasCharge	Silencieux
En dehors	HeuresLibre	HautBruit	HautLumière	HautCharge	Sonnerie avec vibreur
En dehors	HeuresLibre	HautBruit	HautLumière	BasCharge	Silencieux

RÉSULTATS DE L'ALGORITHME D'APPRENTISAAGE POUR LE SÉNARIO D'APPLICATION (suite)

Localisation	Temps	Bruit	Lumière	Batterie	Forme
En dehors	HeuresLibre	HautBruit	BasLumière	HautCharge	Sonnerie avec vibreur
En dehors	HeuresLibre	HautBruit	BasLumière	BasCharge	Silencieux
En dehors	HeuresLibre	BasBruit	HautLumière	HautCharge	Sonnerie
En dehors	HeuresLibre	BasBruit	HautLumière	BasCharge	Silencieux
En dehors	HeuresLibre	BasBruit	BasLumière	HautCharge	Sonnerie
En dehors	HeuresLibre	BasBruit	BasLumière	BasCharge	Silencieux

LISTE DE RÉFÉRENCES BIBLIOGRAPHIQUES

Abowd, G., C. Atkeson, J. Hong, S. Long, R.Kooper et M. Pinkerton. 1997. « Cyberguide: A mobile context-aware tour guide ». *Wireless Networks,* vol. 3, n° 5, p. 421-433.

Aksit, M., et Z. Choukair. 2003. « Dynamic, Adaptive and Reconfigurable Systems Overview and Prospective Vision ». In *Proceedings of the ICDCSW'03* (May, 2003). p. 84-92. Providence, Rhode Island, USA.

Albin, T.S. 2003. *The art of software architecture design methods and techniques*. Wiley Publishing, Inc. .

Babar, M. Ali, et I. Gorton. 2004. « Comparison of Scenario-Based Software Architecture Evaluation Methods ». In *Asia-Pacific software engineering conference (APSEC 2004)*. p. 600-607. Busan , south Corea.

Baldauf, M., S. Dustdar et F. Rosenberg. 2007. « A Survey On Context-Aware Systems ». *International Journal of Ad Hoc and Ubiquitous Computing,* vol. 2, n° 4, p. 263-277.

Balsamo, S., et M. Marzolla. 2003. « A simulation-based approach to software performance modeling ». In. Tech. Rep. TR SAH/44, MIUR Sahara Project.

Bardram, J. E. 2005. « The Java Context Awareness Framework (JCAF) - A Service Infrastructure and Programming Framework for Context-Aware Applications ». In *the 3rd International Conference on Pervasive Computing (Pervasive 2005)*, sous la dir. de Verlag, Springer. Vol. volume 3468 of Lecture Notes in Computer Science, p. 98-115. Munich, Germany.

Bauer, J. 2003. « Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic ». In (2003). Diplomarbeit.

Bellucci, S., L. M. Hilty et D. Bütschi. 2002. « Informatique omniprésente: TA-SWISS en analyse les conséquences pour la santé et l'environnement ». In *Société de l'information*. En ligne. <http://www.ta-swiss.ch/www-remain/reports_archive/press_releases/pressemitteilungen2002/PM020924-Pervasive_Computing_f.pdf>. Consulté le 10 Avril, 2007.

Biegel, G., et V. Cahill. 2004. « A Framework for Developing Mobile, Context-aware Applications ». In *the 2nd IEEE Conference on Pervasive Computing and Communication* (March, 2004). p. 361-365. Orlando, Florida, USA.

Bouzy, B., et T. Cazenave. 1997. « Using the object oriented paradigm to model context in computer go ». In *Context`97*. Rio, Brazil.

Brezillon, P., M. R. Borges, J.A. Pino et J.-Ch. Pomerol. 2004. « Context-Awareness in Group Work: Three Case Studies ». In *2004 IFIP Int. Conf. on Decision Support Systems (DSS 2004)* (Juillet, 2004). Prato, Italie.

Brézillon, P., et J. C. Pomerol. 1999. « Contextual knowledge sharing and cooperation in intelligent assistant systems ». *Le Travail Humain,* vol. 62, n° 3, p. 223-246.

Brown, P. J. 1995. « The stick-e document: a framework for creating context-aware applications ». *Electronic Publishing Origination, Dissemination and Design,* vol. 8, n° 2 (JUNE & SEPTEMBER 1995), p. 259-272

Brown, P. J., J. D. Bovey et X. Chen. 1997. « Context-aware applications: from the laboratory to the marketplace ». *IEEE Personal Communications,* vol. 4, n° 5 (October 1997), p. 58-64.

Carthy, J. Mc, et S. Buvac. 1994. *Formalizing context (expanded notes).* . Coll. « STAN-CS-TN-94-13 ». Stanford University.

Chalmers, M. 2004. « A historical view of context ». *Computer supported cooperative work : CSCW,* vol. 13, p. 223-247.

Chen, G., et D. Kotz. 2000. *A Survey of Context-Aware Mobile Computing Research*. TR2000-381. Dartmouth: Dartmouth College Computer Science

Chen, H. 2004. « An Intelligent Broker Architecture for Pervasive Context-Aware systems, ». Baltimore County, University of Maryland

Chen, H., T. W. Finin et A. Joshi. 2004. « Semantic web in the context broker architecture ». In *PerCom 2004*, sous la dir. de Society, IEEE Computer. p. 277-286. Orlando, Florida, USA.

Chen, H., F. Perich, T. W. Finin et A. Joshi. 2004. « Soupa : Standard ontology for ubiquitous and pervasive applications ». In *MobiQuitous*. p. 258-267. Boston, Massachusetts, USA

Cheverest, K., N. Davis, K. Mitchel, A. Friday et C. Efstratiou. 2000. « Developing a Context-aware Tourist Guide: Some Issues and Experiences ». In *International conference on Computer Human Interaction (CHI'2000)* (April 2000). p. 17-24 The Hague, Netherlands: ACM Press.

Cliquet, G. 2007. « Informatique Ambiante: Nouveaux défis pour le Design ». In *Recherche 2000*. <<http://turing.lecolededesign.com/~gcliquet/>>. Consulté le Mars 2007.

« CPN Tools: Computer Tool for Coloured Petri Nets ». En ligne. (12 Mars, 2009). <http://wiki.daimi.au.dk/cpntools/cpntools.wiki>.

Dey, A., G. D. Abowd et D. Salber. 2001. « A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications ». *Human-Computer Interaction,* vol. 16, n° 2-4, p. 97–166.

Dey, A. K. 2001. « Understanding and using context ». *Personal and ubiquitous computing,* vol. 5, p. 4-7.

Dey, A. K., et G. D. Abowd. 2000. « Towards a Better Understanding of Context and Context-Awareness ». In *Workshop on the What, Who, Where, When, and How of Context Awareness Conference on Human Factors in Computer Systems (CHI2000)*. The Hague, Netherlands.

Dey, A., J. Mankoff, G. D. Abowd et S. Carter. 2002. « Distributed mediation of ambiguous context in aware environments ». In *the 15th Annual Symposium on User Interface Software and Technology (UIST)*. p. 121-130. Paris, France.

Efstratiou, C., K. Cheverst, N. Davies et A. Friday. 2001. « An Architecture for the Effective Support of Adaptive Context- Aware Applications ». In *Proceedings of the. 2nd Int. Conf. in Mobile Data Management (MDM'01)* (January 2001). p. 15-26. Hong Kong.

« Encyclopédie Larousse ». En ligne. (5 Avril, 2008). <<http://www.encyclopedie-larousse.fr/>>.

Fahy, P., et S. Clarke. 2004. « CASS - a middleware for mobile context-aware applications ». In *MobiSys Workshop on Context Awareness* (June 2004). p. 304-308. Boston, Massachusetts, USA

Fayad, M., et P. Marshall Cline. 1996. « Aspects of software adaptability ». *Communications of the ACM,* vol. 39, n° 10, p. 58-59.

« Grand Dictionnaire : Office québécois de la langue française ». En ligne. <<http://www.granddictionnaire.com/>>. Consulté le 20 Mai, 2008.

Gu, T., X. H. Wang, H. K. Pung et D. Q. Zhang. 2004a. « A Middlewar for Context-Aware Mobile Services ». In *IEEE Vehicular Technology Conference* (Spring 2004). Milan, Italy.

Gu, T., X.H. Wang, H. K. Pung et D.Q. Zhang. 2004b. « An Ontology-based Context Model in Intelligent Environments ». In *Communication Networks and Distributed Systems Modeling and Simulation Conference*. San Diego, California, USA.

Gwizdka, J. 2000. « What's in the Context? ». In *Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000)* (April 3, 2000). The Hague, The Netherlands.

H.Wang, X., T. Gu, D. Q. Zhang et H. K. Pung. 2004. « Ontology Based Context Modeling and Reasoning using OWL ». In *Workshop on Context Modeling and Reasoning, In conjunction with the 2nd IEEE Intl Conf PerCom 2004*. Orlando, Florida, USA.

« Hachette Multimédia ». En ligne. (13 Juin, 2007). <<http://www.encyclopedie-hachette.com/W3E/>>.

Held, A., S. Buchholz et A. Schill. 2002. « Modeling of context information for pervasive computing applications ». In *the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI '02)* (July 2002). Orlando, Fla, USA.

Henricksen, K., et J. Indulska. 2004. « Modelling and Using Imperfect Context Information ». In *Workshop Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communications (PerCom2004)* p. 33-37. Orlando, FL, USA.

Henricksen, K., et J. Indulska. 2006. « Developing context-aware pervasive computing applications: Models and approach ». *Journal of Pervasive and Mobile Computing,* vol. 2, n^{o} 1, p. 37-64.

Henricksen, K., J. Indulska, T. Mc Fadden et S. Balasubramaniam. 2005. « Middleware for Distributed Context-Aware Systems ». In *the International Symposium on Distributed Objects and Applications (DOA'05)* (October 2005). p. 846-863. Cyprus: Springer Verlag LNCS 3760.

Henricksen, K., J. Indulska et T. McFadden. 2005. « Modelling Context Information with ORM ». In *OTM Workshops 2005* (2005). Vol. LNCS 3762, p. 626-635. Agia Napa, Cyprus: Springer Verlag.

Henricksen, K., J. Indulska et A. Rakotonirainy. 2002. « Modeling Context Information in Pervasive Computing Systems ». In *Proc. of the First International Conference on Pervasive Computing, Pervasive'2002* (August 2002). Vol. 2414, p. 167-180. Zurich, Switzerland: Lecture Notes in Computer Science, Springer Verlag, LNCS.

Hofer, T., W. Schwinger, M. Pichler, G. Leonhartsberger et J. Altmann. 2002. « Context-awareness on mobile devices - the hydrogen approach ». In *the 36th Annual Hawaii International Conference on System Sciences*. p. 292-302. Hawaii, USA.

Hofer, T., W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann et W. Rets-chitzegger. 2003. « Context-awareness on mobile devices - the hydrogen approach ». In *the 36th Annual Hawaii International Conference onSystem Sciences (HICSS'03)* (January 06-09, 2003). p. 292-302. Hawaii, USA.

Indulska, J., R. Robinson, A. Rakotonirainy et K. Henricksen. 2003. « Experiences in using CC/PP in context-aware systems ». In *the 4th International Conference on Mobile Data Management*. p. 247-261. Melbourne, Australia: Springer Verlag, LNCS 2574.

Jensen, K. 1998. « A Brief Introduction to Coloured Petri Nets ». In *Workshop on the Applicability of Formal Models* (June 1998). p. 55-58. Aarhus, Denmark.

Kagal, L., V. Korolev, H. Chen, J. Anupam et T. Finin. 2001. « Centaurus : a framework for intelligent services in a mobile environment ». In *The 21st International conference on distributed computing system workshop*. p. 195-201. Mesa, AZ, USA: IEEE Computer Society.

Kammanahalli, H., S. Gopalan, V. Sridhar et K. Ramamritham. 2004. « Context-aware retrieval in web-based collaborations ». In *PerCom Workshops*, sous la dir. de Society, IEEE Computer. p. 8-12. Orlando, FL, USA.

Kang, D., H. Lee, E. Ko, K. Kang et J. Lee. 2006. « A Wearable Context Aware System for Ubiquitous Healthcare ». In *The 28th IEEE EMBS Annual International Conference*. p. 5192-5195. New York City, USA.

Kazman, R. , G. Abowd, L. Bass et P Clements. 1996. « Scenario-based analysis of software architecture ». *IEEE Software* vol. 13, n° 6, p. 47-55.

Keeney, J., et V. Cahill. 2003. « Chisel: A Policy-Driven, Context-Aware, Dynamic Adaptation Framework ». In *The 4th International Workshop on Policies for Distributed Systems and Networks* (June 2003). p. 3-13. Lake Como, Italie: IEEE.

Kjær, K. E. 2007. « A survey of context-aware middleware ». In *the 25th conference on IASTED International Multi-Conference: Software Engineering* (February 13-15, 2007). p. 148-155. Innsbruck, Austria.

Korpipää, P., J. Mantyjarvi, J. Kela, H. Keranen et E-J. Malm. 2003. « Managing context information in mobile devices ». *IEEE Pervasive Computing,* vol. 2, n° 3 (July-September, 2003), p. 42-51.

Kown, O. B. 2003. « I know what you need to buy: context-aware multimedia-based recommendation system ». *Expert system with applications,* vol. 25, p. 387-400.

Krause, M., et I. Hochstatter. 2005. « Challenges in Modelling and Using Quality of Context (QoC) ». In *the 2nd International Workshop on Mobility Aware Technologies and Applications (MATA 2005)* (October, 2005). Montreal, Canada: Lecture Notes in Computer Science (LNCS), Springer.

Laforest, F., et F. Le Mouël. 2005/2006. « Systèmes d'information pervasifs ». <liris.cnrs.fr/frederique.laforest/master/CoursSIP07_08.ppt >.

Mattsson, M. , H. Grahn et F. Mårtensson. 2006. « Software Architecture Evaluation Methods for Performance, Maintainability, Testability, and Portability ». In *Second International Conference on the Quality of Software Architectures (QoSA 2006)* (June 27-29, 2006). Västerås, Sweden.

Miraoui, M., et C. Tadj. 2007a. « A Service Oriented Architecture for Context-aware Systems ». *Journal of Research in Computing Science* vol. 29, (Novembre, 2007), p. 236-244.

Miraoui, M., et C. Tadj. 2007b. « A service oriented definition of context for pervasive computing ». In *The 16th International Conference on Computing* (November, 2007). Mexico city, Mexico: IEEE Computer Siciety (to appear).

Miraoui, M., C. Tadj et C. Ben Amar. 2008a. « Architectural Survey of Context-Aware Systems in Pervasive Computing Environment ». *Ubiquitous Computing and Communication Journal (UBICC),* vol. 3, n° 3 (June 2008).

Miraoui, M., C. Tadj et C. Ben Amar. 2008b. « Context Modeling and Context-Aware Service Adaptation for Pervasive Computing Systems ». *Journal of Computer and Information Science and Engineering (IJCISE),* vol. 2, n° 3, p. 148-157

Miraoui, M., C. Tadj et C. Ben Amar. 2009. « Modeling and Simulation of a Multiagent Service Oriented Architecture for Pervasive Computing Systems ». In *Workshop on Middleware for Ubiquitous and Pervasive Systems (WMUPS`2009)* (19 Juin, 2009). Trinity College, Dublin, Ireland (Accepté): ACM.

Mostefaoui, K. G., J. Pasquier-Rocha et P. Brezillon. 2004. « Context-Aware Computing: A Guide for the Pervasive Computing Community ». In *the Proceedings of the 2004 ACS/IEEE International Conference on Pervasive Services (ICPS'2004)* (19-23 July). p. 39-48. Beirut, Lebanon.

Narayanan, D., J. Flinn et M. Satyanarayanan. 2000. « Using history to improve mobile application adaptation ». In *Proceedings of the 3rd IEEE Workshop on Mobile Computing Systems and Applications* (Dec. 2000). Monterey, California, USA.

Noy, N. F., et C. D. Hafner. 1997. « The state of the art in ontology design: a survey and comparative review ». *A.I. Magazine,* vol. 18, n° 3, p. 53-74.

Parsia, B., et E. Sirin. 2004. « Pellet: An OWL DL Reasoner ». In *The third International Semantic Web Conference (ISWC2004)* (November 2004). Hiroshima, Japan.

Pascoe, J. 1997. « The Stick-e Note Architecture: Extending the Interface Beyond the User ». In *International Conference on Intelligent User Interfaces*. p. 261-264.

Pascoe, J. 1998. « Adding Generic Contextual Capabilities to Wearable Computers ». In *Symposium on Wearable Computers (ISWC'98)* (October 1998). p. 92 - 99. Pittsburgh, Pennsylvania USA.

Pellet: The Open Source OWL DL Reasoner. En ligne. <http://pellet.owldl.com/>. Consulté le 12 Aout, 2008.

Petrelli, D., E. Not, C. Strapparava, O. Stock et M. Zancanaro. 2000. « Modeling Context is Like Taking Pictures ». In *CHI'2000 Workshop on Context Awareness* (April 1-6, 2000). The Hague, Netherlands.

Preuveneers, D., J. V. den Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, Y. Berbers, K. Coninx, V. Jonckers et K. De Bosschere. 2004. « Towards an extensible context ontology for ambient intelligence ». In *Ambient Intelligence: Second European Symposium*. Vol. 3295, p. 148-159. Eindhoven, The Netherlands.

Ranganathan, A., R. H. Campbell, A. Ravi et A. Mahajan. 2002. « ConChat: A Context-Aware Chat Program ». *IEEE Pervasive Computing,* vol. 1, n° 3, p. 51-57.

Razzaque, M. A., S. Dobson et P. Nixon. 2005. « Categorization and modeling of quality in contextinformation ». In *the IJCAI 2005 Workshop on AI and Autonomic Communications* (August 2005). Edinburgh, Scotland.

Reisig, W. 1985. « Petri nets: An Introduction ». In *EATCS Monographs on Theoretical Computer Science*. Vol. 4. Springer-Verlag.

Riveill, M. 2002. « Ubiquitous computing: les challenges logiciels ». Université de Nice-ESSI. En ligne. <http://rangiroa.essi.fr/cours/info-mobile/02-ubiquitous-computing.pdf>

Robert, P. 1991. *Le petit Robert 1: dictionnaire alphabétique et analogique de la langue francaise*. Dictionnaires le Robert.

Robinson, R., K. Henricksen et J. Indulska. 2007. « XCML: A runtime representation for the Context Modelling Language ». In *The 4th International Workshop on Context Modelling and Reasoning (CoMoRea)*. New York, USA: IEEE Computer Society.

Ryan, N. 2007. « Contextml: Exchanging contextual information between a mobile client and the field note server ». En ligne. <http://ww.cs.kent.ac.uk/projects/mbicomp/fndConteXtML.html>.

Ryan, N., J. Pascoe et D. Morse. 1997. « Enhanced Reality Fieldwork: The Context-aware Archaeological Assistant ». In *Computer Applications in Archaeology*. Oxford, UK.

Sadeh, N. M., F. L. Gandon et O. B. Kwon. 2005. « Ambient Intelligence: The *MyCampus* Experience ». *Carnegie MellonUniversity, Technical Report CMU-ISRI-05-123*, (July 2005).

Saha, D., et A. Mukherjee. 2003. « Pervasive computing : a paradigm for the 21st century ». *IEEE Computer journal*, (Mars 2003), p. 25-31.

Satyanarayanan, M. 2001. « Ubiquitous Computing : les challenges logiciels ». *IEEE Personnal communication*, (August 2001), p. 10-17.

Schilit, B. N., N. Adams et R. Want. 1994. « Context-Aware Computing Applications ». In *Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA,* (Decenber 1994). p. 85-90. Santa Cruz, CA, : IEEE Computer Society.

Schilit, B. N., et M. M. Theimer. 1994. « Disseminating Active Map Information to Mobile Hosts ». *IEEE Network, 8(5)*, (September/October 1994), p. 22-32.

Schmidt, A., K. Asante Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven et W. Van de Velde. 1999. « Advanced Interaction in Context ». In *Handheld and ubiquitous computing. International symposium No1* (27-29 September 1999). Vol. 1707, p. 89-101. Karlsruhe , ALLEMAGNE Lecture notes in computer science

Segara, M.T., et F. André. 2000. « A Framework for Dynamic Adaptation in Wireless Environments ». In *Proceedings of the Technology of Object Oriented Languages and systems (TOOLS 33)* (June 2000). St. Malo, France.

Singh, A., et M. Conway. 2006. *Survey of Context aware Frameworks - Analysis and Criticism*. UNC-Chapel Hill ITS, The University of North Carolina. <http://its.unc.edu/teap/tap/core/caf_review.pdf>.

South, G., A.P. Lenaghan et R.R. Malyan. 2000. *Using reflection for service adaptation in mobile clients (t4)*. Kingston University, UK.

Strang, T., et C. Linnhoff-Popien. 2004. « A Context Modeling Survey ». In *Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing* (September 2004). Nottingham, England, UK.

Strang, T., C. Linnhoff-Popien et K. Frank. 2003. « CoOL: A Context Ontology Language to enable Contextual Interoperability ». In *4th IFIP International Conference on Distributed Applications & Interoperable Systems* (November 17-21, 2003). Paris, France.

Virgilio, R. De, et R. Torlone. 2006. « Modeling heterogeneous context information in adaptive web based applications ». In *The 6th international conference on Web engineering* (2006). p. 56-63. Palo Alto, California, USA

Want, R., A. Hopper, V. Falcao et J. Gibbons. 1992a. « The Active Badge Location System ». *ACM Transcation on Information Systems* vol. 10, n° 1, p. 42-47.

Want, R., A. Hopper, V. Falcão et J. Gibbons. 1992b. « The Active Badge Location System ». *ACM Transaction on Information Systems,* vol. 10, (Octobre 1992), p. 91-102.

Want, R., B. N. Schilit, N. I. Adams, R. Gold, K. Petersen, D. Goldberg, J. R. Ellis et M. Weiser. 1995. « An overview of the ParcTab ubiquitous computing experiment ». *IEEE Personal Communications,* vol. 2, n° 6, p. 28-43.

Ward, A. J., et A. Hopper. 1997. « A New Location Technique for the Active Office ». *IEEE Personnal Communications,* vol. 4, n° 5 (October 1997), p. 42-47.

Weiser, M. 1991. « The Computer for the Twenty-First Century ». *Scientific Journal*, (September), p. 94-104.

Weiser, M. 1993. « some computer science issues in ubiquitous computing ». In *Communications of the ACM*. Vol. 36, p. 74-84.

« Weka 3: Data Mining Software in Java ». En ligne. (22 Juin, 2008). <http://www.cs.waikato.ac.nz/ml/weka/>.

Winograd, T. 2000. « Architectures for Context ». *Human Computer Interaction,* vol. 15, n° 4, p. 263-322.

Yarvis, M., Peter L. Reiher et Gerald J. Popek. 1999. « Conductor: A framework for distributed adaptation ». In *Workshop on Hot Topics in Operating Systems*. p. 44-49.