HAL
open science

Vers une approche d'adaptation dynamique et temps-réel du contenu informationnel d'une interface utilisateur dans un environnement ubiquitaire

Imen Ben Ismail Dhrif

To cite this version:

Imen Ben Ismail Dhrif. Vers une approche d'adaptation dynamique et temps-réel du contenu informationnel d'une interface utilisateur dans un environnement ubiquitaire. Informatique mobile. ECOLE NATIONALE DES SCIENCES DE L'INFORMATIQUE, 2014. Français. NNT : . tel-01334765

HAL Id: tel-01334765

https://hal.science/tel-01334765v1

Submitted on 21 Jun 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
UNIVERSITE DE LA MANOUBA
ECOLE NATIONALE DES SCIENCES DE L'INFORMATIQUE

THÈSE

En vue de l'obtention du diplôme de

DOCTEUR EN INFORMATIQUE

Par
Imen BEN ISMAIL DHRIF

Vers une approche d'adaptation dynamique et temps-réel du contenu informationnel d'une interface utilisateur dans un environnement ubiquitaire

Réalisée au sein du
Laboratoire CRISTAL

Soutenue le 01/12/2014 devant le jury composé de

Président : Prof. Belhassen ZOUARI (SUPCOM, Tunis)
Rapporteur : Prof. Christophe KOLSKI (Université de valencienne, France)
Rapporteur : Prof. Sadok BEN YAHIA (Faculté des Sciences de Tunis)
Examinateur : MC. Sadok BOUAMAMA (Université de la Manouba, ENSI)
Directeur de thèse : MC. Faouzi MOUSSA (Faculté des Sciences de Tunis)

"The reasonable man adapts himself to the world; the unreasonable one persists in trying to adapt the world to himself. Therefore, all progress depends on the unreasonable man"

George Bernard Shaw.

Remerciements

*Je souhaite remercier en premier lieu mon directeur de thèse, Mr. **Faouzi MOUSSA**, Maître de Conférences à la Faculté des Sciences de Tunis, pour son attention de tout instant sur mes travaux, pour ses conseils avisés et son écoute qui ont été prépondérants pour la bonne réussite de cette thèse. Son énergie, sa confiance et sa gentillesse ont été des éléments moteurs pour moi. J'ai pris un grand plaisir à travailler avec lui.*

****

*Les travaux présentés dans cette thèse ont été réalisés au sein du Laboratoire Centre de Recherche en Réseau, Image, Système, Architecture et multimédia (CRISTAL), dirigé par Mr. **Faouzi GHORBEL**, Professeur à l'Ecole Nationale des Sciences de l'Informatique, à qui j'adresse mes remerciements pour son accueil.*

****

*Je voudrais également remercier les rapporteurs de cette thèse : **Mr. Christophe KOLSKI**, Professeur des Universités de Valencienne (France), et **Mr. Sadok BEN YAHIA**, Professeur à la Faculté des Sciences de Tunis, pour l'intérêt qu'ils ont porté à mon travail.*

*J'associe à ces remerciements **Mr. Sadok BOUAMAMA**, Maître de conférences à l'école nationale des sciences de l'informatique (ENSI) pour avoir accepté d'examiner mon travail.*

*Je remercie **Mr. Belhassen ZOUARI** d'avoir accepté de présider le Jury de ma soutenance.*

****

*Je désire remercier spécialement Mme **RIAHI Meriem** et Mlle **RIAHI Ines** pour leur sympathie, leur amitié. J'ai eu beaucoup de plaisir à travailler avec elles.*

****

*Je remercie ma fille **Zeineb** pour le bonheur qu'elle me procure en la voyant grandir. Je lui adresse toutes mes excuses pour les moments où j'ai du la laisser seule.*

Une seule ligne ne saurait transcrire ma gratitude envers mon père, ma mère, ma sœur, mon frère et mon mari qui m'ont offert leurs encouragements et leur soutien pendant toutes ces années d'études.

Résumé

Titre de la thèse

«Vers une approche d'adaptation dynamique et temps-réel du contenu informationnel d'une interface utilisateur dans un environnement ubiquitaire»

Résumé

Dans un environnement ubiquitaire, un utilisateur mobile pourrait voir sur son interface diverses informations provenant des différents dispositifs disséminés partout dans l'environnement dans lequel il évolue. Toutefois, à un moment donné et dans un contexte bien déterminé, l'utilisateur aurait grandement besoin de certaines informations mais trouvera d'autres complètement inutiles ; ce choix dépend impérativement de ses préférences et de son contexte d'utilisation. En conséquence, l'utilisateur mobile devrait avoir l'information la plus appropriée pour achever sa tâche. Rappelons ainsi le but ultime de l'informatique ubiquitaire qui consistait à **faciliter les activités de l'utilisateur** – notamment celles pénibles et fastidieuses – ainsi que de remplacer son intervention dans toute tâche pouvant lui perdre son temps sans qu'il perçoive cette assistance. Notre contribution est ainsi la conception et la réalisation d'un système ubiquitaire permettant une adaptation dynamique et temps-réel du contenu informationnel de l'interface utilisateur en procurant à l'utilisateur l'information qui répond au mieux à ces exigences. Ce fonctionnement doit donc être régi par une stratégie d'adaptation se basant sur **des modèles évolutifs et dynamiques.** Ceci passe implicitement par détecter le contexte courant de l'utilisateur et l'aider à atteindre ses objectifs en modifiant éventuellement le contenu et le comportement de l'interface selon le changement du contexte de cet utilisateur tout en veillant à lui fournir l'information dont il a besoin sous la forme qui est la mieux adaptée. En résumé, la construction du modèle global d'une tâche utilisateur s'effectue à partir de structures de base modélisant les différentes actions élémentaires de l'utilisateur en se basant sur l'application de différentes opérations de composition. L'intégration des données contextuelles s'appuie sur certaines caractéristiques du modèle réseau de Petri représentant le comportement de l'utilisateur en fonction du contexte d'utilisation. De la même manière, nous procédons afin d'associer à certaines transitions, dans ces dits modèles, les variables adéquates qui se réfèrent aux besoins informationnels de l'utilisateur. Outre la déduction des besoins utilisateurs en termes d'information notre contribution fait intervenir à la nature dynamique des environnements ubiquitaires, à savoir le changement du contexte d'usage de l'utilisateur mobile. Nous cherchons par cette approche à suivre l'utilisateur dans l'environnement où il se situe à travers ses activités. Notre approche est basée, spécifiquement, sur la variation des activités de l'utilisateur en fonction du contexte.

Mots clés

Informatique ubiquitaire, Contexte, Sensibilité au contexte, Modélisation dynamique, Approche basée sur les modèles, Interface homme-machine, Réseaux de Petri, Ontologie, OWL, OWL-S.

Keywords

Ubiquitous computing, Context, Context-awarness, Dynamic modeling, User Interface Adaptation, OWL, OWL-S, Ontology, Petri-nets.

TABLE DES MATIERES

Introduction

1. AVANT PROPOS 13
2. L'INFORMATIQUE UBIQUITAIRE 13
3. CONTEXTE ET SENSIBILITE AU CONTEXTE 14
4. POSITIONNEMENT DE LA PROBLEMATIQUE 15
5. CONTRIBUTIONS 16
6. CAS D'ETUDE 16
7. ORGANISATION DU MEMOIRE 17

Chapitre 1. Adaptation au contexte dans les environnements ubiquitaires

1. INTRODUCTION 20
2. NOTION DU CONTEXTE DANS LE CADRE DE L'INFORMATIQUE UBIQUITAIRE 20
   2.1 Définition du Contexte 20
   2.2 Sensibilité au contexte 21
   2.3 Classification des paramètres contextuels 22
   2.4 Adaptation au contexte dans un environnement ubiquitaire 23
3. PLATEFORMES ET ARCHITECTURES D'ADAPTATION AU CONTEXTE 23
   3.1 Structure générale d'un système sensible au contexte 24
      3.1.1. Couche des capteurs : "Sensing layer" 24
      3.1.2. Couche de traitement du contexte : « Contextual data processing layer » 25
      3.1.3. Couche d'applications : « Application layer » 26
   3.2 Description fonctionnelle des plateformes et architectures existantes 27
      3.2.1. Context Information Service 27
      3.2.2. Context toolkit 28
      3.2.3. HyCon 28
      3.2.4. CoBrA : Context Broker Architecture 29
      3.2.5. Java ContextAware Framework 30
      3.2.6. Service-Oriented Context-Aware Middleware SOCAM 31
      3.2.7. SECAS : Simple Environment for Context-aware Systems 32
      3.2.8. User-Adaptive and Context-aware Architecture for Mobile and Desktop Training Applications 32
      3.2.9. Architecture-based Approach to Context-aware Adaptive Software Systems 34
   3.3 Discussion et synthèse sur les plateformes sensibles au contexte 35
   3.4 Besoins opérationnels d'un système d'adaptation au contexte 38
   3.5 Conclusion 39
4 CONCLUSION GENERALE 40

Chapitre 2. Cadre conceptuel pour une adaptation dynamique et temps réel au contexte d'utilisation

1. INTRODUCTION 42
2. ETUDE DE CAS ET PROBLEMATIQUES 42
   2.1. Aperçu général sur le cas d'étude : l'hôpital ubiquitaire 42
   2.2. Analyse fonctionnelle et identification des problèmes engendrés 46
3. CADRE CONCEPTUEL PROPOSE POUR L'ADAPTATION DYNAMIQUE ET TEMPS-REEL 47
   3.1. Objectifs du système RADEM 47
   3.2. Aperçu général du système RADEM 48
      3.2.1 Gestion et analyse des données contextuelles 48

3.2.2 Approche d'adaptation aux données contextuelles *48*
3.2.3 Génération de l'interface utilisateur graphique *50*
3.3 RADEM : Décomposition et description fonctionnelles **50**
3.3.1 Aperçu sur le composant « AE : Adaptation Engine » *51*
3.3.2 Aperçu sur le composant « Analysis and Query Building » *51*
3.3.3 Description du composant « KR : Knowledge Repositories » *52*
3.4 Description détaillée du fonctionnement du composant « *Adaptation Engine* » **53**
4. GESTION ET MODELISATION DES PARAMETRES CONTEXTUELS **55**
4.1 Principales techniques de modélisation des données contextuelles **56**
4.1.1 Modèle Attributs/Valeur *56*
4.1.2 Les modèles de Marquage *56*
4.1.3 Modèle basé sur les ontologies *57*
4.1.4 Modèles orienté-objet *58*
4.1.5 Modèles basés sur la logique *58*
4.1.6 Modèles graphiques *58*
4.2 Synthèse et choix de la technique de modélisation des données contextuelles **58**
4.3 Gestion des données contextuelles **60**
5. GENERATION DE L'INTERFACE UTILISATEUR **64**
5.1 Classification du contexte : une étape fondamentale **65**
5.2 Génération du contenu **66**
6. CONCLUSION **68**

Chapitre 3. Approche d'adaptation du contenu informationnel d'une interface utilisateur

1. INTRODUCTION **71**
2. CADRE GENERAL ET POSITIONNEMENT DE L'APPROCHE **71**
2.1 Objectifs **71**
2.2 Adaptation au contexte d'utilisation : étude synthétique **72**
2.2.1 Taxonomies fonctionnelles de l'adaptation *72*
2.2.2 Présentation de l'approche adoptée : Approche à base de modèles *73*
2.3 Positionnement de l'approche **75**
3. APPROCHE PROPOSEE **76**
3.1 Présentation de l'approche proposée **78**
3.2 Identification temps-réel des besoins utilisateur **79**
3.2.1 Modélisation du comportement de l'utilisateur avec les RdPIs *79*
3.2.2 Déduction des besoins utilisateurs *83*
3.1 Prise en compte du changement d'activités **87**
4. GESTION DYNAMIQUE ET TEMPS-REEL DES MODELES D'INTERACTION DANS UN ENVIRONNEMENT UBIQUITAIRE **92**
4.1 Correspondance entre une spécification d'un modèle basé sur RdPI avec la spécification des services OWL-S **98**
4.1.1 Modélisation générique d'une action élémentaire *100*
4.1.2 Modélisation d'une activité *103*
4.1.3 Illustration *103*
4.2 Cadre conceptuel et fonctionnel de l'approche **105**
5. ALGORITHMES DE FONCTIONNEMENT **111**
5.1 Algorithme spécifique au mode de fonctionnement normal **111**
5.2 Les autres algorithmes **112**
5.2.1 Algorithmes de préparation des actions élémentaires *112*
5.2.2 Algorithmes de composition et exécution du modèle *115*
6. DISCUSSION ET CONTRIBUTION **116**
6.1 Travaux connexes **116**
6.2 Domaines de contribution **117**
7. CONCLUSION **118**

Chapitre 4. Expérimentation et évaluation de l'approche proposée

1. INTRODUCTION **122**
2. PLATEFORME UTILISEE POUR L'EXPERIMENTATION DE L'APPROCHE PROPOSEE **122**
3. IMPLEMENTATION DE LA PARTIE « KNOWLEDGE REPOSITORIES » **124**
3.1 Conceptualisation des données contextuelles spécifiques au cas d'étude **126**
3.1.1 Modèle Conceptuel de l'ontologie HealthONT *127*
3.1.2 Modèle Conceptuel de l'ontologie DiabetesDiseaseONT *127*
3.2 Ontologisation **129**
3.3 Opérationnalisation **132**
4. IMPLEMENTATION DE LA FACETTE FONCTIONNELLE **134**

4.1 Génération du code WSDL de l'action « *GlucoseRateMeasure* » 135
4.2 Création de l'ontologie de format des données et enrichissement des services par la sémantique 138
4.3 Mise en œuvre des actions élémentaires (Grounding) 141
5. IMPLEMENTATION DE LA PARTIE SYSTEME (PROTOTYPE) 143
5.1 Tests et Résultats 144
5.1.1 Déduction en temps-réel des besoins utilisateur (en termes d'information) 145
5.1.2 Adaptation du contenu selon les spécificités des utilisateurs 146
5.1.3 Adaptation du contenu selon le changement du contexte 146
6. CONCLUSION 147

Conclusion générale et perspectives

1. INTRODUCTION 149
2. CONCLUSION SUR LES TRAVAUX EFFECTUES 149
3. CONTRIBUTION 149
4. PERSPECTIVES ET FUTURS TRAVAUX 150

Références bibliographiques et webographiques 153

Liste des figures

Chapitre 1

Figure 1.1 Architecture basique d'un système sensible au contexte 24
Figure 1.2 Context Information Service (Pascoe, 1999) 27
Figure 1.3 L'architecture "Context toolkit" (Dey, 1999) 28
Figure 1.4 L'architecture HyCon (Niels, 2003) 29
Figure 1.5 L'architecture CoBrA (Chen, 2004) 30
Figure 1.6 Java ContextAware Framework (Bardram, 2005) 30
Figure 1.7 L'architecture SOCAM (Gu, 2005) 32
Figure 1.8 La plateforme SECAS (Chaari, 2006) 33
Figure 1. 9 L'architecture CAAMDTA (Buttussi, 2008) 34
Figure 1.10 Architecture to Context-aware Adaptive Software Systems (Hussein, 2011) 34
Figure 1.11 Processus de développement des systèmes d'adaptation 37
Figure 1.12 Evolution des objectifs des systèmes d'adaptation au contexte 37

Chapitre 2

Figure 2.1 Simulation des différentes parties constituant un Hôpital Ubiquitaire 43
Figure 2.2 Un exemple de message émis par le système 45
Figure 2.3 Cadre Général du système RADEM 49
Figure 2.4 Modèle conceptuel et fonctionnel du système RADEM 51
Figure 2.5 Flux de données entrant et sortant du moteur d'adaptation 54
Figure 2.6 Différentes approches de modélisation du contexte 56
Figure 2.7 Informations décrites avec UAProf 57
Figure 2.8 Informations décrites avec UED 57
Figure 2.9 Mécanisme pour le traitement et la gestion des données contextuelles 62
Figure 2.10 Synchronisation et mise-à-jour de l'ontologie 62
Figure 2.11 Identification du contenu de l'interface en se basant sur le type de widget 67
Figure 2.12 Génération de l'interface utilisateur 68
Figure 2.13 Evolution des systèmes d'adaptation en fonction du temps 69

Chapitre 3

Figure 3.1 Positionnement de l'approche proposée 77
Figure 3.2 Réseau de Petri interprété 80
Figure 3.3 Structure modélisant l'état d'un utilisateur face à une Action Elémentaire 81
Figure 3.4 Composition Séquentielle 82
Figure 3.5 Structures de la Composition Parallèle 82
Figure 3.6 Composition Parallèle 83
Figure 3. 7 Modélisation du comportement de l'utilisateur 85
Figure 3.8 Intégration des besoins utilisateurs à l'étape EA2 86
Figure 3. 9 Modélisation du comportement de l'infirmier lors de traitement d'un cas d'hypoglycémie (après avoir effectué les transformations nécessaires) 88

Figure 3. 10 Modélisation du comportement de l'infirmier avec mise en évidence des besoins utilisateur dans leurs emplacements correspondants 89
Figure 3. 11 Nouvelle exigence de la modélisation de l'interaction Homme-Système dans les environnements dynamiques 90
Figure 3. 12 Modélisation RdPI d'interaction (situation ancienne : Patient conscient et nouvelle situation : patient inconscient) 91
Figure 3. 13 Etat des lieux des principaux facteurs intervenant dans un environnement ubiquitaire 92
Figure 3. 14 Représentation de l'ontologie d'un service 94
Figure 3. 15 Classes et propriétés du Service Profile 95
Figure 3. 16 Mapping entre OWL-S et WSDL 95
Figure 3. 17 Niveau haut de l'ontologie des processus 96
Figure 3.18 Modèle conceptuel de l'ontologie « RDPIModelOntology » 99
Figure 3.19 Représentation partielle de l'ontologie « RdPIModelOntology » 100
Figure 3. 20 Modélisation générique d'une action élémentaire EAi 101
Figure 3. 21 Instanciation d'une action élémentaire en une spécification OWL-S 102
Figure 3. 22 Paramétrisation de l'action élémentaire : GlucoseRateMeasure 104
Figure 3. 23 Paramétrisation de l'action élémentaire : CarbohydrateIngest 104
Figure 3. 24 Flux de données entre les deux actions élémentaires envisagées 105
Figure 3. 25 Le système d'adaptation : architecture fonctionnelle réduite 105
Figure 3. 26 Modèle Conceptuel de l'approche de composition dynamique d'un modèle d'activité 107
Figure 3.27 Enregistrement des actions élémentaires 108
Figure 3. 28 Fonctionnement basique et protocole de communication 111
Figure 3. 29 Le mode de fonctionnement normal 112
Figure 3. 30 Algorithme de présélection 113
Figure 3. 31 Algorithme de Sélection 113
Figure 3. 32 Correspondance (Matching) entre la requête système et la spécification OWL-S d'une action élémentaire 114
Figure 3. 33 Algorithme de construction du modèle 115
Figure 3. 34 Fonction de fusion 116
Figure 3. 35 Système global d'adaptation dynamique et temps-réel 120

Chapitre 4

Figure 4. 1 Plateformes d'exécution 123
Figure 4. 2 Les différentes facettes de la simulation d'un hôpital «ubiquitaire» 124
Figure 4. 3 Les Différentes Composantes Ontologiques 125
Figure 4. 4 Démarche pour la création et l'opérationnalisation de l'ontologie 126
Figure 4. 5 Modèle Conceptuel de l'ontologie HealthONT 128
Figure 4. 6 Modèle Conceptuel de l'ontologie de la maladie de diabète DiabetesDiseaseONT 129
Figure 4. 7 Extrait de l'ontologie HealthONT 130
Figure 4. 8 Extrait de l'ontologie de la facette contextuelle de la maladie de diabète (DIABETOnt) 130
Figure 4. 9 Définition de quelques individus et quelques propriétés 131
Figure 4. 10 Définitions des classes et sous-classes de la facette contextuelle 132
Figure 4. 11 (a) Définitions de quelques propriétés de types et propriétés d'objets 133
Figure 4. 12 (a) Extrait OWL de l'ontologie globale (définition de classes et sous-classes) 133
Figure 4. 13 Les différents niveaux de description d'un service web 134
Figure 4. 14 Implémentation de la partie fonctionnelle de l'action élémentaire GlucoseRateMeasure 135
Figure 4. 15 Génération des SW (Interface D'assistance Eclipse) 136
Figure 4. 16 Test de la présence du service à partir du moteur Axis 136
Figure 4. 17 Génération du code WSDL de l'action élémentaire 137
Figure 4. 18 Exemple de l'interface de saisie de facettes des propriétés dans Protégé 138
Figure 4. 19 (a) Extrait de l'ontologie d'un modèle « IPetriNetModel » - Datatype « hasModelName » 138
Figure 4. 20 Définition des types de paramètres qui servent de base pour la composition dynamique 139

Figure 4. 21 Définition des Input/Output du processus « GlucoseRateMeasureProcess »..... 139
Figure 4. 22 Définition des instances de l'action « GlucoseRateMeasure » 140
Figure 4. 23 Définition du type d'exécution du processus « ConsciousPatientInterventionActivityProcess »... 140
Figure 4. 24 Représentation graphique du processus « ConsciousPatientInterventionActivityProcess »..... 141
Figure 4. 25 Représentation graphique de l'implémentation générale des trois processus du test..... 141
Figure 4. 26 Création de l'instance de grounding :WsdlAtomicprocessGrounding 142
Figure 4. 27 Mise en correspondance entre les paramètres d'une description OWL-S et les éléments d'une description WSDL..... 142
Figure 4. 28 Etapes de construction d'un modèle et les modules intervenant..... 143
Figure 4. 29 Exemple de contenu constituant une requête SysRequest 143
Figure 4. 30 Spécification owl-s du processus ConsciousPatientInterventionActivityProcess..... 144
Figure 4. 31 Extrait du modèle généré 145
Figure 4. 32 Déduction des besoins utilisateur dans un environnement ubiquitaire 146
Figure 4. 33 Déduction des besoins utilisateur selon ses spécificités 146
Figure 4. 34 Déduction des besoins utilisateur selon changement du contexte..... 147

Liste des tableaux

Tableau 1. 1 Architecture d'un système sensible au contexte 24
Tableau 1. 2 Classification par couches des différentes plateformes et architectures d'adaptation 36
Tableau 1. 3 Caractéristiques de l'adaptation au sein des différentes plateformes 39
Tableau 2. 1 Cas particuliers et problématiques associées 47
Tableau 2. 2 Comparaison sur les modèles de description des données contextuelles 60
Tableau 2. 3 Classification des données contextuelles selon les types de la mise-à-jour 63
Tableau 3. 1 Informations requises selon l'étape du traitement (Patient Conscient) 84
Tableau 3. 2 Information requise selon l'étape du traitement (Patient Inonscient) 91
Tableau 3. 3 Structures de contrôle du langage OWL-S 97
Tableau 3. 4 Correspondance macroscopique entre RdPI et OWL-S 98
Tableau 3. 5 Représentation d'une action élémentaire en tant qu'un service web sémantique 101

INTRODUCTION GENERALE

1. Avant propos

Au cours des dernières décennies nous avons assisté à un développement prodigieux des technologies de l'information et de la communication : apparition des ordinateurs de poche, ordinateurs portables sous forme de stylos, de montres ou autres accessoires de l'habillement des assistants numériques (PDA) de plus en plus sophistiqués, la téléphonie mobile extrêmement perfectionnée, etc. En conséquence, un nouveau défi prend naissance : rendre intelligents ces dispositifs ; et un nouvel axe de recherche apparaît, l'Ubiquité Numérique. Dans la littérature, ce paradigme est connu sous plusieurs noms[1] : Informatique ubiquitaire, Informatique omniprésente, Informatique diffuse, Informatique envahissante, Informatique nomade, Informatique ambiante et récemment le « partout », etc. Il s'agit d'un nouveau domaine de recherche aussi bien théorique que pratique et il est en plein effervescence, fondé en 1993 par Marc Weiser (Weiser, 1993) et suggéré comme une solution à nombreux besoins. L'Ubiquité est un mot dérivé du latin "Ubique" qui signifie partout. C'est la faculté d'être présent partout à la fois (Maxidico, 1997). En général, c'est la capacité d'être présent en plusieurs lieux simultanément. Ce nouveau paradigme consiste à intégrer, partout et d'une façon transparente, des environnements numériques au monde physique. L'informatique ubiquitaire est également définie comme *"la technologie invisible à des utilisateurs avec lesquels elle entretient des interactions permanentes"*. Weiser a appelé ce principe le "Calm Computing" (Weiser, 1993). Ainsi, l'interaction avec l'environnement devient plus « naturelle» et plus intuitive. Cette nouvelle technologie vise non seulement à ce que l'interaction avec l'utilisateur soit implicite mais également elle exige que l'assistance au cours de cette interaction doive avoir lieu sans que l'utilisateur n'ait conscience.

2. L'informatique ubiquitaire

L'informatique ubiquitaire est un axe de recherche et d'application en pleine effervescence. Il a pris naissance avec l'arrivée de la miniaturisation des dispositifs électroniques ainsi que l'essor de la mobilité et les réseaux sans fil (Greenfield, 2006). Les concepts basiques de ce nouveau paradigme ont été fondés en 1993 par Mark Weiser dans son fameux papier "*Some Computer Science Issues in Ubiquitous Computing*" (Weiser, 1993). Techniquement parlant, il s'agit d'incorporer et disséminer des différents dispositifs électroniques partout dans notre environnement physique et notre vie quotidienne en les rendant ainsi imperceptibles à l'utilisateur (Tran, 2009) (Kranz, 2010) (Syväniemi, 2011). Le but ultime derrière ce nouveau paradigme consiste à faciliter les activités de l'utilisateur – notamment celles pénibles et fastidieuses – ainsi qu'à remplacer son intervention dans toute tâche pouvant lui faire perdre son temps sans qu'il discerne cette assistance.

[1] ***Terminologie en Anglais :*** *Ubiquitous computing, Ubicomputing, Pervasive computing, Ubicomp, Ambient computing, EveryWare, Things that think, Haptic computing.*
Terminologie en Français : *Informatique diffuse, Informatique envahissante, Informatique nomade, Informatique ambiante, Informatique omniprésente, Informatique pervasive , Ubiquité numérique, Le « partout »*

Un environnement ubiquitaire intègre un nombre important de dispositifs (généralement des capteurs ou actionneurs) incrustés dans divers objets physiques formant ainsi un réseau d'information ubiquitaire. Par conséquent, cet environnement regroupe une très grande quantité d'informations et un contenu hétérogène, provenant de différentes ressources comme les capteurs, les utilisateurs, et même les résultats déduits de l'interprétation de ces informations. Nous citons à titre d'exemple: l'identité de l'utilisateur ainsi que les identités des personnes qui l'entourent, ses intérêts, son niveau intellectuel, ses préférences, sa profession, sa situation sociale, les dispositifs électroniques utilisés, la saison, la date, l'heure, le niveau d'éclairage, la température, etc.

Considérons par exemple le cas d'un environnement ubiquitaire domestique, l'éclairage dans un tel environnement est contrôlé par des capteurs biométriques (individuels, cousus dans l'habillement) permettant ainsi de moduler l'éclairage et le chauffage dans une chambre (Syväniemi, 2011). L'exemple le plus significatif nous est fourni par les réfrigérateurs « avertis » de leurs contenus (Rouillard, 2012). Ces dispositifs avertissent les consommateurs sur les produits ou nourritures périmés ou avariés. Egalement, la porte du réfrigérateur refuse de s'ouvrir si l'utilisateur a abordé la quantité de calories pour la journée. Ou encore, la voiture peut s'arrêter toute seule pour laisser passer des véhicules d'urgence. Un téléphone mobile peut savoir que son propriétaire se trouve actuellement dans la salle de réunion, et qu'il s'est assis. Ce téléphone peut conclure que l'utilisateur se trouve actuellement dans une réunion et qu'il s'agit de rejeter tous les appels sans importance.

Le défi posé par cette forme d'applications est immense. Comme avec n'importe quelle application, le premier défi est d'identifier ce qu'elle doit accomplir comme fonctionnalités et ce qu'elle doit fournir à l'utilisateur afin de l'aider à bien achever ses activités quotidiennes. En effet, un utilisateur mobile pourrait voir sur son interface diverses informations provenant des différents dispositifs disséminés partout dans l'environnement dans lequel il évolue. Toutefois, à un moment donné et dans un contexte bien déterminé, l'utilisateur aurait grandement besoin de certaines informations mais trouvera d'autres complètement inutiles ; ce choix dépend impérativement de ses préférences et de son contexte d'utilisation.

3. Contexte et sensibilité au contexte

Dans le cadre de l'informatique ubiquitaire, la notion du contexte représente toute information pouvant caractériser la situation d'une entité. Une entité est une personne, un lieu ou un objet considéré comme pertinent pour l'interaction entre un utilisateur et une application (Zhanga, 2010) (Rasch, 2011). L'activité courante de l'utilisateur peut constituer également le contexte et il est appelé à ce niveau le contexte d'utilisation. Une définition plus détaillée sera exposée dans le premier chapitre.

Quant à la sensibilité au contexte (en anglais *Context-Awareness*), elle est relativement considérée comme une notion très récente fortement liée au domaine de l'informatique ubiquitaire. Elle caractérise l'adaptation du fonctionnement d'un système

à son milieu d'usage. Autrement dit, un système est dit sensible au contexte s'il se base sur les données contextuelles pour fournir des informations et des services pertinents pour l'utilisateur, où la pertinence dépend généralement de la tâche demandée par l'utilisateur (Syväniemi, 2011). En effet, le système commence par « *percevoir* » les informations contextuelles, traite et évalue ces informations en second lieu. Enfin, il décide du meilleur service à fournir à l'utilisateur. Ce principe est connu sous le nom de l'adaptation au contexte.

4. Positionnement de la problématique

L'adaptation de l'information ou du service à fournir à l'utilisateur au contexte de l'utilisation au sein d'un environnement ubiquitaire est un axe de recherche en plein essor, il trouve sa justification dans la diversité des utilisateurs et celle des situations d'utilisation. La complexité de ce domaine est en croissance perpétuelle vue son interférence avec deux principaux autres domaines de recherche. En effet, le progrès potentiel des réseaux de la quatrième génération et les technologies associées telles que les réseaux sans fils (LAN WiFi, UMTS, Bluetooth, GPRS, RFID, etc) ainsi que les progressions prodigieuses des dispositifs électroniques tels que les PC portables, les PDA, les tables interactives, les iPhone et les iPad, les ordinateurs mettables, etc, cette interaction a approfondi le challenge dans ce domaine. Ainsi, l'adaptation dans ce contexte n'est autre que rendre intelligents ces dispositifs. Le défi est de rendre ces interfaces beaucoup plus « *attentives* » et « *conscientes* » des besoins de leurs utilisateurs.

Dans ce travail, nous ne voulons aucunement proposer une n^{ième} architecture pour l'adaptation au contexte. Notre objectif primordial est de fournir une approche « innovante » dans la manière d'aborder le problème mais également dans les propriétés d'apports et de contributions de notre travail par rapport à ceux existants. Suite à une étude bibliographique, nous avons constaté que la majorité des travaux consacrés à l'adaptation au contexte, notamment les premiers travaux, considèrent une adaptation généralisée, dans le sens où elle n'est pas spécifique à certaines caractéristiques de l'utilisateur à savoir son profil, ses préférences, ses critères individuels et intellectuels, etc. Ensuite, nous avons pris note de la relative rareté et du manque d'engouement autour des études consacrées à l'adaptation en temps-réel. En effet, dans un environnement ubiquitaire où l'information est en perpétuelle effervescence, le facteur « temps-réel » s'avère la principale contrainte à prendre en compte dans ces systèmes et qui devrait nécessairement caractériser l'adaptation afin d'obtenir l'information demandée d'une manière instantanée.

Finalement, cette étude nous a permis de révéler la nature de ces systèmes où l'adaptation se base sur un modèle fixe de fonctionnement. En conséquence, l'enjeu de notre approche s'appuie principalement sur ce majeur point faible dégagé des études existantes. Concevoir une architecture permettant une adaptation dynamique et temps-réel de l'information requise par un utilisateur mobile est la principale contribution de cette thèse.

5. Contributions

Dans un environnement ubiquitaire caractérisé par une nature fortement dynamique, l'utilisateur nomade se trouve en face d'un volume considérable d'information et un contenu fort hétérogène. L'objectif dans le courant de recherche relatif aux systèmes informatiques ubiquitaires est de rendre intelligente l'interaction entre cet utilisateur et ce type de systèmes. Autrement exprimé, les appareils mobiles utilisés dans ces environnements ubiquitaires doivent devenir implicitement « sensibles » aux besoins de leurs utilisateurs. Dès lors, les modèles de fonctionnement de cette interaction doivent être construits avec certaines capacités de sensibilité au contexte général de l'utilisateur. A ce niveau, proposer un modèle fixe de fonctionnement pour une adaptation dynamique et temps réel s'avère inopportun et une solution inappropriée. Il serait donc avantageux d'améliorer le processus d'adaptation en procurant des modèles évolutifs et flexibles et pouvant changer de structures à n'importe quel moment d'exécution afin d'offrir à l'utilisateur l'information idoine au moment approprié.

Notre première contribution est ainsi la conception et la réalisation d'un système ubiquitaire permettant une adaptation dynamique et temps-réel du contenu informationnel de l'interface utilisateur en procurant à l'utilisateur l'information qui répond au mieux à ces exigences. Ce fonctionnement doit être régi par une stratégie d'adaptation qui se base sur des modèles évolutifs et dynamiques. Nous avons choisi une modélisation basée sur les *Réseaux de Petri*. Dans cette approche une particulière focalisation a été accordée à l'étude de la faisabilité de la « *mutation* » d'un modèle basé sur les Réseaux de Petri en fonction du temps et du contexte d'utilisation. Selon l'adaptation conçue et établie, il ne saurait faire de doute que cette approche est porteuse de considérables avantages pour ce domaine, notamment la capacité des réseaux de Petri à exprimer de façon fiable les aspects de la concurrence, le parallélisme et le dynamisme qui sont des caractéristiques fondamentales d'un environnement ubiquitaire. Ces avantages se révèlent principalement dans la déduction en temps réel des besoins utilisateurs, par conséquent la déduction des fonctionnalités de l'interface. Ceci a fait l'objet de notre deuxième contribution qui décline de la façon suivante : nous avons tout d'abord choisi de modéliser les informations contextuelles sur la base des ontologies. La question qui s'est posée à ce niveau est : « *comment peut-on représenter l'aspect dynamique des réseaux de Petri lors de l'exécution d'un modèle*». L'application de la technique des ontologies de services s'est démontrée comme une solution plausible pour cette étude. Ceci facilitera, entre autres, le déploiement de l'architecture proposée à une large échelle, où l'*extensibilité* et l'*évolutivité* sont des critères à prendre en compte dans les environnements ubiquitaires.

6. Cas d'étude

L'informatique ubiquitaire commence à avoir une place de plus en plus importante au sein du monde médical. En conséquence, les systèmes ubiquitaires offrent la possibilité non seulement d'améliorer considérablement la qualité de vie des patients mais également de simplifier les nombreuses tâches du personnel médical. Par ailleurs, la

dernière décennie a connu un développement considérable dans la surveillance et l'intervention médicales temps-réel, y compris la surveillance de la glycémie et l'administration de l'insuline chez les patients atteints de diabètes aigus, la surveillance des personnes cardiaques, le contrôle à distance des défibrillateurs automatiques, la surveillance des patients épileptiques via le contrôle des stimulateurs du cerveau multi-programmables, etc. Notre étude de cas s'inscrit dans ce cadre et a pour objet fondamental la surveillance temps-réel de l'évolution de l'état des patients atteints de diabètes dans un service d'endocrinologie et diabétologie. Ce suivi est rendu possible grâce à des capteurs biologiques implantés sous la peau, et contrôlent périodiquement les niveaux de glucose chez les patients. Le système ubiquitaire doit vérifier, en permanence, l'évolution des états de chaque patient, l'enregistre, la valide et guide une éventuelle intervention médicale.

Beaucoup sont les problématiques qui peuvent découler d'une telle étude de cas. Une des problématiques est la suivante : comment un infirmier peut-il connaître les patients sollicitant une intervention immédiate et urgente. Et comment faut-il procéder pour apporter au mieux les premiers soins pour un cas compliqué. En effet, il s'agit de l'un des objectifs du système ubiquitaire qui devrait répondre aux besoins (implicites) des utilisateurs (Infirmiers, médecins, personnel paramédical, personnel administratif, etc) en leur procurant une information temps-réel et adéquate à leurs préférences, à leurs profils, à leurs activité en cours. Le système ubiquitaire devrait « guider » l'utilisateur à accomplir sa tâche en lui procurant l'information appropriée à chaque étape de traitement et pour chaque patient. Ensuite, supposant que l'état du patient en question avait changé (dans notre cas, il s'agit d'un passage de l'état conscient à l'état inconscient), comment le système doit se comporter pour mener à bien son assistance ?

7. Organisation du mémoire

Pour répondre à notre problématique, nos travaux, nos réflexions ainsi que nos résultats sont proposés et présentés via quatre chapitres.

Chapitre 1 : *L'ADAPTATION AU CONTEXTE DANS LES ENVIRONNEMENTS UBIQUITAIRES*

Dans ce chapitre, nous traitons la question de la conception des systèmes d'adaptation dans les environnements ubiquitaires. L'objectif de ce chapitre est de d'élucider les principales fonctionnalités et critères qui devraient caractériser un système d'adaptation. En outre, cette partie présente une analyse de la problématique de l'adaptation du contexte à un utilisateur mobile situant dans un environnement ubiquitaire et dresse un état de l'art des travaux qui s'apparentent à cette problématique.

En proposant une nouvelle classification des travaux selon une nouvelle vision, nous avons démontré la relative rareté des recherches qui tiennent en considération ces trois principaux critères - *Temps-réel*, *Besoins utilisateur* et *Activité en cours*- lors du processus l'adaptation.

Chapitre 2 : *CADRE CONCEPTUEL POUR UNE ADAPTATION DYNAMIQUE & TEMPS-REEL AU CONTEXTE D'UTILISATION*

Ce chapitre fait l'étude des principales fonctionnalités du système proposé pour l'adaptation. Ce système présente les solutions que nous avons développées pour répondre aux contraintes précédemment identifiées. Une étude critique est également dressée dans ce chapitre concernant les méthodes de modélisation du contexte afin d'aboutir à conclure le modèle approprié pour représenter les données contextuelles. Nous avons identifié les principaux critères sur lesquelles une comparaison sera convenablement établie entre les méthodes.

Chapitre 3 : *APPROCHE D'ADAPTATION DU CONTENU INFORMATIONNEL D'UNE INTERFACE UTILISATEUR*

Le troisième chapitre décrit notre principale contribution, la méthode proposée s'appuie sur l'aspect dynamique des environnements ubiquitaires. Comme nous l'avons déjà spécifié en mettant en avant la caractéristique de dynamicité et en octroyant le facteur temps-réel au fonctionnement du système d'adaptation proposé, notre stratégie ne peut, par son objet même, se conduire d'une façon fixe et prévue. Au contraire, les modèles qui régissent le fonctionnement du système sont susceptibles d'évoluer constamment au fur et à mesure du changement du contexte et l'évolution de l'environnement. En retour, le fonctionnement du système envers l'interaction avec l'utilisateur se verra régulièrement réorienté en fonction des changements du cadre général de son activité en cours. En outre, ce chapitre essaie de justifier l'utilisation des ontologies de services afin de mettre en œuvre cet aspect dynamique de la modélisation de l'interaction, basé à son tour sur les réseaux de Petri. Enfin, ce chapitre présente les algorithmes fonctionnels régissant le fonctionnement du système proposé.

Chapitre 4 : *EXPERIMENTATION & EVALUATION DE L'APPROCHE PROPOSEE*

Il nous reviendra dans ce chapitre d'aborder les développements informatiques que nous avons réalisés pour mettre en œuvre les travaux conceptuels exposés dans les deux chapitres qui précèdent. Une phase de simulation s'est également avérée nécessaire pour illustrer les résultats.

Chapitre 1

ADAPTATION AU CONTEXTE DANS LES ENVIRONNEMENTS UBIQUITAIRES

1. Introduction

L'émergence des études relatives aux environnements ubiquitaires a amplement encouragé la capacité de concevoir et de construire des systèmes qui prennent en charge la sensibilité au contexte. Plusieurs plateformes et applications ont été proposées dans la littérature. En revanche, l'adaptation en temps réel de l'information devant satisfaire les besoins de l'utilisateur, ainsi que la considération des préférences individuelles et des comportements particuliers de chaque utilisateur ne sont pas encore suffisamment abordées dans ces systèmes. Ce chapitre fait un tour d'horizon sur les travaux existants des systèmes sensibles au contexte sans souci d'exhaustivité, mais plutôt de représentativité. Nous analysons les approches abordées par chaque plateforme pour supporter l'adaptation au contexte d'usage. En premier lieu, nous présentons une synthèse des définitions trouvées dans la littérature sur la notion du « *Contexte* » ainsi que la notion de « *Sensibilité au contexte* ». En deuxième lieu, nous établissons une comparaison entre les principales architectures et systèmes proposés tout en identifiant et justifiant les critères de cette comparaison. Finalement, nous concluons ce chapitre par une synthèse sur les besoins nécessaires pour le bon fonctionnement d'un système ubiquitaire.

2. Notion du contexte dans le cadre de l'informatique ubiquitaire

2.1 Définition du Contexte

La notion du « CONTEXTE », dans le domaine de l'ubiquité numérique et dans un cadre d'une application ubiquitaire, est apparue dès les années 1990 avec Boy (Boy, 1991) ou encore Schilit dans les années 1994 (Schilit, 1994). Dès lors, les recherches sur cette notion prennent de plus en plus d'ampleur. Le but ultime est de trouver une meilleure définition du mot « contexte » ainsi que ses propres concepts. Ensuite, Dey (Dey, 2000) a proposé une définition beaucoup plus amendée et qui a été reprise dans de nombreuses recherches ultérieures. En revanche, et jusqu'aujourd'hui, nous ne trouvons pas une définition claire et concise du « contexte ». Après avoir exploré les principales définitions proposées dans la littérature, nous avons constaté, d'une part, une évolution dans les définitions du mot « contexte » allant du plus spécifique au plus généralisée. D'autre part, nous avons remarqué que la majorité de ces définitions tournent autour des définitions académiques proposées dans les dictionnaires et les encyclopédies. Par exemple, la définition proposée dans le « Larousse » décrit le contexte par un « ensemble des circonstances dans lesquelles se produit un évènement, se situe une action ». L'Encarta, propose cette définition : « le contexte est un environnement structurel et évènementiel qui sert de cadre (à quelque chose) ». Parmi les premières définitions, nous citons celle proposée par Schilit (Schilit, 1994). Il définie le « contexte » par des facteurs très particuliers : la localisation de l'utilisateur, les identités et les états des personnes et des objets qui l'entourent. Ensuite, nous citons la définition de Brown (Brown, 1997): « *location, time of day, season of the year, temperature, and so forth*». De même, Brown

définit le « contexte » par des termes très spécifiques à des situations bien déterminées : la localisation, le temps, la saison, la température, etc. Quant à Ryan (Ryan, 1997), il restreint la définition du contexte à la localisation, le temps, la température et l'identité de l'utilisateur : « *Information about its environment, such as location, time, temperature or user identity* ». A ce niveau, nous constatons que les définitions proposées diffèrent en expression mais sont similaires en contenu. En effet, quatre principaux termes ont été mis en évidence dans chacune de ces définitions et c'est le cas pour d'autres auteurs (Pascoe, 1998) (Schmidt, 1999) où les définitions sont très reliées à des applications bien déterminées et appartiennent à un cadre d'étude très restreint. Ces termes sont : (a) l'identité de l'utilisateur, (b) sa localisation, (c) le temps et (d) l'état de l'environnement. Cette constatation était reprise par de nombreuses recherches dans ce domaine, et dès lors, les définitions du « contexte » prennent d'autres formes et deviennent de plus en plus généralisées. La définition la plus significative nous est fournie par Dey (Dey, 2000): « *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves* ». Cette définition est considérée par la plupart des auteurs comme une référence et une transition d'une définition spécifique à une définition généralisée. Winograd (Winograd, 2001) a proposé également une définition simple et significative : « Le contexte est un ensemble structuré d'informations, il est partagé, il évolue et il sert pour l'interprétation», nous avons de ce fait adopté cette définition.

2.2 Sensibilité au contexte

Un système est caractérisé par la sensibilité au contexte lorsqu'il se base sur les différentes informations contextuelles détectées ou déduites de l'environnement, traite et évalue ces informations et éventuellement décide du meilleur service à fournir à l'utilisateur. Autrement dit, la sensibilité au contexte caractérise l'adaptation du fonctionnement d'un système à son milieu d'usage. La définition la plus adaptée dans la littérature est celle de (Dey, 2000) : « Un système est dit sensible au contexte s'il utilise le contexte pour fournir des informations et des services pertinents pour l'utilisateur, où la pertinence dépend de la tâche demandée par l'utilisateur ». A partir de cette définition, trois principales catégories d'applications sensibles au contexte peuvent être envisagées :

a. Applications liées à la présentation de l'information et service à l'utilisateur. Dans cette catégorie, le système présente des informations contextuelles ou propose des choix d'actions correspondant aux changements du contexte.
b. Applications liées à l'exécution automatique de services. Ce type d'applications vise à reconfigurer le système, selon le changement du contexte, d'une manière autonome et sans l'intervention de l'utilisateur.
c. Applications liées à la modélisation et au stockage de l'information pour des éventuelles utilisations.

Cette caractéristique – la sensibilité au contexte – était différemment définie par (Thevenin, 2001) (Calvary, 2003). Ces auteurs parlent particulièrement de la plasticité

d'un système. Un système est dit plastique lorsqu'il peut s'adapter dynamiquement aux différents changements de l'environnement tout en préservant son utilisabilité. Dans de tels systèmes, les activités de l'utilisateur contrôlent le système d'information, ce dernier perçoit son environnement et adapte ses traitements aux conditions existantes. Ces auteurs ont également évoqué le terme « domaine de plasticité ». Ils l'ont défini comme étant l'ensemble de variations du contexte que le système est capable de gérer par lui-même. Ces variations peuvent être :

a. Un changement de support de visualisation (assistant personnel, téléphones portables, écrans plasma, écrans classiques…)
b. Un changement de type d'utilisateur (novices, experts, personne ayant un handicap)
c. Un changement d'environnement (à l'extérieur, à l'intérieur, en mouvement… Exemple : aéroport, gares, centres commerciaux).

2.3 Classification des paramètres contextuels

Dans un but de mieux analyser et bien interpréter ces informations et déduire par la suite le contexte convenable et spécifique, la classification de ces informations s'avère nécessaire. Les différentes classes obtenues sont appelées : « paramètres contextuels ». Dey (Dey, 2001) identifie trois principales entités qui caractérisent un environnement ubiquitaire[2] : « places, personnes et objets ». Il désigne par « places » la localisation géographique, par « personnes » des individus ou des groupes de personnes et par « objets » : des entités physiques ou des composants logiciels.

Afin de décrire ces entités, différentes catégorisations ont été proposées dans la littérature. En revanche, nous trouvons essentiellement quatre principales catégories :

a. Identité : il s'agit d'un identifiant explicite, qui caractérise d'une façon unique, l'entité en question.
b. Localisation : représente les données géographiques de l'entité (sa position, son orientation, coordonnées des entités voisines…).
c. Etats : représentent les différentes propriétés perçues sur/par l'utilisateur. Par exemple, les propriétés d'une place peuvent être la température ambiante, le niveau de l'éclairage, niveau du bruit, etc. Pour une personne, ces propriétés peuvent être l'activité courante de l'utilisateur, son état psychique…
d. Temps : présente la date et l'heure. Par exemple : heure du travail.

D'autres auteurs ont tenté de décrire le contexte et de classifier les paramètres contextuels autrement (Cheung, 2006) : En effet, trois catégories de contexte ont été distinguées. Le contexte peut être physique, il s'agit des informations externes mesurées à partir des capteurs matériels. Si ces informations sont spécifiées par l'utilisateur ou bien capturées par un système de contrôle, le contexte est dit logique ou interne. La dernière

[2] « *places*, *people* and *things* »

catégorie consiste à caractériser les informations contextuelles par : explicites (si elles sont fournies par l'utilisateur) ou implicites (si elles sont déduites).

2.4 Adaptation au contexte dans un environnement ubiquitaire

L'adaptation est un processus complexe qui dépend d'une multitude de variables incluant la situation, le contexte d'utilisation, l'activité en cours ainsi que les ressources qui sont à la disponibilité de l'utilisateur. En particulier, une adaptation diffère en fonction de l'utilisateur en question ainsi que sa situation particulière à un moment donné. L'adaptation des interfaces utilisateurs – notamment celles mobiles – au contexte de l'utilisation et dans des environnements ubiquitaires est un domaine de recherche en plein effervescence. Ainsi, le développement des modèles et des stratégies d'adaptation de ces interfaces est devenu à son tour de plus en plus complexe. Dans un but de rendre ces interfaces plus « attentives » et mieux « conscientes » des besoins de leurs utilisateurs, les applications et les modèles de fonctionnement devraient être basés sur le concept de la sensibilité au contexte. L'interface doit donc adapter les informations qu'elle fournit de façon implicite découlant des besoins instantanés de l'utilisateur ainsi que son contexte d'utilisation.

Deux notions différentes ont émergé de l'adaptation des interfaces utilisateurs (*IU*) à savoir les interfaces adaptables et les interfaces adaptatives (Alvarez, 2007). Généralement, les interfaces adaptatives contrôlent l'interaction de l'utilisateur avec le système et adapte les composants de l'interface en se basant sur des modèles d'utilisations bien définis. Une interface adaptative doit donc être capable de recueillir des données sur l'utilisateur ; se baser ensuite sur ces données afin d'identifier les besoins utilisateur en terme d'information. Tandis que les interfaces adaptables fournissent des outils qui permettent aux utilisateurs la modification directe de leurs interfaces jusqu'à ce qu'ils rencontrent leurs préférences. Nous pouvons en conclure le principal intérêt des interfaces adaptatives qui nécessitent peu d'effort de la part de l'utilisateur, contrairement aux systèmes adaptables qui exigent de l'utilisateur un contrôle total.

3. Plateformes et architectures d'adaptation au contexte

Les caractéristiques propres au développement d'un système sont ceux qui souvent dirigèrent la répartition des tâches entre le système et l'utilisateur. Pour les systèmes d'adaptation au contexte, la nature changeante du contexte opérationnel est la principale caractéristique qui doit régir cette répartition. En effet, il s'agit de l'objectif primordial d'un système d'adaptation au contexte.

Cette section présente une revue de la littérature des plateformes et des architectures des systèmes sensibles au contexte. La liste n'est pas exhaustive. Nous analysons les approches abordées par chaque plateforme pour tenir en compte l'adaptation. Nous présentons ensuite l'architecture générale d'un système sensible au contexte conclue à partir d'une synthèse des travaux existants.

3.1 Structure générale d'un système sensible au contexte

Il est à signaler que la première structure d'un système sensible au contexte a été fournie par Dey (Dey, 1999). Son architecture a constitué une référence pour tous les travaux qui portent sur cet aspect. Il a proposé une architecture à cinq couches où chacune d'elles est responsable d'une fonction bien déterminée, à savoir la couche application, la couche dissémination, la couche stockage, la couche interprétation et la couche capture (Tableau1.1) : Le but majeur visé par Dey était de séparer l'acquisition et la gestion du contexte de son utilisation dans les applications. Cette séparation rend facile le développement de plateformes génériques, la réalisation et déploiement d'applications sensibles au contexte.

Tableau 1. 1 Architecture d'un système sensible au contexte

La couche Application	Il s'agit de l'application qui offre ses services aux différents clients concernés.
La couche Dissémination du contexte	Définir les mécanismes de transmission des informations contextuelles vers la couche application.
La couche Stockage du contexte	Organiser les données capturées et interprétées et les stocker pour une utilisation ultérieure.
La couche Interprétation du contexte	Analyse et transformation des données provenant des capteurs en un niveau de compréhension plus élevé
La couche Capture du contexte	Composée d'une collection de capteurs qui fournissent des données brutes

Après une première vue sur les architectures proposées dans la littérature, nous avons remarqué qu'elles diffèrent principalement en « nom », en « fonction » et en « emplacement des couches », mais se basent forcément sur les cinq couches fournies par Dey. En effet, la première et la dernière couche existent dans la plupart des travaux. Par contre, nous avons constaté que les trois couches du milieu sont traitées de manières différentes au niveau de ces recherches, bien qu'il s'agisse du traitement du contexte dans ces trois couches. Par conséquent, nous avons regroupé ces trois couches en une seule (Figure 1.1). Ci-dessous, nous détaillons les fonctionnalités de chaque couche.

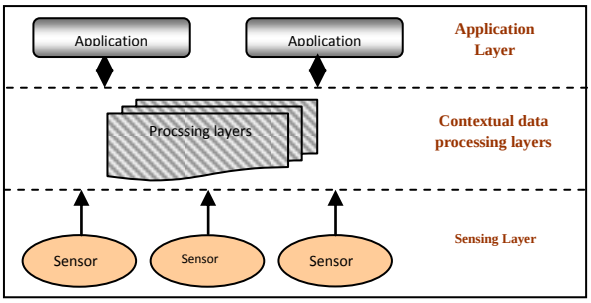


Figure 1.1 Architecture basique d'un système sensible au contexte

3.1.1. Couche des capteurs : "*Sensing layer*"

La couche des capteurs est considérée comme un module commun et basique pour n'importe quelle architecture sensible au contexte. Dans le processus de développement d'une application sensible au contexte, la première et l'indispensable étape est celle de l' « acquisition du contexte ». Elle se base, en un premier lieu, sur les informations

« interceptées » des capteurs. D'où cette couche représente un ensemble de capteurs considérés comme les principales sources de données contextuelles pour le reste du système.

Un capteur est une source matérielle ou logicielle qui peut contenir ou générer une information contextuelle. Il y a différentes catégorisations de capteurs, nous choisissons de présenter brièvement quelques types qui s'articulent autour de l'intérêt de l'information qu'ils procurent :

- Les capteurs **physiques** : sont considérés les plus simples en termes d'information fournie. Ces dispositifs transforment, tout simplement, une grandeur physique en un signal électrique exploitable, il s'agit en fait de la fonction de base d'un capteur.

- Les capteurs **numériques** : ajoutent à la fonction primaire d'un capteur des opérations numériques afin d'améliorer et amender cette fonction. Par exemple, la numérisation de la mesure en vue de son utilisation par une centrale d'acquisition via une liaison série. Les capteurs incrémentaux et les codeurs absolus sont des capteurs numériques typiques.

- Les capteurs **intelligents** : sont les plus dominants aujourd'hui. Ces capteurs implémentent des fonctionnalités de mémorisation et de traitement de données afin d'améliorer les performances métrologiques. En plus, il y a différentes évolutions qui sont apparues pour les capteurs intelligents avec une intégration plus ou moins grande de fonctionnalités. Ces fonctionnalités concernent principalement l'amélioration de la qualité de l'information fournie, comme par exemple des fonctions de compensation, de validation, d'auto-diagnostic, etc. Ces fonctionnalités avancées peuvent aller jusqu'à sélectionner les données à transmettre et à prendre des décisions.

3.1.2. Couche de traitement du contexte : « Contextual data processing layer »

Cette couche est responsable de la préparation des données contextuelles qui vont être « consommées » par l'application. Il s'agit d'une étape fondamentale afin de bien accomplir le but ultime des systèmes sensibles au contexte, à savoir la pertinence de l'information fournie à l'utilisateur. Cette couche est constituée de trois sous-parties : l'interprétation des données contextuelles, la modélisation de ces données et enfin leur diffusion.

a. Module d'interprétation du contexte : " Context Interpretation"

Cette étape doit assurer, aussi bien l'analyse que la transformation des données brutes provenant des capteurs quelque soit leur type. Une méthode d'interprétation des données doit donc être définie. En effet, ces capteurs produisent, généralement, des données qui ne sont pas exploitables par des applications informatiques. La méthode

d'interprétation fournie par cette couche doit transformer les données brutes en données de « plus haut niveau » afin de rendre ces données plus compréhensibles par les couches supérieures (Dey, 1998), (Kiciman, 2000). Par exemple : un certain chiffre fourni par un GPS peut être transformé en une adresse formée par le nom de la rue, la cité ainsi que le nom du pays (Chaari, 2007). Outre les données brutes inexploitables fournies par les capteurs, ces derniers peuvent également donner des résultats contradictoires ou pouvant aboutir à des situations imprécises, d'où la couche d'interprétation et d'analyse doit veiller à la résolution de conflits causés par l'utilisation de plusieurs sources de contexte. Elle doit donc avoir une certaine intelligence d'interprétation pour résoudre ces conflits (Dey, 1999).

b. Module de représentation du contexte : « Context Modeling »

Après avoir interprété les données détectées, ces dernières doivent être sauvegardées sous une forme idoine. En fait, nous devons définir un moyen approprié afin d'organiser et enregistrer ces données pour un ultérieur traitement. En d'autres termes, nous devons définir un modèle de contexte. Ce modèle permettra la réutilisation et l'exploitation du contexte décrit dans d'autres applications similaires. Les documents fournis par ce module seront exploités par le module d'adaptation. Dans le chapitre 2, nous détaillons les approches de modélisation les plus fréquemment utilisées pour représenter le contexte.

c. Module de diffusion des informations contextuelles : « Context diffusion »

Cette phase du processus représente une interface de communication entre la couche de stockage et les applications. Elle permet en fait la transmission des différentes informations contextuelles à l'application. Par conséquent, nous devons définir, dans cette étape, la manière dont l'information sera diffusée entre les différentes composantes du système. En d'autres termes, il faut décrire comment attribuer les paramètres contextuels pertinents pour chaque service. Ce module doit, non seulement, fournir des services et des fonctionnalités qui rendent facile le développement d'applications sensibles au contexte, mais également définir des moyens de communications afin de notifier les applications des nouveaux changements du contexte.

3.1.3. Couche d'applications : « Application layer »

La couche applicative est considérée comme un « Client » où le « serveur » est toute la plateforme décrite précédemment. Les mécanismes d'adaptation ainsi que les réactions face aux éventuels changements du contexte sont implantés dans cette couche (Baldauf, 2007a). Mais par rapport à un environnement ubiquitaire, si on considère que l'utilisateur est le « client » qui cherche l'information adaptée à ces besoins, dans ce cas cette couche joue le rôle d'un « serveur » qui offre ses services aux différents clients concernés. Chaque application doit interagir avec le module de diffusion du contexte afin de récupérer les différentes informations contextuelles et être avertie des éventuels changements (Chaari, 2007).

3.2 Description fonctionnelle des plateformes et architectures existantes

Dans cette section, nous nous préoccupons de l'état de l'art des plateformes ou architectures qui existent pour supporter la sensibilité au contexte. Cette étude sera suivie d'une partie d'analyse et de synthèse mettant exclusivement en valeur les éventuelles déficiences de ces plateformes.

3.2.1. Context Information Service

Context Information Service (CIS) est une architecture orientée-objet qui supporte des applications sensibles au contexte (Pascoe, 1999). C'est une architecture indépendante de la plateforme et qui offre un accès partagé aux ressources. L'objectif de CIS est de maintenir un modèle d'information contextuelle qui facilite l'obtention des données contextuelles. Elle est constituée de plusieurs composants. Le plus important est le composant World (Figure 1.2) : il est considéré comme le cœur de l'architecture CIS. Il comprend certains sous-composants :

- Les « Monitors » : Ces composants contrôlent le passage des données contextuelles dès leurs acquisitions à partir des capteurs jusqu'aux leurs propres destinations désignées par « Artefact ».
- Les « Artefact » : Ce sont des objets composés de nom, type et un ensemble d'états contextuels.
- Les « Sensors » : Ce sont des programmes qui extraient l'information contextuelle à partir de dispositifs.

Un « monitor » est associé à un état d'artefact unique, et il essaie de chercher le « Sensor » qui peut lui fournir l'information dont il a besoin. Une application – considérée comme un client CIS – peut accéder à n'importe quel état de n'importe quel artefact.

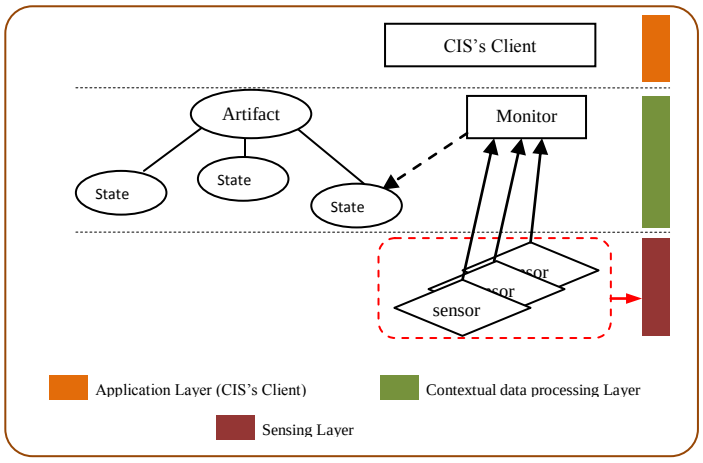


Figure 1.2 Context Information Service (Pascoe, 1999)

3.2.2. Context toolkit

L'architecture du « ContextToolkit » a été proposée par A. Dey (Dey, 1999). Le but ultime de cette architecture consiste à séparer l'acquisition du contexte de la manière dont il est présenté et utilisé par les autres composants de l'architecture. L'architecture ContextToolkit est orientée objet. EIle est composée d'un ensemble d'outils et d'entités qui permettent le développement d'applications sensibles au contexte. EIle se compose de trois modules (Figure 1.3) :

a. Widgets (ou Context-widget) : Les « widgets » sont des composants logiciels utilisés pour rendre abstrait le contexte détecté. Ils représentent une interface entre les données brutes et les applications qui utilisent le contexte. Ces composants enregistrent, en premier lieu et d'une manière automatique, le contexte acquis et en deuxième lieu, ils rendent ces données disponibles et compréhensibles pour les applications. En effet, les « widgets » encapsulent les informations du contexte et fournissent des méthodes pour y accéder.
b. Interpréteur (Interpreter) : Ce module est responsable de l'interprétation des données contextuelles fournies par les Widgets et les transforme en un niveau d'abstraction plus élevé susceptible d'être mieux compris et traité.
c. Aggregator : Ce module représente un intermédiaire entre les widgets et applications. Il est considéré comme une méta-widget qui réunit les données provenant de plusieurs widgets.

Le Discover est un composant qui encapsule des données sur la manière d'accéder aux différents widgets ainsi que les types de données fournies par ces composants. Par conséquent, si un agrégateur « veut contacter » un autre composant, il trouvera l'information adéquate en interrogeant le Discover.

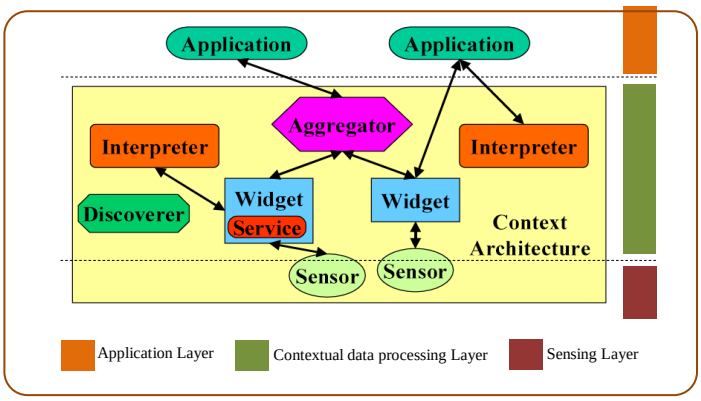


Figure 1.3 L'architecture "Context toolkit" (Dey, 1999)

3.2.3. HyCon

HyCon est une architecture qui fournit une plateforme favorable particulièrement pour les simulations des systèmes hypermédias dans un environnement ubiquitaire mobile (Niels, 2003). Elle est divisée en quatre couches : la couche de stockage, la couche serveur, la couche terminale et la couche des capteurs (Figure 1.4). La première couche : « couche de stockage » est responsable de l'enregistrement des données

contextuelles. Elle comprend une interface qui facilite la communication avec les composants de la couche supérieure, la « couche serveur ». Le but de la couche « serveur » est de fournir l'information adéquate et les fonctionnalités nécessaires pour la couche supérieure : « Terminal ». La distinction entre les composants réutilisables et ceux spécifiques aux applications est constatée au niveau de la couche « Terminal ». Les composants de cette couche comprennent les interfaces de communications avec la couche serveur, tandis que les applications implémentent les fonctionnalités de l'interface utilisateur spécifique au dispositif électronique utilisé (laptops, tabletPCs, PDAs, téléphone portable, etc).

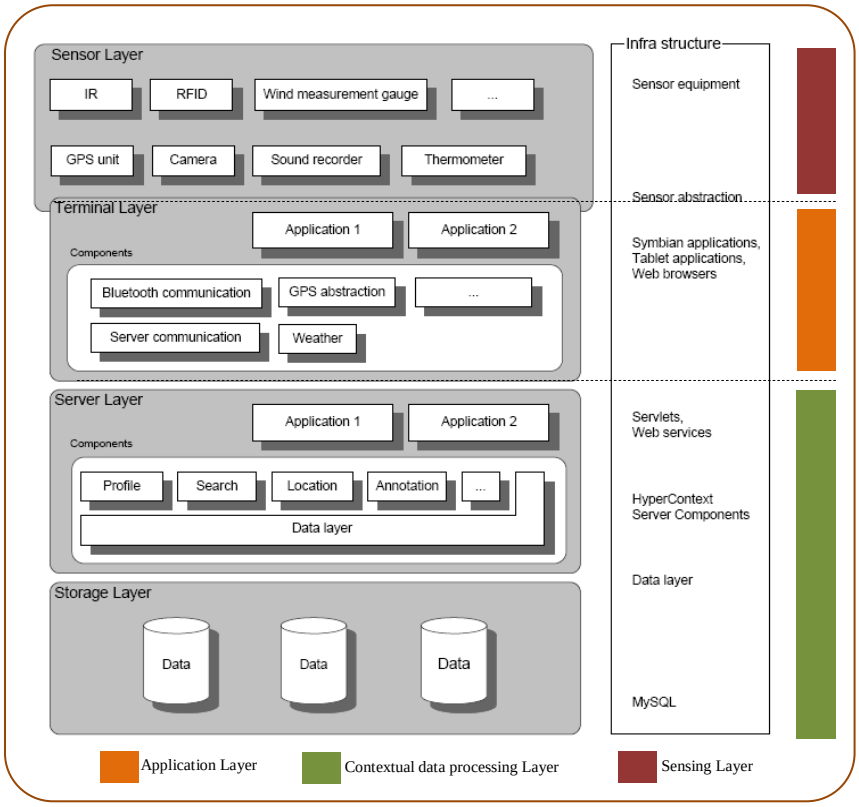


Figure 1.4 L'architecture HyCon (Niels, 2003)

3.2.4. CoBrA : Context Broker Architecture

C'est une architecture basée sur les systèmes multi-agents (Figure1.5), proposée par Chen (Chen, 2004). Son fonctionnement repose essentiellement sur un agent intelligent appelé « Context Broker » qui détient et gère un modèle du contexte. Cet agent est composé de quatre éléments principaux : la base de connaissances du contexte, un moteur de raisonnement « Context Reasoning Engine », un module d'acquisition du contexte et un module de gestion de données privées.

a. Le module d'acquisition du contexte : « Context acquisition module », Le « broker » rassemble les informations provenant des différentes sources, et contrôle les changements d'informations dans ces différentes sources.

b. « ContextKnowledge management » : Le « Broker » contrôle les connaissances contextuelles qu'il maintient afin d'assurer à chaque instant la cohérence de la base de connaissance.
c. « Knowledge sharing » : Le « Broker » permet aux agents de partager les connaissances contextuelles, et il permet de répondre à leurs interrogations.
d. « User privacy protection » : Avant de partager les connaissances contextuelles, le « Broker » négocie avec les utilisateurs des politiques de confidentialité.

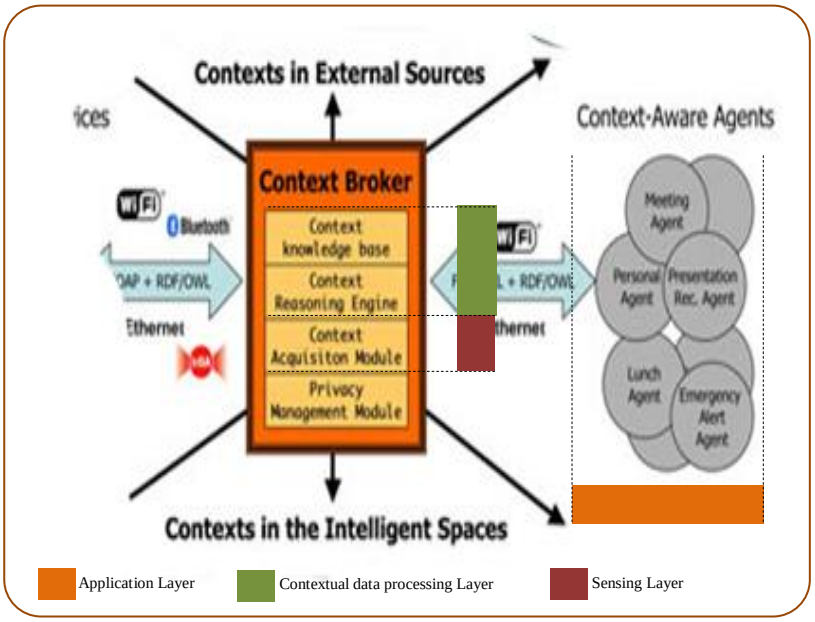


Figure 1.5 L'architecture CoBrA (Chen, 2004)

3.2.5. Java ContextAware Framework

Le Java ContextAware Framework (JCAF) (Bardram, 2005) est une plateforme basée sur la technologie Java, ainsi que le modèle « MVC » (Model View Controller). JCAF est une architecture orientée services. Ses principales composantes sont décrites en Figure1.6.

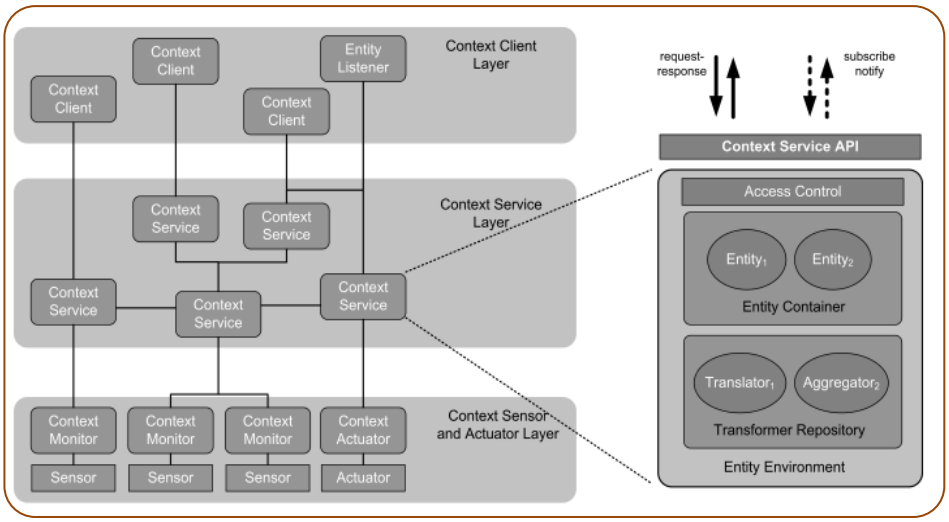


Figure 1.6 Java ContextAware Framework (Bardram, 2005)

- « ContextService » : Constitué par un ensemble d'Entités. Une entité est contrôlée par son conteneur (Container) dans lequel elle a été ajoutée. Ce Container est responsable d'informer les différents composants clients des changements. Le « Service's entity container » est un composant qui gère l'entité en question et les informations contextuelles qu'elle détient.
- « ContextClient » : Ces composants sont considérés comme des applications sensibles au contexte qui utilisent la plateforme JCAF en accédant à un ou plusieurs services. Ces applications peuvent accéder aux entités et leurs informations contextuelles. Elles peuvent également ajouter ou supprimer ces informations.
- « Context monitors/Actuators » : Ce sont des types particuliers de clients du contexte « Context Clients ». Les « Monitors » sont des interfaces d'acquisition d'informations contextuelles dans un environnement ubiquitaire. Elles coopèrent avec certains types d'équipements de capteurs. Les « Actuators » détectent les changements du contexte.

3.2.6. Service-Oriented Context-Aware Middleware SOCAM

Cet intergiciel est désigné en particulier pour faciliter le prototypage des services sensibles au contexte dans des environnements ubiquitaires. L'architecture décrite au sein de cet intergiciel (Gu, 2005) semble la plus analogue à celle présentée par Dey. La majorité des noms des couches ainsi que leurs fonctions respectives sont quasi-similaires. L'architecture de SOCAM, présentée dans la figure 1.7, comprend les composants suivants (qui évoluent en tant que services indépendants les uns des autres) :

- *Context providers* : Ils capturent des informations contextuelles utiles de sources hétérogènes de l'environnement de l'utilisateur et de l'application. Ensuite, ils les convertissent en des représentations OWL pour que le contexte puisse être partagé et réutilisé par d'autres composants de l'architecture.
- *Context interpreter* : Il fournit des services de raisonnement logique sur les représentations OWL du contexte en appliquant des enchaînements de règles d'interprétation. Il fournit aussi un service d'interrogation intelligent qui permet de résoudre les conflits d'interprétation du contexte.
  L'ontologie "pervasive" décrivant l'environnement de l'application et les instances des ontologies spécifiques au domaine de l'application décrivant l'environnement de l'utilisateur.
- *The Context Database :* Cette base représente la couche de stockage. Elle stocke les différents éléments de l'ontologie décrivant l'environnement d'application.
- *Context-aware services* (services sensibles au contexte) : Ces services utilisent les différentes informations stockées dans la base de données du contexte (context-database) pour modifier leurs comportements selon le contexte courant.
- *Service-locating service* (service de localisation de services) : Ce service fournit un mécanisme avec lequel des utilisateurs ou des applications peuvent localiser les fournisseurs et les interpréteurs du contexte.

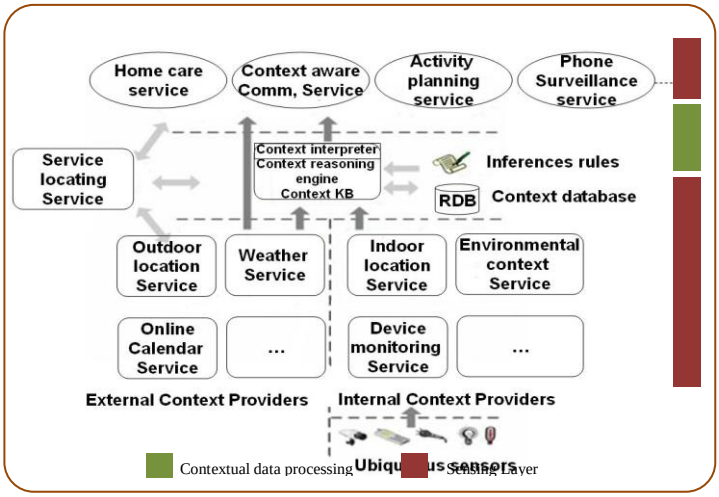


Figure 1.7 L'architecture SOCAM (Gu, 2005)

3.2.7. SECAS : Simple Environment for Context-aware Systems

Le fonctionnement de l'architecture SECAS, représentée par la figure 1.8 (Chaari, 2006), est basé sur quatre composants : le noyau de l'application, la couche d'adaptation, le système de gestion du contexte et la partie terminal. Dans cette architecture l'auteur a mis l'accent sur le composant de la gestion du contexte même s'il a prévu un composant tout entier pour l'adaptation. Ce composant, « G*estion du contexte* » se compose de quatre modules : le « C*ontext-provider* » qui est responsable du traitement physique des données brutes transmises par les capteurs, il représente la couche capteur du modèle générique d'un système d'adaptation. Le deuxième module, « *Context-interpreter* », traduit le contexte obtenu du premier module d'une représentation bas-niveau à une présentation haut-niveau (Par exemple les coordonnées géographiques d'un point se traduisent par la forme quotidienne d'une adresse, nom-rue, cité, ville, CP). La phase du stockage est assurée par le module « *Context-repository* ». Finalement, afin d'obtenir les données appropriées pour la couche adaptation, l'application de l'adaptation doit interroger le « *Context-broker* ». Ce module est responsable de la diffusion de la donnée pertinente à chaque service de l'application d'adaptation.

3.2.8. User-Adaptive and Context-aware Architecture for Mobile and Desktop Training Applications

La même stratégie de classification des couches semble très claire au sein de l'architecture décrite dans (Buttussi, 2008). La couche de détection est un ensemble de capteurs (par exemple un moniteur de fréquence cardiaque, un appareil GPS, un Accéléromètre 3D, etc.). La couche de traitement contextuel des données est représentée par les trois modules suivants : « *Context Analyzer* », « *UM & CM Service* » et « *User Model & Context Model databases* » (Figure 1.9). La couche d'adaptation est représentée par le module « *Adaptation Engine* ». Comme son nom l'indique, le premier module collecte et analyse les données brutes provenant des différents capteurs. Ce module est également responsable de la déduction des informations de haut-niveau relatives en particulier à l'état physiologique des utilisateurs et leurs mouvements. Le second module,

UM & CM Service[3], recueille les informations précédemment identifiées et les stocke par la suite dans la base de données des modèles (le troisième module). De plus, il fournit l'information adéquate à l'analyseur et au moteur de l'adaptation. L'aspect nouveau dans cette architecture est la présence d'une couche d'adaptation fonctionnelle qui se compose d'un moteur d'adaptation, d'une interface utilisateur et de sous-systèmes virtuels de l'utilisateur. Le moteur d'adaptation assure la phase d'adaptation proprement dite en appliquant des règles d'inférence à l'information reçue et déduire les décisions adéquates pour l'utilisateur (Par exemple, l'augmentation automatique de la difficulté d'un exercice particulier ou bien des suggestions pour éviter la fatigue). La partie « User Interface » met en œuvre l'information appropriée provenant du moteur d'adaptation. Enfin, « Virtual Human » est une partie chargée de fournir à l'utilisateur des conseils et des démonstrations (en 3D) virtuelles.

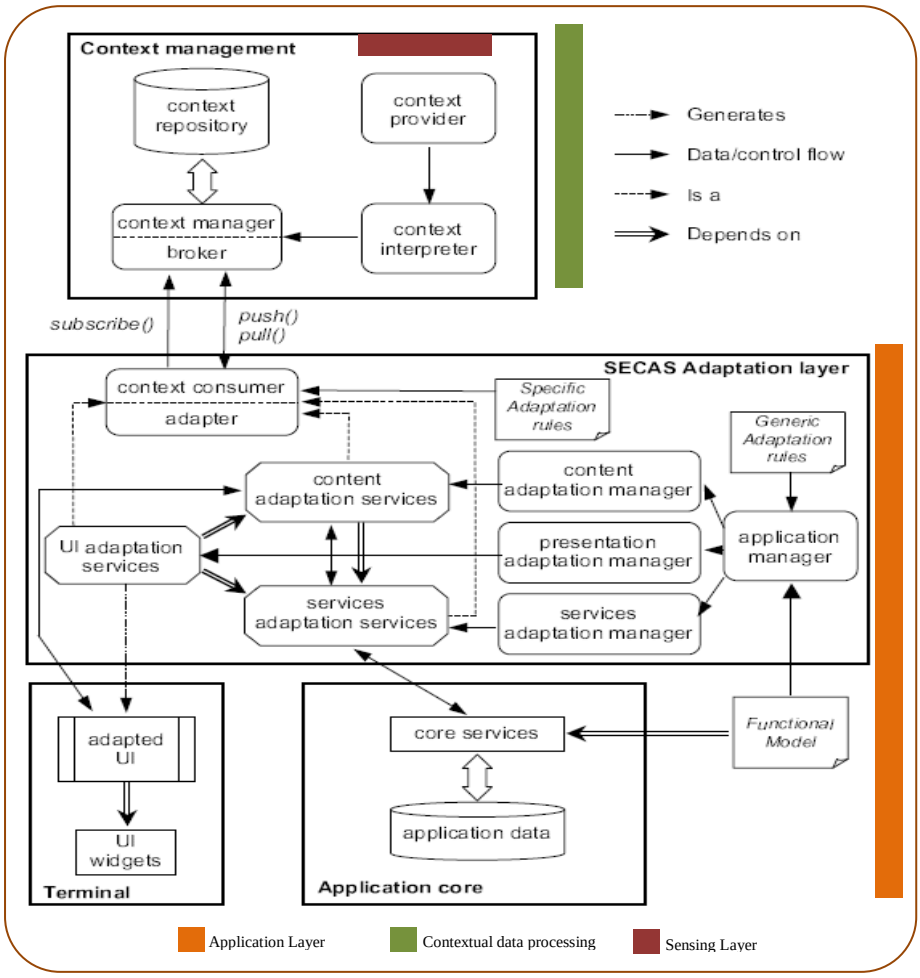


Figure 1.8 La plateforme SECAS (Chaari, 2006)

[3] UM : User Model, CM : Context Model

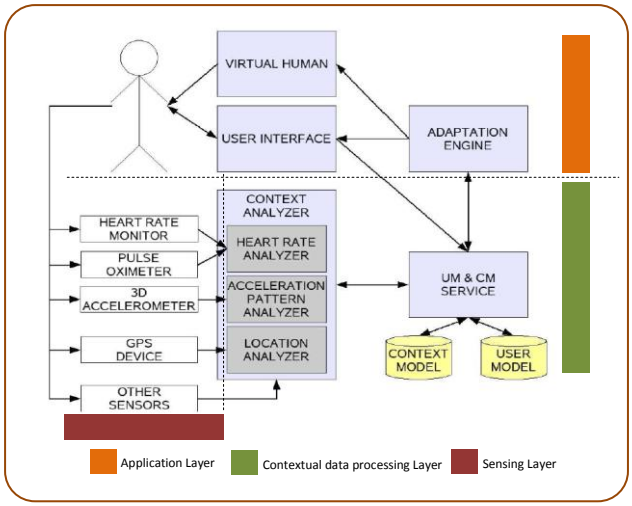


Figure 1. 91 L'architecture CAAMDTA (Buttussi, 2008)

3.2.9. Architecture-based Approach to Context-aware Adaptive Software Systems

Selon la description donnée par l'auteur (Hussein, 2011), l'architecture peut être représentée uniquement par les deux couches du modèle générique. La première est la couche d'adaptation, la seconde est la fusion des deux couches : la couche traitement des données contextuelles et celle de la capture du contexte (Figure 1.10). Dans la couche d'adaptation, la stratégie de l'adaptation est assurée par trois principales opérations. Les scripts d'adaptation représentent un ensemble d'actions d'adaptation prédéfini en vue de répondre à tout changement du contexte. L'opération du traitement « Processing » est utilisée pour la déduction des informations contextuelles de haut-niveau. L'opération d'analyse « Analyzing » est chargée de vérifier la cohérence entre l'état du système en cours d'exécution, l'état de l'environnement et les besoins du système afin de lancer l'opération de décision « *Deciding* » en cas d'absence de cohérence.

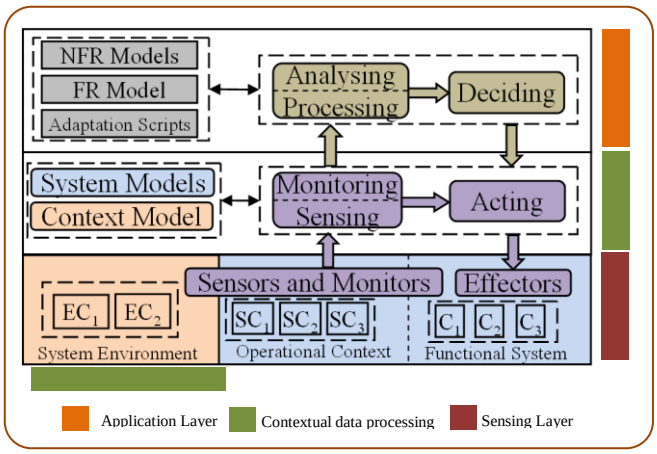


Figure 1.10 Architecture to Context-aware Adaptive Software Systems (Hussein, 2011)

La dernière opération spécifie l'action nécessaire pour la réponse à un changement du contexte. Le contexte est modélisé dans la deuxième couche à savoir le système et sa représentation contextuelle. Cette couche a deux principales fonctionnalités (opérations).

La première consiste en une opération de contrôle et de capture « *Monitoring/Sensing* ». Elle est responsable essentiellement de la mise à jour du modèle de contexte ainsi que les modèles du système. La deuxième fonctionnalité est l'opération « *Acting* ». Elle est responsable d'exécuter les actions spécifiées par la couche précédente. Enfin, la couche capture du contexte est représentée par un ensemble de capteurs et de moniteurs qui détectent tout changement dans le système. Cette couche informe également les couches supérieures de prendre les mesures requises pour l'adaptation.

3.3 Discussion et synthèse sur les plateformes sensibles au contexte

Dans cette section, un aperçu sur certaines plateformes et architectures supportant des systèmes sensibles au contexte a été présenté. Une comparaison des couches conçues pour la description fonctionnelle des architectures a été établie. Nous avons montré que toutes ces architectures sont majoritairement monolithiques et peuvent être présentées avec les mêmes couches fonctionnelles. Mais, chaque auteur possède sa propre façon pour les traiter en fusionnant deux ou trois couches ensemble.

En effet, nous retrouvons toujours les trois principales couches de l'architecture générique. Par exemple, la couche des capteurs existe pour pratiquement toutes les plateformes. Elle est assurée par les « *Context-widgets* » dans l'architecture ToolKit, par les « *Context-providers* » dans la plateforme SOCAM. Elle est assurée de la même manière par le sous-système « *Context-acquisition* » dans l'architecture CoBrA. Pour la plateforme JCAF, la couche capteur est apparue comme une sous-tâche du composant « *Context monitor/actuators* » qui assure également la partie interprétation du contexte. Elle peut être vue ainsi comme une fusion de deux couches. Il en est de même pour la couche traitement du contexte. Elle existe évidemment pour tous les systèmes décrits dans cette section. Dans l'architecture CoBrA, c'est le composant « *Context-Broker* » qui réalise cette couche. Pour la plateforme JCAF, le composant « *Context-service* » accomplit le fonctionnement de cette couche. Pour d'autres, cette tâche est subdivisée suivant les trois principaux stages de traitement du contexte, c'est-à-dire l'interprétation, le stockage et la diffusion du contexte. Par exemple, le composant « *Interpreter* » de ToolKit est le composant responsable de l'interprétation du contexte. Le composant « *Context-analyzer* » assure cette tâche dans la plateforme CAADTA.

Aussi, faut-il noter qu'il s'agissait de la même classification des couches de stockage du contexte et de la diffusion du contexte. La première est assurée par différents composants plus ou moins sophistiqués pour la sauvegarde du contexte, allant d'un simple composant d'archivage (*World archive*, *Storage*, *Context-database*…), passant par des modèles de fonctionnement (*Context-model*, *System-model*) jusqu'aux bases de connaissances (*Context- knowledge-base*).

La couche de diffusion est également assurée par des différents composants, mais elle est caractérisée dans les premiers travaux de recherche en assurant la diffusion de l'information appropriée aux différents services de la couche applicative, elle joue ainsi le

rôle de l'adaptation dans ces plateformes. Le tableau 1.2 résume le rôle de tous les composants des travaux présentés dans cette section et présente une certaine classification par couche de ces travaux.

Tableau 1. 2 Classification par couches des différentes plateformes et architectures d'adaptation au contexte

	Couche capture du contexte	**Couche traitement des données contextuelles**			**Composants Supplément-aires**	**Technique d'implémentation**
		Interprétation	**Stockage**	**Diffusion**		
Context Information Service	Sensors array	World (Synthesizers & Monitors)	World Archive	World (Monitors through States' artifacts)		Object & Service oriented architecture
Context Toolkit	Context widget	Interpreter	Discover & Context widget	Aggregators & Context widget		Object oriented architecture
HyCon	Sensors	Servers	Storage	Terminal		Service-oriented architecture
CoBrA	Context broker				Privacy management module	Agent based architecture
	Context Acquisition	Context Reasoning Engine & Knowledge sharing	Context knowledge base	Context Reasoning Engine & Knowledge sharing		
JCAF	Context monitor / actuators	Context Service			Access Control	Service-oriented architecture
		Context Monitors/Actuators	Translator	Aggregators		
SOCAM	Context Providers	Context Interpreter	Context Database	Context-aware services	Service-locating service	Service-oriented architecture
SECAS	Context provider	Context interpreter	Context repository	Context broker	Application core	Module based architecture
CAAMDTA	Specific sensors	Context analyzer	Context model / user model	UM and CM service		Module based architecture
CAASS	Functional system & Operational context					Module based architecture
	Sensors, monitors and effectors	Analyzing and processing operation	System and Context model	(ensured by the sensing and monitoring component)		

Cette classification nous a fourni un moyen pour conclure sur le processus de développement des systèmes sensibles au contexte. Le processus est constitué de quatre étapes principales. Une première étape d'analyse du système ubiquitaire est indispensable. Cette étape fournit un document contenant les paramètres contextuels décrivant l'environnement et ses différents éléments. L'étape suivante consiste à traiter ces données contextuelles, elle est considérée comme l'étape de préparation au niveau de laquelle les données doivent être analysées et traitées. La troisième étape consiste à

établir la stratégie d'adaptation qui doit être appliquée à ces données afin de répondre aux exigences de l'utilisateur. La dernière étape consiste à spécifier et générer l'interface utilisateur. Cependant, les utilisateurs et les interfaces font partie du système ubiquitaire et constituent la source cruciale d'information pour l'analyse du système comme pour l'étape de traitement des données contextuelles. Pour cette raison, le processus peut être considéré comme étant un cycle (Figure 1.11).

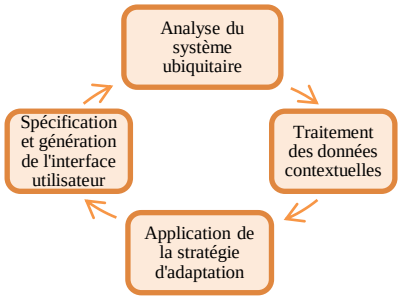


Figure 1.11 Processus de développement des systèmes d'adaptation

D'un autre côté, au début les premiers travaux de recherche engagés dans le domaine de l'informatique ubiquitaire ont porté essentiellement sur la gestion des données contextuelles ainsi que leurs descriptions, raison pour laquelle cette couche était mise en valeur (Dey, 2001) (Pascoe, 2001) (Gu, 2005). Les chercheurs étaient en compétition en vue de concevoir la meilleure solution pour décrire et gérer cette couche. De cette façon, les premiers systèmes offraient des intergiciels qui réalisent des interactions entre les informations du contexte et les applications ; à savoir : la collecte, le stockage, l'interprétation et l'analyse des données contextuelles. L'adaptation, dans ces systèmes, portait particulièrement sur l'adaptation à des situations du contexte. La stratégie adoptée pour l'adaptation au contexte n'est pas détaillée dans ces travaux ou bien elle est totalement négligée. Prenons comme exemple l'architecture CIS, l'objectif primaire de cette architecture est de rassembler, modéliser et fournir les informations contextuelles. L'architecture SOCAM est proposée pour convertir les divers espaces physiques, d'où les informations contextuelles peuvent être transformées en un espace sémantique et ainsi elles peuvent être partagées entre des services sensibles au contexte. L'élément essentiel dans l'architecture CoBrA détient un modèle de contexte. En revanche, les architectures récentes considèrent de façon prioritaire cet aspect et mettent l'accent sur la couche application en fournissant une stratégie d'adaptation (Chaari, 2006) (Buttussi, 2008) (Hussein, 2011). Ce qui montre que cette étape mérite plus d'intérêts et plus d'études (Figure 1.12).

Figure 1.12 Evolution des objectifs des systèmes d'adaptation au contexte

3.4 Besoins opérationnels d'un système d'adaptation au contexte

Si nous revenons avec une minutieuse interprétation des différentes notions liées à l'informatique ubiquitaire citées précédemment, nous discernons tout d'abord que l'informatique ubiquitaire doit faciliter les activités humaines et remplacer son intervention dans toute tâche pouvant lui perdre son temps sans qu'il perçoive cette assistance. Ensuite, un système sensible au contexte utilise le contexte pour fournir des informations et des services pertinents pour l'utilisateur, où la pertinence dépend généralement de la tâche demandée par l'utilisateur. Enfin, dans un environnement ubiquitaire, l'utilisateur aurait grandement besoin de certaines informations mais trouvera d'autres complètement inutiles ; ce choix dépend impérativement de ses préférences.

Nous retenons en conséquence trois principales exigences pour la pertinence et l'efficacité du fonctionnement d'un système d'adaptation au contexte :

- Le fonctionnement d'un système d'adaptation au contexte doit être ***centré utilisateur***. Autrement dit, le fonctionnement doit être spécifique au profil de l'utilisateur, à ses préférences subjectives, ses caractéristiques intellectuelles, individuelles, etc.
- L'adaptation de l'information doit être en **temps réel**. Selon son contexte d'utilisation, l'utilisateur a besoin, en plus d'une information adéquate à ses préférences, d'une **information instantanée**.
- L'adaptation de l'information dans un environnement ubiquitaire – où l'utilisateur est généralement engagé dans la réalisation d'une activité – doit nécessairement **prendre en considération cette activité**. Le but consiste à assister l'utilisateur dans la réalisation de sa tâche courante en lui fournissant l'information idoine à l'accomplissement de cette tâche, et même sans qu'il perçoive cette assistance.

La question qui se pose à ce niveau est : « Si ces trois principaux facteurs sont absents, un système sensible au contexte peut-il mener à bien son rôle ? ». Le tableau 1.3 présente et discute essentiellement la prise en compte de ces trois critères dans les travaux de recherche précédemment décrits. Nous constatons tout d'abord que ces trois critères ne sont guère considérés dans les premières architectures. En revanche, ils commencent à apparaître dans les dernières architectures. Ce qui explique l'importance de la prise en compte de ces critères pour répondre aux exigences d'un système d'adaptation.

En effet, nous avons remarqué que le facteur temps-réel est négligé pour la majorité des systèmes décrits bien qu'il soit un facteur prioritaire dans un environnement ubiquitaire. Pour la plateforme «CAAMDTA», ce facteur n'était pas mentionné explicitement. Par contre, nous avons implicitement conclu sa présence à partir de la nature du fonctionnement du système. De même pour la prise en compte des préférences de l'utilisateur, elle est explicitement négligée dans la majorité des travaux.

Pour la plateforme « SECAS » le composant « *Context-provider* » est responsable de la gestion des aspects du contexte liés à l'utilisateur, tels que le profil et l'historique ; mais la prise en compte de ces facteurs dans le processus d'adaptation n'est pas

réellement mentionnée. A cet égard, la plateforme « CAAMDTA » a prévu un module responsable de la déduction des informations de haut niveau relatives en particulier à l'état physiologique des utilisateurs et à leurs mouvements.

Tableau 1. 3 Caractéristiques de l'adaptation au sein des différentes plateformes

	Prise en compte de l'activité courante	Dynamique du fonctionnement	Prise en compte des spécificités de l'utilisateur	Considération du Temps-réel de l'adaptation	Remarques sur l'adaptation
Context Information Service	Non	Non	Non	Non	Le « monitor » cherche le capteur qui peut lui fournir l'information adéquate.
Context Toolkit	Non	Non	Non	Non	Adaptation du contexte aux demandes de l'application
HyCon	Non	Non	Non	Non	Adaptation physique
CoBrA	Non	Non	Les politiques de confidentialité	Non	Raisonnement sur le contexte
JCAF	Non	Non	Non	Non	Il y a une couche d'adaptation du contexte assuré par le « service client »
SOCAM	Activité en générale et non celle en cours	Non	Non	Non	Les context-aware services changent leur fonction selon le changement du contexte.
SECAS	Activité en générale et non celle en cours	Non	Oui	Non	Adaptation du contenu Adaptation de la présentation
CAAMDTA	Oui	Non	Oui	Oui	Adaptation du contenu Adaptation de la présentation
CAASS	Non	Non	Non	Non	Adaptation du contenu

3.5 Conclusion

En analysant les travaux de recherche sur les plateformes et les architectures d'adaptation au contexte, nous avons tout d'abord constaté que pour les premières architectures proposées, la partie traitement et gestion du contexte était mise en évidence comme une fonctionnalité prioritaire d'un système ubiquitaire. L'expression exacte caractérisant l'adaptation dans ces architectures est « *l'adaptation du contexte*[4] » et non « *l'adaptation au contexte* [5]». L'adaptation du contexte peut se traduire par la déduction de l'information adéquate qui va être utilisée par la couche applicative. La question que nous posons à ce niveau est : « même si cette information est adéquate, est-ce qu'elle va correspondre effectivement à l'information dont l'utilisateur aurait besoin à ce moment là ? ». Cette constatation nous amène à traiter deux problèmes majeurs suivants :

- Problème de l'introduction du facteur « temps-réel » dans le processus d'adaptation au contexte dans un système ubiquitaire ;
- Problème de la déduction des besoins utilisateurs en termes d'information selon ses préférences et ses caractéristiques.

Ensuite, nous avons montré à travers l'analyse chronologique des plateformes étudiées que la couche adaptation est devenue progressivement une couche intéressante et

[4] C'est à dire une sorte de classification du contexte pour faciliter une ultérieure utilisation.
[5] C'est à dire adapter l'information (ou le service) à fournir suivant le contexte courant.

primordiale dans de tels systèmes, allant même à migrer les fonctionnalités de la couche traitement du contexte vers celle de l'adaptation (cas de l'architecture CAASS).

Finalement, nous avons constaté que tous les travaux de plateformes existants pour l'adaptation se basent sur un modèle fixe de fonctionnement. Même l'architecture CAASS qui a donné une grande importance à la couche adaptation, les scripts responsables de l'adaptation représentent un *ensemble d'actions prédéfini* en vue de répondre à tout changement du contexte. Une autre question se pose à ce niveau : « comment le système doit-il réagir face à une action qui n'existe pas dans cet ensemble ? ».

4 Conclusion générale

L'adaptation au contexte suivant notre cadre d'étude peut être définie par la flexibilité des interfaces utilisateur à faire évoluer au cours de l'interaction en fonction du contexte de l'utilisateur (ses préférences, son profil, son activité courante, etc.) et ce, en prenant en considération les informations dont il a besoin à ce moment. Et puisque généralement la pertinence des modèles dépend de la qualité de l'adaptation dynamique de l'interface, l'adaptation qui se base sur un modèle fixe de fonctionnement n'est plus appropriée dans les environnements ubiquitaires. En effet, le contexte a tendance à varier énormément au cours de l'interaction avec l'utilisateur dans un environnement très dynamique. D'où la déduction de l'information nécessaire au moment approprié pour un utilisateur quelconque exige une plus grande flexibilité opérationnelle. Ainsi une méthode statique de fonctionnement est certainement insuffisante. Cette complexité croît lorsque les informations fournies à l'utilisateur doivent correspondre à certaines caractéristiques spécifiques et individuelles des utilisateurs.

Notre travail de thèse consiste essentiellement à proposer une architecture et une stratégie d'adaptation qui remédie aux problèmes rencontrés lors de l'analyse de l'état de l'art sur ce domaine de recherche. Notre travail a pour objectifs de :

- Concevoir une architecture flexible et dynamique pour l'adaptation de l'information fournie à l'utilisateur à son contexte d'usage et en particulier à son activité courante.
- Etablir une stratégie d'adaptation qui permet essentiellement de déduire et d'identifier les besoins de l'utilisateur en terme d'information et ce en temps réel.

Le but est de doter notre architecture d'une certaine flexibilité afin qu'elle puisse générer dynamiquement des modèles de l'activité de l'utilisateur et en déduire la meilleure information pouvant l'aider à accomplir sa tâche courante au sein de l'environnement ubiquitaire en question. Ceci passe implicitement par détecter le contexte courant de l'utilisateur et l'aider à atteindre ses objectifs en modifiant éventuellement le contenu et le comportement de l'interface selon le changement du contexte de cet utilisateur tout en veillant à lui fournir l'information dont il a besoin sous la forme la mieux adaptée.

Chapitre 2

CADRE CONCEPTUEL POUR UNE ADAPTATION DYNAMIQUE ET TEMPS-REEL AU CONTEXTE D'UTILISATION

1. Introduction

Au cours du chapitre précédent, et à travers l'étude de l'existant, nous avons pu identifier les couches fonctionnelles nécessaires et suffisantes pour définir un système ubiquitaire ainsi que les différents besoins opérationnels qui devront être satisfaits. Dans cette perspective, l'objectif clé de ces systèmes consistait à fournir à l'utilisateur l'information fiable, pertinente et la mieux adaptée à ses besoins. L'aspect temps-réel qui devrait caractériser le processus d'adaptation en représente clairement l'un des défis majeurs pour ces systèmes. Il s'agit, non seulement de savoir qu'elle est l'information la plus appropriée pour l'utilisateur, mais également de la fournir au bon moment. Cette complexité prend plus d'ampleur lorsque l'information déduite doit correspondre à des caractéristiques spécifiques et individuelles de chaque utilisateur. Certes, un modèle fixe de fonctionnement s'avère inapproprié pour la mise en œuvre effective des systèmes ubiquitaires. En présumant cette hypothèse, le fonctionnement du système proposé devrait être régi par un modèle dynamique. Ce modèle devrait être maintenu et géré en temps-réel afin de mieux satisfaire l'aspect fortement dynamique des environnements ubiquitaires et la très grande variabilité des utilisateurs. La description fonctionnelle du système proposé fait l'objet de ce présent chapitre. La gestion des données contextuelles en représente clairement l'une des composantes-clefs, imposant également le bon choix du modèle de description convenable pour la gestion des systèmes ubiquitaires. Cette partie est détaillée dans la quatrième section. Compendieusement, ce chapitre nous offre un cadre général d'étude (Présentation du système ubiquitaire proposé et modélisation des données contextuelles) pour la proposition d'une méthode d'adaptation présentée au quatrième chapitre. Mais avant tout, nous nous efforçons de préciser et détailler, par l'analyse d'une étude de cas, les principales problématiques étudiées dans le cadre de ce travail.

2. Etude de cas et problématiques

Dans le modèle classique de la recherche, la phase d'étude de cas est une étape intermédiaire entre la construction du modèle de fonctionnement d'un système et l'examen des données choisies pour le tester. Dans la recherche que nous avons menée, il n'en est rien. Nous avons été conduits à considérer différents cas dès le début de notre recherche. L'étude applicative s'est donc déroulée en même temps que la construction de notre objet d'étude. Raison pour laquelle nous avons mis en évidence la déduction temps-réel des besoins utilisateurs comme premier facteur et problématique majeure de l'adaptation au contexte.

2.1. Aperçu général sur le cas d'étude : l'hôpital ubiquitaire

L'analyse des relations entre la problématique et l'évolution de l'approche proposée, d'une part, et l'efficacité et le dynamisme des modèles de fonctionnement, d'autre part, est faite à travers l'étude de cas. Le but des travaux était au début d'orienter les politiques, les

stratégies d'adaptation dans un esprit de faire face aux insuffisances identifiées dans ce domaine. Pour atteindre ce but, l'étude présente les caractéristiques suivantes :

- Domaine d'étude : Médecine
- Environnement ubiquitaire : Hôpital Ubiquitaire
- Utilisateurs : Médecins, Médecins internes, Médecins résidents, Infirmiers, Chef service, Paramédicaux, Techniciens, Radiologues…

L'ubiquité numérique a déjà trouvé des applications dans de nombreux domaines. La médecine, en particulier a pris une place prépondérante dans l'informatique ubiquitaire d'aujourd'hui, comme en témoigne l'éclosion de nombreuses directions de recherche dans ce domaine. De l'échelle moléculaire ou cellulaire, modèles mathématiques et sémantiques permettant de simuler des processus du vivant ; à l'échelle de la société avec l'accès et le partage de l'information, aucun domaine, aucun aspect de la médecine n'est épargné. Le but est d'améliorer la qualité de vie des patients, des soins mais également de tenter de simplifier nombreuses tâches, généralement quotidiennes, du personnel médical.

Un hôpital ubiquitaire ou « *U-hospital* » représente encore un défi majeur estimé être réalisé entre les années 2016-2018. Il est considéré comme un espace ubiquitaire interactif, ambulatoire et intelligent qui a pour objectif principal de s'assurer que l'information dont le personnel médical aurait besoin est disponible immédiatement, quand et où ils en ont besoin. Il peut être équipé par un ensemble de dispositifs d'affichage sophistiqués, des dispositifs spécialisés pour l'interaction, des capteurs de supervision, etc. (Figure 2.1). Un suivi approprié devrait être assuré par le personnel soignant, notamment à l'égard des traitements administrés en consultation ou intervention ambulatoire. Ce même constat s'applique à l'accès au traitement, aux soins et à l'accompagnement.

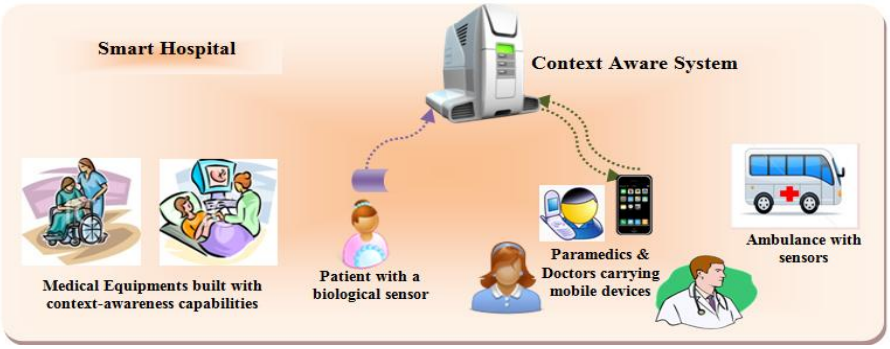


Figure 2.1 Simulation des différentes parties constituant un Hôpital Ubiquitaire

Si par exemple un patient présente des changements indésirables des signes vitaux, une alerte sera transmise directement à l'infirmier qui prend soin du patient en question à travers son dispositif. De même, un infirmier pourrait voir les informations relatives à un patient quelconque sur son dispositif. Plus important encore, un médecin pourrait appeler les derniers résultats du laboratoire d'un patient. En outre, les images sur le petit écran peuvent être transférées vers le grand écran de sorte que le patient peut également voir sa

dernière radiographie (Fuhrer, 2006). Par ailleurs, l'un des principaux buts visés par une telle technologie est la surveillance et l'intervention médicales temps-réel, par exemple : la surveillance des personnes cardiaques, le contrôle à distance des défibrillateurs automatiques, la surveillance des patients épileptiques via le contrôle des stimulateurs du cerveau multi-programmables, la surveillance de la glycémie et l'administration de l'insuline chez les patients atteints de diabètes aigus, etc. Notre premier cas d'étude s'intègre dans ce dernier exemple et consiste à suivre en temps réel l'évolution des états des patients. Ce suivi est rendu possible grâce à des capteurs biologiques[6] implantés sous la peau et contrôlant, périodiquement, les niveaux de glucose chez les patients. Le système doit vérifier, en permanence, l'évolution des états de chaque patient, l'enregistre, la valide et guide une éventuelle intervention médicale.

Ci-après, trois scénarios sont succinctement présentés dans le but d'illustrer implicitement la façon dont le système fonctionne et tout en soulignant la flexibilité qui devrait régir ses interactions. Chacun des cas étudiés présente un aspect original, et des enseignements sur les procédés d'adaptation. Dans cet exemple, les patients souffrant d'hypoglycémie sont particulièrement étudiés. Une hypoglycémie se produit lorsque le taux de glucose (sucre) dans le sang est trop bas. La concentration de glucose sanguin de l'organisme se maintient normalement entre 4,0 mmol/L et 8,0 mmol/L. La constance de ces valeurs est maintenue dans le corps grâce à des mécanismes comme l'hormone insuline ainsi que plusieurs autres hormones fabriquées par le pancréas (Nagati, 2001) (Tremblay, 2006) (Canoe, 2012). Cet état se produit généralement lorsque le patient diabétique administre trop d'insuline ou s'il manque un repas. Une hypoglycémie se définit comme une glycémie inférieure à 4 mmol/L. Une hypoglycémie sévère présente un taux de glycémie généralement inférieure à 2,8 mmol/L ; dans ce cas la personne a besoin de l'aide d'une autre personne ; en plus elle peut facilement perdre conscience ; situation qui nécessite un traitement et des précautions à prendre afin d'éviter les situations potentiellement dangereuses.

Le système surveille l'état généralisé de chaque patient tout en analysant leurs dossiers cliniques et les traitements. Les valeurs des niveaux de glucose sont les facteurs les plus importants qui doivent être suivies dans le cadre d'une intervention d'hypoglycémie sévère. A la lumière de ces valeurs, le système doit vérifier l'évolution de chaque patient. Nous allons supposer qu'à un moment donné, le système détecte qu'une certaine patiente « *Sonia* » présente un taux anormal de glucose ; alors il peut déduire, à partir de sa base de connaissance, qu'elle souffre d'une hypoglycémie. Par conséquent, un message sera diffusé partout dans l'environnement indiquant un état d'urgence qui nécessite une intervention. Par exemple, un message relatif à cette situation peut être : "The PATIENT Id = 202 has ABNORMAL GLUCOSE RATE 2.2mmol". Un infirmier passant près d'un point de diffusion, voit afficher ce message sur l'interface de son dispositif (Téléphone mobile, PDA, PC portable) (Figure 2.2), déduit implicitement la nécessité d'une

[6] Un minuscule dispositif électronique d'à peine deux centimètres carré intégré d'une manière sous-cutanée et qui mesure le niveau du sucre chez les patients (Marinesco, 2008).

intervention urgente pour le cas mentionné. Mais également cet utilisateur aurait le choix d'accepter ou de refuser cette intervention.

D'un autre côté, l'intervention médicale d'un infirmier est en fonction de l'état enregistré du patient. Cette intervention s'incarne en les mesures à prendre (sous forme d'étapes) lors de la détection d'un cas d'urgence (Tremblay, 2006). A chaque étape de traitement, l'infirmier (ou n'importe quel personnel médical) aurait besoin de certaines informations afin d'accomplir la tâche en cours, ces informations sont appelées les informations contextuelles qui représentent des connaissances sur tout l'environnement. D'ailleurs, ces connaissances doivent être sélectionnées et fournies à l'infirmier selon ce qu'il est en train de faire.

Figure 2.2 Un exemple de message émis par le système

a. « Déduction de l'information requise par l'utilisateur suivant la tâche en cours » : Le premier cas concerne les données à fournir ainsi que les mesures à prendre afin de permettre à l'infirmier agissant pour dispenser des soins au patient, établir un diagnostic ou administrer un traitement. Soutenir l'infirmier pour se rappeler du plan de soins et de traitement de ce type de patients ainsi qu'ajuster le traitement à ses besoins constitue l'objectif de ce premier cas.

b. Dans le deuxième exemple, nous revenons au cas décrit ci-dessus, mais nous supposons qu'à un instant *t*, le patient perd connaissance pendant que l'infirmier s'occupait de lui. Comme nous l'avons déjà mentionné, une hypoglycémie sévère peut causer une perte de conscience. Dans ce contexte, le traitement chez une personne consciente diffère de celui chez une personne inconsciente et incapable d'avaler.

c. Le troisième cas, partiellement traité, est centré profil-utilisateur, c'est-à-dire concerne l'infirmier lui-même. Le système doit prendre en compte la spécialité et le niveau de connaissance de l'infirmier (son profil). Autrement exprimé, l'information requise diffère d'un infirmier endocrinologue à un autre non-spécialiste. Par ailleurs, cette information est également en forte relation avec l'état psycho-cognitif du personnel médical. Cette problématique peut être au centre de notre étude plutôt qu'en marge. Un personnel médical peut parfois être surchargé de travail et psycho-cognitivement non-disponible. Cette problématique n'est pas spécifiquement parmi nos ultimes objectifs, en revanche elle fera l'objet des premières perspectives envisagées.

2.2. Analyse fonctionnelle et identification des problèmes engendrés

Au travers ces cas suscités, nous allons essayer d'identifier des problèmes ou des propriétés de situation ou encore des évènements complexes. Le propos est plutôt analytique : il ne s'agit pas d'accumuler les faits ou de développer des hypothèses, mais de conduire une étude à partir de certaines idées. Au départ, l'étude de cas suppose l'existence de problèmes relatifs au domaine d'étude considéré dans ces travaux. En fait, nombreuses sont les problématiques qui peuvent découler d'une telle étude de cas. La question qui en découle naturellement est : comment faut-il procéder pour que l'infirmier apporte au mieux les premiers soins pour un cas compliqué ? Il s'agit des objectifs du système ubiquitaire relatif à ce cas qui devrait **répondre, implicitement, aux besoins des utilisateurs en leurs procurant, au moment opportun, une information temps-réel.** D'un autre côté, le deuxième cas pose la question suivante : comment le système doit-il agir face à un changement brusque de la tâche de l'utilisateur ? Autrement exprimé, est-ce qu'à travers une modélisation classique de l'interaction Homme-Système nous pouvons garantir un résultat satisfaisant qui supporte ce changement.

C'est sans doute là qu'il faut voir le tout début de notre approche d'adaptation. Cette phase sera consacrée à la mise au point d'une technique permettant la construction du modèle d'interaction en temps réel qui révèle une réflexion plus générale sur les difficultés de déduire les besoins informationnels des utilisateurs conditionnés par leurs situations courantes. En particulier, supposant que l'état du patient en question avait changé (dans notre cas, il s'agit d'un passage de l'état conscient à l'état inconscient), comment le système doit-il se comporter pour mener à bien son assistance ? Le troisième cas d'étude révèle un facteur jugé également nécessaire dans le processus d'adaptation de l'information, à savoir le profil de l'utilisateur et notamment d'autres caractéristiques qui peuvent en découler. Le profil joue un rôle capital à ce sujet. Il convient de l'intégrer dans le processus d'adaptation pour pouvoir déduire l'information exacte. Ainsi, nous prolongeons la phase de déduction des besoins informationnels en considérant les profils des utilisateurs, notamment les savoir-faire mis en jeu.

En se référant au cas d'étude, l'information fournie doit être adaptée également à la spécialité de l'utilisateur et son niveau de connaissances. Ce facteur se révèle ainsi l'un des facteurs clefs de détermination de la pertinence de l'information. Cette contrainte est associée aussi à d'autres facteurs importants (résumés dans le tableau 2.1) qui sont souvent liés entre eux, tels que : le niveau de connaissance, la situation courante, l'activité courante, l'état du patient. En effet, si nous partons d'un modèle de fonctionnement basique et fixe, le fonctionnement du système se révèlera inapproprié de par l'objet même de notre vision. Ce problème s'est en effet posé, spécifiquement, lors d'un passage brusque d'une situation à une autre ou d'une tâche à une autre.

Tableau 2. 1 Cas particuliers et problématiques associées

Situation Particulière	Infirmier spécialiste		Infirmier non-spécialistes
	Patient Conscient	Patient devient Inconscient	
Problématique identifiée	Modèle connu ⬇ Information requise	Modèle inapproprié ⬇ Information inutile	Modèle inapproprié ⬇ Information insuffisante
Situation Générale Correspondante	Niveau de connaissance suffisant Situation connue	Niveau de connaissance suffisant Situation a changé	Niveau de connaissance insuffisant

3. Cadre conceptuel proposé pour l'adaptation dynamique et temps-réel

Une des classes de problèmes pour laquelle un système ubiquitaire devrait plus particulièrement apporter des réponses spécifiques, concerne l'architecture proprement dite des systèmes (Touzet, 2002). Le chapitre précédent nous a permis d'identifier les facteurs nécessaires à prendre en compte lors de l'établissement d'un système d'adaptation au contexte. Nous avons également spécifié qu'au niveau des systèmes d'adaptation, une architecture formée au minimum de trois couches est nécessaire et suffisante pour la mise en œuvre de tels systèmes. Cette section décrit l'architecture générale du cadre proposé dans ce travail. Les différents composants de cette architecture et leurs rôles sont tout d'abord analysés avec un accent plus particulièrement porté sur le module responsable, de près, de l'adaptation (c'est à dire *l'Adaptation Engine*) et tout en soulignant la manière adoptée pour faire face aux problèmes ci-dessus mentionnés. Nous avons à faire à une architecture permettant le support d'exécution des modèles de fonctionnement dites dynamiques ou particulièrement des modèles évolutifs. Ainsi, nous avons choisi d'appeler le système : **R**eal Time **A**daptation **D**ynamic based on **E**volutionary **M**odels (désigné sous l'acronyme RADEM).

3.1. Objectifs du système RADEM

Un des objectifs de ce travail consiste à étudier et concevoir un système permettant la gestion dynamique et temps-réel de l'information fournie à l'utilisateur à travers son interface. Pour atteindre cet objectif, un modèle conceptuel du système a été défini et décrit dans cette partie. Le système doit s'adapter dynamiquement aux besoins, en perpétuelle évolution, des utilisateurs. L'interface doit offrir ainsi et en temps réel un contenu très bien adapté à ce que nous avons appelé le contexte courant d'utilisation. Cette architecture devrait gérer des modèles de fonctionnement pertinents et efficaces permettant une adaptation dynamique du comportement de l'interface utilisateur sans oublier que l'ultime objectif est d'aider l'utilisateur à accomplir sa tâche courante. Pour répondre à ces objectifs, nous avons proposé un modèle conceptuel se basant sur les trois principales parties identifiées dans le chapitre précédent. Ceci passe tout d'abord par détecter les objectifs des utilisateurs et les aider à atteindre ces objectifs en modifiant éventuellement l'état et le comportement de leurs interfaces. Ensuite, étudier la possibilité

de leur représenter, sous la forme la mieux adaptée, les informations issues du système. Le fonctionnement est donc guidé par ces objectifs.

3.2. Aperçu général du système RADEM

Une partie de la solution comprendra un premier module pour recueillir et analyser les informations contextuelles de diverses sources dont les utilisateurs et les réseaux d'accès. Les résultats de cette analyse seront utilisés par un module responsable de l'adaptation proprement dite et capable de concevoir et produire en temps réel les informations requises par l'utilisateur. Enfin cette information sera présentée à l'utilisateur. Le processus d'adaptation doit prendre en considération le changement du contexte qui inclue toute information pouvant être cruciale pour améliorer ou transformer le modèle de fonctionnement en cours (Figure 2.3).

3.2.1 Gestion et analyse des données contextuelles

Différentes composantes contribuent donc à ce processus d'adaptation. La gestion des données en représente clairement l'une des composantes-clefs. Elle recouvre de nombreux aspects, notamment l'analyse des données traitées par le système, l'intégration des données ainsi que l'exécution des requêtes dans des environnements distribués fortement hétérogènes. Cette analyse aboutit à un document contenant une organisation ou classification des paramètres contextuels. Ce document sera traduit en une représentation contextuelle en adoptant une des techniques de spécification du contexte. Après une étude bibliographique et critique, le langage de description basé sur les ontologies a été choisi. Nous reviendrons plus en détail sur cette partie ultérieurement.

3.2.2 Approche d'adaptation aux données contextuelles

L'approche d'adaptation suppose de considérer les différentes facettes des données contextuelles de l'utilisateur afin de les adapter à ses besoins. Il s'agit du concept fondamental qui influe sur le processus de fonctionnement du système, typiquement connu sous la dénomination : « stratégie ou politique d'adaptation ». A travers cette politique, l'importance d'ajuster l'information fournie à l'utilisateur à ses besoins devrait être particulièrement soulignée. Cette réponse aux besoins de chaque utilisateur doit s'appuyer sur une évaluation du modèle du fonctionnement qui permet de considérer non seulement les variations du contexte mais également l'attribution à chacun des utilisateurs une information spécifique à ses besoins. Cette assignation est également fortement liée à tous les aspects de la vie sociale, quotidienne et même privée des utilisateurs.

Il revient alors au système d'identifier ces besoins et de prévoir les mesures nécessaires pour y répondre, puisque c'est lui qui connaît le mieux ces utilisateurs ainsi que les situations courantes dans lesquelles ils évoluent. L'actualisation de l'approche individualisée nécessite des ajustements strictement relatifs à leurs situations courantes.

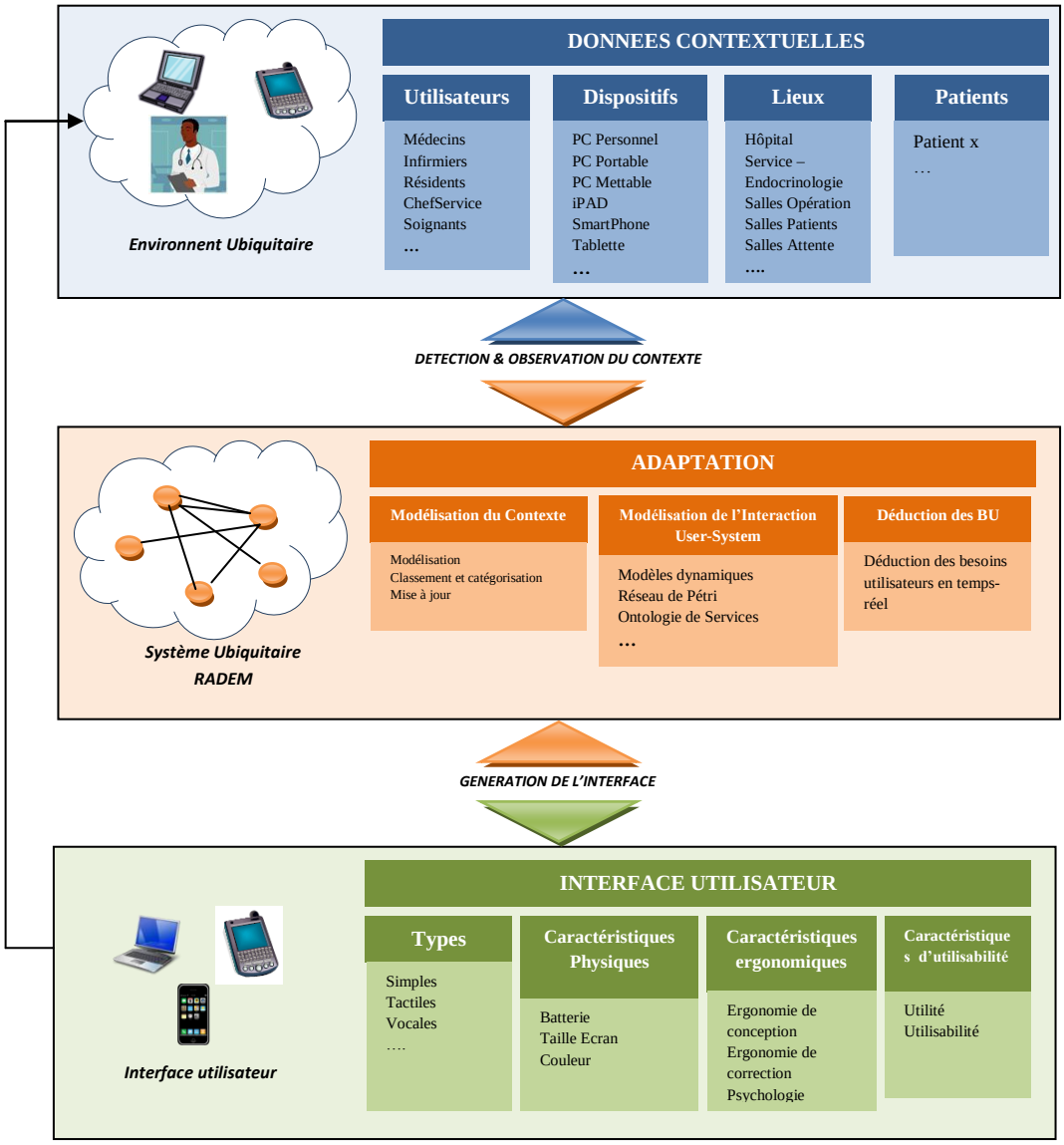


Figure 2.3 Cadre Général du système RADEM

a. Modélisation de l'interaction homme-système

La répartition des tâches entre le système et l'utilisateur est souvent effectuée en fonction des critères propres au développement du système et non en fonction des caractéristiques – par nature changeantes – du contexte. L'informatique ubiquitaire avait relativement changé la relation d'interactivité dans les systèmes interactifs. Cette constatation nous amène à traiter le problème de : La nouvelle manière dont l'utilisateur interagira avec le système tout en considérant d'une part la teneur des informations échangées et d'autre part la nature de cette interaction.

L'amélioration de la communication homme-système avec la possibilité d'intégrer de nouveaux moyens d'interaction en vue d'arriver à bien gérer ces environnements par des modèles de fonctionnement bien appropriés, constitue l'objet principal derrière ce présent travail. Ainsi, dans une deuxième étape les activités des utilisateurs doivent être décrites et analysées. Une activité est composée d'un ensemble d'actions élémentaires.

Ces actions élémentaires sont, au préalable, modélisées au moyen de structures élémentaires de réseaux de Petri interprétés. Ensuite, nous procédons par des compositions élémentaires sous forme séquentielle, parallèle, alternative, itérative, etc. Après, un modèle global sera construit tout en intégrant les principales évolutions des états de fonctionnement du système ubiquitaire. De même, le comportement de l'utilisateur sera modélisé. Cette modélisation décrit l'interaction de l'utilisateur avec l'interface graphique.

Egalement, les réseaux de Petri interprétés sont utilisés en introduisant les événements et les conditions. Ainsi, le modèle RdP va permettre de décrire les enchaînements d'actions et leurs effets sur la déduction des besoins utilisateurs et par la suite la génération de l'interface de son dispositif. Nous reviendrons plus en détail sur cette partie dans le prochain chapitre.

b. Déduction des besoins utilisateurs et identifications des Widgets nécessaires

L'étape qui vient après, permet l'identification des besoins utilisateur en termes d'informations dans un premier lieu, et en termes de Widgets[7] en second lieu.

3.2.3 Génération de l'interface utilisateur graphique

Une fois les éléments nécessaires de l'interface sont identifiés, l'étape suivante consiste à spécifier l'interface graphique en termes d'objets graphiques et d'affichage (Moussa, 2000). En effet, la dernière étape de cette approche est consacrée à la génération automatique et temps réel de l'interface. Cette étape est divisée en deux phases indispensables, à savoir la spécification de l'interface graphique suivie par la génération automatique et temps réel de celle-ci.

3.3 RADEM : Décomposition et description fonctionnelles

Cette section décrit le modèle conceptuel du système d'adaptation proposé en vue de répondre à ces objectifs. La description fonctionnelle du modèle proposé est donnée par la figure (Figure 2.4). Ce système est constitué de trois principaux composants qui sont fonctionnellement dépendants : le composant ***Adaptation Engine*** (AE), le composant ***Knowledge Repositories*** (KR) et le composant ***Analysis and Query Building*** (AQB). Les différents blocs de l'architecture ainsi que leurs rôles sont tout d'abord analysés avec un accent plus particulièrement porté sur la partie responsable de l'adaptation proprement dite.

[7] Widget est une contraction des mots Windows et Gadget (cf. 5ème section). En informatique, le mot widget recouvre deux notions distinctes en relation avec les interfaces graphiques. Il peut alors être considéré comme étant la contraction des termes Windows (fenêtre) et gadget.

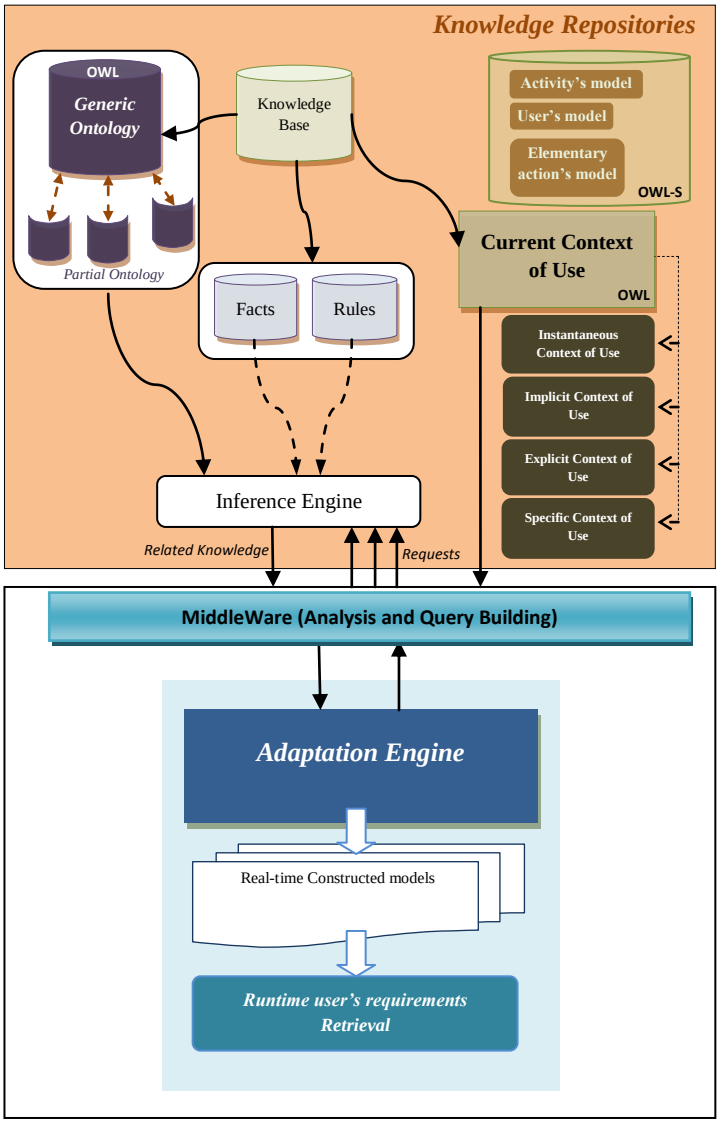


Figure 2.4 Modèle conceptuel et fonctionnel du système RADEM

3.3.1 Aperçu sur le composant « *AE : Adaptation Engine* »

Ce composant est considéré le cœur de fonctionnement du système. C'est lui qui est à la base de la création dynamique des modèles de fonctionnement. Il est responsable de la construction et du suivi des modèles en temps réel en fonction de l'activité de l'utilisateur. En plus, la déduction des besoins utilisateur à partir de ces modèles représente un des principaux rôles de ce module. Il s'agit également du module qui incarne la stratégie d'adaptation. La prochaine section tentera de détailler davantage le fonctionnement de ce composant.

3.3.2 Aperçu sur le composant « *Analysis and Query Building* »

Ce composant est considéré comme une étape primordiale d'analyse de la situation actuelle (Figure 2.4). Il s'agit de vérifier en permanence la différence entre une nouvelle capture des données et la situation antérieure. La sortie est une requête formulant une

situation donnée en vue d'obtenir l'action où l'ensemble d'actions servant de base pour la gestion de cette situation. Rappelons que chaque situation décrite est centrée utilisateur où chaque situation est spécifique à un utilisateur particulier évoluant dans un contexte particulier.

3.3.3 Description du composant « *KR : Knowledge Repositories* »

Le composant « Knowledge Repositories » assure l'espace de stockage des différentes connaissances requises (ou non) par le système. Il permet d'organiser et de classer les différentes données y compris les détails sur les utilisateurs à savoir leurs préférences, leurs intérêts, leurs profils, leurs activités usuelles, également leurs historiques, leurs interactions passées avec le système. Le contenu de ce référentiel sera de plus en plus enrichi en mémorisant et effectuant un certain apprentissage sur les interactions de l'utilisateur et ses comportements. Le système maintient une ontologie générique qui peut être partagée par tous les utilisateurs. Il peut y avoir un accès total ou restreint sur cette ontologie. Lorsqu'il s'agit d'un accès restreint à l'ontologie, on parle de vue partielle. Cet accès est étroitement lié aux profils des utilisateurs. Un des problèmes envisagé par cette politique est la confidentialité qui n'est pas traitée dans ce travail. De la même manière, seulement les informations requises, notamment les informations relatives à l'activité en cours seront disponibles et fournies. De plus, pour s'assurer que les données sont mises à jours, le système "doit être conscient" de la variation des données contextuelles et doit s'assurer des mises à jours nécessaires. Le système doit finalement effectuer une périodique synchronisation des informations actualisées entre l'ontologie générique et celle partielle. Ce module contient deux autres sous-modules qui assurent cette fonction.

Pattern Modelling

Le module « Pattern Modelling » est une abstraction d'une représentation de haut niveau du système en particulier les utilisateurs et leurs interactions. Les réseaux de Petri (Williem, 1988) ont été utilisés comme modèle formel pour représenter les utilisateurs et leurs interactions. Ainsi, ce module comprend des modèles abstraits basés sur ce formalisme de modélisation. Le moteur d'adaptation doit générer un modèle concret et instantané de l'activité de l'utilisateur, ou même changer son état courant selon la variation de l'activité en s'appuyant sur ces abstractions formelles et des règles de transformation. Le modèle généré peut être considéré comme des instances de méta-modèles bien définis et il est considéré comme un ensemble d'actions élémentaires (sous-modèles) possédant certaines propriétés. Sur la base de ces structures, le module « *Adaptation Engine* » doit fournir un modèle basique de l'activité en cours avec un ensemble minimal d'actions élémentaires.

Composition Rules

Cet élément comprend des règles et des stratégies qui contrôlent la création du modèle en temps réel. Une activité est un ensemble d'actions élémentaires liées selon des compositions typiques telles que la composition séquentielle, parallèle, choix, itération

(Moussa, 2000). L'élaboration du modèle global de l'activité de l'utilisateur est améliorée par les compositions fonctionnelles de modèles élémentaires et en se basant sur les règles d'une composition bien définie. Ces règles doivent guider, entre autres, l'ordre et l'exécution conditionnelle de l'action dans le modèle. Les règles générales nécessitent la vérification du déroulement de l'interaction. En effet, le module « Rules » représente des données rassemblées relativement à la situation actuelle, organisées et formalisées sous forme de règles de production. Ces règles sont exploitées par la suite, moyennant un moteur d'inférence capable de décider des éléments appropriés à prendre en compte à un instant donné pour formuler la requête système.

Current Context of Use

Ce module est chargé d'établir, entre autres, une classification dynamique et temps-réel du contexte général et du contexte spécifique d'un utilisateur donné. Il représente une interface entre la partie d'adaptation et l'environnement ubiquitaire physique. Le but d'une telle classification consiste à offrir au système un degré élevé de compréhension afin que l'information requise puisse être déterminée d'une façon cohérente et trouvée d'une façon aisée. Cette catégorisation représente un point de départ pour déterminer les exigences qui doivent guider la gestion des données et peuvent aider ainsi à obtenir des informations sémantiquement pertinentes.

Ce module peut être considéré comme la représentation actuelle de l'environnement, c'est à dire regrouper toute connaissance sur l'environnement. Ses composantes sont principalement : le « contexte spécifique de l'utilisation », le « contexte d'utilisation instantanée », le « contexte implicite d'utilisation » et le « contexte explicite d'utilisation ». Le contexte spécifique d'utilisation est une partie de l'état actuel des connaissances qui détermine les données déjà analysées et pouvant être nécessaires pour une situation spécifique. Les données explicites sont des données obligatoires où une action consciente par un utilisateur permet une intervention, alors que les données implicites sont des informations sans aucune intervention directe de l'utilisateur. Les données instantanées sont déduites à partir des informations requises de la part de l'activité en cours de l'utilisateur. Nous avons également prévu une base de connaissance qui comprend les recommandations ergonomiques pour la spécification et la génération des vues graphiques.

3.4 Description détaillée du fonctionnement du composant « *Adaptation Engine* »

Le moteur d'adaptation est considéré comme la partie de raisonnement du système. Il supervise continuellement la mise à jour des modèles en même temps que l'utilisateur change d'activité ou que le contexte courant d'utilisation change. Cette surveillance sera sous le contrôle direct de l'interaction de l'utilisateur avec son environnement. Cependant, et dans un environnement ubiquitaire, l'activité de l'utilisateur ne suit pas strictement un plan, mais elle est très dépendante de la situation actuelle de l'utilisateur. Comme les activités peuvent changer selon le contexte et les éventuelles conditions, le principal

événement qui doit déclencher et contrôler le fonctionnement du moteur d'adaptation est l'événement « *activité en cours* ». La figure 2.5 décrit le modèle de fonctionnement de ce module ainsi que ses différents blocs et spécifie la relation entre ces blocs.

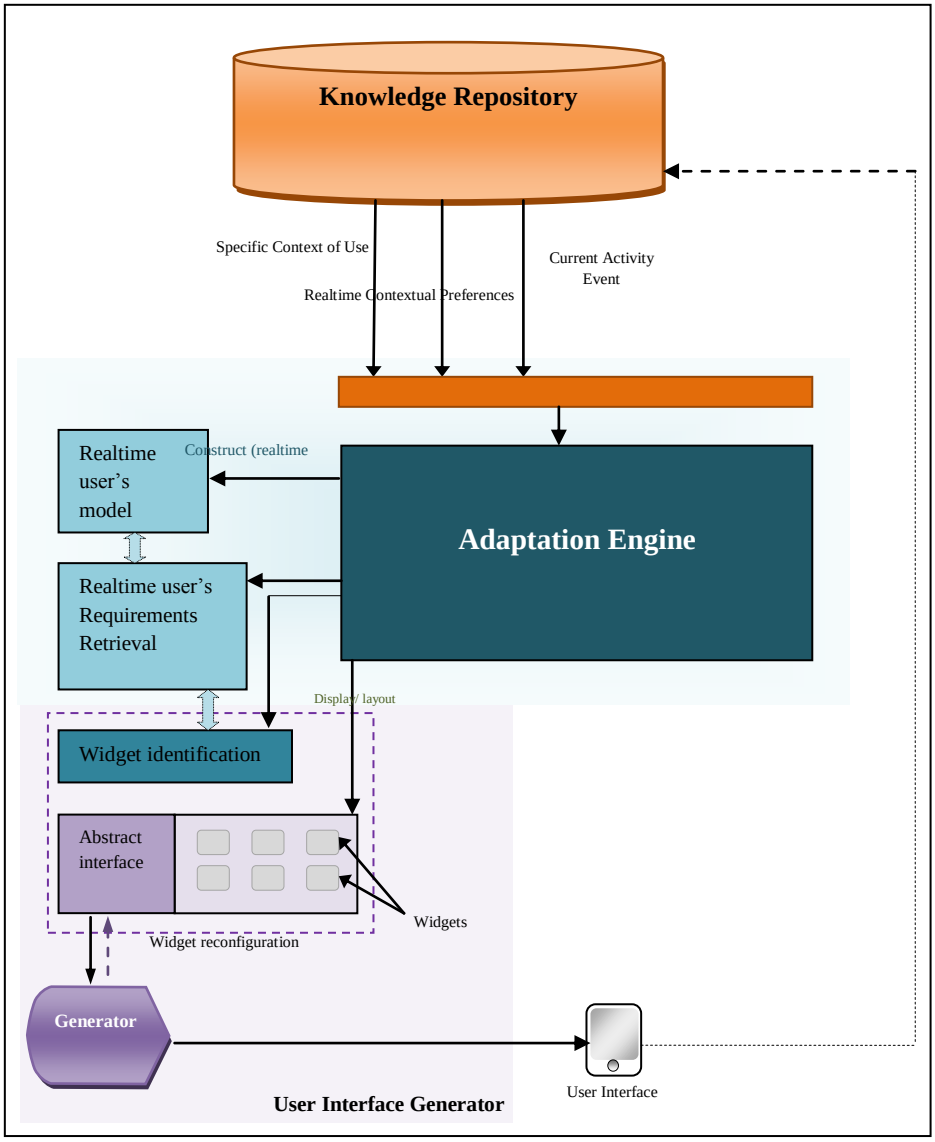


Figure 2.5 Flux de données entrant et sortant du moteur d'adaptation

Le bloc « *Current Context of Use*» est l'intermédiaire entre le composant « *Adaptation Engine* » et l'environnement. Ce composant gère les données contextuelles de façon organisée et stratégique. Il comprend notamment le contexte d'utilisation, l'activité en cours et les informations relatives aux intérêts et préférences spécifiques de chaque utilisateur. A la lumière de cette information, le composant « *Adaptation Engine* » surveille la construction du modèle approprié à l'interaction de l'utilisateur. La composition dynamique du modèle est étroitement liée aux caractéristiques particulières de l'utilisateur. Pour certaines activités, le modèle associé, respectivement les informations fournies, peuvent varier d'un utilisateur à un autre. En outre, pour des activités particulières, le système génère différents modèles en fonction de plusieurs caractéristiques, par exemple, les préférences de l'utilisateur, les intérêts individuels et le

niveau intellectuel. Les contextes actuels d'utilisation ainsi que l'activité correspondante sont vérifiés périodiquement. Chaque fois qu'un changement se produit et est reconnu par le composant « *Adaptation Engine* », il est susceptible de donner lieu à une mise à jour dynamique du modèle d'activité. Elle est suivie par une mise à jour automatique des données connexes, y compris l'historique du dialogue. La mise à jour du modèle est soit d'améliorer la structure du modèle actuel ou bien de reconstruire un modèle approprié. Le passé des interactions et autres sources de connaissances représentées dans les modèles de connaissance gèrent également cette composition dynamique. Grâce à ce processus, le sous-composant « *User Requirements* » maintient en temps-réel des données qui sont étroitement liées au modèle actuel de l'utilisateur. Ces données dépendent implicitement de l'activité actuelle. Le module d'identification des widgets permet de récupérer les widgets les plus adéquats pour répondre aux besoins également il est responsable de la réorganisation automatique de l'interface en déplaçant, supprimant ou ajoutant des widgets. Enfin, le générateur d'interface crée l'interface utilisateur pour la plateforme cible.

4. Gestion et modélisation des paramètres contextuels

Cette section constitue la partie préalable à la présentation de notre stratégie d'adaptation. Elle consiste à modéliser et décrire les données contextuelles sur lesquelles le système doit se baser pour la phase du traitement du contexte. Nous avons utilisé les ontologies pour représenter et traiter ces données, en particulier le sous-langage de OWL à savoir **OWL-Lite.** Les ontologies que nous considérons, pourront évoluer vers un autre sous-langage lorsque le système sera capable de prendre en compte le niveau de formalisation spécifié. Cette section comprend et décrit essentiellement les différentes étapes nécessaires pour développer le modèle d'ontologie proposé représentant le contexte de l'application. Un exemple d'ontologie, relative au domaine d'application étudié (domaine de la médecine : précisément celui de diabète) a été conçu et mis en œuvre afin de valider le modèle proposé et sera détaillé dans le dernier chapitre.

L'utilisation des ontologies offre une grande richesse sémantique et plus de possibilité d'inférences et d'interprétation du contexte par rapport aux autres modèles de contexte. Cependant, leur implémentation reste difficile à réaliser surtout dans les cas d'utilisations de plusieurs ontologies comme dans le cas de SOCAM (Gu, 2005). L'utilisation d'un modèle trop simpliste comme les couples (attribut, valeur) est aussi à éviter car il est source de conflits d'interprétation et de description du contexte actuel de l'application et de l'utilisateur. La gestion d'historique du contexte est aussi un autre critère important dans les systèmes sensibles au contexte. En effet, l'historique permet d'implémenter des algorithmes d'apprentissage pour fournir des services hautement adaptables au contexte. De plus, avec ce genre d'algorithmes, des actions proactives peuvent être automatiquement déclenchées pour un certain nombre de services à l'utilisateur sans qu'il formule une demande explicite. Un autre aspect important dans ces systèmes concerne la gestion de la sécurité et de la confidentialité des données. En effet, des concepts doivent être spécifiés pour définir à qui appartient l'information

contextuelle. CoBrA (Chen, 2004) utilise le langage *Rei* pour définir les politiques de sécurité en termes de droits et d'autorisations d'accès au contexte. Le Context Toolkit (Dey, 1999) implémente le concept d'appartenance du contexte à un utilisateur ou à une entité. Ainsi, une information contextuelle n'est accessible qu'à l'utilisateur ou l'entité à laquelle elle appartient.

4.1 Principales techniques de modélisation des données contextuelles

Afin d'être mieux organisées et interprétées ultérieurement, les données contextuelles captées doivent être représentées et stockées sous une forme appropriée. Autrement dit, un modèle du contexte doit être défini. Le résultat de cette description est un document qui sert de base pour l'étape de l'adaptation. Dans cette partie, nous présentons d'une manière brève les approches de modélisation les plus fréquemment utilisées (Figure 2.6).

4.1.1 Modèle Attributs/Valeur

Ce modèle est considéré comme la technique de modélisation la plus simple dans la littérature. Il était le premier à être utilisé dans les applications sensibles au contexte. Dans ce modèle, les données contextuelles sont représentées sous forme d'un ensemble d'Attributs/Valeurs, où «Attribut» représente généralement le nom du paramètre contextuel et «Valeur», évidemment sa valeur.

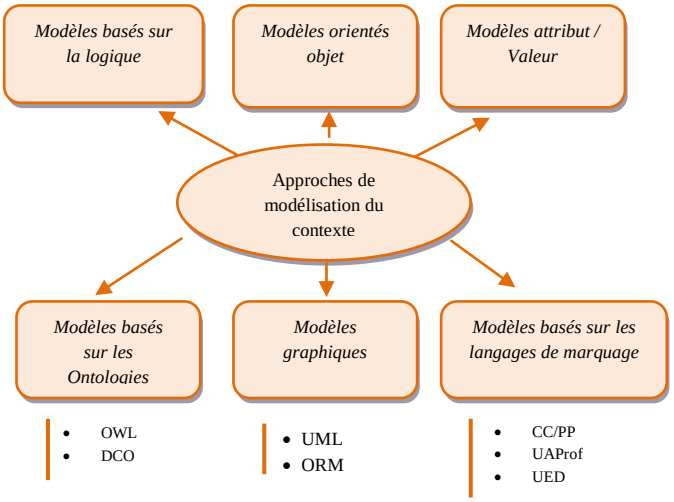


Figure 2.6 Différentes approches de modélisation du contexte

4.1.2 Les modèles de Marquage

Ces modèles ont une structure hiérarchique de données composée de balises et attributs. Il y a plusieurs approches de modélisation du contexte basées sur la technique de balisage. CC/PP est un standard W3C conçu pour la description des caractéristiques du dispositif et les préférences de l'utilisateur (Kiss, 2009), il est basé sur la technique RDF (Resource Description Framework[8]). Généralement, dans ces approches de modélisation,

[8] www.w3.org/RDF/

les auteurs étendent ces modèles avec leurs propres vocabulaires afin de les adapter à leurs objectifs. Nous pouvons citer à titre d'exemple, le modèle Augmented-CC/PP (ACC/PP) (Hanmansetty, 2004) ou CC/PP Context Extension (Indulska, 2003) qui ont ajouté au modèle CC/PP d'autres données contextuelles telles que le rôle d'utilisateur et le comportement des utilisateurs. Comprehensive Structured Context Profiles (CSCP) (Held, 2002) est un exemple qui a donné des solutions pour remédier à quelques lacunes du CC/PP, comme la hiérarchie fixe des données en la rendant plus flexible. UAProf est l'extension la plus importante de CC/PP. Elle est basée également sur la technique RDF et est supportée par l'organisme Open Mobile Alliance[9] (OMA). Ce modèle permet de capturer des données sur les capacités et les caractéristiques des dispositifs mobiles (Figure 2.7). Enfin, le modèle UED (Usage Environment Description) est spécialement basé sur XML-Schema (Vetro, 2005) (Figure 2.8). Il est caractérisé par un vocabulaire riche et simple d'utilisation.

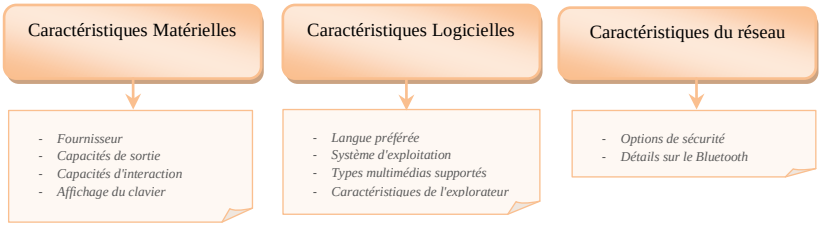


Figure 2.7 Informations décrites avec UAProf

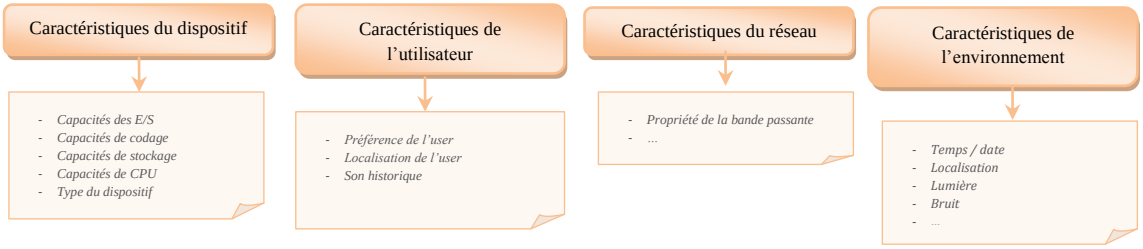


Figure 2.8 Informations décrites avec UED

4.1.3 Modèle basé sur les ontologies

Une ontologie est définie, d'une manière générale, comme une description de concepts et de relations qui peuvent caractériser une entité ou un ensemble d'entités (Briand, 2008). Ce modèle permet une représentation simple des connaissances, et rend facile le partage ainsi que la réutilisation de ces connaissances. Et ce, en particulier au sein d'un système ouvert et dynamique. En se basant sur une ontologie, l'information contextuelle sera décrite par une organisation rigoureuse et hiérarchique regroupant toute entité pertinente pour l'application. Un modèle basé sur une ontologie est défini par un ensemble de termes et d'états décrivant le contexte (Brut, 2008) (Hernandez, 2006). Cette spécification est basée sur les langages du web sémantique. OWL est le langage le plus fréquemment utilisé (Ontology Web Language). Un document OWL est composé

10 http://openmobilealliance.org/

d'ensembles de classes, hiérarchies de classes, de propriétés, les contraintes sur ces éléments et les types de relations permises sur ces entités.

4.1.4 Modèles orienté-objet

Ces modèles (Masmoudi, 2013) exploitent les avantages de la technologie et la conception Orientée-Objet. Ce modèle se base sur les propriétés de l'orienté-objet pour décrire l'information contextuelle telles que l'héritage, l'encapsulation, la réutilisation et le polymorphisme.

4.1.5 Modèles basés sur la logique

Ces modèles de spécification de contexte ont un degré de formalisation très haut (Baldauf, 2007). Ils se basent sur des faits, des expressions et des règles. En effet, l'information contextuelle est représentée d'une manière formelle sous forme de faits. Un avantage de ce modèle est qu'il permet la modélisation et la déduction de nouveaux faits en se basant sur les règles définies. Ainsi, un niveau logique plus élevé sera créé. Exemple de modèles logiques : la formalisation du contexte proposée par McCarthy (McCarthy, 1993, 1997), la théorie de la représentation des situations fournie par Akman and Suray (Akman et al., 1997).

4.1.6 Modèles graphiques

Parmi les modèles graphiques pour la représentation du contexte le langage de modélisation UML (Unified Modeling Language) reste le plus approprié (Kubski, 2013). En réalité, UML dispose d'une forte représentation graphique (Les diagrammes UML) et d'une structure rigoureuse et générique (Baldauf, 2007) qui facilitent la description du contexte. Henricksen et al. (Henricksen, 2003) introduisent un autre modèle graphique pour la représentation du contexte qui est une extension de l'approche ORM (Object-Role Modeling) (Halpin, 2001). Un diagramme ORM représente des objets (Types d'entités), les relations (Types de fait) entre eux, les rôles de ces objets, les contraintes sur ces objets et éventuellement des exemples (appelés tableaux type de fait).

4.2 Synthèse et choix de la technique de modélisation des données contextuelles

En analysant des travaux antérieurs portant sur la sensibilité au contexte, nous avons conclu que la plupart de ces travaux se basent ou bien sur les langages de marquage ou sur les ontologies au niveau de la partie description du contexte. En outre, la majorité des études comparatives faites à ce niveau mettent en évidence ces deux techniques de modélisation de contexte (Strang, 2004) (Timmerer, 2009) (Najar, 2009). En plus, l'applicabilité de ces techniques de modélisation dans des environnements ubiquitaires doit être prise en considération. Par conséquent, nous allons nous contenter de bâtir notre comparaison sur ces deux types (marquage et ontologie) de langage de spécification notamment qu'ils sont des normes et des recommandations W3C. Notre

choix est principalement porté sur ces modèles, considérés comme les plus appropriés pour nos travaux : CC/PP, UAProf et OWL.

En premier lieu, il est indispensable d'identifier des critères de comparaison qui peuvent orienter notre choix. En même temps, nous sommes tenus à étudier l'applicabilité de ces techniques de modélisation dans un environnement ubiquitaire. Parmi les caractéristiques principales d'un environnement ubiquitaire, nous citons le dynamisme et l'ouverture. Nous devons prendre en considération ces deux critères lors du choix du modèle du contexte. En fait, les deux types de modélisation, c'est-à-dire les langages basés sur les ontologies ainsi que les modèles de marquage, permettent d'assurer le dynamisme et l'ouverture (Strang, 2004). Les modèles basés sur les ontologies permettent le traitement et le partage de l'information dans les systèmes ouverts et dynamiques. Les langages basés sur le marquage ont montré une grande applicabilité dans les systèmes et les plateformes ubiquitaires (Strang, 2004). Ces langages permettent une transmission efficace dans un réseau sans fil.

Outre le dynamisme et l'ouverture, un autre critère doit être pris en considération, c'est celui de la mobilité. En réalité, le domaine de l'informatique ubiquitaire est très lié à la technologie mobile et sans fil. Par conséquent, le modèle choisi doit permettre la description des dispositifs mobiles. Ce critère est assuré par les langages de balisages (Timmerer, 2009). En outre, ils permettent la description des préférences de l'utilisateur, les caractéristiques des utilisateurs et d'autres capacités énumérées dans le paragraphe précédent telles que les caractéristiques du réseau. D'autre part, les modèles de marquage ainsi que ceux basés sur l'ontologie sont basés sur la technologie RDF (CC/PP et UAProf sont basés sur RDF, OWL est basé sur RDF/s). A son tour, la plateforme RDF permet la description des capteurs ainsi que les données provenant des capteurs sur les dispositifs mobiles (Korpipaa, 2003).

En plus, l'informatique ubiquitaire est un domaine qui se caractérise par une évolution considérable de logiciels, de matériels (Smartphone, iPhone, PDA très sophistiqué, etc). En conséquent, l'extensibilité du langage de description du contexte sera un critère important à prendre en considération. De même, cette notion est soutenue par les techniques en train d'être étudiées mais avec des niveaux d'extensibilité qui diffèrent d'un modèle à un autre. Les modèles basés sur l'ontologie sont généralement les plus extensibles. La représentation des informations contextuelles complexes est un autre critère qu'un langage de représentation doit mettre en relief. Une autre fois, les langages basés sur l'ontologie se sont avérés les mieux à décrire des données complexes. Dans plusieurs cas, les modèles basés sur l'ontologie montrent être plus adaptés et plus structurés que d'autres. Ces modèles exigent une organisation très rigoureuse et se basent sur une modélisation sémantique du contexte. Par conséquent, ils facilitent le raisonnement et l'interprétation des informations contextuelles.

Finalement, nous pouvons conclure que les ontologies s'avèrent être des techniques prometteuses pour la spécification du contexte, grâce d'une part, à leur niveau élevé de formalité et d'expressivité et d'une autre part à la possibilité d'appliquer des techniques

de raisonnement sur les ontologies (Baldauf, 2007), le tableau 2.2 présente des caractéristiques supplémentaires des modèles décrits précédemment. Une autre fois, les modèles basés sur les ontologies semblent être les plus appropriés candidats pour décrire les données contextuelles relatives à notre travail, en particulier le langage OWL.

Dans un cas général et en tenant compte de l'état actuel du domaine de la sensibilité au contexte, nous considérons que l'utilisation d'une ontologie pour modéliser le contexte est un choix très ambitieux et prometteur. Egalement, l'utilisation des ontologies offre une grande richesse sémantique et plus de possibilité d'inférences et d'interprétation du contexte par rapport aux autres modèles de contexte. En plus, elles permettent de standardiser le vocabulaire, d'uniformiser le langage d'échange entre les différents acteurs, de comparer les différents systèmes, de structurer la connaissance pour simplifier l'analyse et la synthèse des connaissances d'un domaine et de spécifier un contexte (Ikeda, 1998). Dans le contexte d'adaptation d'information à l'utilisateur, l'ontologie est devenue une solution incontournable. L'ontologie permet la construction de modèles de connaissance complexes qu'il est possible d'utiliser pour modéliser à la fois les utilisateurs et le domaine de manière intelligible pour tout type d'utilisations et d'acteurs. Basée sur les outils du Web sémantique, l'adaptation du Web à l'utilisateur devrait permettre le développement d'un Web plus évolué, que l'on peut nommer le Web adaptatif (Patonnier, 2013).

Tableau 2. 2 Comparaison sur les modèles de description des données contextuelles

	Ambiguïté sémantique	Niveau de Formalité	Implémentation concrète	Méthode de retrieval	Traitement des conflits	Applicabilité à des environnements ubiquitaires
Les Ontologies	- -	+ +	Complexe	Raisonnement	+	Très bien
Modèles de marquage (CC/PP en particulier)	-	+	Plus ou moins facile (Dépend de l'extension)	Markup query languages	-	Bien
Modèles basés sur la Logique	+	+ +	Complexe	Inférences	NM[10]	Bien
Modèles orientés objets	+	+	Facile	Algorithme	NM	Plus ou moins
Modèles basés sur des graphes	+	+	Facile	Transformation	NM	Plus ou moins
Modèles attribut/valeur	+ +	- -	Très facile	Recherche linéaire	-	Non

4.3 Gestion des données contextuelles

Le traitement des données contextuelles est une étape d'analyse nécessaire et primordiale. La sortie de cette étape est un document décrivant les informations contextuelles. Il représente les données qui seront utilisées plus tard par la stratégie d'adaptation. Dans l'architecture proposée, ces informations ont été spécifiées par l'ontologie écrite en OWL (Ontology Web Language). En particulier, ces spécifications sont basées sur le sous-langage OWL-DL. Dans cette section, les mesures envisageables dans ce processus seront décrites, ainsi que le fonctionnement du middleware associé.

[10] Non mentionné

Les applications doivent, d'une part, être en mesure de capturer les différents éléments de contexte utiles et intéressants et d'autre part, pouvoir les synthétiser afin d'en déduire un contexte global utile. Le module permet la vérification temps réel des informations contextuelles ainsi que la validation de ces informations par rapport aux besoins de l'utilisateur.

Architecture fonctionnelle de la gestion des données contextuelles

L'architecture représentée par la figure 2.9, prend en charge trois niveaux de gestion des données : l'étape de validation, l'étape d'analyse de dysfonctionnement et la catégorisation des données. Le but est de transformer l'information utile en une connaissance sur l'environnement, les utilisateurs, leurs exigences, etc. Ce but est atteint par le filtrage des données contextuelles à chaque étape afin de déterminer quelles informations doivent être stockées pour une utilisation accrue de la gestion des connaissances.

Le système maintient un modèle de contexte globale basé sur l'ontologie soit une « ontologie générique ». Cette ontologie sera partagée par tous les utilisateurs finaux, en particulier les utilisateurs qui peuvent avoir accès à une partie spécifique de l'ontologie générique selon leurs profils, il s'agit de l'ontologie partielle (PVO[11]). Ainsi, c'est seulement l'information requise, qui comprend des données contextuelles spécifiques notamment des informations sur l'activité en cours, sera fournie. De plus, pour s'assurer que les données sont relativement exactes et mises à jour comparées à l'ontologie générique, le système doit être conscient de la variation de l'exécution des données contextuelles et effectue les mises à jour nécessaires. En outre, le système doit synchroniser périodiquement ou sur demande, les informations mises à jour entre l'ontologie générique et celles partielles (Figure 2.10).

Phase de validation : un environnement ubiquitaire comprend un volume considérable d'informations et, par conséquent, s'affronter à la quasi-totalité de ces informations s'avère une tâche presque impossible. Pour cette raison, il pourrait être utile dans un premier temps, de lancer la numérisation des données ainsi que d'identifier quelles sont les données que le système a captées.

La sortie de cette phase, dite de validation, est censée être un ensemble de données valables et utiles. Le suivi de l'exécution ainsi que la validation des données contextuelles par rapport au contexte courant doivent être effectués. L'objectif est de s'assurer que le système traite en continu des données valides. Cette étape peut être considérée comme une étape d'analyse des données plutôt que d'une analyse ontologique. Par exemple, la valeur reçue <GR>[12] 8g/l est interprétée comme une expression valide. Elle indique que le patient en question a une valeur de GR égale à 8 g/l, bien que cette valeur soit sémantiquement incorrecte.

[11] PVO : Partial View of the Ontology
[12] GR : Glucose Rate (Le niveau du glucose mesuré chez le patient)

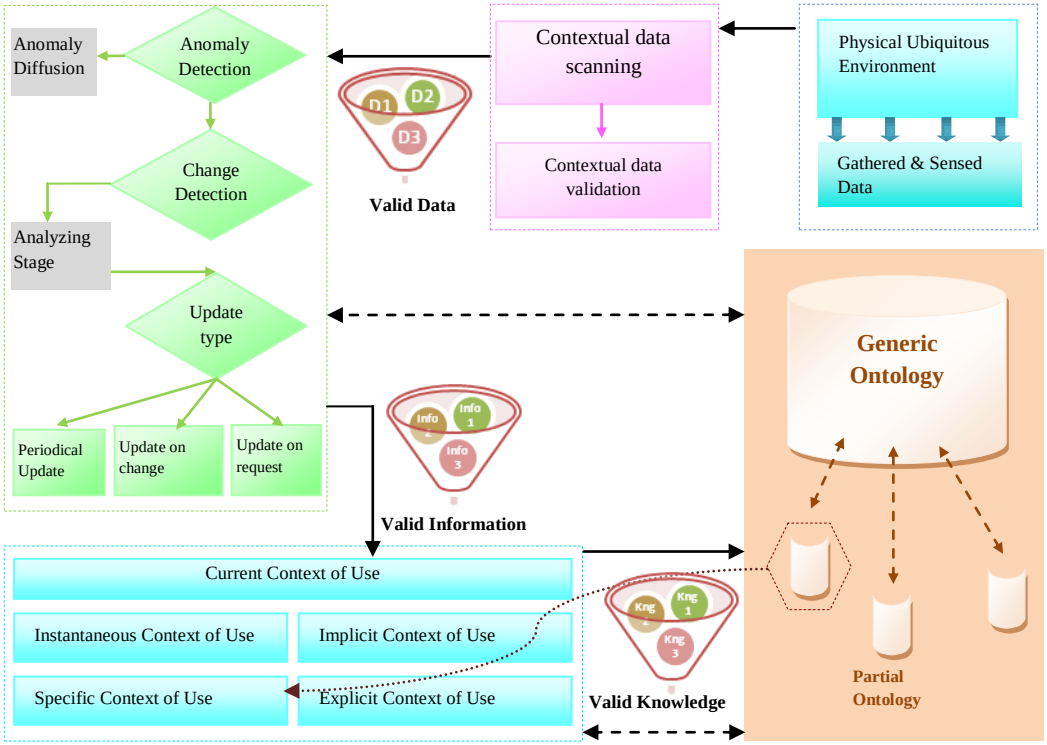


Figure 2.9 Mécanisme pour le traitement et la gestion des données contextuelles

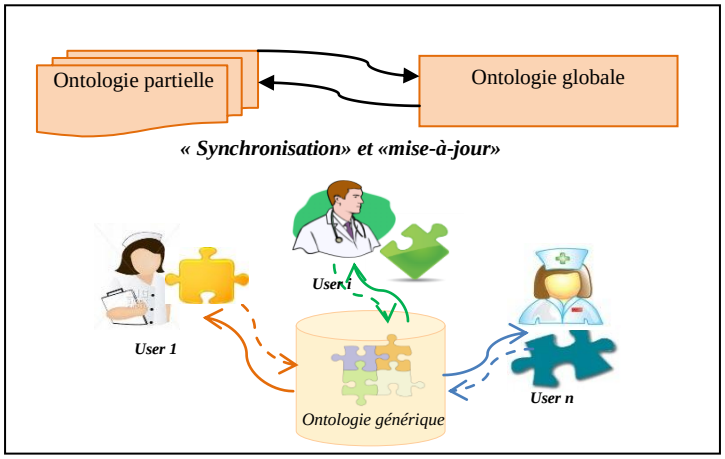


Figure 2.10 Synchronisation et mise-à-jour de l'ontologie

Les informations contextuelles comme par exemple la localisation de l'utilisateur, ses préférences, etc., vont être utilisées pour conduire la décision de l'information à fournir à l'utilisateur au moment approprié. En conséquence, trois procédures de transfert des informations du contexte peuvent être envisagées :

a. Le contexte des informations peut être transmis périodiquement,
b. Chaque fois que l'état actuel de l'utilisateur change,
c. Ou bien une seule fois.

Analyse du dysfonctionnement : Le suivi et le processus d'analyse du changement de contexte se fait à la deuxième étape d'une manière systématique et continue. La première étape de ce processus consiste à détecter les erreurs sur les données reçues. Certes, bien que ces données soient utiles, elles peuvent contenir des erreurs ou des anomalies. Par conséquent, un processus de contrôle dont le but principal consiste à détecter les erreurs possibles est nécessaire. À cette fin, un processus de gestion du dysfonctionnement y compris la détection des données erronées se révèle fondamental. Par exemple les données précédemment reçues sont interprétées comme erronées. En fait, le taux de glucose maximal ne doit pas être supérieur à 7g/l. Ainsi, le système doit être conscient d'une telle erreur afin de prendre les actes correctifs appropriés.

La deuxième étape de ce processus consiste à détecter tout changement pouvant avoir lieu sur les données reçues. Eventuellement, il y a une partie du contexte qui ne change pas fréquemment. Il serait donc inutile de faire continuellement et automatiquement des mises à jour sur la base de données. Par conséquent, trois types de mise à jour des processus ont été identifiés : (i) mettre à jour périodiquement, (ii) mettre à jour sur demande et (iii) mettre à jour suivant le changement. En d'autres termes, les données seront mises à jour en fonction de leurs types. Cet attribut est également variable et est lié à un contexte spécifique d'utilisation. Le tableau 2.2 présente une liste non exhaustive des paramètres contextuels classés selon le type de leurs mises à jour. Pour cette raison, le système dispose de deux types de fichiers de données contextuelles : les fichiers de données statiques et ceux dynamiques. Le premier type contient des données qui ne changent pas souvent, comme le profil de l'utilisateur. Par contre, les fichiers de données dynamiques contiennent des données qui changent périodiquement. Ces informations représentent en particulier les données détectées à partir des capteurs.

Tableau 2. 3 Classification des données contextuelles selon les types de la mise-à-jour

Mise-à-jour périodique	Mise-à-jour selon les changements	Mise-à-jour selon la demande
Position courante Temps *(Exemple date et heure)* Température { *En général les paramètres spécifiques, Exemple. GlucoseRate* }	Activité courante Préferences Profiles Relations Sociales Historique de l'utilisateur	Les objectifs de l'utilisateur Les éventuelles interventions de l'utilisateur (l'utilisateur peut recommander une correction)

L'objectif principal du traitement des données contextuelles est d'organiser toutes les informations requises par le système d'adaptation et mettre ainsi le processus d'adaptation sur la bonne voie. Cette étape a tendance à être peu étudiée par de nombreux chercheurs en ce sens que toutes les données contextuelles seront traitées. Le but ultime de ce middleware proposé se concentre sur le traitement des connaissances judicieuses et non pas uniquement des données simples. En outre, notre approche basée sur le filtrage et la classification des données permet, entre autres, d'obtenir des données significatives et utiles. Il devrait être évident à partir de cette approche que le processus de filtrage est plutôt basé sur un mécanisme de gestion des connaissances. En fait, la première étape

d'analyse des données permet de transformer les données recueillies en des données utiles et valables. La seconde étape, transforme les données utiles et valables en des informations valides.

La troisième étape convertit les informations utiles en des connaissances sur l'environnement ubiquitaire, y compris les utilisateurs. De cette manière, l'analyse des éventuels contenus peut passer par la classification (ou raffinement) suivante : des données, des informations et des connaissances. Les données sont des représentations simples quoiqu'elles n'aient pas de signification. Elles peuvent exister sous une forme quelconque, utilisable ou non. Généralement, une donnée est le résultat direct d'une mesure. Raison pour laquelle une donnée seule ne permet pas de prendre une décision ou d'effectuer un raisonnement. Le raisonnement basé sur ces données est volontairement trop simplifié. Toutefois, une information est une donnée à laquelle un sens et une interprétation ont été donnés. De cette manière, elle permet éventuellement au système de prendre une décision. La connaissance est un assemblage approprié d'informations dans un but d'être utiles et profitables. D'où, cette représentation possède un sens utile et est étroitement liée à la pratique et implique un savoir-faire et la compréhension. La connaissance que possède un système sera considérée comme un résultat de son « expérience », et englobe les normes dont il évalue les nouvelles contributions de ses environs (Davenport, 2000).

Notre idée était de profiter de ce raffinement des données et d'appliquer ses principes à la gestion des données contextuelles. Rappelons notamment notre objectif ultime à travers cette étape : se préparer d'une meilleure façon possible les connaissances consommables par la stratégie de l'adaptation. Néanmoins, l'aspect crucial d'un tel filtrage consiste essentiellement à détecter les erreurs dans les données reçues. Cette étape d'analyse que nous caractérisons par l'analyse de dysfonctionnement fait l'objet de futurs travaux. En fait, nous considérons que l'utilisation d'une méthode inductive afin de rechercher les éventuelles erreurs s'avère indispensable ; par exemple, « Failure Mode » and « Effects Analysis » (FMEA). Nous envisageons ainsi l'utilisation de la méthode déductive pour une analyse plus précise afin de repérer les causes de chaque défaut, par exemple « Fault Tree ».

5. Génération de l'interface utilisateur

Une interface utilisateur est généralement définie par la présence d'un ou plusieurs widget(s). Un « *widget* » est défini à l'origine comme une représentation graphique affichée sur l'interface utilisateur. Techniquement parlant, ces objets seront identifiés de façon dynamique et assemblés à la volée sous la "direction" du moteur d'adaptation. Cette étape supervise la réorganisation en temps réel et la disposition des widgets nécessaires et disponibles. Elle est en relation harmonique avec le module d'identification des widgets. L'objectif est de construire la disposition de l'interface la plus appropriée. Tout un projet est consacré pour atteindre cet objectif, cette partie –génération de l'interface- en représente la troisième phase. Cependant, bien qu'elle constitue une partie indispensable, pour nous cette partie sera succinctement présentée dans ce travail. Toutefois, elle est

envisagée dans une des perspectives des travaux faits. Tous les efforts seront particulièrement consacrés à la stratégie d'adaptation, qui fera l'objet du prochain chapitre.

5.1 Classification du contexte : une étape fondamentale

La dernière partie consiste à classifier le contexte d'utilisation. Le but derrière cette classification consiste à fournir un haut degré de compréhension permettant une reconnaissance et un repérage systématique et facile des informations requises ; si bien que répondre aux besoins spécifiques des utilisateurs, devient beaucoup plus aisée. Cette catégorisation va être examinée afin de déterminer les exigences imposées à la gestion des données et elle peut aider à obtenir sémantiquement les informations pertinentes. Elle peut être considérée également comme une représentation instantanée de l'environnement ubiquitaire, c'est-à-dire l'ensemble des connaissances sur cet environnement. Pour cette raison cette étape revêt une importance considérable en elle-même et représente particulièrement un point fort pour déclencher le processus d'adaptation.

Les quatre composants sont les suivants : le contexte spécifique d'utilisation, le contexte immédiat de l'utilisation, le contexte implicite de l'utilisation et le contexte explicite de l'utilisation. Néanmoins, toutes les informations sont des informations potentiellement utiles.

- Le contexte spécifique d'utilisation est une partie de l'état courant des connaissances qui détermine des données analysées et jugées nécessaires pour une situation spécifique. Il ne peut être consulté que sur une demande,
- Les données *implicites* se sont des variables d'information obtenues sans une intervention directe de l'utilisateur.
- Les données *explicites* : Se sont des variables d'information obtenues lorsqu'une action de l'utilisateur déclenche une interaction.
- Les données instantanées sont des informations déduites à partir de l'activité courante de l'utilisateur. Les commandes incomplètes sont elles aussi transmises à ce module qui seul dispose des connaissances nécessaires pour les compléter ou effectuer un retour adapté vers l'utilisateur.

Les données contextuelles dans les environnements intelligents peuvent comprendre une grande quantité d'information hétérogène, à savoir l'identité de l'utilisateur et celles des gens autour de lui, ses intérêts, son niveau intellectuel, ses préférences, sa profession, son statut social, les dispositifs, la saison, la date, le temps, le niveau de lumière, la température ambiante, etc. En réalité, ces données contextuelles ne peuvent pas être limitées. Il est nécessaire de classer cette information pour mieux répondre aux différents paramètres contextuels et aux exigences variées. Les catégories obtenues sont appelées « *paramètres contextuels* ». Trois entités principales caractérisant un environnement ubiquitaire et intelligent ont été identifiées (Schilit, 1994), (Brown, 1997), (Ryan, 1997), (Schmidt, 1999) : les lieux, les personnes et les objets. Pour décrire

ces entités des taxonomies différentes ont été proposées. Toutefois, quatre catégories principales identifiées par Dey (Dey, 2001b), ont été retenues à savoir l'identité, la localisation, le temps et l'état. Cependant, cette classification ne semble pas être utile pour identifier la pertinence de l'information destinée à être présentée à l'utilisateur. Par conséquent, la catégorisation envisagée vise à mettre en évidence le type de l'information en fonction des besoins de l'utilisateur.

- Les données contextuelles qui peuvent informer l'utilisateur en lui procurant des données utiles sont classifiées des paramètres d'informations (***InfoParam***).
- Les données contextuelles qui peuvent aider l'utilisateur à prendre une décision dans une situation donnée ou d'activités de collaboration sont classifiées des paramètres décisionnels (***DeciParam***).
- Certaines données contextuelles peuvent apparaître inutiles pour l'utilisateur, mais pour une situation donnée, elles s'avèrent indispensables. Ces données sont classifiées des paramètres d'alerte (***AlerParam***).

Chacun de ces types est transmis à l'interface utilisateur à travers des différentes formes en fonction de leurs contenus spécifiques, comme décrit ci-dessous :

• Alergets : (alerte-widgets) le contenu de ces widgets est considéré comme un avertissement pour évoquer un état d'urgence et ainsi demander une intervention directe de l'utilisateur.

• Infogets : (informational-widgets) le contenu de ces widgets est étroitement lié aux besoins de l'utilisateur.

• Decigets : (decisionnels-widgets) le contenu de ces widgets est considéré comme une recommandation assistant à une prise de décision.

5.2 Génération du contenu

A partir de la catégorisation des paramètres contextuels précédemment décrite, le contenu de l'interface assistera l'utilisateur à réaliser ses activités plutôt qu'être une simple information parfois difficile à interpréter. Cette catégorisation pourrait également conduire à améliorer la précision et la concision de l'information à fournir à l'utilisateur. Techniquement parlé, en se basant sur le modèle courant de l'activité de l'utilisateur et en tenant compte des paramètres contextuels identifiés, il sera facile d'en déduire tous les widgets nécessaires à une interface utilisateur utile. Ceci est simplement réalisé en associant les widgets appropriés à chaque paramètre correspondant (Figure 2.11).

Notons que le générateur d'interfaces est implémenté sur une plateforme tierce (Figure 2.12). Après avoir récupéré les données physiques à partir de l'appareil de l'utilisateur, le module « User Interface Generation » génère l'interface puis la transfère sur la plateforme cible[13] désirant l'adapter. Cette méthode permet de créer des interfaces

[13] La plateforme cible ou « la cible » représente l'équipement destiné à recevoir l'interface.

en accord avec les possibilités de la cible et les habitudes de l'utilisateur. Généralement, la génération automatique d'interfaces utilise des langages à balises dérivés de XML[14].

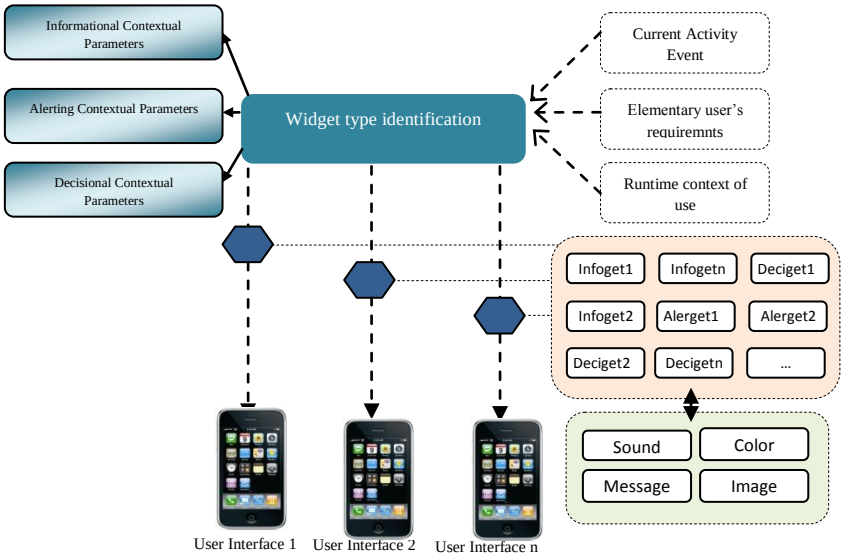


Figure 2.11 Identification du contenu de l'interface en se basant sur le type de widget

UiML[15] (Abrams, 2004) ou XiML[16] (Puerta, 2004) sont deux exemples de langage orienté interface utilisateur basé sur XML. C'est sous cette forme que les fonctions de l'équipement à piloter vont être récupérées par le "générateur d'interface". Il utilisera ensuite la DTD[17] et la feuille de style XSL[18] propre à la cible pour générer l'interface ad hoc. En effet la DTD est un document permettant de décrire un modèle de document XML. Une DTD est une sorte de grammaire de l'application XML. Elle permet de définir quelles sont les balises autorisées, leur ordre et quels éléments elles peuvent contenir. Ainsi les balises qui ne concernent pas la cible seront écartées lors de la génération de l'interface. La feuille de style XSL est quant à elle un fichier qui décrit comment doivent être présentés, affichés ou lus, les documents XML basés sur une même DTD.

Rappelons que le modèle de l'interface est une représentation de l'interface ou d'un aspect de l'interface comportant un certain degré d'abstraction. Il contient les informations qu'on retrouve dans l'interface sous forme graphique, notation ou description de haut niveau. Par contre l'interface finale est typiquement composée du code et de tous les détails permettant son exécution sur une plateforme spécifique. Il s'agit d'utiliser un modèle pour la génération de l'interface utilisateur : nous parlons de la *génération d'interface basée sur les modèles*.

[14] XML: e**X**tensible **M**arkup **L**angage.
[15] UiML: **U**ser **I**nterface **M**arkup **L**anguage.
[16] XiML: e**X**tensible **I**nterface **M**arkup **L**angage.
[17] DTD: **D**ocument **T**ype **D**efinition
[18] XSL: e**X**tensible **S**tylesheet **L**angage

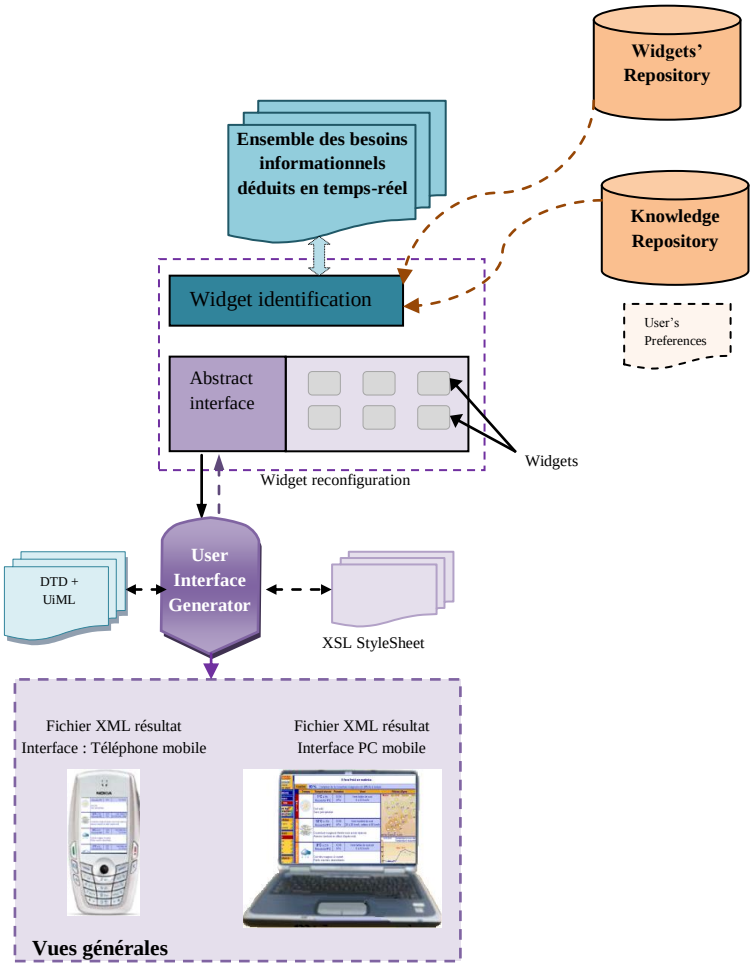


Figure 2.12 Génération de l'interface utilisateur

Finalement, une interface concrète est une forme descriptive, contenant les détails sur la présentation et la disposition des composants de l'interface. L'adaptation peut être effectuée lors de la génération de l'interface finale ou au moment de l'exécution. En particulier l'approche adoptée pour la spécification de la présentation de l'interface est celle des systèmes à base de connaissances permettant de décider des vues et objets graphiques adéquats en fonction des besoins identifiés, en ajoutant des recommandations ergonomiques.

6. Conclusion

Pour des raisons méthodologiques, l'étape de présentation d'un cadre conceptuel de l'approche d'adaptation se révèle indispensable dans la mesure où il faut définir les éléments nécessaires intervenant, mais également d'élucider la structure et le modèle adoptés pour représenter les données contextuelles qui devront être adaptées. Au cours de ce chapitre nous venons de présenter le système d'adaptation. L'objectif spécifique du système est d'améliorer le processus d'adaptation au contexte d'utilisation en gérant dynamiquement des modèles de fonctionnement en temps réel. En particulier, nous avons veillé à souligner l'importance du facteur temps-réel lors du traitement et de la

présentation de l'information à l'utilisateur. En conséquence, notre système d'adaptation peut être situé parmi les systèmes dynamiques et temps-réel (Figure 2.13). L'objectif est d'établir et d'appliquer à la volée, les déductions et les décisions sur les widgets qui seront affichés sur les appareils mobiles et pour un utilisateur particulier.

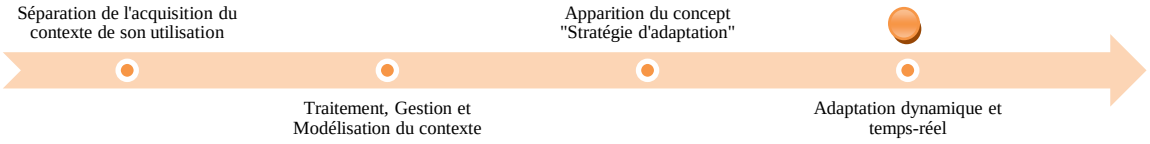


Figure 2.13 Evolution des systèmes d'adaptation en fonction du temps

Il devrait être évident à partir du cadre conceptuel proposé que le système pourrait avoir une représentation adéquate et temps réel de l'utilisateur dans une situation particulière. Par conséquent, il en déduit le modèle approprié ou bien à partir de l'analyse de l'interaction de l'utilisateur-système et le contexte courant d'utilisation construire le modèle idoine.

L'introduction de la déduction des besoins utilisateur comme un paramètre à part entière dans la conception d'un système ubiquitaire est considéré l'élément le plus novateur. En conséquence, le premier challenge de développement d'un tel système est de le doter d'une certaine capacité pour supporter ce dynamisme, ce phénomène a été appelé ***mutation des modèles*** (Ismail, 2012). Ceci nous amène à discuter dans le chapitre suivant l'approche proposée et adoptée pour définir la stratégie d'adaptation régissant le comportement du système.

Chapitre 3

APPROCHE D'ADAPTATION DU CONTENU INFORMATIONNEL D'UNE INTERFACE UTILISATEUR

1. Introduction

L'interaction homme-système dans les environnements ubiquitaires et intelligents revêt aujourd'hui une préoccupation majeure de ce domaine, une nouvelle forme d'interaction émerge : l'**interaction ubiquitaire** (Schlegel, 2011) (Ismail, 2012b) (Caon, 2013). Un système qui réagit en temps-réel comme prévu par l'utilisateur final contribue énormément à la fiabilité et promeut la finalité de cette interaction (Tran, 2009). Un des défis émanant de ce type d'interaction et que nous avons particulièrement identifié en analysant l'état de l'art est l'adaptation de l'information au contexte d'usage d'un utilisateur évoluant dans un tel environnement. Autrement exprimé, il s'agit de savoir au bon moment l'information la plus appropriée pour l'utilisateur pour qu'il mène à bien son activité courante. Jusqu'à aujourd'hui il n'y a pas encore unanimité quant à une stratégie de fonctionnement idéale pour les systèmes adaptatifs. Néanmoins, il est admis qu'une stratégie d'adaptation est un ensemble d'actions entrepris par le système pour répondre à un événement donné. Autrement dit, il s'agit d'associer à chaque événement au moins une stratégie de traitement. En particulier, il s'agit d'extraire du contexte les informations nécessaires pour créer et mettre à jour un modèle courant de l'utilisateur et de son environnement. Ce chapitre présente et discute notre approche proposée pour faire face à ce défi. Il comporte en conséquence quatre sections principales. La clarté de la présentation de la stratégie nous conduit à regrouper, dans la première et la deuxième section, les éléments relevant de notre démarche pour aborder le sujet et le cadre de cette démarche. La présentation du modèle conceptuel et fonctionnel de la stratégie fait l'objet de la troisième section. La quatrième section détaille davantage les algorithmes de fonctionnement.

2. Cadre général et positionnement de l'approche

Le but de cette première section consiste à concevoir la mise en cohérence entre les objectifs de notre recherche et la démarche que nous mettons en œuvre pour confronter notre problématique.

2.1 Objectifs

Pour être doté du critère ubiquitaire, un système doit en permanence maintenir un modèle de l'environnement, de ses utilisateurs et de leurs activités (Brdiczka, 2005). D'une autre part, le comportement de l'utilisateur évolue au cours du temps et son activité dépend de la situation courante (Suchman, 1987) et ne suit pas nécessairement un plan. Des nouvelles activités apparaissent pendant que d'autres disparaissent (Brdiczka, 2005). Afin de s'adapter et de proposer des réponses appropriées à ces changements, nous nous intéressons particulièrement aux changements engendrés par l'activité de l'utilisateur dans un environnement ubiquitaire. Par conséquent, afin de répondre de manière appropriée à l'activité de l'utilisateur, une stratégie d'adaptation dynamique

s'avère un besoin essentiel pour la mise en œuvre d'une interaction homme-machine satisfaisante dans un environnement ubiquitaire. En guise de préliminaires, il nous semble opportun de présenter une vue d'ensemble contextualisée et synthétique sur la notion d'adaptation.

2.2 Adaptation au contexte d'utilisation : étude synthétique

Le terme « adaptation » est utilisé dans différents domaines avec des sémantiques légèrement variées. Nous pouvons citer à titre d'exemple son application dans le domaine de l'ingénierie logicielle, l'interaction homme-machine (IHM) et l'intelligence artificielle. Afin d'éviter les ambiguïtés, nous allons définir dans cette section ce que nous entendons par adaptation et ses objectifs. L'adaptation peut être définie comme une caractéristique exprimée au niveau d'un système – pour notre cas il s'agit d'un système informatique – qui reflète sa capacité à se modifier structurellement en réaction à certains évènements bien identifiés (Andresen, 2005). L'adaptation dépend de différents facteurs : le type d'adaptation, l'objectif de l'adaptation et les données sur la base desquelles l'interface utilisateur s'adapte. Le processus d'adaptation peut être situé dans l'équipement mobile de l'utilisateur, ou bien lors de la génération de l'interface finale ou au moment de l'exécution.

2.2.1 Taxonomies fonctionnelles de l'adaptation

Trois grands types d'interfaces utilisateur pour les équipements mobiles ou "nomades" se dégagent de l'ensemble de ce qui existe. Les interfaces préprogrammées ou préconçues, les interfaces dynamiquement calculées et les interfaces hybrides. Pour les interfaces pré-calculées, l'adaptation est entièrement calculée à la conception et se rapporte à une cible spécifique. Elle est appliquée à l'exécution. Ces interfaces sont rigides et peu ou pas adaptables aux réels besoins ou habitudes de l'utilisateur. Quant aux interfaces dynamiquement calculées, l'adaptation est réalisée à l'exécution en fonction des attributs de la cible (mais ces attributs sont figés et pris en compte lors de la conception). Finalement, pour les interfaces hybrides il s'agit de combiner les deux méthodes dans le sens où une cible est reconnue (PDA par exemple), dont l'adaptation va être affinée en fonction de ses caractéristiques propres (écran couleur par exemple). Pareillement, trois grandes classes d'adaptation de systèmes interactifs en général, ou d'interfaces utilisateur en particulier peuvent être distinguées. Nous parlerons d'interface adaptée. L'adaptation ici vise des critères d'ergonomie pour lesquelles elle a été conçue. Cette adaptation considère uniquement un type d'utilisateur bien déterminé, une tâche spécifique et un contexte bien déterminé. Nous parlons également d'interface adaptable. A ce niveau l'intervention humaine est nécessaire pour enclencher le processus de modification. Autrement dit, ce type d'interface répond aux demandes de personnalisation ou de paramétrage de l'utilisateur. Finalement, si l'interface s'adapte automatiquement à l'utilisateur, à ses gouts, ses habitudes, etc., nous parlons d'interfaces adaptatives ou auto-adaptatives (Hocine, 2005).

2.2.2 Présentation de l'approche adoptée : Approche à base de modèles

Dans cette partie, nous n'avons pas l'intention de lister toutes les techniques d'adaptation, mais plutôt de passer en revue différentes formes d'adaptation. Le but est de mettre en œuvre les éléments récurrents dans les travaux antérieurs en vue de positionner notre stratégie.

Durant les années quatre-vingt, la conception de systèmes et interfaces adaptés à l'utilisateur a été réalisée de façon à ce que l'accomplissement de sa tâche soit optimal. Bastien et Scapin (Bastien, 2001) en représentent des références ; cependant leur objectif vise uniquement l'ergonomie de l'interface adaptée. Egalement, l'adaptation est souvent dédiée à un type d'utilisateur donné ; les utilisateurs non ciblés se sont retrouvés doublement handicapés par l'interface proposée et par leurs limites sensorielles et intellectuelles (Brdiczka, 2005). Cette limite devient plus complexe lorsque nous parlons des utilisateurs mobiles. Il faut non seulement créer des interfaces qui s'adaptent à l'utilisateur mais également prendre en compte le contexte d'interaction.

Au début, l'adaptation était incarnée dans différents dispositifs informatiques mais nous ne rendons pas compte de sa présence. Comme en témoigne particulièrement les navigateurs web (Stephanidis, 2001), ainsi que les systèmes d'information basés sur l'internet et fonctionnant sur le Web (Stephanidis, 2004). Il en va de même pour les communautés virtuelles (Antona, 2006) et les jeux électroniques qui ont beaucoup bénéficié de ce paradigme (Graamenos, 2007) (Hocine, 2011). Dans le cadre proprement dit de l'adaptation de l'interface utilisateur (IU), certaines approches ont été développées. Les premières formes d'adaptation considèrent le concept d'accessibilité et de facilité d'utilisation. Cette adaptation intéresse spécifiquement les personnes handicapées ou les personnes âgées. L'objectif est de faciliter l'accès aux données pour les utilisateurs handicapés (Savidis, 2004). Par exemple un utilisateur malentendant aura grandement besoin d'un discours de conversation sous forme de texte, et *vice versa* un utilisateur présentant des problèmes de vision préfèrera plutôt une conversion en parole d'un discours sous forme textuelle.

Avec l'apparition des interfaces mobiles qui sont généralement dotés de capacités restrictives, certaines approches ont été développées dans le cadre d'adapter ces interfaces à leurs contraintes physiques (Leondis, 2012). Ces contraintes prises en compte sont spécialement les capacités de mémoire, les tailles d'écran ainsi que la résolution des écrans (Ye, 2005). L'objectif visé est de minimiser le décalage entre la présentation d'une interface utilisateur et les capacités d'un dispositif pour la présenter. A l'instar de la notion de plasticité, cette technique permet la migration dynamique des interfaces utilisateurs entre les différents dispositifs hétérogènes (Calvary, 2005) (Coutaz, 2004). L'adaptation a pris de l'ampleur avec (Thevenin, 2001) : il s'agit de satisfaire l'utilisateur dans sa situation d'interaction. Nous rappelons que la situation d'interaction est le triplet <utilisateur, utilisation, contexte>. Concernant l'adaptation au profil et aux préférences de l'utilisateur, elle s'est basée simplement sur les données fournies par le dispositif. Il

s'agit également d'offrir aux utilisateurs nomades un accès à l'information ambiante en fonction de leurs dispositifs. Cette adaptation dépend nécessairement des caractéristiques du dispositif d'accès au moment de connexion (Ramos, 2005).

La réalité aujourd'hui est malheureusement autre, notamment avec l'apparition des environnements intelligents et ubiquitaires. Les interfaces utilisateurs ont tendance à s'approprier à de nouvelles formes d'interaction. En alliance avec l'intelligence artificielle, l'adaptation des interfaces utilisateurs a connu un essor considérable allant des systèmes à bases des règles aux systèmes se basant sur les réseaux Bayésiens (Alvarez, 2007) (Tomlinson, 2007). Les techniques utilisées impliquent l'apprentissage du comportement des utilisateurs dans l'adaptation en se référant à des déductions faites en utilisant les connaissances acquises par le suivi des utilisateurs.

Plusieurs familles de supports sont proposées pour adapter un système au changement du contexte d'utilisation. (Samaan, 2006) a proposé une étude détaillée et a regroupé ces approches en quatre classes d'approches d'adaptation, à savoir la traduction des interfaces homme-machine, la réingénierie et la migration des applications, l'utilisation de langages abstraits de description des interfaces et enfin les approches à base de modèles. La conversion et la réingénierie, s'insèrent dans une logique ascendante d'adaptation. Elles partent des interfaces déjà existantes pour un contexte spécifique pour en retirer de nouvelles interfaces. Concernant la conversion, il s'agit d'une solution qui se base sur l'idée de la réutilisation des interfaces produites pour un contexte spécifique et de les traduire en vue de leur utilisation dans d'autres contextes. Ces solutions sont employées largement dans la traduction des pages web (notamment celles écrites en HTML) vers d'autres langages à balises (XSL, XHTML, XML, etc). Quant à la réingénierie des interfaces c'est une technique qui consiste à analyser un produit pour déterminer la manière dont celui-ci a été conçu et identifier ses composants et leurs dépendances (Müller, 2000). Une telle méthode rend possible de produire plusieurs interfaces pour plusieurs contextes d'utilisation à partir de spécifications obtenues dans la phase d'abstraction.

Les deux dernières approches font partie des démarches descendantes de construction des interfaces. Elles partent d'une spécification du système (à différents niveaux d'abstraction), pour en extraire l'interface de l'application. Pour les approches basées sur les langages à balises, ce sont particulièrement des techniques qui rendent possible la séparation du contenu d'un document de sa forme, mettant ainsi en œuvre les principes architecturaux basiques défendus par la communauté des chercheurs en Interaction Homme-Machine (Samaan, 2006). Cette séparation a fait l'objet d'une première technique d'adaptation à travers la possibilité de recourir à des feuilles de style utilisées pour l'externalisation de la définition de la présentation d'un document (les couleurs, les polices, le rendu et d'autres caractéristiques) ; tout d'abord avec les CSS (Cascading Style Sheets) (Lie, 1999) associées à HTML puis avec XSL et les langages associés, proposés autour de XML. Dès lors, la création de différentes feuilles de style (qui contiennent les spécifications des adaptations souhaitées) conduit à la production de

différents documents et cela pour le même contenu. Par contre, les a*pproches à base de modèles* sont des approches qui s'appuient sur un ensemble de modèles (tâches, domaine, présentation, dialogue, etc.) pour la conception et la génération des systèmes interactifs. Dans une démarche descendante, ces approches procèdent à une concrétisation progressive des spécifications de modèles pour la génération de l'interface finale du système interactif. L'idée de ces approches est de prendre en considération, dans le processus d'adaptation au contexte d'utilisation, l'ensemble des modèles sous-jacents à un système interactif et de ne pas se limiter à l'adaptation du modèle de la présentation.

Dans ce cadre, l'approche à base de modèle s'avère la plus prometteuse. Certes ces approches ont été largement employées pour la conception de systèmes interactifs sensibles au contexte d'utilisation (Van den Bergh, 2004). La conception et le développement d'interface utilisateur basées sur les modèles ont suscité beaucoup d'intérêts ces dernières décennies, comme l'avait décrit Szekely (Szekely, 1996) "model-based user interface development tools trace their roots to work on user interface management systems done in the early 1980's." En effet, depuis les années 90, un intérêt progressif autour des approches de conception à base de modèle (Model Based Design/Development) a été remarqué, s'accélérant suite à la proposition de l'approche MDA (Model Driven Architecture) par l'OMG. Cette classe d'approche implique forcément le recours aux modèles (procédé, tâche, interaction, etc.) dans le processus de conception. A ce niveau, le développement d'interfaces utilisateur est constitué d'un ensemble de paradigmes pour la construction des interfaces. En fait, les développeurs n'ont plus recours aux fonctions prédéfinies de bibliothèques d'outils pour la programmation d'interfaces, mais plutôt ils ont à rédiger une spécification au moyen d'un langage de haut niveau, tels des diagrammes de transition d'état, des grammaires ou bien des représentations basées sur des événements. Cette spécification sera automatiquement traduite en un programme exécutable. La spécification peut également être interprétée de façon à générer automatiquement l'interface utilisateur.

2.3 Positionnement de l'approche

Une des premières définitions stipulant d'un modèle est : « *la description d'un modèle doit représenter précisément la solution à un problème* » [19]. Ainsi, le modèle montre qu'il propose une solution à un problème spécifique d'utilisabilité [20] dans un contexte donné. Le second reproche qui pourrait être fait aux modèles c'est d'être figés et une fois la description du modèle effectuée son utilisation reste limitée à une application précise dont le cadre est défini par le concepteur. L'utilisateur de l'interface ainsi conçue doit en accepter les règles et les modes de fonctionnements pour lesquels elle a été conçue.

[19] La première définition déclarée dans un de workshop de la conférence CHI 2000.

[20] La norme ISO 9241 la définit l'utilisabilité comme « le degré selon lequel un produit peut être utilisé, par des utilisateurs identifiés, pour atteindre des buts définis avec efficacité, efficience et satisfaction, dans un contexte d'utilisation spécifié ». Les trois critères de l'utilisabilité soulignés par cette définition sont l'efficacité, l'efficience et la satisfaction.

Pour pouvoir aller plus loin dans l'adaptation des interfaces dans des environnements dynamiques et ubiquitaires, partir d'un modèle de description même générique et plus abstrait de l'interface n'est plus satisfaisant. L'objectif est de créer une description dynamique facilement utilisable dans des environnements hétérogènes. Notre approche s'inscrit dans ce contexte et se base sur les modèles évolutifs et applique les réseaux de Petri pour modéliser l'interaction utilisateur-système. Ces modèles peuvent être créés, tout en intégrant progressivement une série de modèles élémentaires ou subir des modifications et des changements à la suite d'interactions avec l'utilisateur et à travers des réinterprétations de modèles existants stockés par l'acquisition de connaissances qui précède. Ce principe a été appelé mutation des modèles.

L'approche proposée tire profit de l'ontologie de service écrit en OWL-S pour décrire l'aspect dynamique des modèles basés sur les réseaux de Petri, en particulier la composition en temps réel et automatique de ces modèles. En bref, un modèle élémentaire est formulé en utilisant un processus OWL-S atomique. Puis progressivement un ensemble d'actions élémentaires sera composé afin de construire l'ensemble du modèle RdP. En résumé, Bien que l'adaptation physique attire beaucoup de chercheurs dans ce domaine, l'adaptation du contenu constitue certainement l'un des mécanismes centraux dans la mise en œuvre effective des systèmes ubiquitaires. La figure 3.1 permet d'élucider notre position de recherche et tenter de situer notre approche de l'adaptation tout en encadrant la problématique de la relation entre ces deux concepts. Il s'agit d'apparenter notre étude à l'intersection de deux domaines : Interface Homme-Machine (IHM) et Adaptation.

3. Approche proposée

Selon l'étude précédente, nous constatons que le domaine d'étude de ***l'Adaptation*** s'avère très vaste. Nous avons donc choisi de nous intéresser au rôle clé de **l'architecture logicielle** qui supporte l'adaptation. Ainsi dans une **approche à base de modèles**, nous proposons d'utiliser un **modèle dit d'interaction** comme modèle clé dans le processus de construction et d'adaptation d'application interactive. Ce modèle donne une vision globale sur l'ensemble des composants des applications. Le modèle d'interaction d'une part, gère la façon dont l'utilisateur interagit avec le système, et d'autre part, il assure le lien avec le moteur d'adaptation autrement dit le noyau fonctionnel du système. Ce modèle a été souvent négligé dans les démarches d'adaptation (Samaan, 2006) (Bazargan, 2005). Ce modèle est basé sur un découpage explicite en plusieurs modèles, repris dans de nombreux autres systèmes. Nous pouvons citer tout d'abord le modèle de domaine qui précise les fonctions et concepts utilisés dans le noyau fonctionnel et décrit comment le domaine d'application est structuré. Egalement, le modèle de l'utilisateur qui porte sur les caractéristiques des utilisateurs ciblés ; ces modèles ont été particulièrement exploités dans les systèmes d'aide intelligents et les interfaces adaptatives.

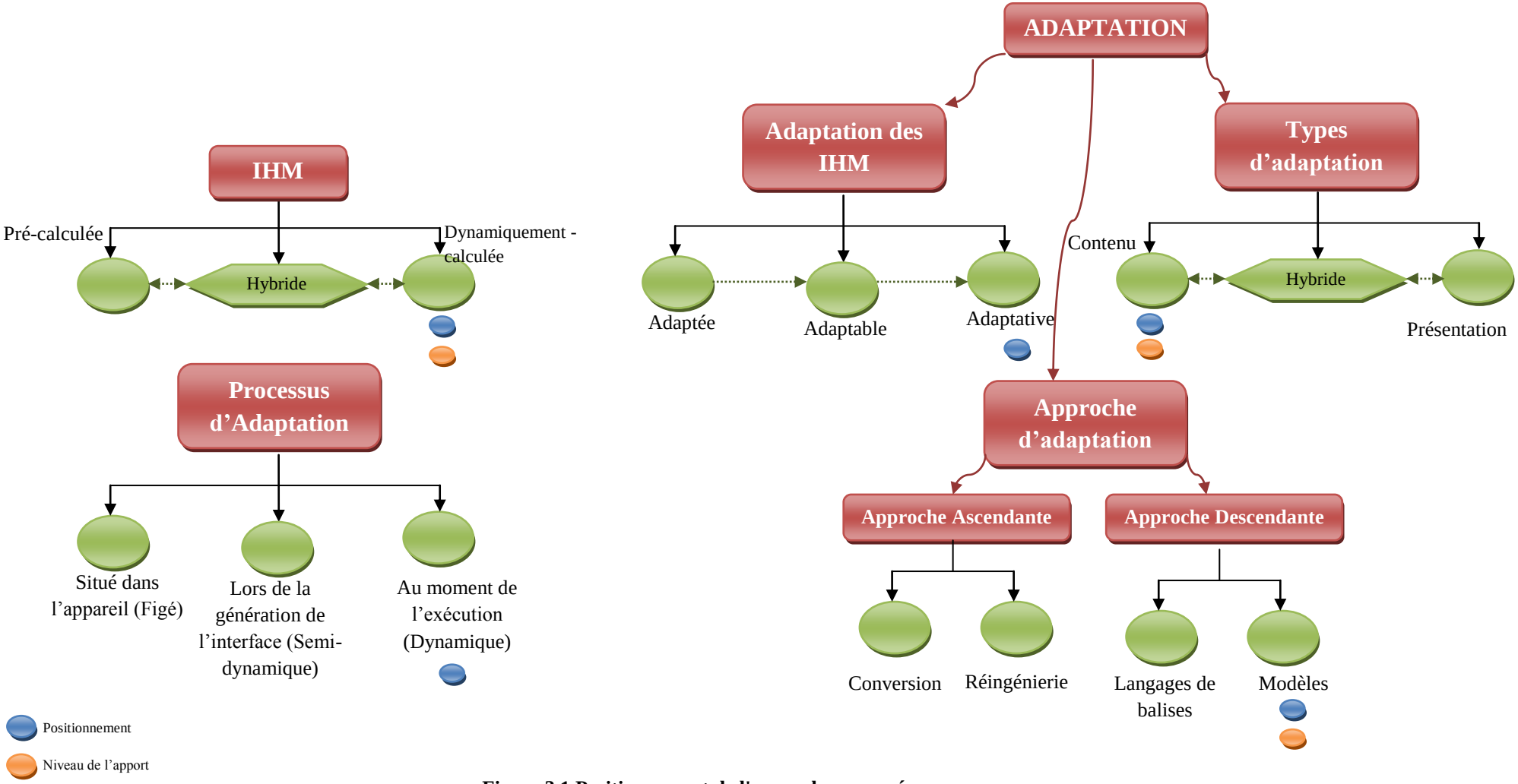


Figure 3.1 Positionnement de l'approche proposée

Le modèle des tâches est un modèle basé sur l'analyse de l'activité, il spécifie les tâches que l'utilisateur désire accomplir. Le modèle de dialogue : détermine la structure du dialogue entre l'utilisateur et le système. Il découle du modèle des tâches et des modèles du domaine. Le modèle d'application quant à lui porte sur la sémantique du noyau fonctionnel. Enfin, le modèle de présentation montre le niveau interaction physique du modèle d'architecture, soit la partie perceptible de l'interface. Outre ces modèles pré-mentionnés, nous pouvons citer le modèle d'environnement qui concerne le contexte d'utilisation de façon générale, ainsi que le modèle de la plate-forme qui concerne les caractéristiques du dispositif de restitution visé. Finalement, un autre modèle a été défini afin de spécifier les règles d'ergonomie utilisées lors de la génération ou de l'évaluation, il s'agit du modèle de l'ergonomie. Le modèle de l'utilisateur, le modèle des tâches, le modèle d'environnement et le modèle de la plate-forme sont considérés généralement comme des modèles utilisables pour reconnaître une situation d'adaptation considérée.

3.1 Présentation de l'approche proposée

Un modèle doit être créé et gardé à jour pour représenter la connaissance, les intérêts, les préférences, les buts et objectifs, l'historique des actions, le type, le style et autres propriétés qui pourraient être utiles pour l'adaptation. Un modèle de tâche fournit une description sémantique des objets qui seront manipulés pendant l'interaction en prenant en compte les relations conceptuelles entre ceux-ci. Ainsi, ce modèle doit veiller à fournir souvent des informations sur les objectifs de l'utilisateur par l'utilisation de la même structure conceptuelle (Samaan, 2006) (Bazargan, 2005). Le modèle d'adaptation contient des mécanismes d'adaptation de contenus et de liens en se basant sur le modèle utilisateur et le modèle du domaine. Il peut aussi mettre à jour le modèle de l'utilisateur en fonction du comportement de ce dernier. Dans cet axe, Nous préconisons plutôt une approche dynamique fondée sur des modèles de fonctionnement évolutifs construits et modifiés dynamiquement à l'exécution. Nous plaidons en faveur de la création de ce modèle à l'exécution et à sa modification selon le changement du contexte. La construction de modèles pertinents et efficaces qui permettent une adaptation dynamique et temps-réel du comportement de l'interface portant essentiellement sur le contenu. L'acquisition et l'adaptation d'un modèle existant de haut niveau reflétant le comportement d'un utilisateur est déjà un problème complexe (Brdiczka, 2005). Si nous remplaçons cette adaptation d'un modèle qui sera créé en temps-réel, le problème devient vraiment très difficile. Autrement, cette problématique de voir le moment d'adaptation a le mérite de distinguer l'instant de son calcul (à la conception ou à l'exécution) de celui de son identification (toujours à la conception) il n'y a pas de méthode d'adaptations à des situations complètement nouvelles. A ce niveau, l'objectif est de proposer une approche pour la modélisation du comportement de l'utilisateur tout en tenant compte de son contexte d'utilisation, en particulier son activité courante.

Déduction des besoins utilisateur en termes d'information en temps-réel selon son activité en cours : ces besoins informationnels changent si l'activité change.

Le travail présenté dans cette partie s'inscrit dans cette lignée et représente une évolution de la méthode proposée dans les travaux précédents de l'équipe (Moussa, 2002) (Riahi, 2001) dans un contexte de spécification d'interaction Homme-Machine pour les applications de contrôle de processus interactifs. Initialement, l'objectif de cette contribution consistait à intégrer cette approche de modélisation dans le domaine ubiquitaire (Ismail, 2010). Autrement dit, il s'agissait de rendre accessible, réutilisable et partagé ce processus de modélisation. Cette approche consiste en une modélisation formelle du comportement du système. Elle permet de déduire les besoins utilisateurs et d'identifier ensuite les *Widgets* nécessaires pour la spécification et la génération d'une interface utilisateur. Cette méthode pourrait, au terme de cette étude donner lieu à un nouvel axe de recherche portant sur la spécification temps réel de l'interface utilisateur, notamment celle mobile. Cette innovation est fortement dirigée par la nature des environnements ubiquitaires. Nous promouvons ainsi les modèles existants par le biais de l'exécution dynamique des RdP temps-réel que nous avons, *a priori*, appelée la **dynamicité des réseaux de Petri Temps-réel**. Le cadre conceptuel et contextuel de cette approche avait particulièrement recours à l'ontologie des services web sémantiques tout en élargissant le rôle d'une action élémentaire représentée par un réseau de Petri à un service web atomique et autonome. Ainsi la dynamicité des RdP s'incarne par le concept fondamental des services Web sémantiques qui est la composition dynamique des services web (Ismail, 2011). Le reste de ce chapitre essayera de débroussailler les différentes facettes de cette approche, mais dans une première étape il s'agit d'examiner ses notions de base.

3.2 Identification temps-réel des besoins utilisateur

Comme nous l'avons précédemment précisé, il s'agit ici du plus important maillon dans la chaîne du processus d'adaptation. L'étape de déduction des besoins de l'utilisateur consiste à associer au niveau de chaque état les paramètres nécessaires (associés à chaque besoin) et avec lesquels il pourrait accomplir sa tâche. L'objectif ultime est de déduire ces besoins à chaque état du fonctionnement du système.

3.2.1 Modélisation du comportement de l'utilisateur avec les RdPIs

Les Réseaux de Petri (RdP) représentent un modèle formel et abstrait pour spécifier et modéliser un flot de données. Il a été proposé par Carl Adam en 1962 (Petri, 1962). Depuis, les réseaux de Petri ont constitué une efficace et rigoureuse méthode de modélisation et de description du comportement dynamique de systèmes divers. En effet, cette technique est dédiée essentiellement à la description des systèmes dynamiques, distribués ainsi que la modélisation du parallélisme et de la synchronisation dans de tels systèmes. La raison essentielle derrière ce choix est le fait que ces modèles peuvent profitablement décrire des aspects de concurrence, parallélisme et dynamisme qui

représentent des caractéristiques fondamentales de l'environnement ubiquitaire (Brdiczka, 2005). En outre, les Réseaux de Petri représentent, en particulier, un outil approprié pour la modélisation de l'interaction Homme-Système (Matej, 2010). La modélisation avec les réseaux de Petri évolue au fil des années, notamment avec la modélisation des interfaces Homme-Machine (Abed, 2001). Au début, c'est uniquement la description des tâches qui était basée sur les RdP mais plus tard, ces derniers deviennent un outil approprié pour la modélisation du dialogue Homme-Machine. Nous pouvons citer les travaux de Willem et Biljon (Willem, 1988) les premiers à avoir utilisé les RdP pour modéliser le dialogue H-M, de même que ceux de Palanque et Bastide (Palanque, 1996). Avec l'émergence des réseaux de Petri de haut niveau (Decker, 2011), l'étude d'un large éventail d'applications devient plus aisée. Ces nouveaux modèles ont été principalement basés sur les concepts de base des réseaux de Petri, le modèle initial a été enrichi par des extensions et une variante de modèles surgit. Parmi ces modèles nous citons : les Réseaux de Petri colorés, les Réseaux de Petri synchronisés, les RdP stochastiques et RdP interprétés. Les RdP proposés ici sont les RdPI (Réseaux de Petri interprétés) (Moalla, 1985).

Les Réseaux de Petri interprétés : Certaines de ces extensions consistent à introduire la prise en compte du facteur temps, la synchronisation des évolutions des marquages sur les occurrences d'évènements et la superposition aux RdP d'une interprétation (traitements ou « Actions » associés aux places, conditions ou « réceptivités » associées aux transitions). Plusieurs modèles, réunissant plus ou moins ces trois extensions, ont été définis sous le sigle RdPI (Réseaux de Petri Interprétés). Ce type de réseaux introduit les notions d'événements et de conditions ainsi que la notion d'actions. En effet, à chaque transition Tj d'un RdPI on associe une condition de passage (Cj), un événement de déclenchement (Evj) et une action éventuelle (Aj) : (Figure 3.2).

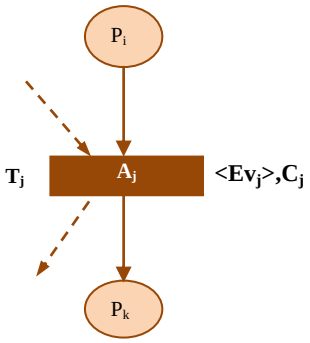


Figure 3.2 Réseau de Petri interprété

Pour modéliser l'interaction homme-machine moyennant les $RdPI_S$, nous avons convenu d'utiliser les places pour représenter « l'état du comportement de l'utilisateur » en fonction de l'évolution du système, plus précisément en fonction de l'évolution de l'environnement ubiquitaire et de ses changements. Il convient de noter ici que toute modification de l'état de l'environnement peut affecter l'affichage au niveau de l'interface utilisateur. En d'autres termes, cette modification peut impliquer des

changements au niveau de l'affichage graphique tout en affectant les paramètres des Widgets et/ou le contenu (apparition ou disparition de quelques Widgets, etc).

Cette technique de modélisation nous permet de déduire les besoins utilisateurs en fonction du contexte d'utilisation courant, y compris la tâche actuelle de l'utilisateur.

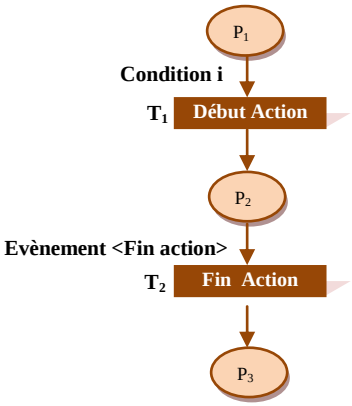


Figure 3.3 Structure modélisant l'état d'un utilisateur face à une Action Elémentaire

Une tâche utilisateur est composée d'un ensemble organisé d'actions élémentaires. La structure modélisant une action élémentaire est donnée en Figure 3.3. La validation de la condition i (transition T_1) modélise le fait que l'utilisateur va commencer l'exécution de l'action relative à cette condition. L'apparition, par la suite, de l'événement "*fin action*" (transition T_2) exprime le fait que l'action de l'opérateur a été exécutée et a pris fin. La place P_2 exprime un état d'attente de la fin de l'exécution de l'action, alors que les places P_1 et P_3 modélisent l'état de l'opérateur avant et après l'exécution de son action.

La construction du modèle global d'une tâche utilisateur s'effectue à partir de structures de base modélisant les différentes actions élémentaires de l'utilisateur. Elle se base sur l'application de différentes opérations de composition. Son principe réside dans le fait que la construction du modèle de la tâche n'utilise que des structures et des règles de composition définies. Cela est important pour assurer par avance de bonnes propriétés au modèle obtenu.

Toutes les actions de l'opérateur (élémentaires ou non) sont ordonnées selon des compositions typiques : séquentielle, parallèle, alternative, de choix, itérative ou de fermeture. Par ailleurs, l'application des règles de composition peut conduire à des réseaux dans lesquels apparaissent des simplifications évidentes qui peuvent être effectuées pour réduire la taille du modèle tout en conservant ses bonnes propriétés. Ces simplifications concernent, en fait, des états instables. C'est le cas par exemple de la composition séquentielle (Figure 3.4) de tâches élémentaires non soumises à des conditions explicites de déclenchement. Notons, enfin, que chaque changement au niveau de l'état du système doit absolument affecter l'interface de contrôle afin d'informer l'opérateur de la situation et lui faciliter sa tâche de commande. Cela implique un changement au niveau des vues graphiques affectant les paramètres des objets graphiques

(couleurs, formes, etc.) voire même un changement au niveau du contenu de la vue (apparition et/ou disparition de certains objets, etc.).

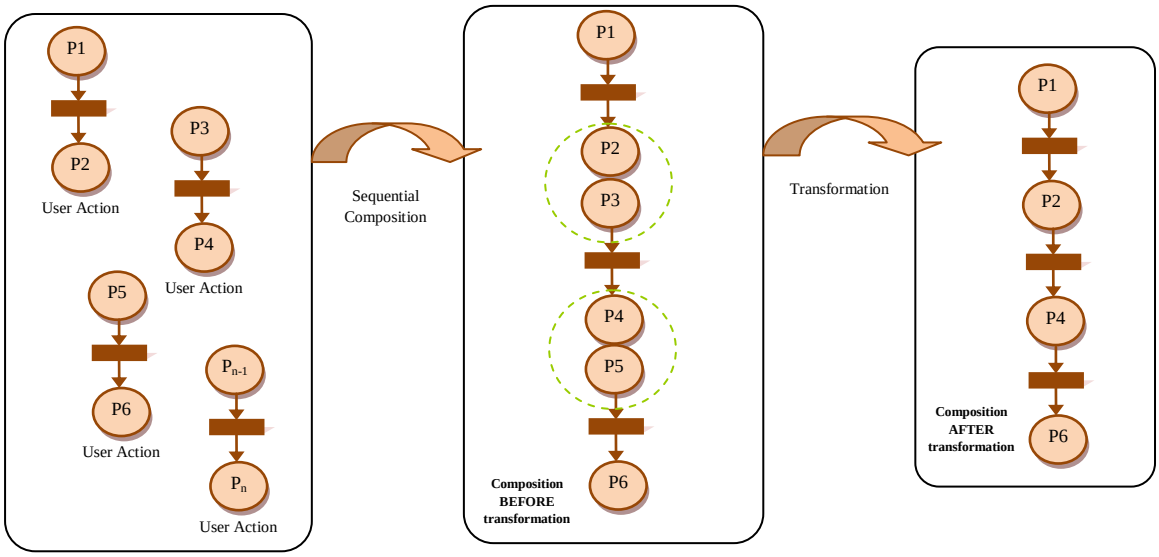


Figure 3.4 Composition Séquentielle

La composition séquentielle de n actions traduit l'enchainement de leurs réalisations. La composition de n actions est assurée par la fusion de la place fin du réseau modélisant l'action i, avec la place début de l'action i+1. Il faut noter à ce niveau que cet enchaînement des actions décrites doit être possible. La figure 3.5 montre un exemple de ce type de composition.

Figure 3.5 Structures de la Composition Parallèle

La composition parallèle permet l'exécution simultanée de deux ou plusieurs actions. Une place d'entrée, appelée place synchronisation, active en même temps toutes les places d'initialisation des actions parallèles à exécuter. Il est à noter ici que le parallélisme n'aura lieu que si les actions à exécuter ne demandent pas les mêmes ressources en même temps. Sinon, une séquence partielle ou totale serait nécessaire. La composition parallèle de n réseaux relatifs à n actions fait intervenir deux structures de composition PAR_1 et PAR_2. La structure PAR_1 modélise une synchronisation de départ de n réseaux. La structure PAR_2 modélise une synchronisation à l'arrivée de n réseaux. Evidemment, le nombre de places P_n dans les deux structures PAR_1 et PAR_2 devrait être égal au nombre des actions parallèles Ai. Ainsi, pour assurer la composition parallèle des actions, il est nécessaire de synchroniser les places d'entrée et celles de sortie de ces actions (Figure 3.6). Dans le cas où une ou plusieurs ressources peuvent être sollicitées

par l'exécution des branches (actions) parallèles, une analyse fine doit juger de l'opportunité de la parallélisation. A signaler que, d'une façon générale, la gestion des ressources partagées s'effectue d'une façon tout à fait indépendante des tâches.

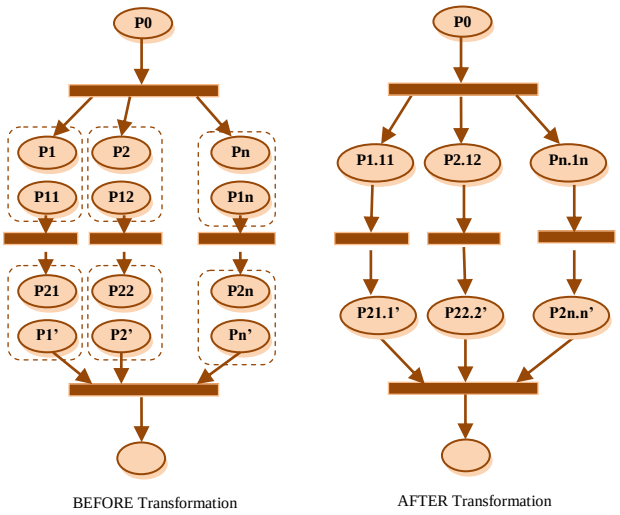


Figure 3.6 Composition Parallèle

3.2.2 Déduction des besoins utilisateurs

Afin d'élucider cette étape préliminaire de l'approche proposée, nous allons illustrer le processus de déduction des besoins utilisateur à l'aide du cas d'étude antérieurement présenté. En premier lieu, il convient de rappeller les différentes actions élémentaires que l'infirmier est supposées effectuer afin de rétablir l'état normal du patient en question. Ensuite, il s'agit de modéliser ces actions tout en y intégrant les paramètres contextuels. A l'issue de cette phase de modélisation, il sera aisé d'en déduire les **besoins utilisateurs**.

Tout d'abord, une intervention médicale dépend nécessairement de l'état enregistré du patient. Un ensemble d'actions qui indiquent les mesures nécessaires pour traiter un diabétique hypoglycémique doit être respecté (Tremblay, 2006). Le tableau 3.1 présente brièvement ces étapes et identifie les informations contextuelles nécessaires à chaque étape. Nous considérons comme première étape une intervention indispensable suite à une hypoglycémie sévère, en particulier chez une personne consciente : il faut tout d'abord et si possible, faire une glycémie au bout du doigt dans les plus brefs délais. Ensuite, ingérer 20 g de glucides, de préférence du glucose ou du sucrose, en comprimés ou sous forme liquide ; attendre 15 minutes. Et puis, refaire une glycémie. Si la glycémie demeure inférieure à 4 mmol/L, ingérer de nouveau 15 g de glucides. Si la glycémie est égale ou supérieure à 4 mmol/L, prendre sa collation ou son repas habituel. Si la collation ou le repas est prévu dans plus d'une heure, prendre une collation contenant 15 g de glucides et une source de protéines.

Nous constatons qu'à chaque étape du traitement (Tableau 3.1), l'infirmier (*Utilisateur*) aurait besoin de certaines informations afin de rendre facile et accélérer

l'accomplissement de sa tâche. Ces connaissances doivent être fournies à l'infirmier selon son action courante.

Tableau 3. 1 Informations requises selon l'étape du traitement (Patient Conscient)

Etapes	Actions élémentaires à entreprendre	Information(s) requise(s)	Description
EA1	Mesure la glycémie au bout du doigt	**UGR** {Usual glucose rate}	L'infirmier a besoin de savoir le taux habituel de glucose chez le patient en question
EA2	Ingérer 20gr de glucides si GR (Glucose Rate[21]) est inférieur à 4 mmol/l	**Available**, **NotAvailable**, **CarboHydrateType**, **CarboHydrateAvailableAt**, **CarboHydrateAmount.**	L'infirmier devrait avoir des informations sur le stock disponible de glucides, le type existant (comprimé ou liquide) ainsi que l'emplacement. Si tout va bien, l'infirmier peut avoir besoin de savoir la quantité nécessaire.
EA3	Attendre quinze minutes	**TimeRemainingM**	L'infirmier a besoin de savoir le temps restant (en minute par exemple)
EA4	Appeler le médecin en charge de ce patient.	**DoctorName**, **DoctorPhone**	L'infimier peut avoir besoin de consulter le médecin responsable du patient en question pour plus d'information
EA5	Refaire l'étape EA1	**UGR** {Usual glucose rate }	L'infirmier a également besoin de connaître le taux habituel de glucose chez le patient en question
EA6	Si GR est encore inférieur à 4 mmol/l ingérer de nouveau 15gr de glucides	**CarboHydrateAmount**	L'infirmier peut avoir besoin de connaître la quantité nécessaire à ingérer.
EA7	Sinon, donner au patient sa collation habituelle.	**RequestForMeals**, **MealsAvailableAt**	L'infirmier peut demander la collation du patient ou bien l'apporter lui-même.

Citons par exemple, l'étape EA_2 où l'infirmier devrait ingérer une certaine quantité de glucide (20gr) si le niveau de glucose mesuré est inférieur à 4mmol/l. A cette étape, l'infirmier devrait voir sur son interface des informations indispensables pour accomplir sa tâche à savoir la quantité à ingérer, l'endroit où il pourrait trouver cette substance et avant tout, est-ce qu'elle est disponible ou non. Considérons également le cas de la troisième étape qui exige de la part de l'infirmier d'attendre 15mn pour reprendre son traitement. A cet égard le temps restant après avoir effectué l'action précédente représente, sans doute, l'information dont l'utilisateur aurait besoin dans un tel contexte.

Modélisation du comportement de l'utilisateur

Supposons que toute l'activité nécessaire à entreprendre face à un cas d'hypoglécymie sévère s'appelle : ***« EmergencyInterventionHG »***. Afin de l'accomplir, les actions précédemment mentionnées (EA_1, EA_2, EA_3, EA_4, EA_5, EA_6 et EA_7) doivent être achevées également. Notons que toutes ces actions sont successives à part EA_3 et EA_4 qui peuvent être effectuées simultanément (voir f1).

[21] Indique le niveau du glucose instantané

	Modélisation des Structures Elémentaires	**Modélisation du comportement de l'infirmier suite à un cas d'hypoglycémie** *(Avant la Transformation)*
Structure modélisant l'état de l'infirmier face à la première action à entreprendre (**EA1**).	P1 → T1 : Begin EA1 → P2 → *<End EA1>* T2 : End EA1 → P3	
Structure modélisant l'état de l'infirmier face à la deuxième action à entreprendre (**EA2**).	P4 → T1 : Begin EA2 → P5 → *<End EA2>* T2 : End EA2 → P6	
Structure modélisant l'état de l'infirmier face à la troisième action à entreprendre (**EA3**).	P7 → T1 : Begin EA3 → P8 → *<End EA3>* T2 : End EA3 → P9	
Structure modélisant l'état de l'infirmier face à la quatrième action à entreprendre (**EA4**).	P10 → T1 : Begin EA4 → P11 → *<End EA4>* T2 : End EA4 → P12	
Structure modélisant l'état de l'infirmier face à la cinquième action à entreprendre (**EA5**).	P13 → T1 : Begin EA5 → P14 → *<End EA5>* T2 : End EA5 → P15	
Structure modélisant l'état de l'infirmier face à la sixième action à entreprendre (**EA6**).	P16 → T1 : Begin EA6 → P17 → *<End EA6>* T2 : End EA6 → P18	

Figure 3. 7 Modélisation du comportement de l'utilisateur

EmergencyInterventionHG ← {EA1} Seq[22] {EA2} Seq {EA3 Par[23] EA4} Seq {EA5} Seq {EA6} Seq {EA7} (f1)

Le comportement de l'utilisateur (l'infirmier) face à chacune de ces actions peut être modélisé avec les structures élémentaires (RdPIs) associées (Figure 3.7). Ainsi, le modèle est construit en composant l'ensemble de ces actions élémentaires selon la notification antérieurement mentionnée (f1).

Intégration des données contextuelles

L'intégration des données contextuelles s'appuie sur des caractéristiques du modèle réseau de Petri représentant le comportement de l'utilisateur en fonction du contexte d'utilisation. De ce fait, nous considérons la transition « Begin Action » dans le modèle de comportement de l'utilisateur. Nous associons à ces transitions les variables adéquates qui se réfèrent aux besoins informationnels de l'utilisateur (Figure 3.8).

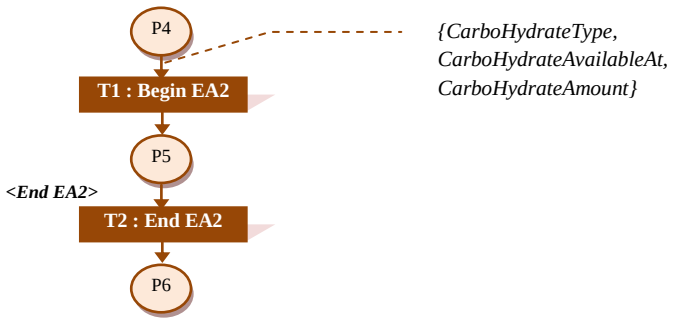


Figure 3.8 Intégration des besoins utilisateurs à l'étape EA2

En procédant ainsi, nous considérons les places P2, P4, P7, P10, P14 et P17 qui représentent des états relatifs à la présence des événements particuliers en attente de l'accomplissement des actions d'intervention. La liste de ces besoins associés à chacun de ces états est :

- Pour l'état P2, l'infirmier a besoin de connaître le taux habituel de glucose du patient concerné. Cette information est décrite par la variable d'information **UGR**.
- Pour l'état P4, L'infirmier devrait avoir des informations sur le stock disponible de glucides, le type existant (comprimé ou liquide) ainsi que l'emplacement. Ces informations sont respectivement fournies par les variables **Available**, **NotAvailable**, **CarboHydrateType**, **CarboHydrateAvailableAt**.
- Pour l'état P7, L'infirmier a besoin de connaître le temps restant, cette information sera incarnée par la variable informationnelle **TimeRemainingM**.
- Pour l'état P10, l'infirmier peut avoir besoin des conseils du médecin responsable du patient en question, le système doit lui fournir des détails sur ce médecin notamment

[22] Deux actions séquentielles
[23] Deux actions parallèles

son nom et son numéro de téléphone (**DoctorName**, **DoctorPhone**). Ces derniers représentent les besoins informationnels de cette étape.

- Pour l'état P14, l'infirmier peut avoir besoin de savoir la quantité nécessaire à ingérer : **CarboHydrateAmount**.
- Finalement, pour l'état P17, L'infirmier peut demander la collation du patient ou bien l'apporter lui-même, **RequestForMeals** et **MealsAvailableAt** sont les deux variables informationnelles de cette étape.

Identification des besoins informationnels, en conséquence identification de l'IU

Une fois que le comportement de l'utilisateur a été modélisé (Figure 3.9), les besoins des utilisateurs peuvent être déduits comme indiqué ultérieurement (Figure 3.10). Une fois ces paramètres contextuels identifiés, ces informations lui seront fournies à travers différents Widgets (Messages, Valeurs numériques, Graphiques, Alarme, etc.). Il ne reste qu'à déduire les objets nécessaires de l'interface. Pour chaque variable d'information, nous associons un objet graphique (Widget). Et puisque ces objets d'information sont relatifs au contexte (spécialement les paramètres contextuels) de l'environnement ubiquitaire où se situe l'utilisateur, identifier les paramètres contextuels relatifs à chaque état s'avère une étape indispensable.

3.1Prise en compte du changement d'activités

Outre la déduction des besoins utilisateurs en termes d'information, notre contribution fait intervenir également la deuxième problématique qui est fortement reliée à la nature dynamique des environnements ubiquitaires, à savoir le changement du contexte d'usage de l'utilisateur mobile. D'une certaine façon, nous cherchons par cette approche à suivre l'utilisateur dans l'environnement où il se situe à travers ses activités. Notre approche est basée, spécifiquement, sur la variation des activités de l'utilisateur en fonction du contexte. Il va de soi que l'utilisateur est impliqué dans une variété d'activités au cours de son travail. Ces activités peuvent être des activités routinières, des activités imprévues, des activités qui changent à leur tour selon le contexte (Par exemple les conditions de travail). Dans tous les cas, l'activité sollicite des informations spécifiques afin d'être convenablement accomplie.

Comme nous l'avons déjà spécifié en mettant en avant la caractéristique de dynamicité et en octroyant le facteur temps-réel au fonctionnement du système d'adaptation proposé, notre approche ne peut, par son objet même, se conduire d'une façon fixe et prévue. Au contraire, les modèles qui régissent le fonctionnement du système sont suceptibles d'évoluer constamment au fur et à mesure du changement du contexte et de l'évolution de l'environnement (Figure 3.11). En retour, le fonctionnement du système envers l'interaction avec l'utilisateur se verra régulièrement réorienté en fonction des changements du cadre général de son activité en cours. Autrement exprimé, la modélisation de cette interaction et son exécution ne se situent plus dans un ordre séquentiel.

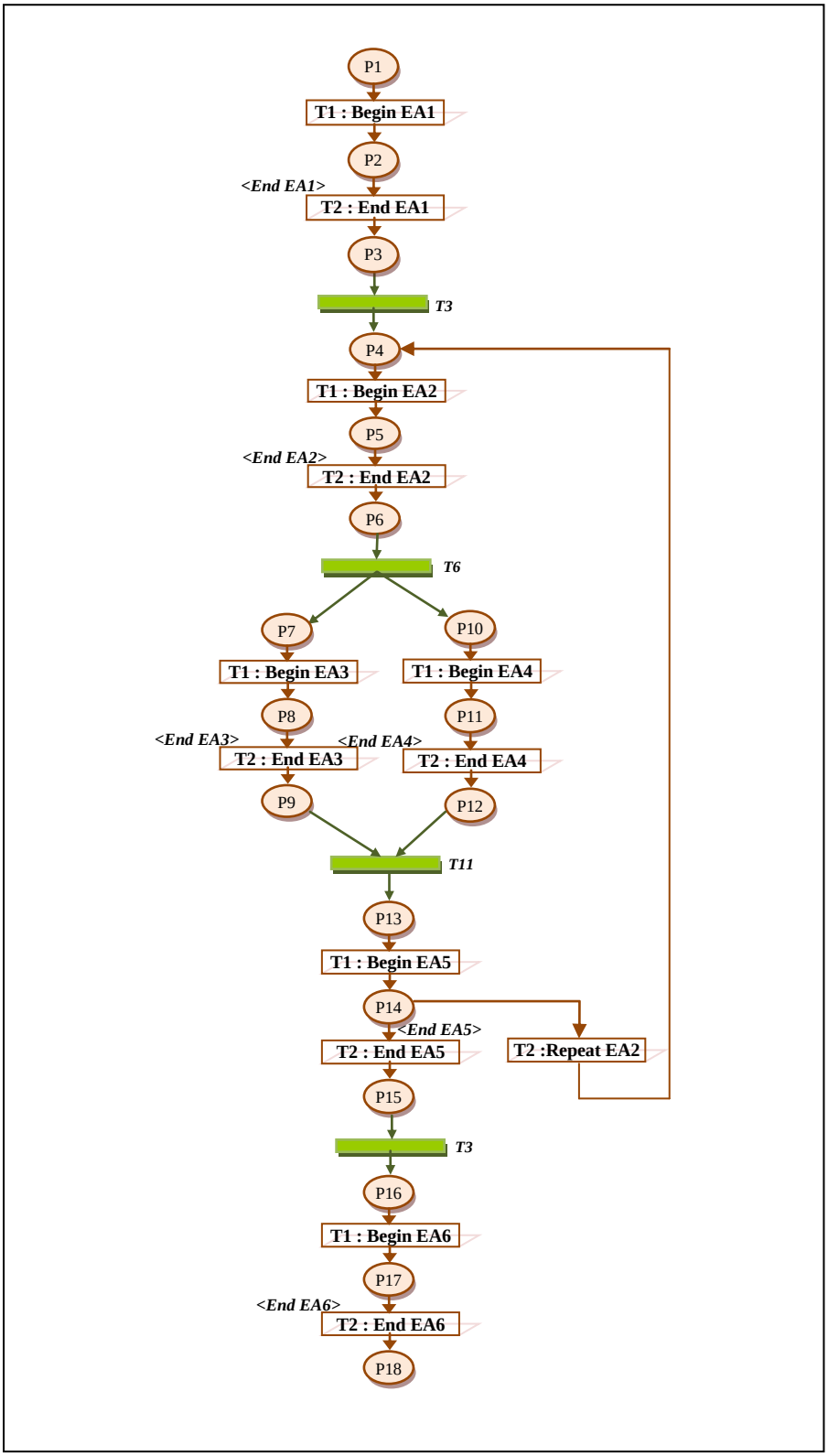


Figure 3. 9 Modélisation du comportement de l'infirmier lors de traitement d'un cas d'hypoglycémie
(après avoir effectué les transformations nécessaires)

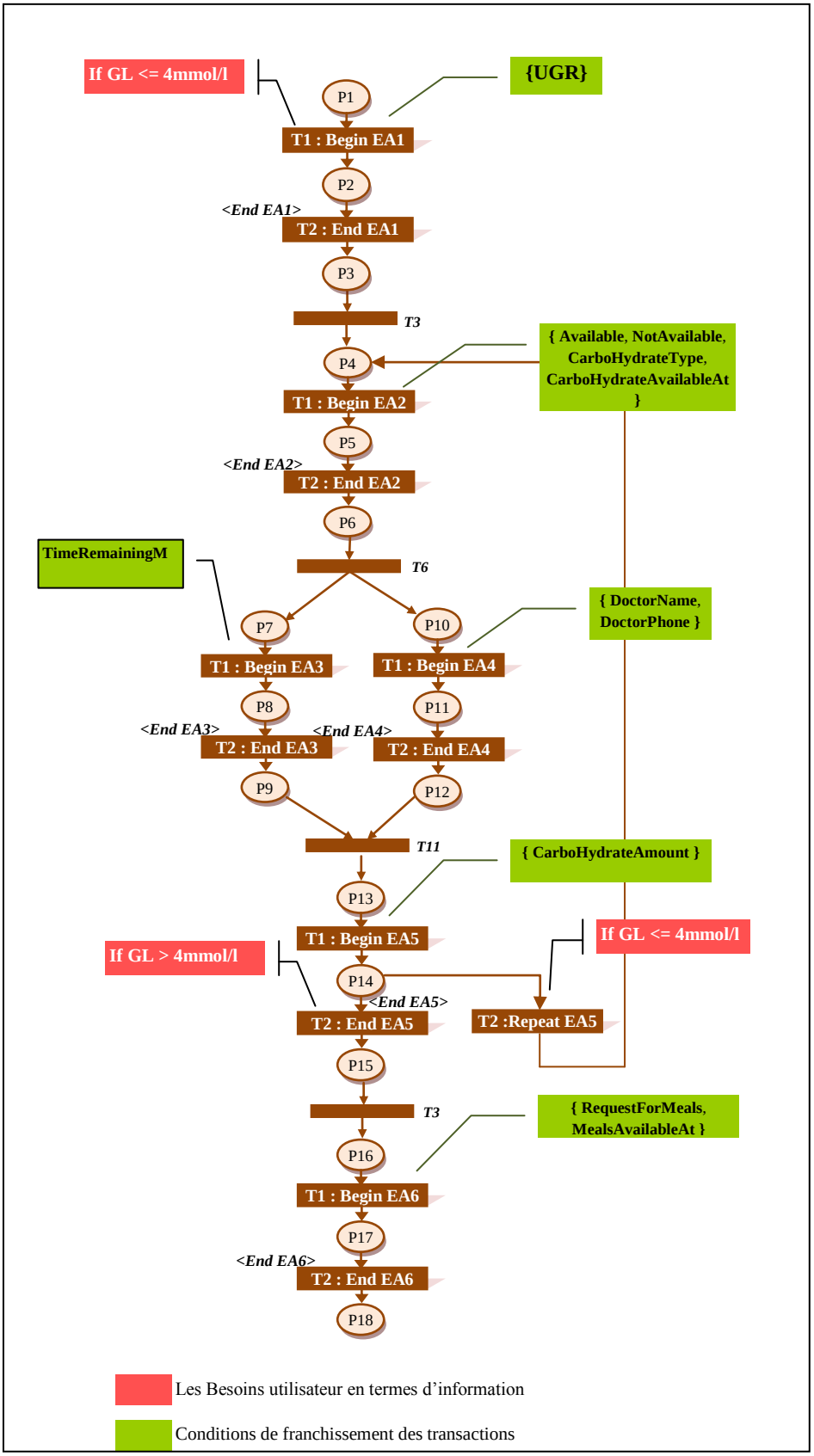


Figure 3. 10 Modélisation du comportement de l'infirmier avec mise en évidence des besoins utilisateur dans leurs emplacements correspondants

Changement d'activité donc changement de l'information requise

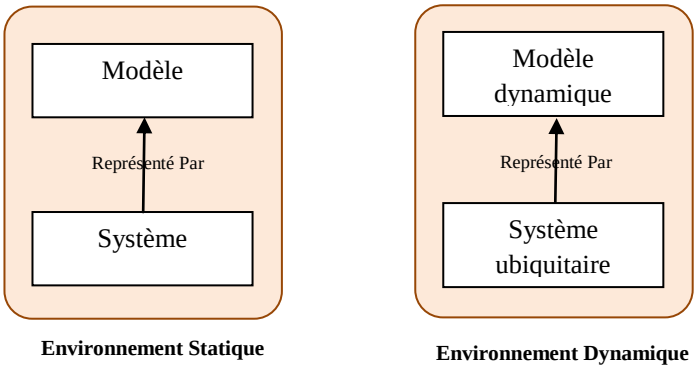


Figure 3. 11 Nouvelle exigence de la modélisation de l'interaction Homme-Système dans les environnements dynamiques

3.3.1 Description du Changement de l'information requise en conséquence du changement de l'activité en cours

Nous nous situons toujours dans le cadre d'étude de cas précédemment décrit. Nous poursuivons avec la même situation. Toutefois nous allons considérer que le patient perd conscience[24] pendant le traitement. Dans ce cas de figure, le traitement adéquat pour un patient atteint d'une hypoglycémie sévère avec perte de conscience change partiellement, et dans certaines conditions il peut changer totalement. Les questions qui s'imposent à ce niveau sont les suivantes : comment faut-il procéder pour mener à bien l'activité de « prise des soins » d'un patient ? D'une manière générale, comment le système doit-il réagir face à ce changement du contexte ? Techniquement parlant, le modèle adapté par le système lui permettant de guider l'infirmier n'est plus valable, par conséquent les besoins informationnels qui vont être déduits à partir de ce modèle sont pareillement inappropriés.

Les étapes du nouveau contexte sont brièvement résumées dans le tableau suivant (Tableau 3.2). Durant la première phase il faut mesurer la glycémie au bout du doigt dans les plus brefs délais. Ensuite, il serait préférable de coucher la personne sur le côté et éviter de la faire boire ou manger. Injecter par la suite environ 1 mg de glucagon comme c'est indiqué sur la boîte. Vérifier la glycémie 15 minutes après l'injection de glucagon. Si la personne reprend conscience et que la glycémie est inférieure à 4 mmol/L, ingérer 15g de glucides ; sinon, si elle est égale ou supérieure à 4 mmol/L, prendre sa collation ou son repas habituel. Si la collation ou le repas est prévu dans plus d'une heure, prendre une collation contenant 15g de glucides et une source de protéines.

Si la personne ne reprend pas conscience, est confuse ou fait des convulsions, téléphoner immédiatement au médecin responsable et un autre traitement s'avèrent nécessaire.

[24] Dans de telles situations une hypoglycémie sévère peut éventuellement provoquer une perte de conscience

Tableau 3. 2 Information requise selon l'étape du traitement (Patient Inonscient)

Etapes	Actions élémentaires à entreprendre	Information(s) requise(s)
EA1	Faire une glycémie au bout du doigt	**UGR**
EA2	Coucher la personne sur le côté, Eviter de la faire boire ou manger	
EA3	Injecter ensuite environ 1 mg de glucagon	**CarboHydrateAmount**
EA4	Vérifier la glycémie 15 minutes après l'injection de glucagon	**TimeRemainingM**
EA5	Si la personne reprend conscience et que la glycémie est inférieure à 4 mmol/L, ingérer 15 g de glucides	**CarboHydrateAmount**
EA6	si elle est égale ou supérieure à 4 mmol/L, prendre sa collation ou son repas habituel	**RequestForMeals**, **MealsAvailableAt**
EA7	Si le repas est prévu dans plus d'une heure, prendre une collation contenant 15 g de glucides et une source de protéines.	**CarboHydrateAmount**
EA8	Si la personne ne reprend pas conscience téléphoner au docteur responsable	**DoctorName**, **DoctorPhone**

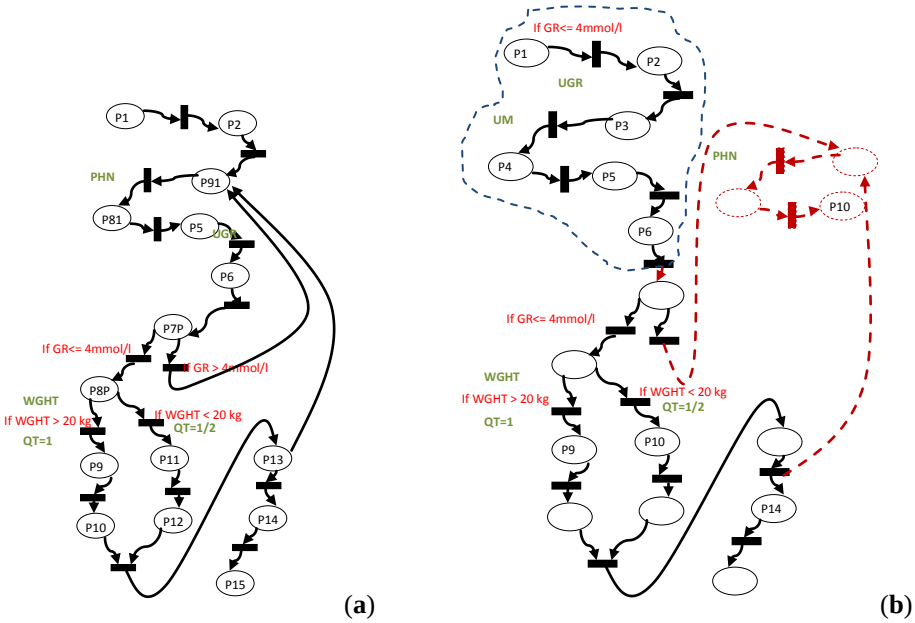


Figure 3. 12 Modélisation RdPI d'interaction (situation ancienne : Patient conscient et nouvelle situation : patient inconscient)

Les deux figures 3.12 (a) et (b) représentent respectivement le modèle d'interaction en cours d'exécution et qui paraît inapproprié pour la nouvelle situation et le modèle qui pourrait régir l'interaction au sein du nouveau contexte courant, par conséquent jugé être le modèle idoine. La phase solution de notre approche peut être divisée en deux parties. Une partie méthodologique vise à spécifier la méthode adoptée pour porter une solution au problème du contrôle du changement de contexte. La deuxième partie est technique : elle vise à spécifier la technique adoptée pour mettre en œuvre ces aspects. De fait, concernant la méthodologie et comme nous l'avons déjà décrit, elle consiste à spécifier pour chaque situation un modèle d'interaction spécifique.

À ce titre, si la situation est connue, alors le modèle sera identifié et peut être extrait et exécuté ; sinon, la « construction » du modèle au fur et à mesure de l'évolution du

contexte au sein de l'environnement se révèle nécessaire. Il s'agit de combiner dynamiquement les composants élémentaires d'un modèle dans un but de construire le modèle global. La technique à adopter pour mettre en œuvre cet aspect - que nous avons particulièrement appelé la composition dynamique et temps-réel d'un modèle - consiste à représenter ces actions élémentaires par des services web sémantiques atomiques et à décrire ces actions en s'appuyant sur l'ontologie de service web spécifiée par le langage standard OWL-S. Ainsi, la composition du modèle n'est autre que l'application de la technique de la composition dynamique et automatique des services web (CDSW). Ceci, peut représenter un aspect de la solution qui concorde parfaitement avec l'ensemble des facteurs intervenants dans cette approche : ontologie OWL, technologie des RdP pour modéliser les interactions, environnement distribué dynamique, interaction ubiquitaire, temps-réel (Figure 3.13).

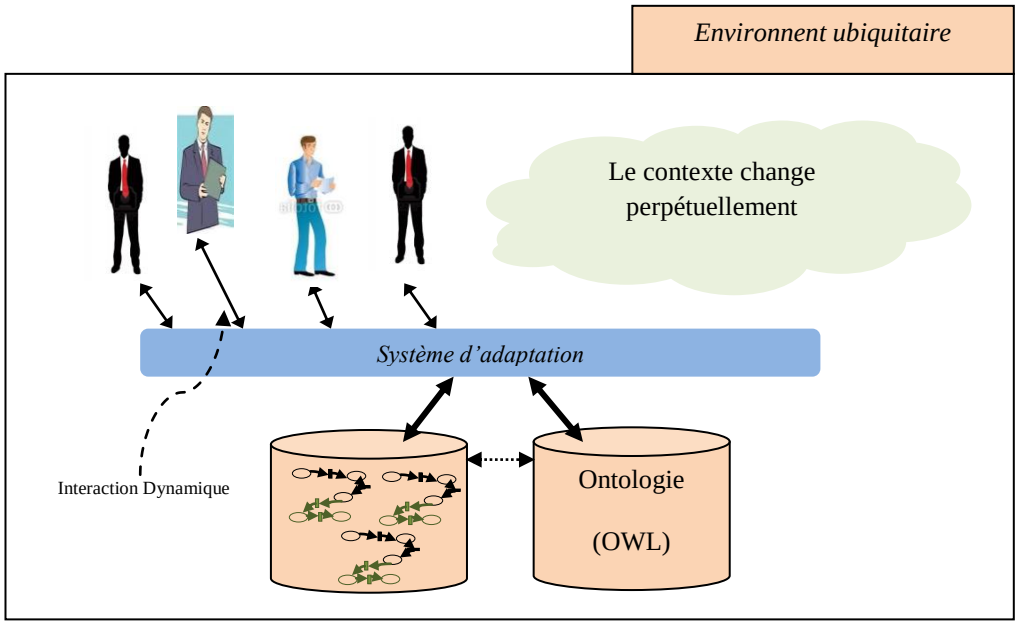


Figure 3. 13 Etat des lieux des principaux facteurs intervenant dans un environnement ubiquitaire

4. Gestion dynamique et temps-réel des modèles d'interaction dans un environnement ubiquitaire

Nous avons introduit dans la section précédente l'utilisation de la technologie de la composition dynamique des services web sémantiques pour gérer dynamiquement l'interaction avec l'utilisateur se situant dans un environnement ubiquitaire. Il s'agit dans cette section d'appréhender de manière plus approfondie cette proposition. Toutefois, nous reviendrons tout d'abord et d'une manière brève sur les concepts clés de cette technologie adoptée, allant de simples définitions jusqu'à l'architecture la supportant. La première partie présente la démarche d'étude de l'adoption de cette technique que nous avons conduite ; elle consiste à décomposer le modèle d'activité global (M_c) que nous appelons « **modèle composite** » en des sous-modèles que nous avons précédemment nommés les « actions élémentaires » (EA_i). Ces actions élémentaires seront ainsi

spécifiées par des **services web sémantiques atomiques** modélisés avec la **technologie OWL-S**. Par la suite, la construction en temps réel du modèle composite dynamique n'est autre que la composition dynamique des services atomiques en profitant des deux caractéristiques fondamentales à savoir la **découverte** et **l'invocation** dynamiques des services web. Ensuite, une partie sera consacrée à la description du cadre conceptuel et fonctionnel de l'approche pour donner finalement un aperçu sur les différents algorithmes de fonctionnement. N'est-il pas vrai à travers cette étude de proposer une technique de composition de services web mais plutôt de tirer parti de certaines critères afin de mettre en œuvre ce que nous avons appelé : composition dynamique et temps-réel d'un modèle.

4.1 Service web sémantique : concepts clés

Un service web est un programme informatique permettant la communication et l'échange des données entre applications et systèmes hétérogènes dans des environnements distribués. Il s'agit donc d'un ensemble de fonctionnalités exposées sur internet ou sur intranet, par et pour des applications ou machines, sans intervention humaine, et en temps réel (OWL-S, 2004). Ces services peuvent proposer des fonctions très simples (du type requête/réponse) ou un ensemble complet d'outils, permettant d'aller jusqu'à la composition des services pour proposer une application complète (Abouzaid, 2010). Le consortium W3C [25] définit un service web comme étant une application ou un composant logiciel qui vérifie les propriétés suivantes (Booth, 2004) :

- Il est identifié par un URI[26] ;
- Ses interfaces et ses liens (binding) peuvent être décrits en XML ;
- Sa définition peut être découverte par d'autres services web ;
- Il peut interagir directement avec d'autres services web à travers le langage XML et en utilisant des protocoles Internet.

L'objectif ultime de l'approche service web est de transformer le Web en un dispositif distribué de calcul où les programmes (services) peuvent interagir de manière intelligente en étant capables de se découvrir automatiquement, de négocier entre eux et de se composer en des services plus complexes (Kellert, 2004) (McIlraith, 2001).

Services Web Sémantiques

L'idée consiste à faire converger le Web sémantique et les services Web afin de proposer des services et des procédés qui prennent en compte la connaissance que l'on peut avoir d'un service ou d'un procédé afin de pouvoir profiter des travaux sur le raisonnement à partir de connaissances qui ont été mises en avant par le Web sémantique. Et ce, afin de surmonter les limitations des services Web en ajoutant une sémantique explicite. Son essor a même ouvert de nouvelles possibilités et des défis à la conception

[25] http://www.w3.org/2002/ws/
[26] Uniform Resource Identifier, littéralement identifiant uniforme de ressource.

d'une génération de systèmes adaptatifs, rendant possible la modélisation du profil des utilisateurs (Razmerita, 2003).

Les services Web sémantiques sont des services Web dont la description est améliorée par des langages empruntés au Web sémantique, tel que RDF et OWL. Cet emprunt au Web sémantique permet à ces services Web d'être découverts et sélectionnés automatiquement par des machines ou d'autres services Web distants (Lee, 2001) (McIlraith, 2003). OWL-S découle de travaux issus du W3C qui tentent d'apporter une solution standard en termes de description de services Web sémantiques.

Services Web sémantiques spécifiés avec OWL-S

Le but initial du langage OWL-S[27] est de mettre en œuvre des services Web sémantiques. Cette mise en œuvre inclut un grand nombre d'objectifs, rendus possibles par le biais de l'expressivité héritée d'OWL et de l'utilisation de la logique de description. Ces objectifs sont :

- La description de services Web sémantiques ;
- L'**invocation automatique de ces services**, par le biais de la détection et de l'interprétation automatique de la localisation et des paramètres d'Entrées/Sorties.
- La **composition automatique de services** (description et invocation) et la surveillance de l'exécution de la composition.

En outre, le langage OWL-S organise la description d'un service en trois zones conceptuelles : le profil (*Profile*), le modèle de processus (*ProcessModel*) et les liaisons avec le service (*Grounding*) (Figure 3.14).

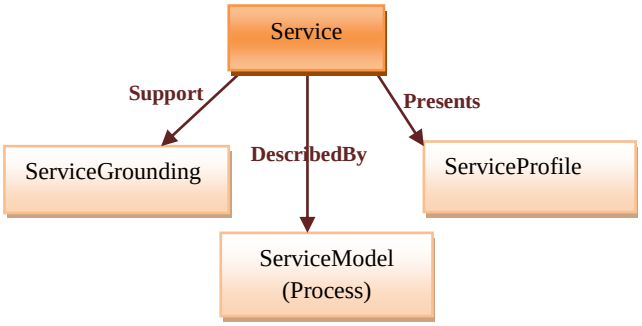


Figure 3. 14 Représentation de l'ontologie d'un service

D'une manière générale, la classe *ServiceProfile* (Figure 3.15) comprend une description du service, de son comportement fonctionnel ainsi qu'une description des attributs fonctionnels donnant par la suite les informations nécessaires pour publier ou découvrir ce service. Les profils des services sont généralement organisés sous la forme

[27] OWL-S (Ontology Web Language for Services) est un langage issu des travaux de la DARPA et de son programme Agent Markup Language (DAML) et prend la suite de DAML-S (DARPA Agent Markup Language Service). Il a été intégré au consortium W3C en 2004, au sein du groupe d'intérêt sur les services Web sémantiques, lors de la recommandation du langage OWL.

de taxonomies, qui constituent le premier niveau de discrimination lors de la recherche d'un service Web spécifique. La classe *ServiceModel* donne les conditions nécessaires à l'exécution d'un service ainsi que ses conséquences. Elle inclut des informations concernant les **entrées**, **sorties**, **préconditions** et **effets** d'un service. Dans le cas d'un service complexe (i.e., composé de plusieurs étapes dans le temps), le modèle de processus détaille la décomposition du service en composants plus simples et en explicitant les liaisons et structures de contrôle permettant l'enchaînement de ces composants. La classe *ServiceGrounding* spécifie la manière d'accéder au service concerné. Typiquement, elle spécifie les protocoles de communication à utiliser ainsi que les éléments spécifiques au service comme, par exemple, les ports à utiliser : elle fait le lien entre la description OWL-S et la spécification WSDL (Figure 3.16). De plus, la classe *ServiceGrounding* doit définir, pour chaque type abstrait spécifié dans le *ServiceModel*, une façon non ambigüe d'échanger des données de ce type avec le service. A la classe *ServiceModel* est associée une ontologie qui permet de décrire les services en termes de processus.

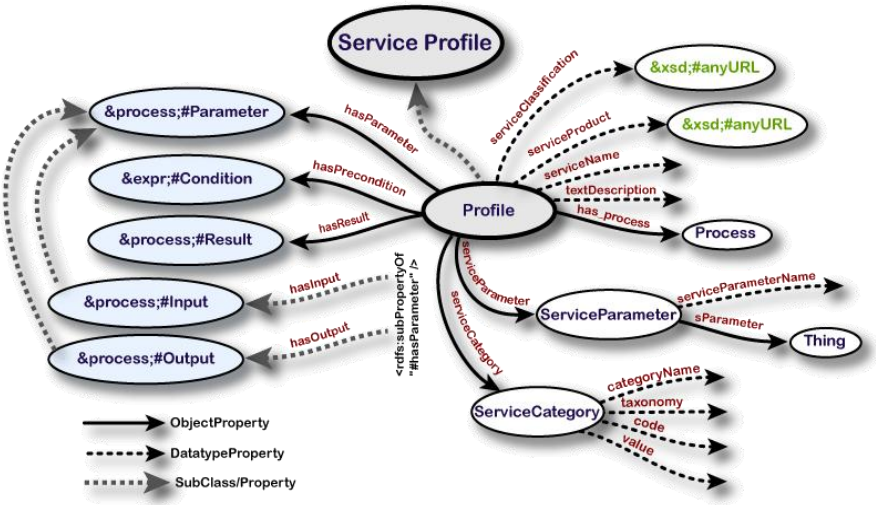


Figure 3. 15 Classes et propriétés du Service Profile

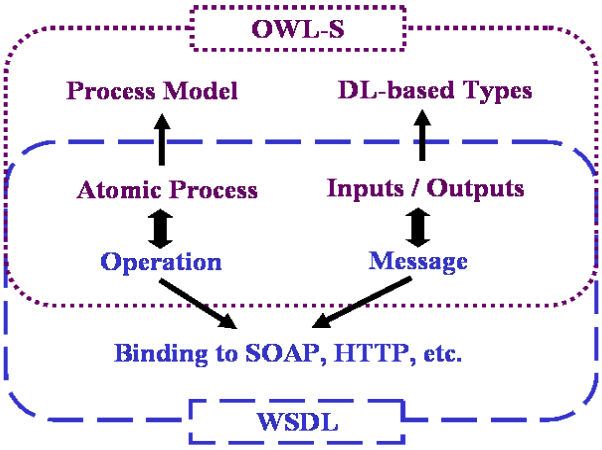


Figure 3. 16 Mapping entre OWL-S et WSDL

Le modèle de processus (*ProcessModel*) fournit une base pour spécifier le comportement d'un service. Pour avoir une vue détaillée du fonctionnement d'un service Web, celui-ci peut être vu comme un processus. OWL-S fournit une ontologie de processus (Figure 3.16) dans laquelle les processus sont vus de deux façons : un processus produit une transformation à partir d'un ensemble de données d'entrée vers un ensemble de données de sortie ; un processus produit une transition d'un état du monde vers un autre état du monde. Cette transition est décrite en termes de pré-conditions et d'effets.

Pour chaque processus, OWL-S permet donc de modéliser un ensemble d'entrées et de sorties. Il permet également de spécifier les conditions sous lesquelles les données de sortie et les effets seront produits. Les entrées, sorties, préconditions et effets jouent un rôle central dans la caractérisation des processus. Les entrées sont nommées et typées à l'aide de classes OWL ou de types de données primitifs (exemple : chaîne de caractères, entier, etc.). Ces entrées sont spécifiées comme étant des propriétés OWL auxquelles sont associés des domaines de restriction. Elles spécifient également le contenu des messages d'invocation du processus. De la même façon, les sorties sont nommées, typées et précisent les informations renvoyées par le service lors de son exécution. Les préconditions et les effets permettent de se rendre compte de l'interaction du service avec son environnement : les préconditions doivent toutes être vérifiées pour que le service puisse être invoqué, les effets spécifient les modifications de l'environnement suite à l'exécution du service.

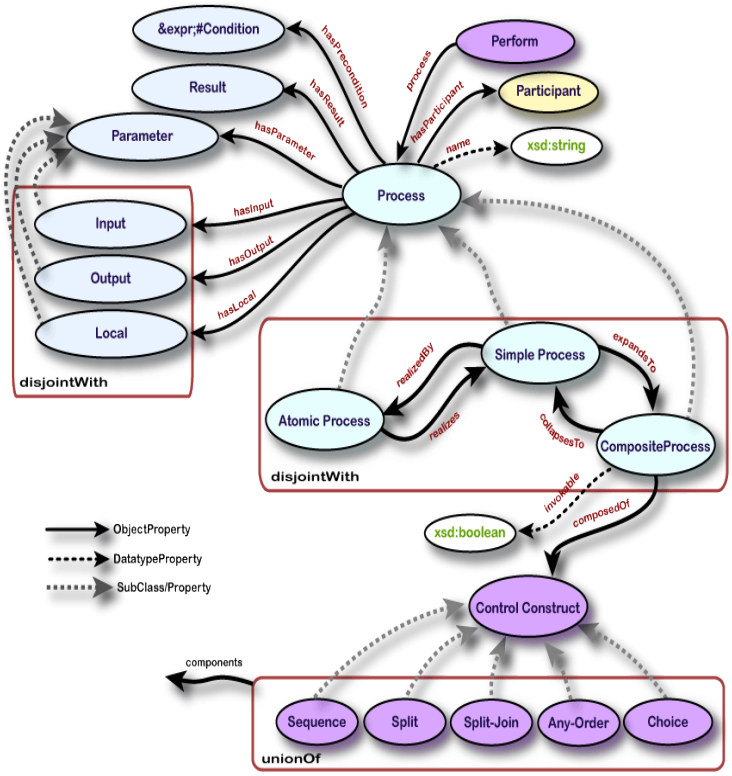


Figure 3. 17 Niveau haut de l'ontologie des processus

Comme représenté sur la figure 3.17, les processus peuvent être de trois types différents :

- les processus atomiques (AtomicProcess). Ils correspondent à des services simples directement invocables et s'exécutant en une seule fois sans intervention de la part de l'utilisateur ;
- les processus composites (CompositeProcess) qui sont décomposables en processus atomiques et composites. Ils permettent de spécifier l'enchaînement de leur différents composants à l'aide de structures de contrôle comme Sequence ou If-Then-Else.
- les processus simples (SimpleProcess) qui ne sont pas invocables et qui correspondent à des abstractions de processus atomiques ou composites. Contrairement aux processus atomiques, les processus simples ne sont pas associés à un ServiceGrounding.

OWL-S définit différents types de structures de contrôle qui régissent l'enchaînement des composants (atomiques ou composites) d'un processus composite. Par exemple, la structure de contrôle If-Then-Else contient une condition (i.e., est reliée à une condition par une propriété particulière) et deux sous-processus, l'un étant exécuté lorque la condition est vérifiée et l'autre dans le cas contraire.

Le tableau 3.3 présente les différentes structures de contrôle disponibles dans le langage OWL-S ainsi que leurs sémantiques. OWL-S a pour objectif de fournir une plus grande expressivité en permettant la description des caractéristiques des services afin de pouvoir raisonner dessus dans le but de découvrir, invoquer, composer et gérer les services web de façon la plus automatisée possible.

Tableau 3. 3 Structures de contrôle du langage OWL-S

Structure	Description
Sequence	Exécute une liste de processus séquentiellement
Concurrent	Exécute les éléments d'un ensemble de processus en concurrence
Split	Invoque les éléments d'un ensemble de processus
Split+Join	Invoque les éléments d'un ensemble de processus de façon synchronisée
Choice	Choisit parmi plusieurs alternatives et exécute l'élément choisi
If-then-else	Si la condition est vérifiée alors exécute le « then » sinon exécute le « else »
Repeat-Until	Itère l'exécution d'un ensemble de processus tant que la condition est vérifiée
Repeat-While	Itère l'exécution d'un ensemble de processus jusqu'à ce que la condition soit vérifiée

4.1 Correspondance entre une spécification d'un modèle basé sur RdPI avec la spécification des services OWL-S

A l'origine, l'idée consistait à faire correspondre la modélisation (RdPI) de l'interaction système-utilisateur à la technologie de la composition dynamique et automatique des services web sémantiques. Nous avons utilisé OWL-S pour faire associer la description sémantique aux services web développés en utilisant les termes définis dans l'ontologie de domaine créée. Ainsi, un modèle basé sur les réseaux de Petri interprétés sera concordé à la spécification OWL-S des services web sémantiques. L'idée de cette correspondance revient non seulement à la similitude des caractéristiques entre ces deux concepts mais également à la haute affinité de leurs fonctionnements, particulièrement lorsqu'il s'agit d'un environnement distribué fortement dynamique. Il revient donc à spécifier le mappage entre l'ontologie du système et les ontologies de domaine utilisées pour la description des services et identifier les critères de sélection de services, qui fournit un moyen de sélectionner de manière dynamique et classer les services basés sur les aspects fonctionnels. Ce mappage sera utilisé pour sélectionner l'action élémentaire en fonction des besoins de l'utilisateur, qui sont ensuite classés en fonction des préférences définies par l'utilisateur.

Tableau 3. 4 Correspondance macroscopique entre RdPI et OWL-S

Ontologie de modélisation RdPI	Ontologie de service OWL-S
Action élémentaire (EA)	Service web atomique *(AWS)*
Activité	Service web composite
Construction dynamique d'un modèle	Composition dynamique de services web
Invocation d'une EA	Invocation dynamique de services web

Dans ce contexte, une action élémentaire sera modélisée par un service web atomique en associant certaines propriétés d'un service web OWL-S à celles d'une action élémentaire. Ensuite, pour composer un modèle d'activité (Tableau 3.4) – représenté par un ensemble d'actions élémentaires encapsulées par les dits AWS – il suffit d'appliquer les opérateurs de composition idoines.

Pour cela, nous allons tout d'abord commencer par définir l'ontologie du domaine que nous désignons par « **RdPIModelOntology** [28] » qui décrit, entre autres, la spécification d'un modèle RdPI représenté par un ensemble d'actions élémentaires. Cette ontologie est représentée par les deux figures suivantes illustrant respectivement la conceptualisation (Figure 3.18) et l'ontologie correspondante (Figure 3.19). Il s'agit d'un modèle de données résultant d'une analyse du fonctionnement d'un réseau de Petri. Dans ce contexte-ci, la technologie UML a été utilisée pour construire un modèle du domaine qui correspondait à une abstraction du modèle RdPI.

[28] http://www.owl-ontologies.com/RDPIModelOntology.owl (URI de l'ontologie)

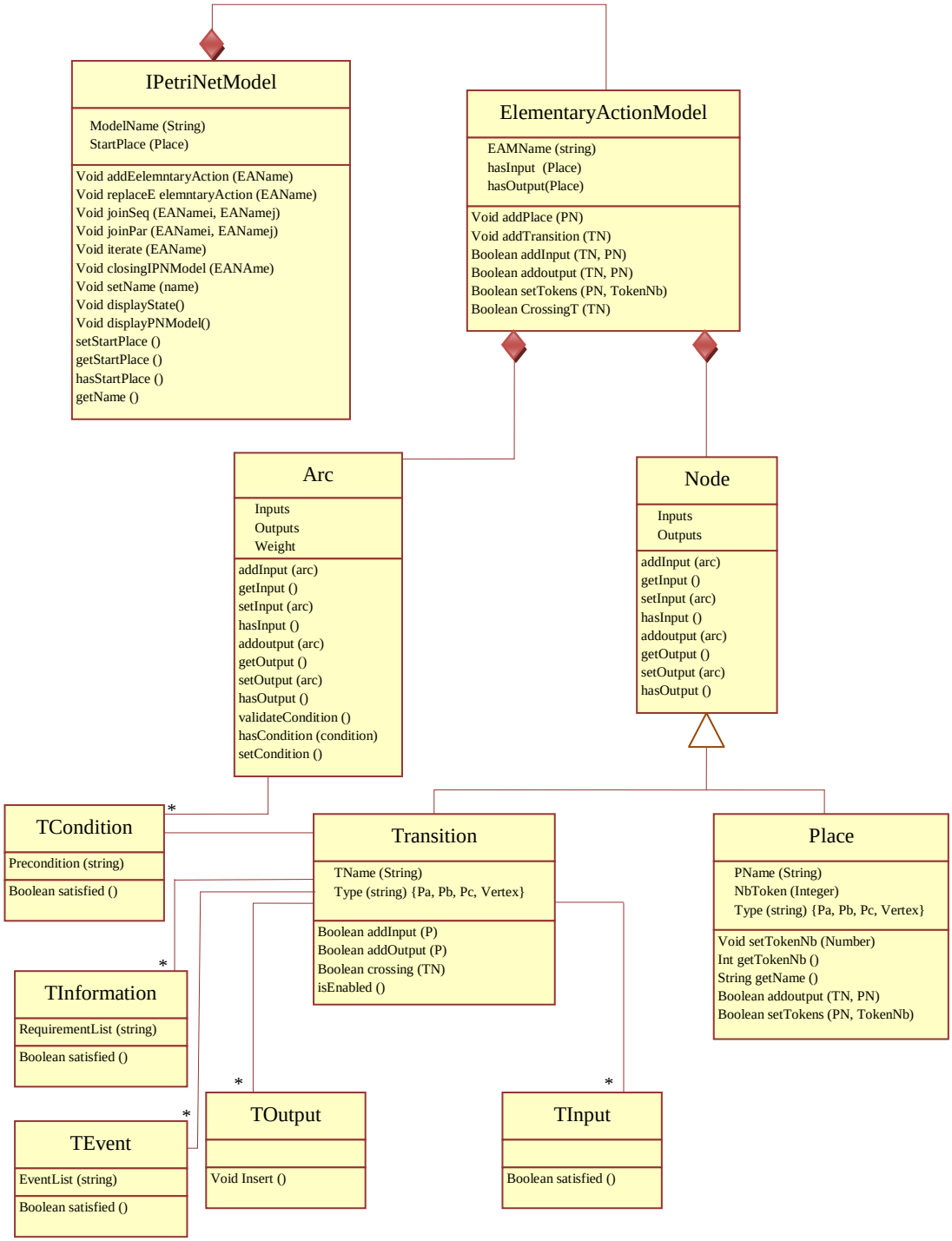


Figure 3.18 Modèle conceptuel de l'ontologie « *RDPIModelOntology* »

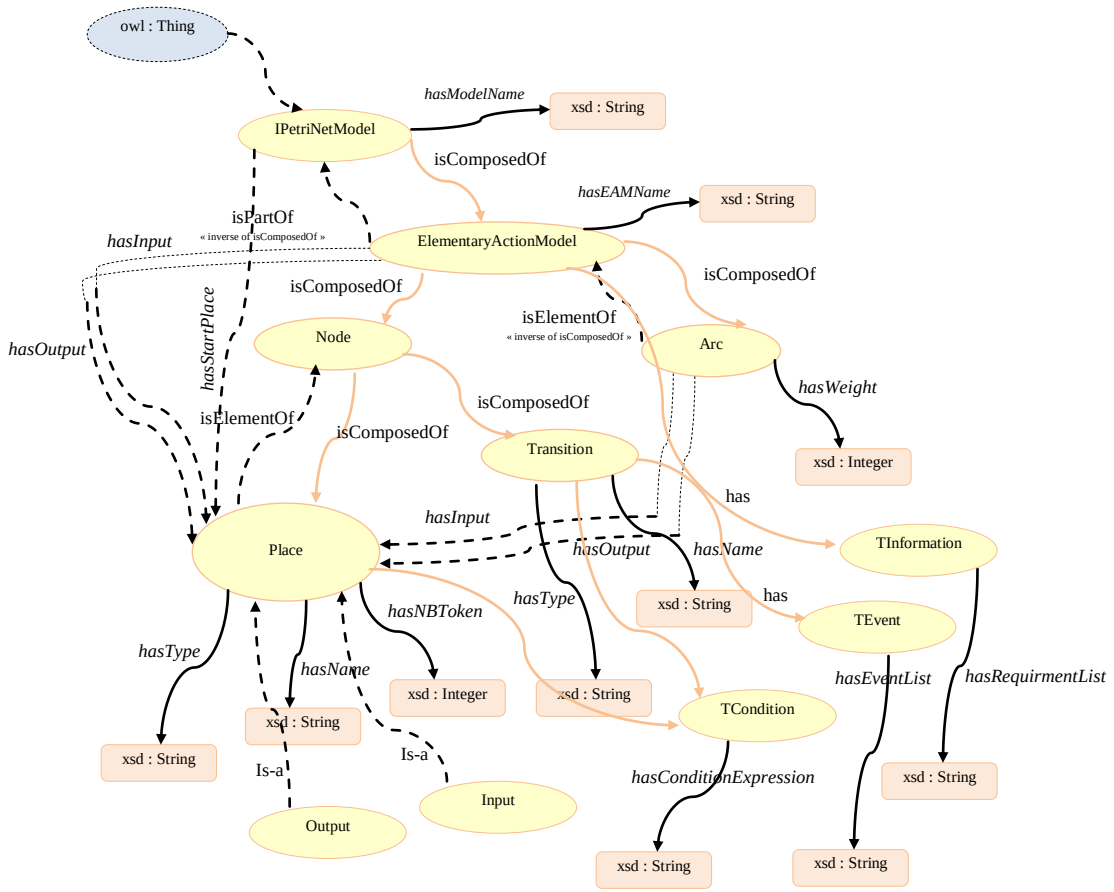


Figure 3.19 Représentation partielle de l'ontologie « RdPIModelOntology »

4.1.1 Modélisation générique d'une action élémentaire

Les descriptions considérées nécessaires dans le contexte de la spécification OWL-S sont désignées, comme nous l'avons déjà mentionné, par l'acronyme IOPR (*Input*, *Output*, *Precondition*, *Result*). En effet, un modèle de service OWL-S décrit le fonctionnement interne du service en termes de processus. Cette partie décrit la transformation entreprise par le service à travers les données (représentées par les entrées et les sorties) et le changement d'état produit par l'exécution du service (représentés par les préconditions et les effets).

Quant à la spécification RdPI d'une action élémentaire (EA_i), les paramètres d'entrées sont les informations nécessaires à l'accomplissement avec succès de l'action, ce sont principalement :

- *Condition i :* conditions de passage qui doivent être vérifiées pour déclencher l'action.
- *BeginAi : d*éclenchement effectif de l'action, en conséquence il s'agit du démarrage de l'action *Ai*.
- *Eventi :* la présence de cet événement exprime que l'action a été exécutée et a atteint sa fin.

- *P*ai : c'est la place qui modélise l'état de l'utilisateur avant l'exécution de l'action (*input place*).

Les paramètres de sortie représentent l'information extraite ou/et produite par l'action qui a été achevée (Figure 3.20) :

- *EndAi :* indique la fin de l'action A_i.
- *Pci :* modélise l'état de l'utilisateur après l'exécution de l'action (output place).
- *Requirements :* un ensemble de paramètres contextuels qui contient les informations requises en chaque transition.

Autres informations pertinentes permettant de caractériser une action telles que :

- *ActionName :* indique le nom de l'action
- *ActionGoal :* indique la fonctionnalité qui sera offerte par cette action élémentaire
- *S :* indique la situation de l'exécution de l'action
- *t :* temps d'exécution de l'action
- *Pbi :* exprime un état d'attente.

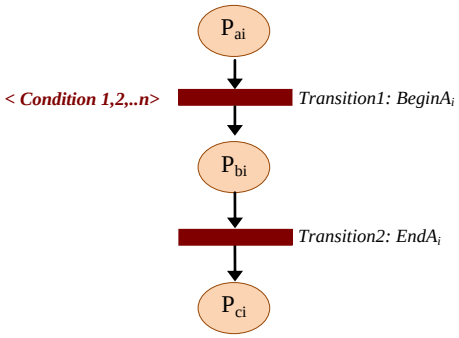


Figure 3. 20 Modélisation générique d'une action élémentaire EA_i

Sous réserve de cette description, la spécification OWL-S d'un service s'applique par analogie à l'action élémentaire comme décrit dans le tableau 3.5. Il s'agit d'intervenir sur le fond et sur la forme de cette analogie. Les similitudes fonctionnelles de la paramétrisation jouent un rôle primordial à ce sujet, il faut pouvoir les utiliser pour ramener toujours la correspondance idoine de ce qui doit être décrit par rapport à ce qui va le décrire tout en cherchant à orienter cette correspondance vers les points pertinents. Tel est le cas par exemple des entrées-sorties qui précisent la transformation des données produites par le processus qui en va de même pour les entrées-sorties de l'action élémentaire. Considérons par exemple le cas de la condition d'exécution où un processus ne peut pas être réalisé avec succès à moins que la condition soit vraie. De même, une action élémentaire possède des conditions de passage qui doivent être vérifiées pour la déclencher. L'exemple le plus significatif nous est fourni par le paramètre *RequirementList* qui représente les besoins informationnels requis durant cette action. Ce paramètre est spécifié par le terme *hasResult* (ou Result). Rappelons également que nous utilisons le terme résultat pour se référer à un effet et une sortie couplés. Aussi, faut-il noter que les changements dans l'environnement sont l'une des sources d'information

dont le système a besoin instantanément. Cet événement est spécifié par le paramètre Effet (*hasEffect*).

Tableau 3. 5 Représentation d'une action élémentaire en tant qu'un service web sémantique

Propriété de l'action élémentaire	Nom de la propriété correspondante	Type	Description
Name	*hasActionName*	String	But de l'action
« Dépend de l'action »	*hasEffect*	String	Changer une valeur dans le monde
Type de l'entrée de l'action → Place Pai	*hasInputType*	Place	Type de données en entrée
Type de la sortie de l'action Pci → Place Pci	*hasOutputType*	Place	Type de données en sortie
Condition i	*hasPrecondition*	*"SWRL-Condition"*	Décrit l'expression qui doit être vraie pour utiliser le service.
Nom de Pai	*hasInput*	Place	Entrée de l'action
Nom de Pci	*hasOutput*	Place	Sortie de l'action
Type du résultat	*hasResultType*	String	Type de sortie
Requirements	*hasResult*	String	Les besoins informationnels demandés
RelatedActivtity	*hasRelatedActivity*	String	Le processus composite susceptible de regrouper cette action
Category	*hasCategory*	String	Généralement représente le domaine globale ou spécifique de l'exécution de l'action
Profile	*hasProbProfile*	string	(Facultatif) indique le profile de l'utilisateur sensé exécuter cette action.
Priority	*hasPriority*	Integer	Indique la priorité de l'exécution de l'action, initialement égale à zéro.
Nom de l'action qui vient après	*hasAntProbAction*	String	Faciliter sémantiquement l'invocation automatique.

Un service décrit avec OWL-S est une instance de l'ontologie OWL-S, nous créons donc des instances des classes, « *Service* », « *Profile* », « *Process* » et « *Grounding* ». Reprenons l'exemple de l'étude de cas de la première action élémentaire susceptible d'être réalisée par l'infirmier, la figure 3.21 décrit l'instanciation de la classe service pour spécifier l'action élémentaire en question.

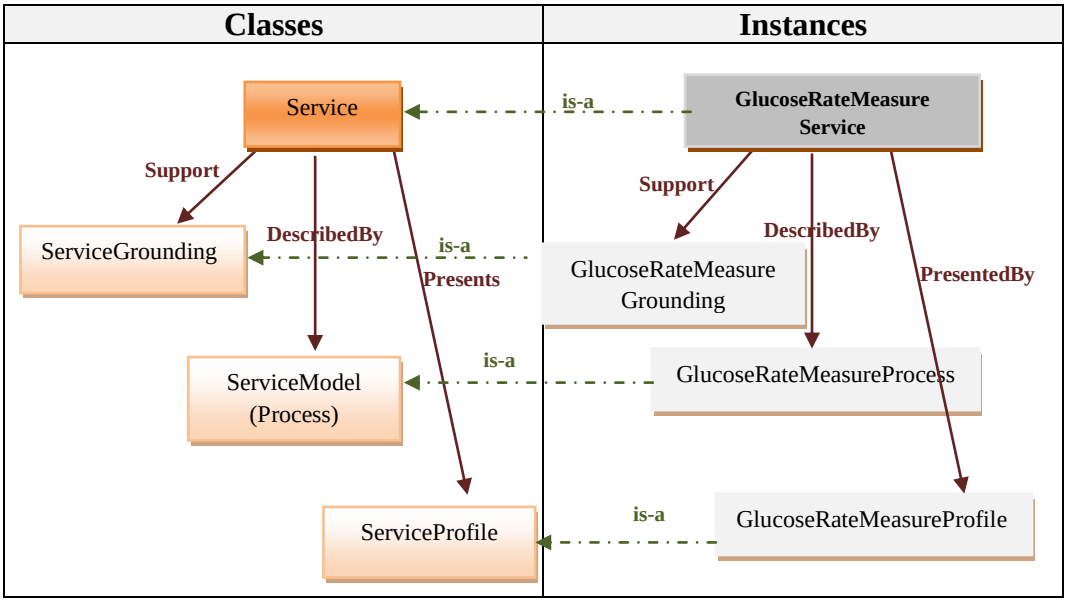


Figure 3. 21 Instanciation d'une action élémentaire en une spécification OWL-S

4.1.2 Modélisation d'une activité

Après avoir traité la question de la spécification d'une action élémentaire, nous passons à la modélisation d'une activité. Une activité est un ensemble d'actions élémentaires ordonnées selon des compositions typiques comme séquentiel, parallèle, choix, itération. L'élaboration du modèle global d'activité de l'utilisateur est basée sur la composition des modèles basiques des actions élémentaires et sur des règles de composition bien définies. De manière analogue à l'action élémentaire qui peut être spécifiée par le processus atomique, une activité peut être représentée par un processus composite. Ceci est rendu possible grâce à de nombreuses fonctionnalités de la spécification OWL-S, tels que les structures de contrôle qui peuvent assurer la composition d'actions élémentaires.

L'agrégation de services constitue l'un des champs de recherche les plus actifs dans le domaine des services Web (Maesano, 2003). Plusieurs courants sont d'actualité, dont l'orchestration et la composition en utilisant des techniques d'intelligence artificielle (Splunter, 2010) (Nevejan, 2010) (Richards, 2003). Cette composition peut se faire avant ou pendant l'exécution des services Web. Le service résultant de cette composition est appelé service composite. Il s'agit de l'approche adoptée pour composer dynamiquement un modèle d'activité (*Service Composite)* sous forme d'agrégat d'actions élémentaires par assemblage des actions déjà existantes nommées dans ce contexte, actions basiques. La construction spécifie quelles AEs doivent être invoquées, dans quel ordre et sous quelles préconditions. Dans ce sens, nous avons déjà proposé une ontologie qui formalise la description sémantique des actions élémentaires. Cette ontologie facilite implicitement la découverte, la sélection, la composition et l'invocation automatique de ces services : il s'agit de la spécificité de l'approche de composition des SW.

Dans ce cadre, un moteur d'enchaînement a été envisagé et qui permet de définir un enchaînement de SW en réponse à un objectif précis fixé par le système d'adaptation. Il tient compte des AEs, de leurs fonctionnalités et du but à atteindre que ce soit avant ou pendant l'exécution du modèle. A cette fin, le système applique cette approche pour créer dynamiquement ces dits modèles en découvrant et en sélectionnant les modèles élémentaires jugés pertinents pour le suivi et le contrôle de l'interaction. La sélection des AEs sera faite à la volée, au moment de l'exécution, en fonction des contraintes imposées par le système. A ce niveau, le modèle globale peut être vu comme un ensemble de services web qui coopèrent entre eux pour fournir certaines fonctionnalités.

4.1.3 Illustration

Nous allons maintenant décrire la construction du processus composite. Nous appelons cette activité « ***consciousPatientInterventionActivity*** » et le processus associé « ***consciousPatientInterventionActivityProcess*** », qui est un processus composite composé de six actions. Nous envisageons la composition, uniquement et à titre d'exemple, des processus de EA_1 et EA_2, qui sont deux processus séquentiels.

A titre d'illustration, nous présentons la composition **séquentielle** des actions élémentaires. En général, la composition séquentielle des actions élémentaires N se fait en fusionnant les places de sortie de l'action i, et les places d'entrée de l'action i +1.

- Elementary action EA_1: ***GlucoseRateMeasure***
- Elementary action EA_2: ***CarbohydrateIngest***

Une description des entrées et des sorties pour chacun des processus atomiques est nécessaire (Figure 3.22 et 3.23). *A1* et *A2* sont respectivement des instances des processus des actions élémentaires EA_1 et EA_2. Les expressions en *italique* représentent les types des paramètres d'entrée et sortie des actions élémentaires.

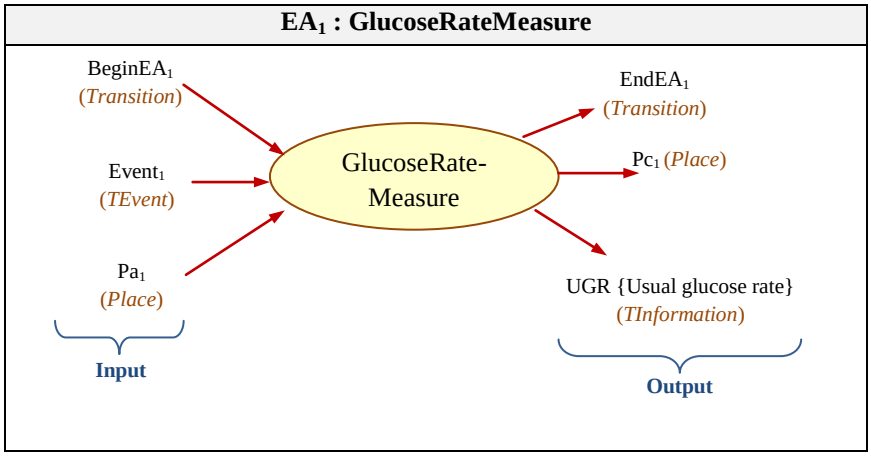


Figure 3. 22 Paramétrisation de l'action élémentaire : GlucoseRateMeasure

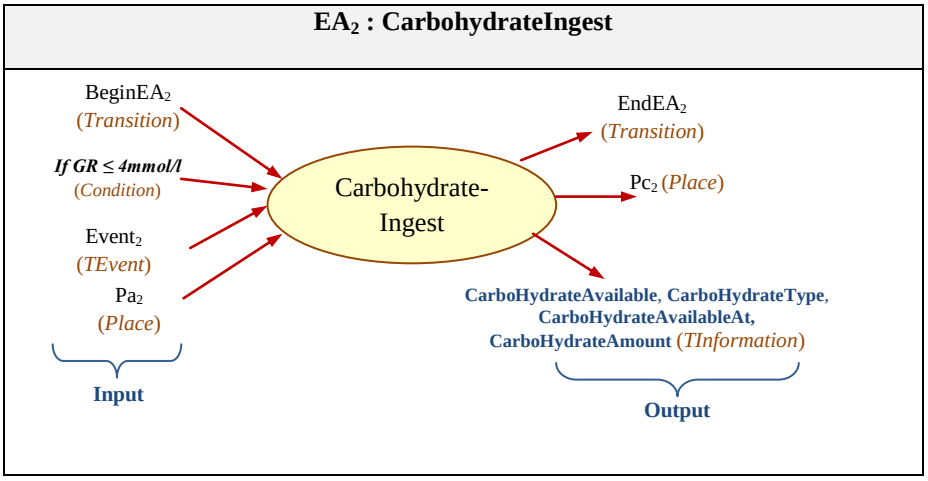
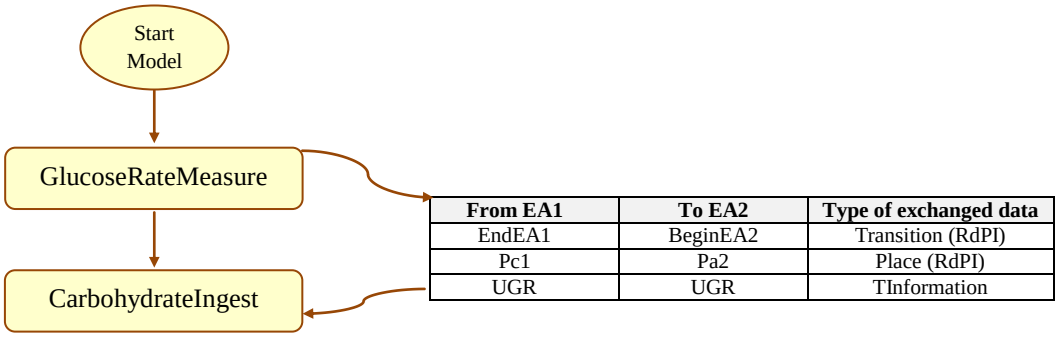


Figure 3. 23 Paramétrisation de l'action élémentaire : CarbohydrateIngest

Dans ce cas, le processus composite associé doit avoir une propriété *composedOf* qui est indiquée par la structure de contrôle « *sequence* », en utilisant un *ControlConstruct*. Ensuite, la spécification du flux de données doit être définie, c'est à dire, les paramètres qui doivent passer de l'action source à l'action destination doivent être spécifiés. Dans notre exemple, UGR représente le paramètre qui doit être transmis.

Ce passage des données sera davantage détaillé dans la section suivante qui décrit les différentes facettes de notre approche.

En fait, dans de nombreux cas, quand un processus est exécuté comme une étape dans un processus plus large, il doit y avoir une description des données en entrée du processus exécuté ainsi que les sorties qui en découle (Figure 3.24). Comme décrit précédemment, le modèle global d'une activité est développé en utilisant les différentes actions élémentaires composées à travers des structures de contrôle bien déterminées. Pour cette raison, nous allons combiner d'une manière séquentielle EA_1 et EA_2 en fusionnant la place de sortie (Pc_1) et la place d'entrée de l'action EA_2 (Pa_2) en une seule place.



From EA1	To EA2	Type of exchanged data
EndEA1	BeginEA2	Transition (RdPI)
Pc1	Pa2	Place (RdPI)
UGR	UGR	TInformation

Figure 3. 24 Flux de données entre les deux actions élémentaires envisagées

4.2 Cadre conceptuel et fonctionnel de l'approche

Implicitement, le système d'adaptation (Figure 3.25) est considéré ici comme le consommateur de ces services, il va donc les utiliser et les réutiliser afin de répondre aux besoins de l'utilisateur. Pour obtenir les services correspondant à ses besoins, le système interroge le registre ***EARegistry***. En effet, la découverte d'un service est réalisée grâce à sa description disponible dans ***EARegistry***. Après avoir sélectionné le service approprié, le système récupère les informations dont il a besoin. Autrement dit, grâce à la description du service, il peut, dès lors, réaliser une liaison et appeler les fonctions du service, notamment le paramètre «*UserRequirements*».

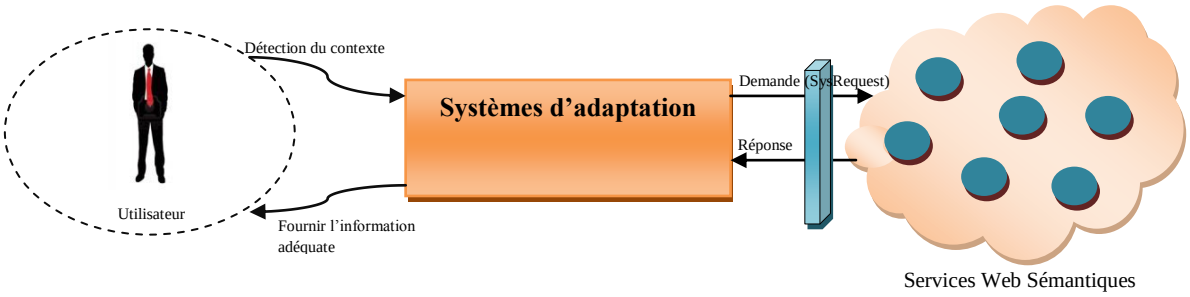


Figure 3. 25 Le système d'adaptation : architecture fonctionnelle réduite

La figure 3.26 représente une vue circonstanciée et analytique des différents composants de l'architecture fonctionnelle de notre approche. Ce modèle conceptuel met l'accent sur le composant « *ChainMaker* » étant donné qu'il représente le cœur de cette architecture. Il s'agit en fait du moteur qui pourvoit la découverte et construit dynamiquement un modèle grâce à la spécification sémantique des concepts définis par l'ontologie de domaine.

Cette architecture conceptuelle distingue trois grandes étapes pour créer dynamiquement une séquence d'un modèle : la découverte des prochaines actions élémentaires nécessaires pour répondre aux besoins du système d'adaptation (« *SysRequest* »), l'identification de la candidate pertinente parmi les dites actions élémentaires candidate et, enfin, effectuer l'opération de correspondance «*matching*» afin de récupérer les résultats fournis par l'action prédestinée. Nous avons proposé d'utiliser les descriptions sémantiques des services Web pour pouvoir comparer les fonctionnalités en utilisant, particulièrement, les concepts « **preconditions** » et « **results** » de la spécification OWL-S. La solution proposée peut être résumée ainsi :

- Identifier les fonctionnalités requises (à travers l'analyse de la requête du système)
- Localiser les différentes actions élémentaires appropriées (***EADiscovery***) pour la tâche en cours identifiée et ce grâce à leurs descriptions sémantiques entreposées dans le registre ***EARegistry***.
- Choisir celle (SEA[29]) qui s'adapte au mieux aux exigences du système (***EASelection***)
- Combiner les services sélectionnés afin d'atteindre l'objectif préfixe (***ChainMaker***).
- Récupérer les résultats ravitaillés par l'action élémentaire sélectionnée (***Requirments' Deduction***).

L'architecture contient donc les composants nécessaires suivants : le composant « *ChainMaker* » assure le chaînage du modèle en cours, le composant « *Knowledge Repository* » assure le stockage des données et des ontologies, entre autres, l'ontologie formalisant la description sémantique des services. Le composant « *EARequester* » qui assure l'envoi des critères pour sélectionner une « *SEA* ».

Enfin, le composant « *EARequest* API » transforme la description de la requête initiale en se référant aux différents termes et concepts de l'ontologie du système dans une description pseudo (et ce en utilisant les termes et les concepts de l'ontologie). Finalement, quand une action est invoquée avec tous ces paramètres d'entrées, il retourne en résultat les paramètres de sortie.

[29] Selected Elementary Action

Ces différents composants se combinent afin d'assurer une opération donnée. En termes de services web proprement dit, cette opération se déroule en trois étapes :

- Les services web sont recherchés et sélectionnés en fonction des besoins à réaliser.
- La composition est effectuée en utilisant la description (syntaxique ou sémantique) des services sélectionnés.
- Une description du service composite (i.e l'enchainement des appels aux services sélectionnés) est créée.

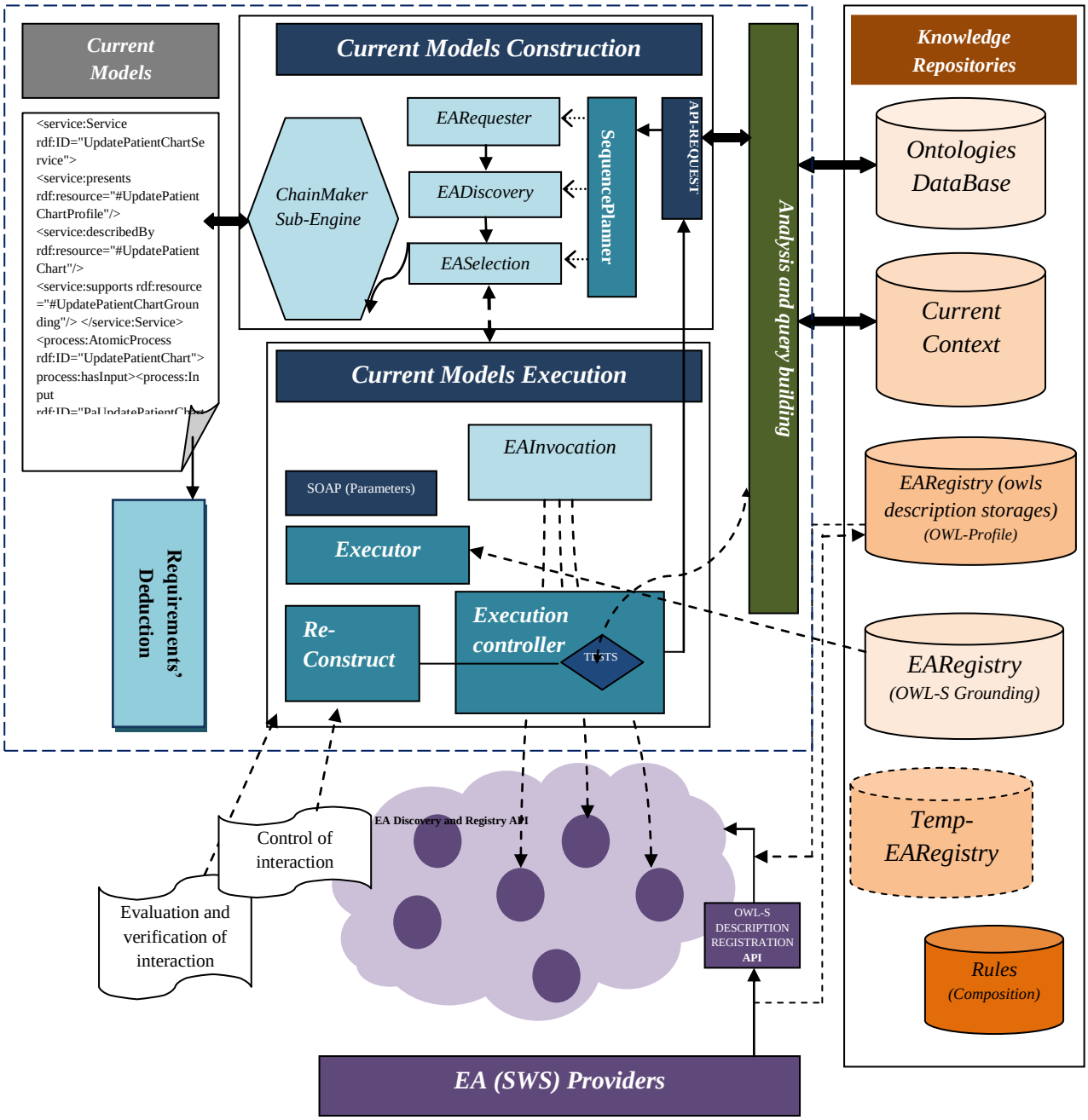


Figure 3. 26 Modèle Conceptuel de l'approche de composition dynamique d'un modèle d'activité

a. Enregistrement des AEs dans EARegistry

Les éléments « EAService providers » peuvent être une personne ou une organisation qui fournit les fonctionnalités sous forme de services. Pour notre cas d'étude, il s'agit d'un personnel médical qui peut spécifier, ajouter, améliorer les fonctionnalités des actions élémentaires. Ces services vont être publiés par la suite dans le registre (EARegistry), un des composants qui sert d'acteur intermédiaire avec le système d'adaptation. Initialement, les services ***providers*** annoncent leurs services (profil, processus, grounding OWL-S). Les actions élémentaires prévues pour être utilisées enregistrent leurs services dans la base de données temporaire.

Au moment de l'enregistrement, le composant (API du sous-composant responsable de l'enregistrement et du traitement des requêtes) (Figure 3.27) analyse les descriptions OWL-S disponibles dans le référentiel. Ensuite, il convertit l'ontologie OWL de description sélectionnée dans une collection de faits stockés sous forme de triplets (c'est-à-dire <subject, predicate, object>). Finalement, il traduit les conditions préalables (**Precondition**), les sorties (**Outputs**) et les effets (**Effects**) dans l'ontologie de description du service en règles stockées dans la KB. Ces règles se composent de certaines expressions conditionnelles, et une série de commandes à exécuter lorsque cette expression est satisfaite.

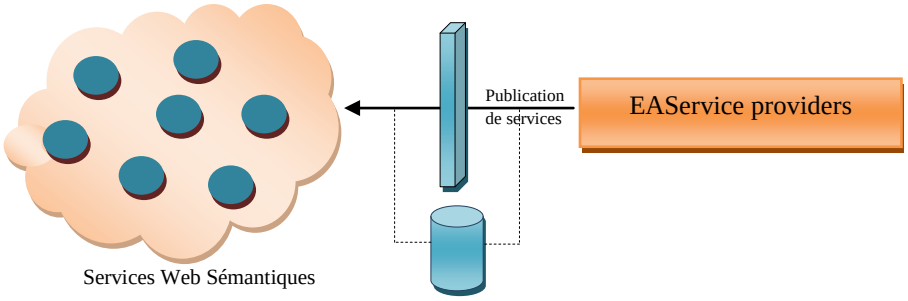


Figure 3.27 Enregistrement des actions élémentaires

Finalement, l'invocation automatique d'un service signifie l'exécution du service par le système d'adaptation qui n'est autre qu'un programme informatique susceptible d'exécuter ces services. Autrement dit, notre système doit être capable d'interpréter les descriptions OWL-S afin de délivrer les données nécessaires à l'exécution du service Web (*UserRequirements*).

- *TempEARegistry* : C'est un répertoire temporaire (référentiel) des actions élémentaires probablement destinées à être utilisées dans un temps relativement court, sélectionnées selon des critères bien déterminés, notamment le contexte d'utilisation courant.

- *EARegistry* : C'est un référentiel qui regroupe un ensemble de fichiers de descriptions de services web, utilisant le langage WSDL, basé sur XML. Le système interroge ce registre à l'aide de mots clés (identifiés à partir de la

requête système) pour obtenir un ensemble de descriptions WSDL (CEASet). Ces descriptions WSDL contiennent toutes les informations nécessaires à l'invocation du service (URL de localisation, description de/des fonction(s) et les types de données). Indirectement sollicité par le système d'adaptation, ce composant sert à trouver les actions élémentaires disponibles potentiellement candidates.

- *EADiscovery :* On entend par la découverte dynamique la possibilité de localiser automatiquement un Service web qui répond à des besoins particuliers. La découverte de la prochaine action élémentaire requise consiste à l'abstraction du but concret de l'utilisateur à un objectif prédéfini, réutilisable et formalisé. Cette technique traite de la mise en correspondance des objectifs formalisés, la sélection de services Web qui peuvent potentiellement être utilisés pour obtenir l'action élémentaire désirée. La dernière étape constitue l'action de la découverte proprement dite. Cette étape fait recours aux services Web adaptés à l'étape précédente pour accéder aux services réels. Enfin, il s'git de vérifier quels sont les services qui s'avèrent nécessaire afin d'atteindre l'objectif visé.

- *EARequester* : Ce composant envoie des requêtes qui répondent à des critères servant de base pour recevoir la prochaine action. En particulier, ce composant spécifie une demande de sélection de service décrite avec la technologie OWL-S. Un autre élément : *reqAPI* transforme la description du service demandé en une pseudo-description.

- *EASelection* : ce composant essai de préciser les services candidats selon qu'ils satisfont les exigences prédéfinies.

- *Chainmaker* : ce composant supporte la démarche de la composition, il assure le chaînage des modèles. Sa principale fonction consiste à trouver des descriptions de services qui répondent à la demande. Cette tâche de correspondance se base principalement sur l'identification des éventuelles similitudes entre les descriptions des services disponibles avec la requête. En particulier, cette tâche est fondée sur la correspondance entre les « Input/Output », les Preconditions ainsi que les Effets du service.En outre, cette correspondance peut tirer profit de la catégorie du service. Ce composant précise également le contexte spécifique pour effectuer la correspondance sémantique entre plusieurs ontologies.

  Cette correspondance servira de base pour sélectionner les services répondant aux exigences du contexte de la requête-système. Par conséquence, les exigences du contexte de l'activité de l'utilisateur et sa situation en général seront prises en compte. Ces services pré-sélectionnés sont ensuite classés en fonction des critères spécifiques. Des contraintes

d'interopérabilité pour spécifier des correspondances entre les ontologies utilisateurs et les ontologies de domaine utilisés seront identifiées (par exemple : une extension de OWL-S qui décrit un service atomique encapsulant l'action élémentaire).

- *API-Request* : ce composant transforme la description de demande de service initiale (en utilisant les termes et les concepts de l'ontologie de l'utilisateur) dans une pseudo-description (en utilisant les termes et les concepts du domaine de l'ontologie). Enfin, la requête d'une action élémentaire répondant aux critères de sélection peut être transformée en une requête traitable par le composant « *ChainMaker* ».

  Le but visé consiste à spécifier les ontologies de domaine en question et l'ontologie de l'action élémentaire. Cette spécification sera par la suite utilisée pour découvrir et invoquer le service correspondant. Egalement, ce composant sera placé sur un intermédiaire situé entre le système et le service. L'utilisation d'une couche de médiation permet une meilleure interopérabilité entre les services et les demandes de services ainsi qu'une plus grande robustesse face aux changements. Par exemple, lorsqu'un service n'est pas disponible au moment de son invocation, la couche de médiation peut se connecter à un autre service.

Un autre composant permet l'interprétation des descriptions WSDL des SWs en vérifiant leurs descriptions par rapport au modèle. Il réalise en parallèle un processus de raisonnement basé sur les règles de composition RdP pour composer automatiquement une séquence de services Web exécutable. Cette séquence est obtenue en suivant le schéma de workflow défini dans le modèle de processus OWL-S et pourvoit une requête de service de haut niveau.

Déploiement des actions élémentaires /activités

Le cycle de vie d'une action se déroule de la façon suivante : une fois créée, l'action élémentaire (ou bien l'activité) sera déployée sur le réseau (local ou internet) puis le système d'adaptation tout en ayant des besoins spécifiques va rechercher le service qui répond à ses exigences à l'aide du référentiel spécialisé. Enfin, une fois l'action trouvée, le système va invoquer le service associé : une communication va être mise en place entre l'utilisateur et le service web. Ce cycle de vie, représenté par la figure 4.28, fait particulièrement intervenir ces technologies :

- SOAP[30] : protocole permettant d'invoquer à distance les opérations offertes par un WS en utilisant des messages XML.

[30] **SOAP** (ancien acronyme de Simple Object Access Protocol) est un protocole de RPC orienté objet bâti sur XML. Il permet la transmission de messages entre objets distants. il autorise un objet à invoquer des méthodes d'objets physiquement situés sur un autre serveur..

- WSDL[31] : un standard permettant de décrire l'interface d'un service web sous la forme d'un fichier de description en XML.

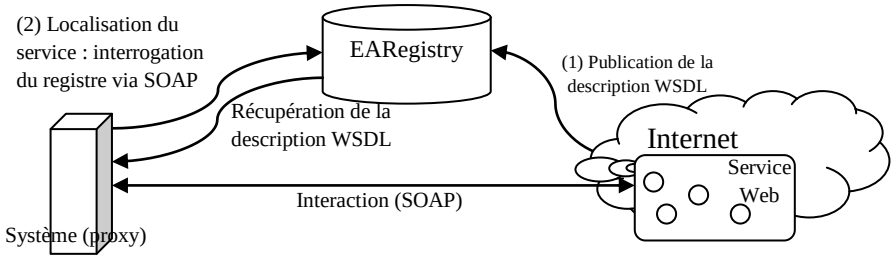


Figure 3. 28 Fonctionnement basique et protocole de communication

5. Algorithmes de fonctionnement

Dans ce qui suit, nous procédons à l'analyse du cadrage fonctionnel de l'approche d'adaptation via les algorithmes et les modes de fonctionnement. Dans ce cadre nous proposons trois principaux cas de fonctionnement qui découlent implicitement de la nature de ce fonctionnement estimée être générique pour de tels systèmes. En effet, le premier mode de fonctionnement envisagé est le mode normal : « ***NormalExecutionMode*** : **0** ». Ensuite, le mode « ***ReplanningExecutionMode*** : **1** » qui aurait lieu lors d'un changement du contexte d'usage de l'utilisateur en question. Enfin, le mode « ***PlanningExecutionMode*** : **2** » supposé avoir lieu lors du traitement d'un cas considéré comme inconnu.

5.1 Algorithme spécifique au mode de fonctionnement normal

Le premier mode de fonctionnement, *NormalExecutionMode*, suppose la présence d'un fonctionnement normal libre de toute exception pouvant survenir durant sa progression (Figure 3.29). Après avoir analysé la situation spécifique à un utilisateur donné et formulé la requête associée à cette situation, le module « *SequencePlanner* » récupère le fichier "owl-s process model" correspondant à la dite requête. Il procède par le parcours de ce fichier en second lieu et y extrait deux listes : la première regroupe les actions élémentaires susceptibles de gérer la situation courante, la seconde organise les séquencements de ces présumées actions.

[31]**WSDL** est une grammaire XML permettant de décrire un Service Web. Le WSDL décrit une Interface publique d'accès à un Service Web, notamment dans le cadre d'architectures de type SOA (Service Oriented Architecture). C'est une description fondée sur le XML qui indique « comment communiquer pour utiliser le service »;
Le WSDL sert à décrire : le protocole de communication (SOAP RPC ou SOAP orienté message) & le format de messages requis pour communiquer avec ce service & les méthodes que le client peut invoquer & la localisation du service. Une description WSDL est un document XML qui commence par la balise <definitions> et qui contient les balises suivantes : <binding> : définit le protocole à utiliser pour invoquer le service web ; <port> : spécifie l'emplacement actuel du service ; <service> : décrit un ensemble de points finaux du réseau.

Algorithm ***PseudoCurrentModel***

A. Find OWL-S Process Model that best mutch with the current situation
B. Extract two data lists from the file (owl-s)

Returns (List) EAList : set of ordered EA
(List) SeqList : set of used control structures

```
Function PseudoCurrentModel
Local EAList, SeqList, (String)DataSet, (Int) LineNb=0
Begin
(owl-s) CurrentSolution=LocalDiscovery(R) /*a discovery procedure (local) call specifications owl-s responding to the concerned request */
While(Mode=0) /*Mode is a global Var*/
If notEmpty(CurrentSolution) then /*Parse the file*/
While (NOF(CurrentSolution))
DataSet = Line(CurrentSolution, LineNb)
EAList := EAList ∪ extractEA(DataSet)
SeqList := SeqList ∪ extractSeq(DataSet)
End While
End While
Return EAList, SeqList
End PseudoCurrentModel
```

Figure 3. 29 Le mode de fonctionnement normal

5.2 Les autres algorithmes

Concernant le deuxième mode de fonctionnement, *ReplanningExecutionMode*, l'algorithme est déclenché lors de la détection d'un changement du contexte. Dans ce cas, l'ensemble des EAi (présélectionnées) est inapproprié. Le module « SequencePlanner » doit re-planifier les actions élémentaires adéquates à la nouvelle situation en activant la découverte et la sélection décrites ci-dessous. Ce fonctionnement demeure le même pour le troisième mode sauf que ce mécanisme commence de cette manière dés le début (ce fonctionnement sera identifié, entre autres, par une variable globale **Mode**=2).

5.2.1 Algorithmes de préparation des actions élémentaires

Initialement, les actions élémentaires destinées à être utilisées par le système d'adaptation en tant que services web atomiques ou composites commencent à se charger dans le référentiel temporaire (***T*** : **TempERegistry**). Cette présélection est basée sur une analyse des données contextuelles de l'environnement où se situe l'utilisateur, des données concernant ses activités habituelles, son profil, etc. Ces paramètres font l'objet de la première requête envoyée par le système (*InitRequest*). Ceci permet d'améliorer aussi bien les performances d'extraction et d'invocation des actions élémentaires que le temps d'exécution. En effet, se référer à chaque fois à un référentiel volumineux risque de ralentir le fonctionnement du système et donc d'altérer ses performances (Figure 3.30).

Algorithm (0) ***EAPreselection***

Find *EAi* that best match with the initial Request (InitRequest) of the adaptation system
Returns set of EA

```
1   Function EAPreselection(InitRequest IR)
2   Local FirstResult ;
3   Begin
4   For All (EA) Do
5   While (EA.hasCategory = IR.Category) OR
    (EA.hasActivity=IR.Activity)
    OR (EA.hasProfile=IR.Profile)
6   FirstResult := FirstResult ∪ {EA}
7   Return FirstResult
8   End EAPreselection
```

Figure 3. 30 Algorithme de présélection

L'étape suivante consiste à charger ces actions résultats dans le référentiel temporaire : **Reload** (FirstResult, T). Une fois que ces dites actions sont décrites et stockées dans le référentiel, il est possible de les rechercher et les récupérer en fonction des besoins du système d'adaptation.

L'étape postérieure consiste à sélectionner les prochaines actions élémentaires candidates pour répondre à la requête système et les ordonner en se basant sur les critères extraits à partir de cette requête. Ce fonctionnement est décrit par l'algorithme suivant (Figure 3.31). Notation : EAc = {EA ⊂ T, avec la même Catégorie}.

Algorithm (1) ***EASelection***

Find *EAi* that best match with the Request (SysRequest) of the adaptation system
Returns set of EA (Ordered)

```
1   Function EASelection(SysRequest SR)
2   Local CurrentResult ;
    Local PriorityDegree ;
3   Begin
    /*the algorithm begins by extracting any feature of
    priority on which he will order the EA*/
4   PriorityDegreeSys := extractPD(SR)
5   For All (EA ϵ EAc) Do
6   If ConceptSim(SR, EAi)
      Then
7     CurrentResult := CurrentResult ∪
      InsertOrdered(CurrentResult, EAi)
8   End if
9   End For
10  Return CurrentResult
11  End EASelection
```

Figure 3. 31 Algorithme de Sélection

Cette deuxième sélection est basée sur une comparaison sémantique des différents concepts, sachant que nous considérons que les *Inputs* et les *Outputs* se réfèrent impérativement aux concepts de l'ontologie de domaine. Comme nous l'avons déjà exprimé, cette phase se base sur le calcul de la similarité sémantique entre la requête système et les descriptions EA. Dans ce cadre, nous avons trouvé utile de se référer à la méthode proposée par (Ahmed, 2007) et appuyée par (Benaboud, 2012) (Benaboud, 2013) permettant de calculer la similarité sémantique de deux concepts X et Y : ***ConceptSim***(X,Y).

Pour effectuer la comparaison de deux concepts X et Y, quatre cas de figure s'imposent. Nous distinguons à ce niveau :

- **Cas1** : où les deux concepts sont similaires ou bien sont définis par la même classe (X=Y), par exemple la classe **Place** de l'ontologie de domaine d'une action élémentaire.
- **Cas2** : le concept X représente implicitement ou explicitement une sous-classe du concept Y (X<Y).
- **Cas3** : le concept Y représente implicitement ou explicitement une sous-classe du concept X (X>Y).
- **Cas4** : le concept X ne possède pas une relation d'association avec Y mais les deux concepts possèdent un concept commun (Parent) Z (X<>Y ; X,Y⊂ Z)

Considérons par exemple le cas de l'identification de l'action élémentaire ***CarbohydrateIngest*** qui peut convenablement correspondre à la requête système suivante (Figure 3.32) :

- ConceptSim (Place, PaCarbohydrateIngest) =1 (étant donné que PaCarbohydrateIngest est une instance de la classe Place),
- ConceptSim (Transition, BeginCarbohydrateIngest) =1 (étant donné que BeginCarbohydrateIngest est une instance de la classe Transition),

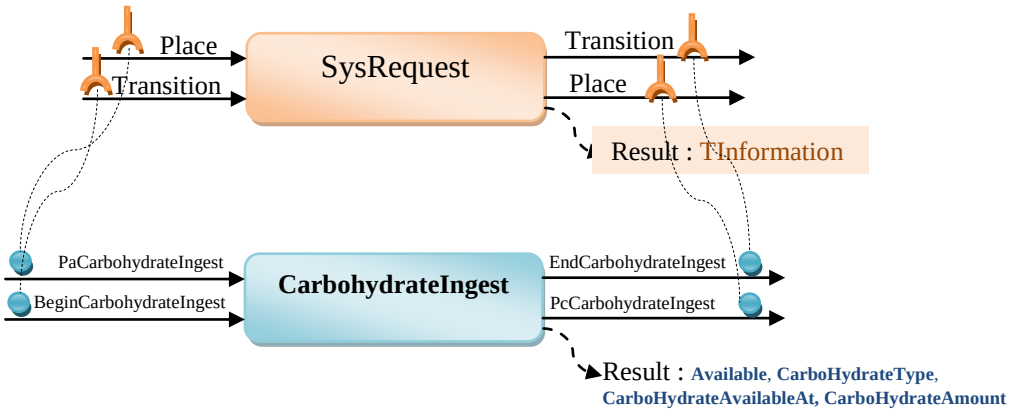


Figure 3. 32 Correspondance (*Matching*) entre la requête système et la spécification OWL-S d'une action élémentaire

5.2.2 Algorithmes de composition et exécution du modèle

Cet algorithme génère les services composites et ce, à partir de la description déclarative fournie par le composant ***EASelection***. Il se base sur des règles de composabilité pour déterminer dans quelles mesures deux EAs peuvent être composées (e.g. Sequentiel, Parallel).

- Règles syntaxiques : incluant des règles pour les types d'opérations possibles et pour des liaisons protocolaires entre les EAs (c'est à dire les *Bindings*).
- Règles sémantiques : incluant des règles régissant la compatibilité des messages échangés et la compatibilité des domaines sémantiques des services.

L'algorithme *Chainmaker* constitue le programme qui agit directement sur le fichier contenant le modèle d'exécution courant (CurrentModel). Ayant comme input l'action élémentaire courante et le mode d'ajout de cette action au modèle courant, ce module permet d'intégrer cette action dans le modèle global après avoir vérifié certaines propriétés (Figure 3.33).

Algorithm *Chainmaker*

It tries to update the model running by adding, deleting, or modifying an elementary action
Returns the current model in execution "CurrentModel.owl"

```
1   Function Chainmaker (CurrentEA,MMode)
2   Begin
3   While (MutEvent := False)
4   /* Open the owl file corresponding to the current
    elementary action */
    owlDoc = Load ("EA.owl ") ;
5   /* Merge two files => add selected content of the
    current action at the end of file CurrentModel*/
    Merge(CurrentModel, CurrentEA, MMode) ▷
    Function Call
6   FileStorage(CurrentModel) ▷ Function Call
7   Out(CurrentModel) /*File Out*/
8   End Function
```

Figure 3. 33 Algorithme de construction du modèle

Les changements dans l'environnement représentent l'une des sources d'information les plus importantes dont le système a besoin de disposer instantanément dans le cadre de son fonctionnement. La variable **MutEvent** représente une condition de franchissement de n'importe quelle transition. Une variable globale à tout le système dont le changement de la valeur est la cause d'un changement dans la situation actuelle et est en même temps la cause de l'interruption de l'exécution du modèle. Cette variable appartient aux *Preconditions* d'un service étant donné que les préconditions présentent les exigences à satisfaire pour garantir la bonne exécution d'un SW. Cette variable

indique n'importe quelles modifications produites par les effets d'une exécution de service. La reconnaissance de cette modification fait l'objet d'une perspective de ces travaux.

```
Algorithm  Merge

It tries to Merge two files
Returns  Updated file
1     Function  Merge(CurrentModel, CurrentEA, MMode)
2     Local     ContentToUpdate= ""<process:composedOf>"
      (String)
3     Begin
4     ReachEOF(CurrentModel) ;
      /*Reading file until reaching the end of the file :
      ("</process:composedOf></process:CompositeProcess>")*/
5     a return line and add the tag according to MMode
      Switch (MMode)
6      Begin
7     Case of  "Choice"
8     ContentToUpdate = ContentToUpadte + "<process:Choice>"
9     Case of  "Seq"
10    ContentToUpdate = ContentToUpadte
      + "<process:Sequence>"
11    Case of  "Par"
12    ContentToUpdate = ContentToUpadte + "<process:Parallel>"
13    ...
14    End Switch
15    ContentToUpadte=
      ContentToUpadte+"</process:composedOf>"
16    End Function
```

Figure 3. 34 Fonction de fusion

6. Discussion et contribution

Cette partie permet de mettre en œuvre où se situent les travaux de cette thèse, d'une part en citant quelques travaux connexes faisant intervenir à la fois les ontologies de services et les RdPs. D'une autre part, nous citons dans cette partie les principaux domaines de contribution de ces travaux.

6.1Travaux connexes

Dans cette section, nous avons choisi de citer des travaux qui ont fait intervenir à la fois le formalisme de modélisation des réseaux de Petri et la technologie de spécification OWL-S afin d'accomplir un but donné. Il faut d'abord noter que les réseaux de Petri avaient généralement été utilisés comme un mécanisme d'analyse formelle et ils ont été particulièrement considérés comme un outil de modélisation efficace (Khadka, 2010).

Cependant, combiné avec l'ontologie des services spécifiée avec OWL-S, le formalisme des réseaux de Petri fournit une sémantique opérationnelle à OWL-S, car il définit les spécifications formelles et exécutables à analyser, à simuler, à vérifier et à valider les spécifications d'OWL-S (Khadka, 2010) (Alonso, 2004). En particulier, les réseaux de Petri sont destinés à analyser le modèle de composition de services Web et de calculer la performance du service. Par exemple, (Brogi, 2009) fournit un traducteur d'une description OWL-S vers un réseau de Petri qui supporte l'analyse formel des services OWL-S grâce à certaines caractéristiques de ces réseaux.

En plus de l'analyse des performances, les réseaux de Petri objets (Harel, 2003) ont été spécifiquement utilisés pour la simulation et la vérification de service (workflow) tout en proposant des règles pour déduire des spécifications OWL-S des spécifications PNO. Un autre exemple porte sur la vérification de l'exactitude de la composition des services web OWL-S. La vérification est principalement fondée sur une logique mathématique rigide des réseaux de Petri (Yu, 2010).

Un autre mode d'utilisation des réseaux de Petri dans ce contexte est décrit dans (Vidal, 2011), où une approche pour la modélisation de la chorégraphie des services web OWL-S a été présentée. Chaque commande de la construction de la chorégraphie OWL-S est représentée par un modèle réseaux de Petri qui décrit les sémantiques opérationnelles. Un service web est modélisé sous forme de réseau de Petri en associant les opérations aux transitions et les états de service en places (Hamadi et al., 2003). Ensuite, pour composer des services web, il suffit d'appliquer des opérateurs de composition aux réseaux de Petri représentant les dits services web ; ceci facilite la définition des opérateurs de composition, ainsi que l'analyse et la vérification de certaines propriétés comme l'accessibilité et l'absence de cycles. Cependant, à notre connaissance, aucune approche de la littérature ne fait usage de l'OWL-S pour modéliser l'interaction homme-système dans un environnement ubiquitaire.

6.2Domaines de contribution

L'exécution indépendante de la plateforme et l'interopérabilité : Le système d'adaptation (considéré le client des services) est situé sur la même machine ou sur une machine distante. Il devra, lui aussi, respecter certains impératifs permettant de répondre au problème principal d'interopérabilité. Cette dernière concerne non seulement le codage des données, mais également les plateformes, elle est présente à plusieurs niveaux. Tout d'abord, entre les services eux-mêmes : en effet, rien n'oblige à utiliser la même plateforme entre les différents services formant une application à part entière. Ensuite, au niveau des clients et des services : l'architecture logicielle et matérielle peut être totalement différente, l'essentiel est la mise à disposition des protocoles des services web sur ces architectures, indépendamment du langage et des logiciels utilisés. L'utilisation des services web pour effectuer l'adaptation facilite l'interopérabilité en bénéficiant de protocoles standards d'échange de données. En effet, le langage qui sous-

tend ces protocoles est XML, considéré généralement comme un moyen très efficace pour l'interopérabilité des systèmes.

La dynamicité et l'automatisation de la modélisation : l'objectif ultime de l'approche Services web est de transformer le Web en un dispositif distribué de calcul où les programmes (services) peuvent interagir de manière intelligente en étant capables de se découvrir automatiquement, de négocier entre eux et de se composer en des services plus complexes. En d'autres termes, l'idée poursuivie avec les Services web, est de mieux exploiter les technologies de l'Internet en substituant, autant que possible, les humains qui réalisent actuellement un certain nombre de tâches, par des machines en vue de permettre une découverte et/ou une composition automatique de services. L'automatisation est donc un concept clé qui doit être présent à chaque étape du processus de conception et de mise en œuvre des Services web.

Utilisation des ontologies : les ontologies permettent de standardiser le vocabulaire, d'uniformiser le langage d'échange entre différents acteurs, de comparer différents systèmes, de structurer la connaissance pour simplifier l'analyse et la synthèse des connaissances d'un domaine, et de spécifier un contexte. Pour Mitsuru Ikeda (Ikeda, 1998), les ontologies peuvent servir de théorie pour supporter la recherche d'information tout comme les mathématiques apportent une base théorique à la physique. S'appuyant sur les ontologies, le développement du Web sémantique ouvre de nouvelles possibilités et des défis à la conception d'une nouvelle génération de systèmes adaptatifs, rendant possible la modélisation du profil des utilisateurs (Razmerita, 2003). Egalement le Web sémantique fait référence à la vision de l'internet de demain. Il est considéré comme un espace étendu dans lequel les utilisateurs et les machines peuvent échanger de ressources permettant ainsi une exploitation de grands volumes d'informations et de services. Dans le contexte d'adaptation d'information à l'utilisateur, l'ontologie est devenue une solution incontournable. L'ontologie permet la construction de modèles de connaissance complexes, qu'il est possible d'utiliser pour modéliser à la fois les utilisateurs et le domaine de manière intelligible pour tout type d'utilisations et d'acteurs. Basée sur les outils du Web sémantique, l'adaptation du Web à l'utilisateur devrait permettre le développement d'un Web plus évolué, que l'on peut nommer le Web adaptatif.

7. Conclusion

A travers ce présent travail nous avons veillé à prouver que la composition dynamique des services web en tant que technologie est perçue comme un outil précieux pour la composition d'un modèle et son exécution d'une manière dynamique. En outre, la description OWL-S qui a été adoptée pour représenter les modèles RdPI élémentaires et composites s'est avérée une méthode prometteuse.

Une approche de modélisation dynamique a été proposée et détaillée dans ce présent chapitre. L'architecture intégrale du système proposé est regroupé dans la figure 3.35. Le but principal consiste à contrôler l'interaction en temps réel des utilisateurs se

situant dans un environnement ubiquitaire. Pour s’assurer que les utilisateurs disposent de l'information la plus appropriée en temps réel nous devons garantir que le système d’adaptation possède pareillement le modèle le plus approprié afin de pourvoir cette information. Le prochain chapitre met en valeur ces objectifs à travers l’expérimentation et l’évaluation de l’approche d’adaptation proposée.

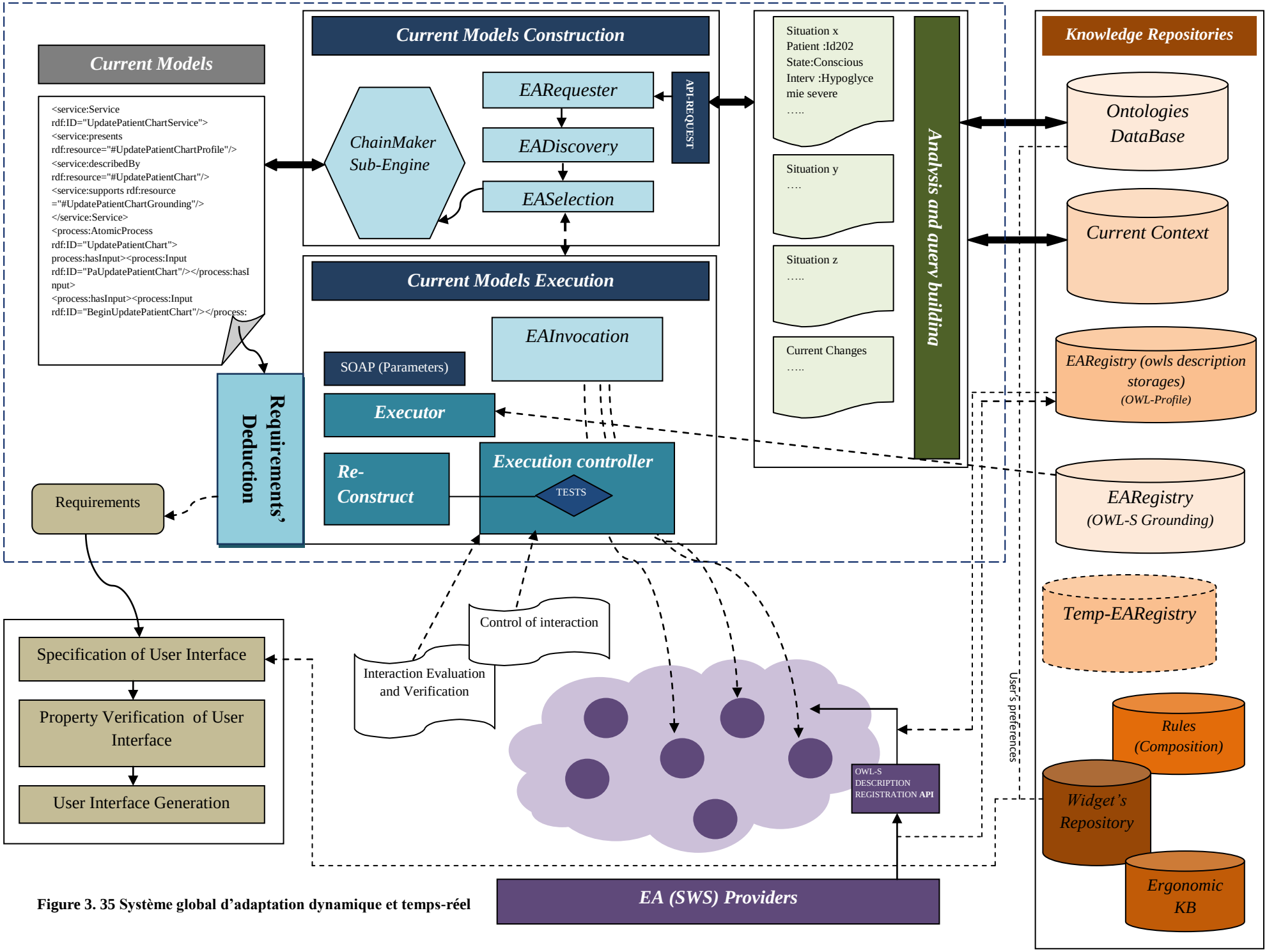


Figure 3. 35 Système global d'adaptation dynamique et temps-réel

Chapitre 4

EXPERIMENTATION ET EVALUATION DE L'APPROCHE PROPOSEE

1. Introduction

La nécessaire étude de faisabilité effective des travaux nous conduit à regrouper dans ce chapitre les éléments relevant de notre démarche de recherche et de sa mise en application. Nous y représentons en particulier les différentes facettes de l'évaluation des propositions conceptuelles de ce travail. Ce chapitre est organisé en trois grandes parties. Le but de la première partie consiste à préparer les données et les connaissances constituant la principale source du système à mettre en place. L'application de la stratégie d'adaptation, définie et détaillée dans le chapitre précédent est incarnée dans la deuxième partie. La dernière partie est une partie d'évaluation qui présente une interprétation des résultats de tests correspondants à chaque scénario d'utilisation étudié.

2. Plateforme utilisée pour l'expérimentation de l'approche proposée

En prenant en considération les propositions conceptuelles spécifiées dans les chapitres 2et 3, la plateforme suivante (Figure 4.1) nous donne une idée sur le support technologique utilisé pour l'expérimentation de ces propositions. En fait, la plateforme d'expérimentation déploie les choix technologiques suivants[32] :

- ***Eclipse :*** Le choix de l'environnement de développement s'est porté sur l'environnement Eclipse. En particulier, nous avons opté pour « Eclipse IDE for Java EE Developers » avec le package « Eclipse Galileo Sr2 Packages (ou Eclipse 3.5)[33] ». Eclipse IDE est un environnement de développement intégré libre, extensible, universel et polyvalent. A plus forte raison, il permet potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation.

- ***Java*** : Le choix du langage de programmation s'est porté sur le langage JAVA vu ses nombreuses caractéristiques. L'un des avantages évidents de ce langage est une bibliothèque d'exécution qui se veut indépendante de la plateforme. Cette propriété est indispensable pour une programmation sur Internet.

- ***Apache-TomCat***[34] : Apache-Tomcat est le serveur d'application Java du projet Jakarta de la fondation Apache. Ce serveur libre, sous licence Apache, permet d'exécuter des applications Web développées avec les technologies Java. Apache-Tomcat représente aujourd'hui un serveur de référence pour les technologies Java EE servlets et JSP. Tomcat est un moteur de Servlets fiable, évolutif et adapté à l'utilisation professionnelle. Il est actuellement utilisé dans le monde entier et mis en application au sein de domaines très variés.

[32] La configuration matérielle présente les caractéristiques suivantes : le système développé est testé sur une machine fonctionnant avec un système d'exploitation 32 bits (Windows7) avec les spécificités physiques suivantes : Intel ® Pentium (R) CPU @2.00GhZ, mémoire (RAM) 2.00Go
[33] http://www.eclipse.org/downloads/packages/release/galileo/sr2
[34] http://tomcat.apache.org/download-60.cgi

- ***Protégé***[35] : représente un des plus performants éditeurs d'ontologie et ce grâce au plugin OWL qui permet de développer des ontologies OWL. Il est très reconnu et a été exploité dans de nombreuses applications académiques et industrielles. Protégé est un logiciel mature qui propose une interface conviviale, pratique et intuitive. En outre, Protégé est un outil libre et open source qui a été développé par l'université de Stanford. Il permet de définir des vocabulaires ou des thésaurus et de représenter des connaissances par le biais d'ontologies. Il supporte les langages standards du W3C tels que XML, RDF et OWL. Le graphe représentant l'ontologie peut être visualisé grâce au plugin OWLViz. Il est également possible de faire des requêtes sur les ontologies avec le plugin Queries ou avec SPARQL.

- ***Axis2 :*** Axis est un ensemble de logiciels créés par Apache Software Foundation, qui vise à faciliter le développement de services Web en technologie SOAP. Il offre un environnement pouvant soit fonctionner comme un serveur SOAP indépendant, soit comme un plug-in de moteur de servlet (en particulier Tomcat). Il possède une API pour développer des services web SOAP RPC ou à base de messages SOAP ainsi que des outils pour déployer, tester et contrôler des services web. Il permet également la sérialisation/désérialisation automatique d'objets Java dans des messages SOAP. En particulier, il offre des outils pour **créer automatiquement les WSDL correspondant à des classes Java**, ou inversement, pour créer les classes Java sur la base d'un WSDL (classe proxy en quelque sorte, qui fait le lien entre l'application Java cliente et le service distant). En plus, il supporte différentes couches de transport : HTTP, FTP, SMTP, POP et IMAP... Axis2 est une réécriture complète qui a pour objectif d'être plus efficace, plus modulaire et plus orienté XML que la version précédente. Un certain nombre de modules sont en cours de développement concernant la sécurité, les transactions, etc.

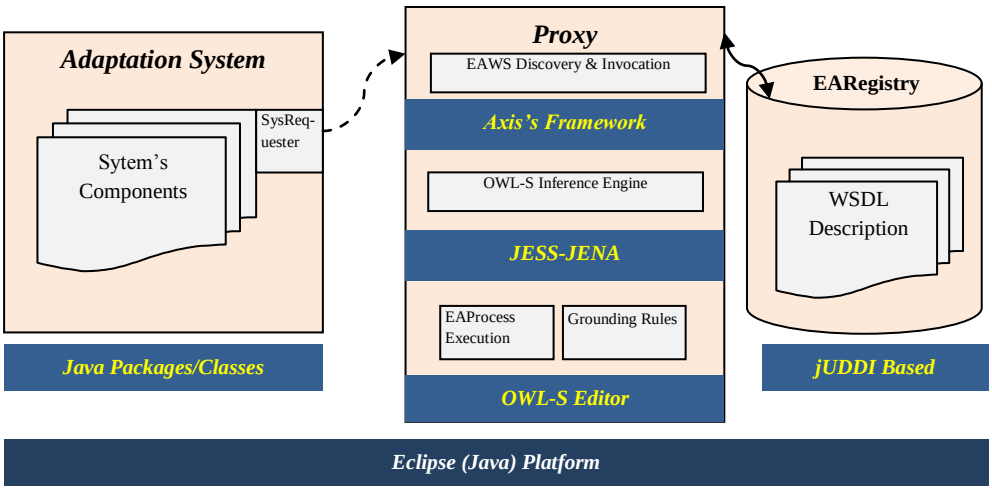


Figure 4. 1 Plateformes d'exécution

[35] http://protege.stanford.edu/

A travers la figure 4.2, nous rappelons les différentes facettes du système d'adaptation, en particulier, les utilisateurs se situant dans l'environnement ubiquitaires. Ces utilisateurs disposent généralement de différents appareils mobiles (iPhone, iPad, Pc-Portable) avec des fonctionnalités et des applications spécifiques assurant l'interaction avec le système. Ce dernier est sensé renvoyer des informations pertinentes, en fonction des besoins spécifiques de chaque utilisateur en temps réel.

Nous supposons également la présence des capteurs comme « MobiDetectSys » qui sert de support à l'observation en temps réel de l'utilisateur en situation d'interaction et à la capture de l'activité en cours. Un capteur biologique « BioSensorSys » qui capte et enregistre des données diverses sur le patient, en particulier le taux du glucose s'avère nécessaire. Ces dits capteurs représentent la principale source d'information pour notre système d'adaptation.

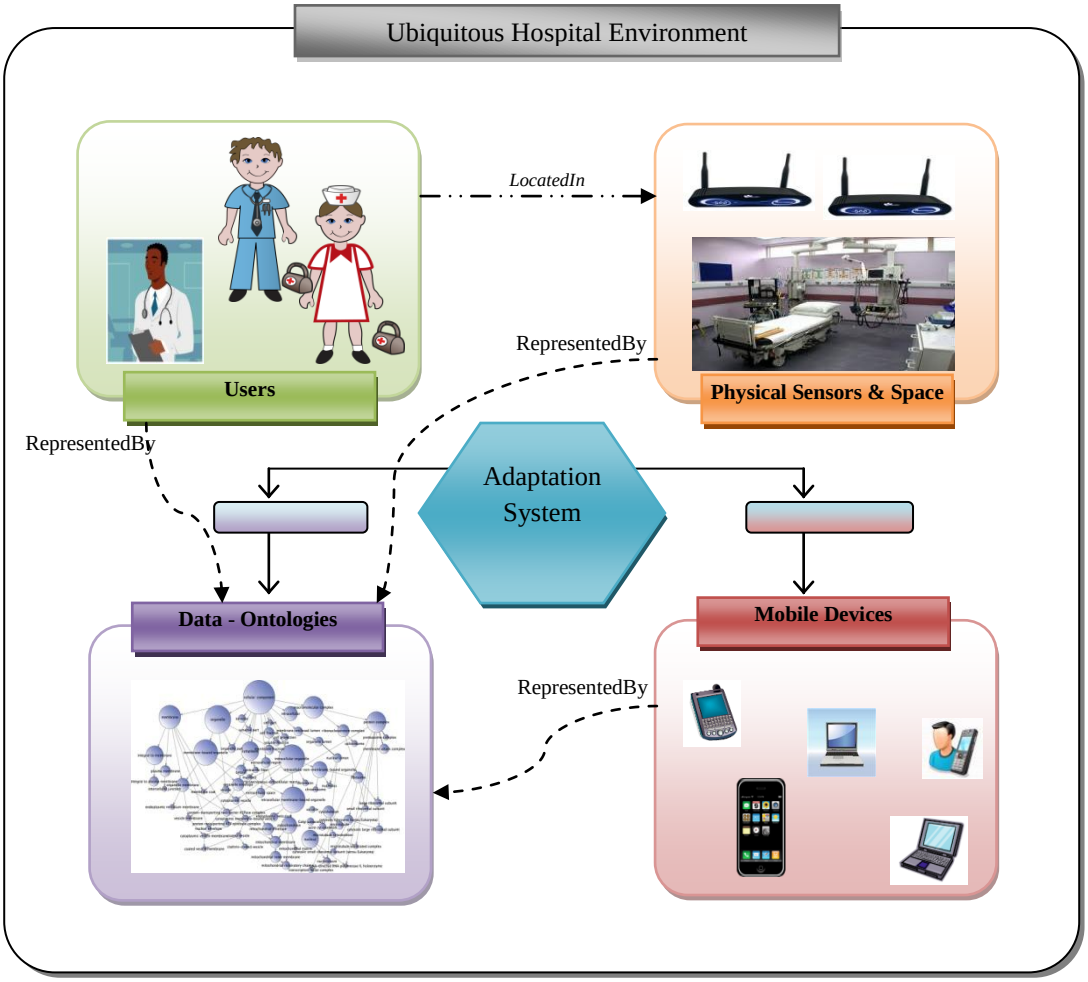


Figure 4. 2 Les différentes facettes de la simulation d'un hôpital «ubiquitaire»

3. Implémentation de la partie « Knowledge Repositories »

Cette section explore les données à consommer par le système et sur la base desquelles il fonctionne. Il s'agit d'une étape primordiale afin de mener l'évaluation des approches conceptuelles exposées dans ce travail. Nous avons considéré, à cet effet, les

composantes ontologiques suivantes (Figure 4.3) tout en notant qu'uniquement celles colorées en rouge étaient mises en place. Dans cette figure nous avons regroupé toutes les ontologies envisagées sous une ontologie globale « ***UbiquitousHospitalEnvironment-ONT*** ». Nous distinguons quatre composantes ontologiques nécessaires à la description d'un tel environnement (hôpital ou clinique ubiquitaire), à savoir :

- La composante « *HealthONT* » qui regroupe tous les concepts en relation avec la santé en générale ;
- La composante « *HospitalManagmentONT* » qui décrit les cadres gestionnaires et administratifs d'une clinique,
- La composante « *HopitalEquipmentONT* », où tout équipement médical est représenté par cette ontologie. Enfin, le système maintient un modèle partagé du contexte basé sur une ontologie. Ce modèle peut être partagé par tous les dispositifs et applications dans l'environnement « *EnvironmentalContextual-DataClassification* » qui contient, entre autres, la composante ontologique qui particulièrement nous intéresse : « *CurrentContextofUse* ».

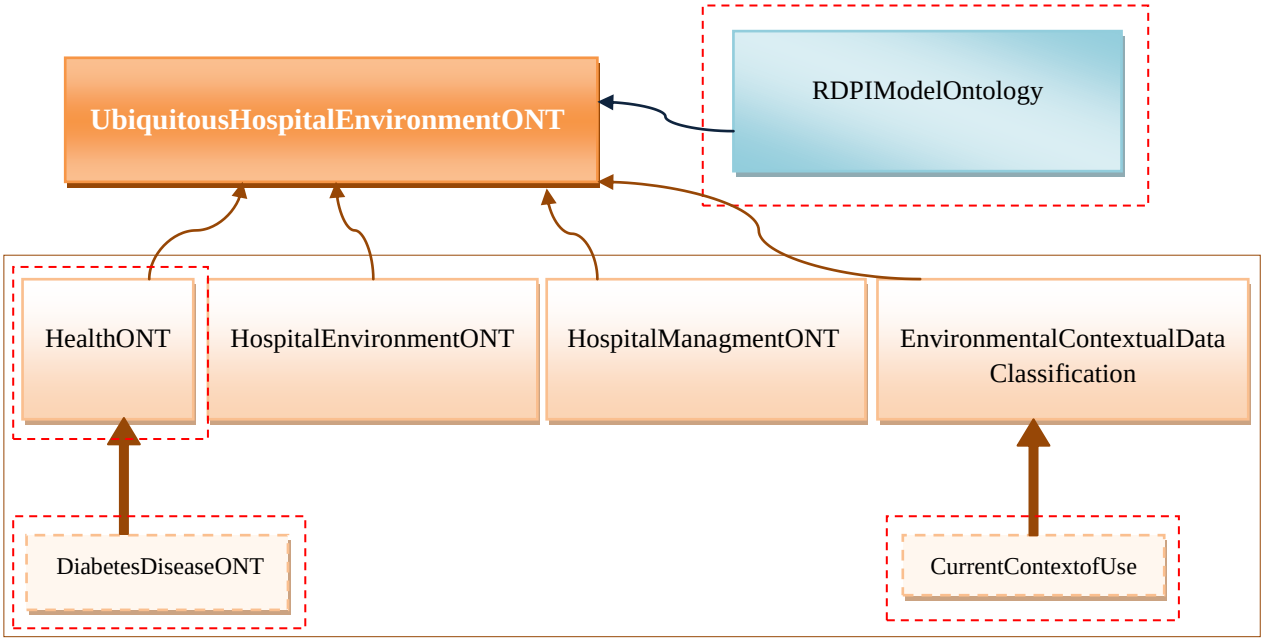


Figure 4. 3 Les Différentes Composantes Ontologiques

Tout d'abord, il faut signaler que la composante ontologique applicative a été élaborée avec l'assistance d'un médecin spécialiste[36]. Nous nous sommes également basées sur de nombreuses ressources dont (Tremblay, 2006) (Nagati, 2001) décrivant des notions basiques relatives en particulier à la *diabétologie* et en relation avec *l'Endocrinologie* en général. La création de l'ontologie est passée par les étapes

[36] Dr. Myrvat Kamoun : Spécialiste en Endocrinologie-Diabétologie et maladies métaboliques depuis 1997. Assistante hospitalo-universitaire au service de Médecine interne Hôpital Charles Nicolle de Tunis de Dec 1998 à Nov 2003. Assistante HU au service d'Endocrinologie-Diabétologie La Rabta de Tunis de Novembre 2003 à Décembre 2006. Professeur Agrégée de Décembre 2006 à Février 2011. Programme national diabète chez DSSB.

présentées dans la figure 4.4 tout en spécifiant les résultats de chaque étape (Chourabi, 2009). Tout d'abord, il faut commencer par la création de la partie déclarative de l'ontologie. Cette partie doit débuter par regrouper le maximum possible de connaissances sur le domaine de préférence ainsi que récupérer ces données à partir d'un expert métier. Nous partons donc de l'hypothèse de disposer d'un document informel regroupant les données et les connaissances nécessaires pour la construction de l'ontologie. En effet, l'étape suivante consiste à établir le modèle conceptuel d'où le terme conceptualisation. L'ontologisation qui consiste à spécialiser les ressources pour l'étude de cas traité constitue l'étape qui succède à la conceptualisation.

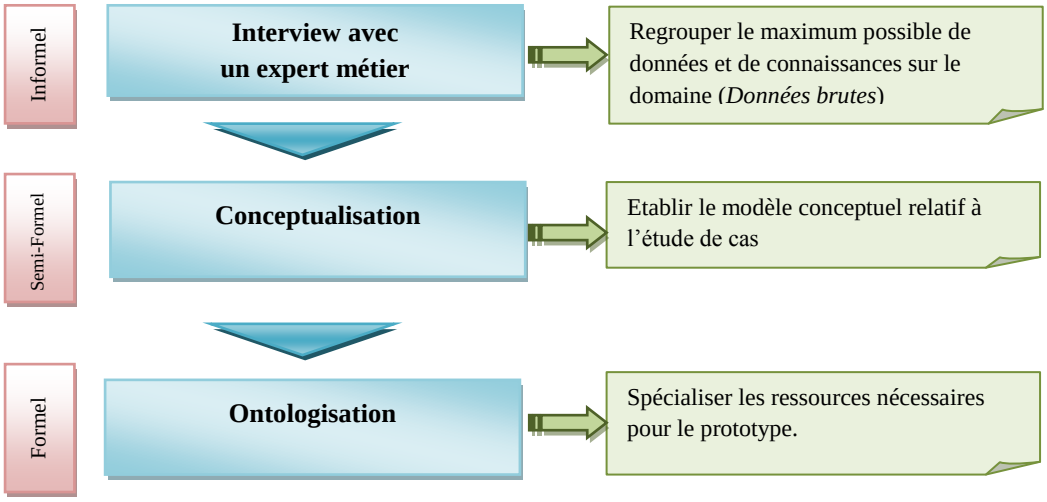


Figure 4. 4 Démarche pour la création et l'opérationnalisation de l'ontologie

L'élaboration écrite d'une première version de l'ontologie se fait au cours de la première phase. La question posée porte essentiellement sur la construction d'une base de connaissances cohérente. Comme première étape, il faut commencer par systématiser les connaissances métiers considérées dans le processus d'analyse des besoins et des spécificités d'un service d'Endocrinologie.

3.1 Conceptualisation des données contextuelles spécifiques au cas d'étude

La conceptualisation est une structure intentionnelle qui code les règles implicites contraignant la structure d'une partie de la réalité ou bien l'union d'un ensemble d'objets existant dans un monde donné et des relations qui existent entre eux (Sowa, 2000). Ce paragraphe décrit les différentes étapes fondamentales aboutissant à créer l'ontologie représentant le contexte d'exécution et le contenu des données. Cette ontologie est généralement présentée en deux parties : déclarative et impérative. Dans notre cas, le point de départ pour la construction de la partie déclarative de l'ontologie est le diagramme de classes. En effet, une première étape de conception doit être établie. Nous

devons y décrire les principales classes[37], propriétés et relations adaptées à la situation générale du prototype du système à mettre en place. Ensuite, nous avons implémenté cette ontologie à l'aide d'un outil approprié. L'outil *Protégé* a été choisi grâce à certaines caractéristiques qui s'avèrent les mieux adoptées à notre application.

3.1.1 Modèle Conceptuel de l'ontologie HealthONT

La figure 4.5 montre le modèle d'ontologie spécifié à l'aide de la représentation graphique basée sur UML. Cette figure dévoile en particulier les classes et les sous-classes basiques pour notre étude de cas. Afin de mener à bien la phase de conceptualisation des connaissances contextuelles, nous avons commencé par analyser les besoins généraux d'un département ou un service dans un hôpital ainsi que les besoins particuliers d'un service d'Endocrinologie. Le but est d'aboutir à identifier les différentes parties et entités de ce système (classes et sous-classes) ainsi que leurs interactions dans le système (relations).

Dans ce modèle les principales composantes d'un hôpital ubiquitaire ont été regroupées (comme par exemple : les départements, le personnel, les patients, les laboratoires…). Des informations nécessaires pour incarner les différentes facettes relatives au cas étudié ont été également identifiées (comme par exemple les informations physiologiques relatives à chaque patient).

3.1.2 Modèle Conceptuel de l'ontologie DiabetesDiseaseONT

La deuxième étape représente les caractéristiques ainsi que les paramètres décrivant une personne diabétique. Nous y explorons les différentes facettes d'une maladie de diabète. Nous obtenons enfin une couverture des différentes typologies des données à consommer par le système. Deux types de diabètes ont été dégagés (Nagati, 2001) : le type 1 et le type 2. Parfois, le diabète se développe aussi pendant la grossesse (diabète gestationnel). Le diabète de type 1 se manifeste soit dès l'enfance, à l'adolescence ou chez les jeunes adultes. Il se caractérise par l'absence totale de la production d'insuline.

Les personnes diabétiques de type 1 dépendent d'injections quotidiennes d'insuline pour vivre. Le diabète de type 2 se manifeste beaucoup plus tard dans la vie, généralement après l'âge de 40 ans. Les symptômes du diabète ne se présentent pas tous de la même manière ni avec la même intensité. La superclasse dans cette ontologie est la classe « *DiabeteDisease*», cette classe qui est dérivée de la classe « *Disease* » regroupe différentes propriétés relatives à cette maladie. Le diagramme de la figure 4.6 présente entre autres les diagnostics et les traitements associés.

[37] Les classes dans l'ontologie sont considérées habituellement comme des ensembles qui regroupent les individus. Généralement les classes sont organisées dans des taxonomies, c'est-à-dire sous une hiérarchie de classes (classes mères, sous-classes…). Le terme concept peut également être utilisé à la place de classe, en fait les classes sont des représentations concrètes de concepts.

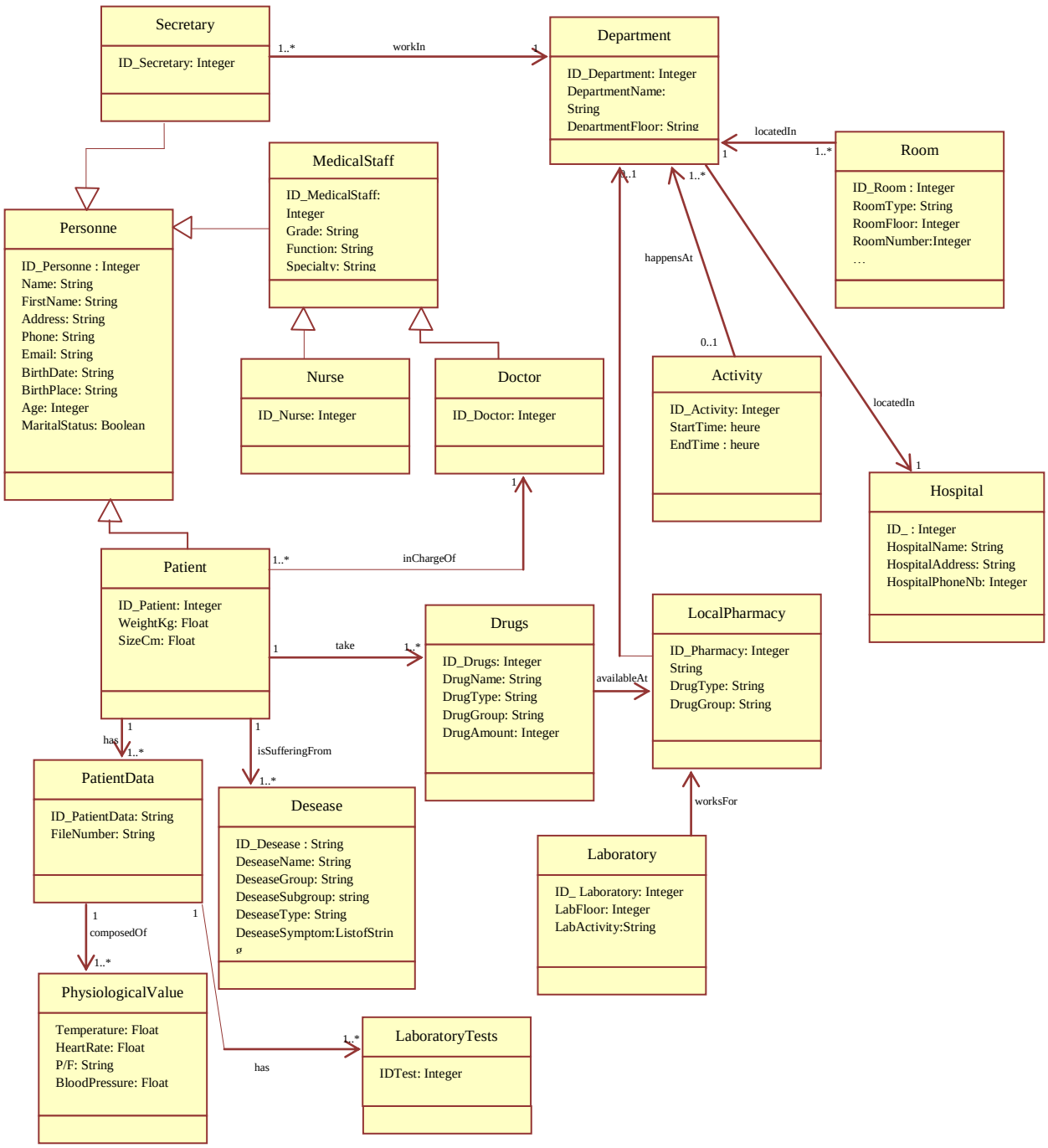


Figure 4. 5 Modèle Conceptuel de l'ontologie HealthONT

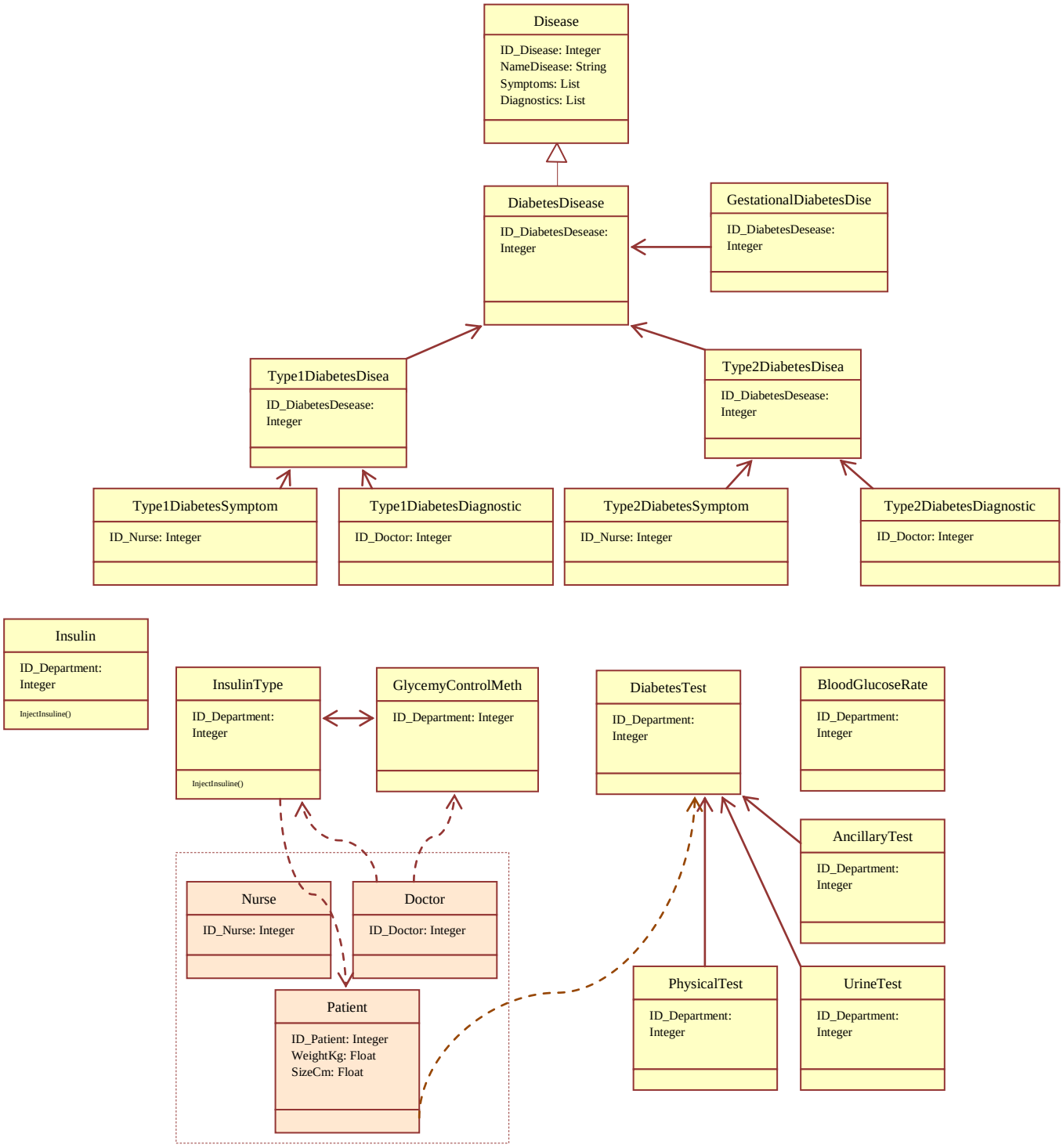


Figure 4. 6 Modèle Conceptuel de l'ontologie de la maladie de diabète DiabetesDiseaseONT

3.2 Ontologisation

Après une phase d'analyse des données disponibles pour construire des modèles métiers de la gestion d'un service endocrinologie qui représente l'environnement ubiquitaire, l'étape suivante consiste à transformer ces modèles en une ontologie formalisée en OWL. Nous y donnons, en un premier lieu, un extrait du graphe de l'ontologie «HealthONT». Décrite sous la forme d'un réseau sémantique, l'ontologie est

composée de nœuds qui représentent les concepts reliés par des arcs, exprimant la relation entre ces concepts. La figure 4.7 met en évidence la partie développée et constitue le noyau de notre application sous forme de hiérarchie de classes et de propriétés. En deuxième lieu, nous avons exposé à travers la figure 4.8 un extrait de l'ontologie « DiabetONT » regroupant particulièrement les concepts identifiés par l'expert.

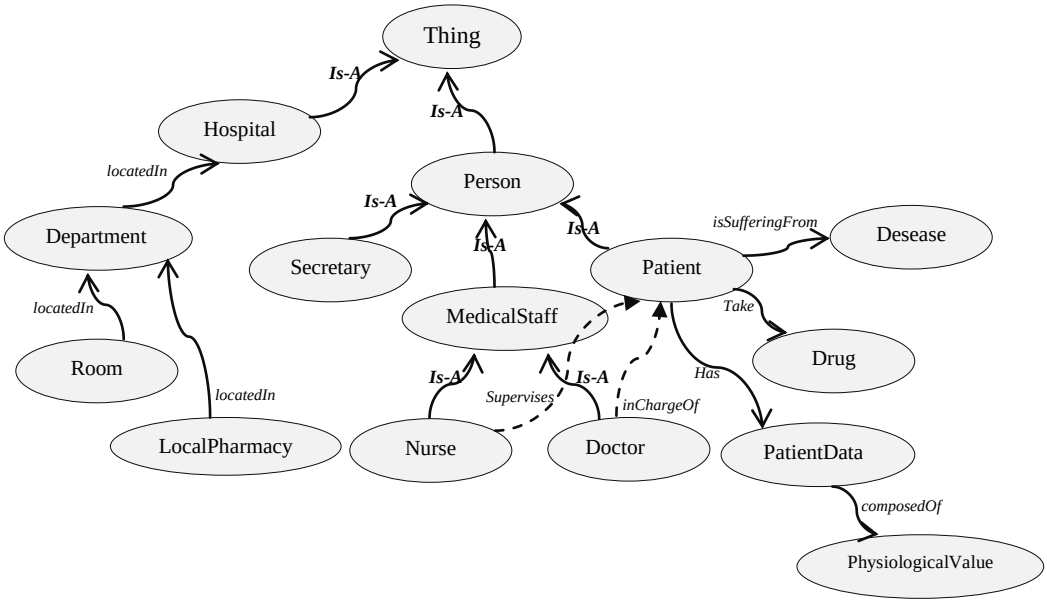


Figure 4. 7 Extrait de l'ontologie HealthONT

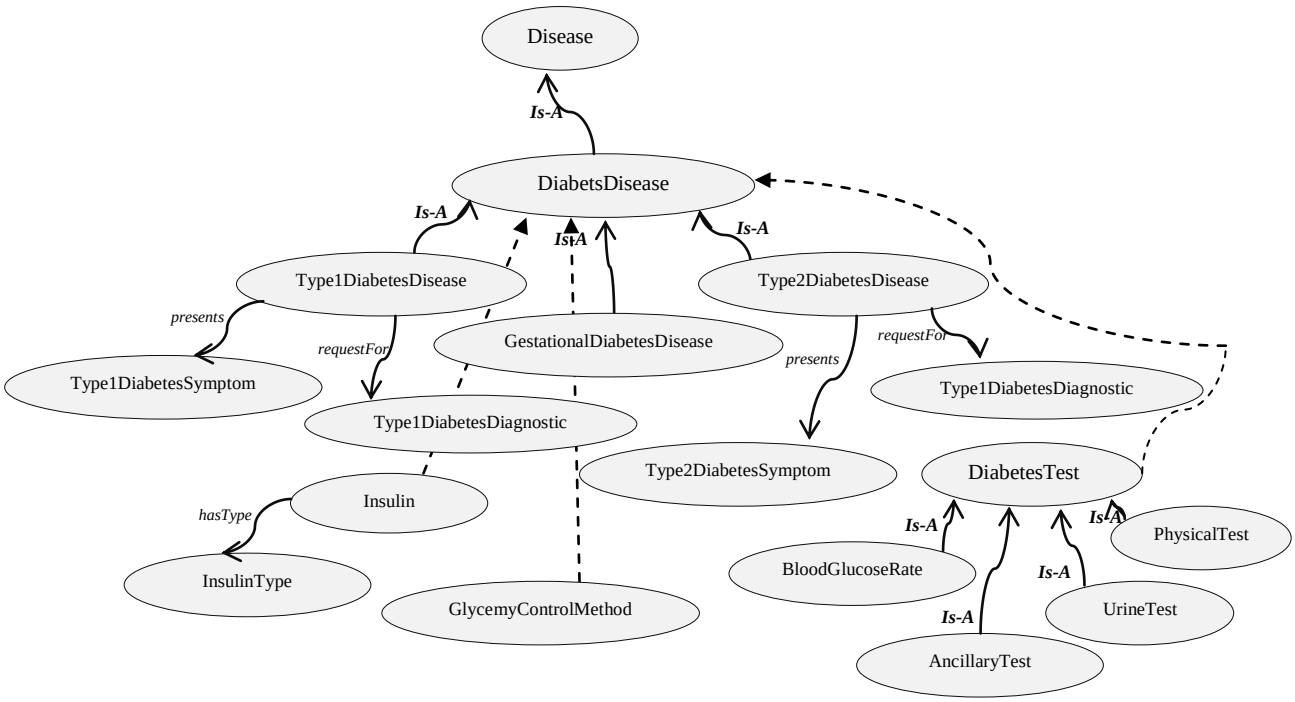


Figure 4. 8 Extrait de l'ontologie de la facette contextuelle de la maladie de diabète (DIABETOnt)

Identification des propriétés et des individus

Une ontologie OWL est constituée essentiellement d'individus qui représentent les entités du domaine étudié, les propriétés et les classes. Les ***individus,*** appelés également les instances, représentent les objets spécifiques du domaine étudié. La figure 4.9 représente quelques individus se situant dans notre environnement ubiquitaire. La partie supérieure « TBox » symbolise les définitions des notions basiques ou dérivées ainsi que la manière dont elles sont reliées entre elles. Ces informations sont considérées génériques ou globales, vraies dans tous les modèles et pour tous les individus. Concernant la partie inférieure « ABox », elle dévoile des informations considéreés comme spécifiques ou locales mais vraies pour certains individus particuliers. Une ***propriété*** est une relation binaire entre individus, en général elle représente un type d'interaction entre les concepts du domaine en question. Elle peut être définie formellement comme n'importe quel sous-ensemble d'un produit de n-ensemble : R dans C1 x C2 x... x Cn. Les propriétés sont connues dans la logique sous le nom de Rôle et dans la représentation UML sous le nom de Relations. La figure 4.9 montre quelques propriétés qui joignent des individus. Par exemple la propriété « *Supervise* » relie l'individu « *Ali* » qui est un infirmier avec l'individu « *Sonia* » qui est une patiente. Une propriété peut avoir un inverse, exemple « *IsSupervisedBy* » est l'inverse de « *Supervise* ». Il existe deux principaux types de propriété, les *propriétés d'objets* et les *propriétés de type de données*. Les propriétés d'objets représentent une relation entre deux individus (Ex. *inChargeOf*), alors que les propriétés de type de données décrivent une relation entre individus et valeurs de données (Ex. *hasAge*).

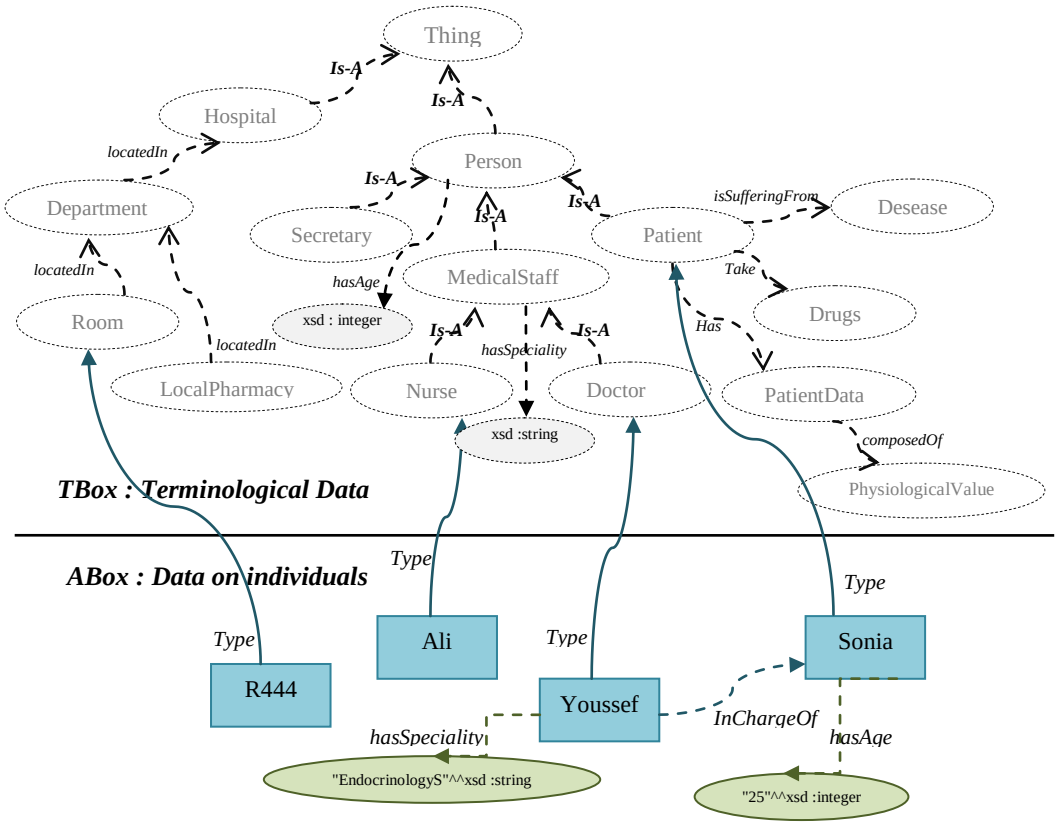


Figure 4.9 Définition de quelques individus et quelques propriétés

3.3 Opérationnalisation

Il s'agit de la mise en œuvre de l'ontologie. Comme nous l'avons mentionné, nous avons utilisé l'éditeur d'ontologie « Protégé» pour définir visuellement toute la partie déclarative de l'ontologie. Les figures 4.10, 4.11 (a et b) présentent un extrait des nomenclatures et les implémentations relatives aux classes, aux propriétés et aux individus préalablement mentionnés.

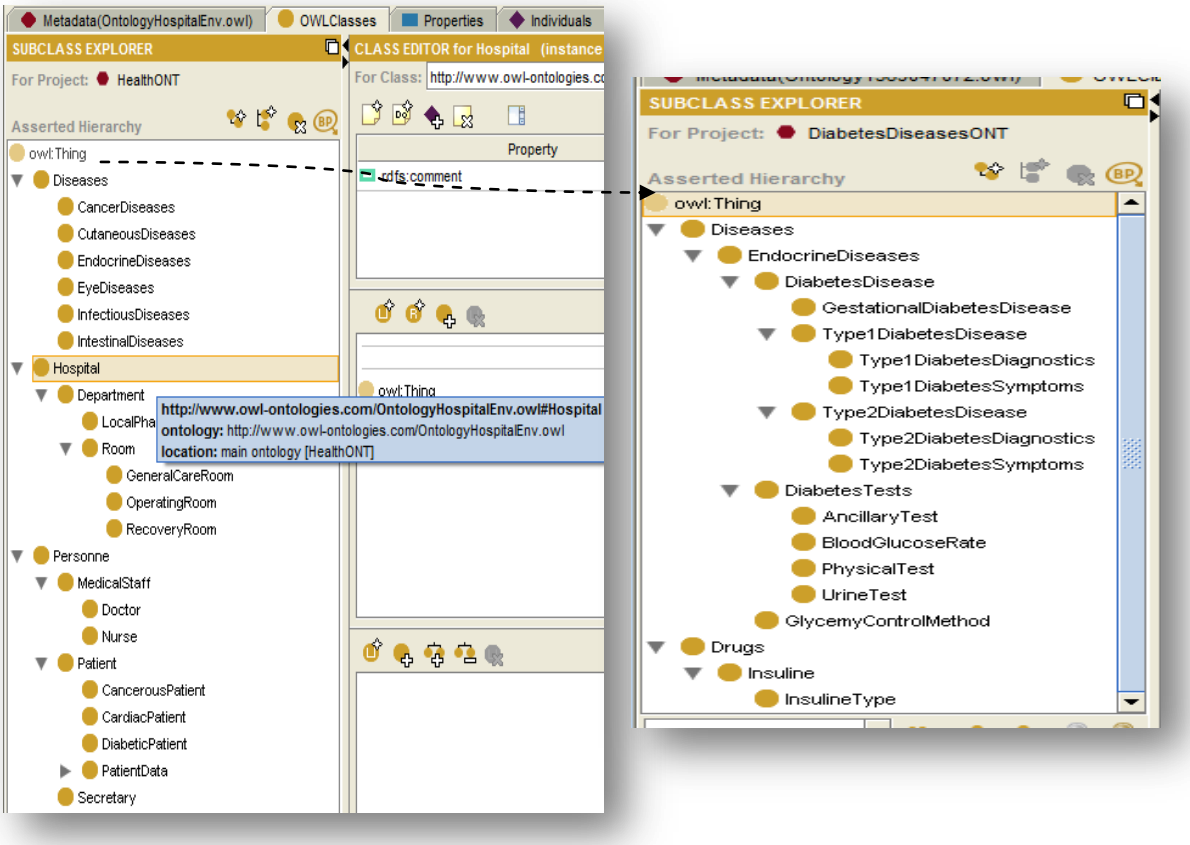


Figure 4. 10 Définitions des classes et sous-classes de la facette contextuelle

La figure 4.10 montre l'éditeur Protégé et le « plugin » OWL : à gauche nous présentons la hiérarchie des classes décrivant l'ontologie « HealthONT », à droite nous présentons la hiérarchie des classes décrivant l'ontologie « DiabetesDiseasesONT ». Quelques propriétés et quelques valeurs des instances sont présentées ci-dessous.

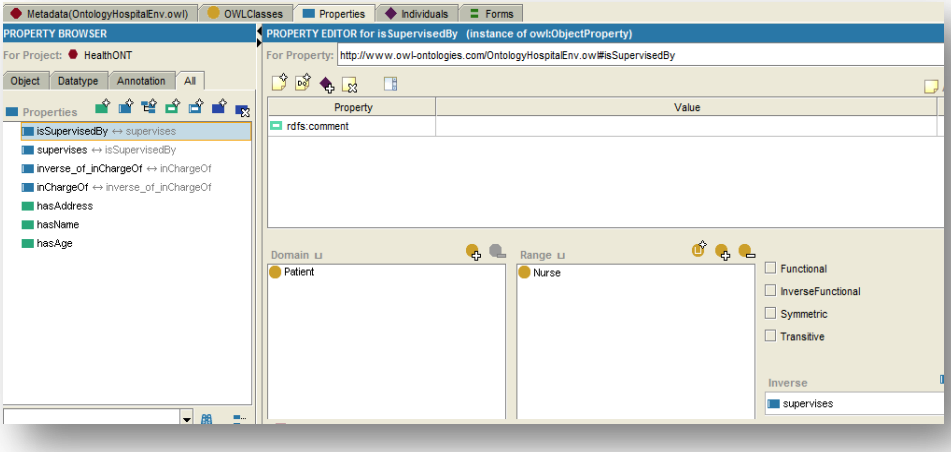


Figure 4. 11(a) Définitions de quelques propriétés de types et propriétés d'objets

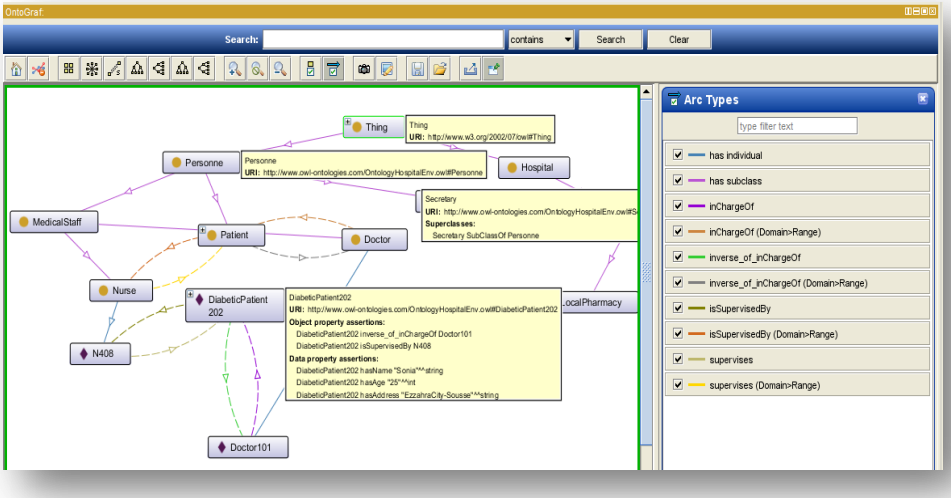


Figure 4.11 (b) Représentation graphique des propriétés et des individus

Les figures 4.12 (a et b) représentent des extraits de l'ontologie développée respectivement. Elles montrent la représentation des classes et sous-classes (a) ainsi que la représentation des individus et des propriétés (b).

```
<!-- http://www.owl-ontologies.com/OntologyHospitalEnv.owl#Patient -->
  <owl:Class rdf:about="http://www.owl-ontologies.com/OntologyHospitalEnv.owl#Patient">
    <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/OntologyHospitalEnv.owl#Personne"/>
  </owl:Class>
...
<!-- http://www.owl-ontologies.com/OntologyHospitalEnv.owl#DiabeticPatient -->
  <owl:Class rdf:about="http://www.owl-ontologies.com/OntologyHospitalEnv.owl#DiabeticPatient">
    <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/OntologyHospitalEnv.owl#Patient"/>
  </owl:Class>
```

Figure 4. 12 (a) Extrait OWL de l'ontologie globale (définition de classes et sous-classes)

```
<!-- http://www.owl-ontologies.com/OntologyHospitalEnv.owl#DiabeticPatient202 -->
<owl:NamedIndividual rdf:about="http://www.owl-ontologies.com/OntologyHospitalEnv.owl#DiabeticPatient202">
  <rdf:type rdf:resource="http://www.owl-ontologies.com/OntologyHospitalEnv.owl#DiabeticPatient"/>
  <hasAge rdf:datatype="&xsd;int">25</hasAge>
  <hasAddress rdf:datatype="&xsd;string">EzzahraCity-Sousse</hasAddress>
  <hasName rdf:datatype="&xsd;string">Sonia</hasName>
  <inverse_of_inChargeOf rdf:resource="http://www.owl-ontologies.com/OntologyHospitalEnv.owl#Doctor101"/>
  <isSupervisedBy rdf:resource="http://www.owl-ontologies.com/OntologyHospitalEnv.owl#N408"/>
</owl:NamedIndividual>
<!-- http://www.owl-ontologies.com/OntologyHospitalEnv.owl#Doctor101 -->
<owl:NamedIndividual rdf:about="http://www.owl-ontologies.com/OntologyHospitalEnv.owl#Doctor101">
  <rdf:type rdf:resource="http://www.owl-ontologies.com/OntologyHospitalEnv.owl#Doctor"/>
  <hasAddress rdf:datatype="&xsd;string">NasserCity-Tunis</hasAddress>
  <hasName rdf:datatype="&xsd;string">Youssef</hasName>
  <inChargeOf rdf:resource="http://www.owl-ontologies.com/OntologyHospitalEnv.owl#DiabeticPatient202"/>
</owl:NamedIndividual>
<!-- http://www.owl-ontologies.com/OntologyHospitalEnv.owl#N408 -->
<owl:NamedIndividual rdf:about="http://www.owl-ontologies.com/OntologyHospitalEnv.owl#N408">
  <rdf:type rdf:resource="http://www.owl-ontologies.com/OntologyHospitalEnv.owl#Nurse"/>
  <hasAge rdf:datatype="&xsd;int">25</hasAge>
  <hasName rdf:datatype="&xsd;string">Ali</hasName>
  <hasAddress rdf:datatype="&xsd;string">NasserCity-Tunis</hasAddress>
  <supervises rdf:resource="http://www.owl-ontologies.com/OntologyHospitalEnv.owl#DiabeticPatient202"/>
</owl:NamedIndividual>
```

Figure 4.12 (b) Extrait OWL de l'ontologie globale (définition d'individus)

4. Implémentation de la facette fonctionnelle

Pour implémenter les actions élémentaires sous forme de services web afin de pouvoir les déployer et les tester, nous devons passer par différentes étapes de description (Figure 4.13). Tout d'abord il faut commencer par créer les classes qui implémentent le fonctionnement de ces actions élémentaires, ensuite, il faut en déduire la description WSDL correspondante à ce service. Développer et compiler le modèle de la description owl-s de ce service constitue l'étape suivante. La dernière étape exige la compilation (a) du modèle de processus qui correspond à l'exécution réelle du service web, (b) de son profil qui permet, entre autres, sa découverte, enfin (c) de son « *Grounding* » lié à son modèle de processus.

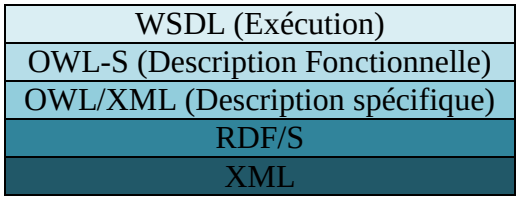


Figure 4. 13 Les différents niveaux de description d'un service web

Le service peut donc être perçu comme étant une interface décrivant une collection d'opérations accessibles via le réseau et dont les paramètres d'entrées et/ou de sorties correspondrons aux paramètres spécifiés par le modèle conceptuel, et qui servent particulièrement pour la découverte et l'invocation. Considérons par exemple le cas de l'action «*GlucoseRateMeasure*» dont les entrées sont : **BeginGlucoseRateMeasure** (Type : Transition) et **PaGlucoseRateMeasure** (Type : Place) ; les sorties sont représentées par : **EndGlucoseRateMeasure** (Type : Transition) et **PcGlucoseRate-**

Measure (Type : Place) ; concernant le résultat, cette action devrait fournir la valeur usuelle du taux du glucose chez le patient en question (Type : TInformation).

4.1 Génération du code WSDL de l'action « *GlucoseRateMeasure* »

- Formulation de la requête sur l'ontologie HealthONT.owl :

```
String QueryForUGR =""  +
                    "PREFIX onto:http://www.owl-
ontologies.com/HealthONT.owl" +
                    "SELECT ?ugr ?patient" +
                    "WHERE" +
                    "{" +
                    "?patient onto:ID_PATIENT "+ PatientId +
                    "?patient onto:ugr ?ugr" +
                    "}";
```

- Exécution de la requête « QueryForUGR » & récupération du résultat :

```
String QService = "http://dbpedia.org/sparql";
QueryExecution Q = QueryExecutionFactory.sparqlService(QService,
QueryForUGR);
        try {
        ResultSet results = Q.execSelect();
        QuerySolution solution = (QuerySolution) results.next();
        UGR= solution.get("?ugr");
       }catch(Exception e){
         e.printStackTrace();
        }finally {
Q.close();}
```

- Création du service correspondant à cette implémentation[38]. Le but demeure toujours la génération de la description WSDL du service. La figure 4.14 présente le code de la classe « GlucoserateMeasure.java » permettant d'aboutir à cette description WSDL.

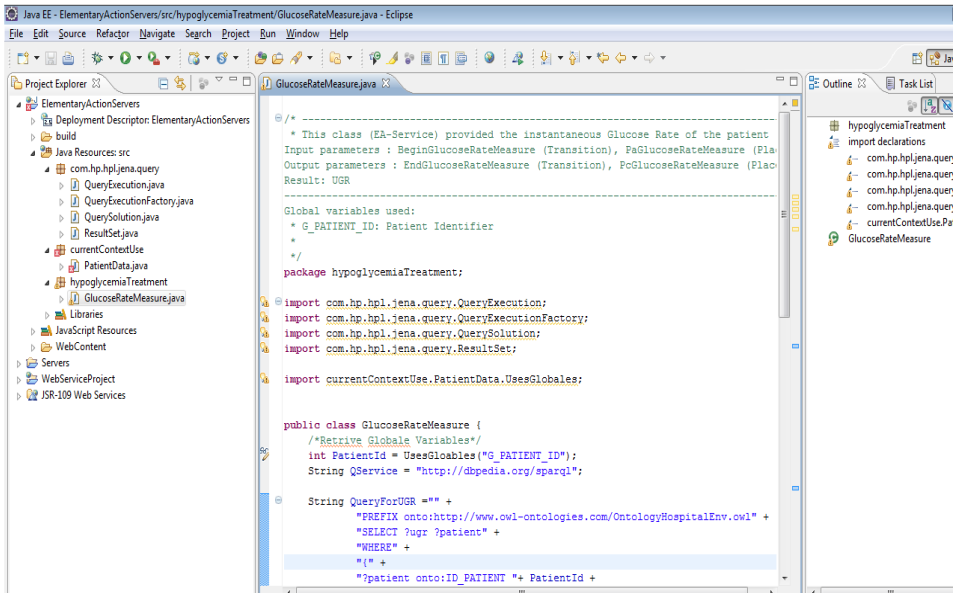


Figure 4. 14 Implémentation de la partie fonctionnelle de l'action élémentaire GlucoseRateMeasure

[38] Cette tâche est réalisée à l'aide de l'outil (Create Web Service) de la plateforme Eclipse.

- La génération passe, tout d'abord, par générer le service en spécifiant certains paramètres (Figure 4.15), lancer et vérifier l'existance du service (Figure 4.16) pour aboutir enfin à générer le code WSDL (Figure 4.17).

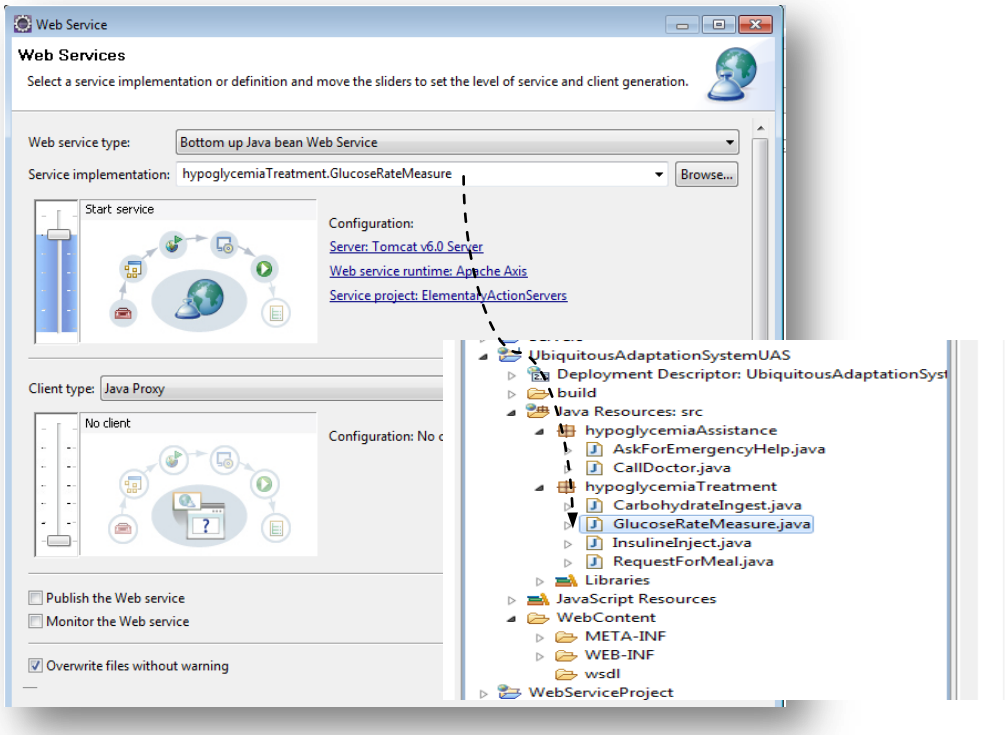


Figure 4. 15 Génération des SW (Interface D'assistance Eclipse)

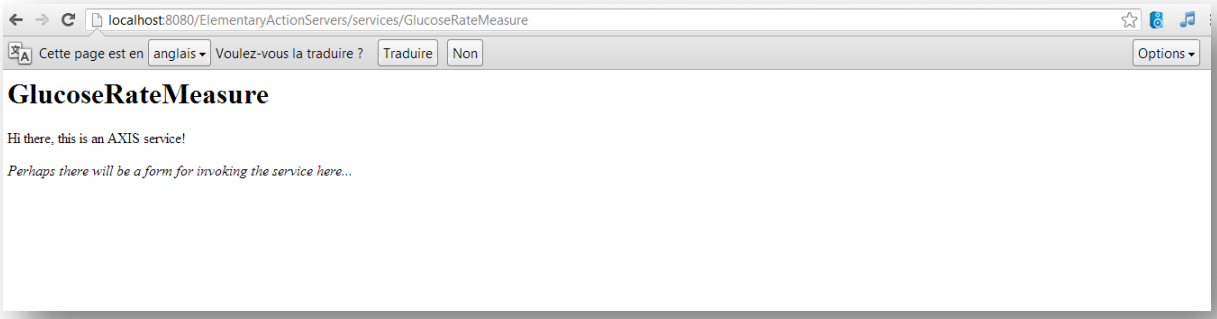


Figure 4. 16 Test de la présence du service à partir du moteur Axis

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://hypoglycemiaTreatment"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://hypoglycemiaTreatment" xmlns:intf="http://hypoglycemiaTreatment"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->
 <wsdl:types>
  <schema elementFormDefault="qualified" targetNamespace="http://hypoglycemiaTreatment"
xmlns="http://www.w3.org/2001/XMLSchema">
   <element name="returnUGR">
    <complexType>
     <sequence>
      <element name="T1" type="Transition"/>
      <element name="P2" type="Place"/>
     </sequence>
    </complexType>
   </element>
   <element name="returnUGRResponse">
    <complexType>
     <sequence>
      <element name="returnUGRReturn" type="xsd:float"/>
     </sequence>
    </complexType>
   </element>
  </schema>
 </wsdl:types>

   <wsdl:message name="returnUGRResponse">
      <wsdl:part element="impl:returnUGRResponse" name="parameters"/>
   </wsdl:message>

   <wsdl:message name="returnUGRRequest">
      <wsdl:part element="impl:returnUGR" name="parameters"/>
   </wsdl:message>

   <wsdl:portType name="GlucoseRateMeasure">
      <wsdl:operation name="returnUGR">
         <wsdl:input message="impl:returnUGRRequest" name="returnUGRRequest"/>
         <wsdl:output message="impl:returnUGRResponse" name="returnUGRResponse"/>
      </wsdl:operation>
   </wsdl:portType>

   <wsdl:binding name="GlucoseRateMeasureSoapBinding" type="impl:GlucoseRateMeasure">
      <wsdlsoap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="returnUGR">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="returnUGRRequest">
         <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="returnUGRResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>

   <wsdl:service name="GlucoseRateMeasureService">
      <wsdl:port binding="impl:GlucoseRateMeasureSoapBinding" name="GlucoseRateMeasure">
         <wsdlsoap:address
location="http://localhost:8080/ElementaryActionServers/services/GlucoseRateMeasure"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

Figure 4. 17 Génération du code WSDL de l'action élémentaire

4.2 Création de l'ontologie de format des données et enrichissement des services par la sémantique

Rappelons que nous avons utilisé la spécification OWL-S pour pourvoir une description sémantique aux services web développés. Ces derniers utiliseront les concepts, en particulier les types (DataType, ObjectType). Les figures 4.19 (a) et (b) représentent respectivement des exemples d'un DataType et d'un ObjectType. Ces concepts sont définis dans l'ontologie de domaine «*RDPIModelOntology*» représentant le format des données du domaine (Figure 4.18). Rappelons que le but de l'OWL-S est d'offrir un moyen pour doter un service web d'une sémantique lui permettant d'être découvert et invoqué automatiquement.

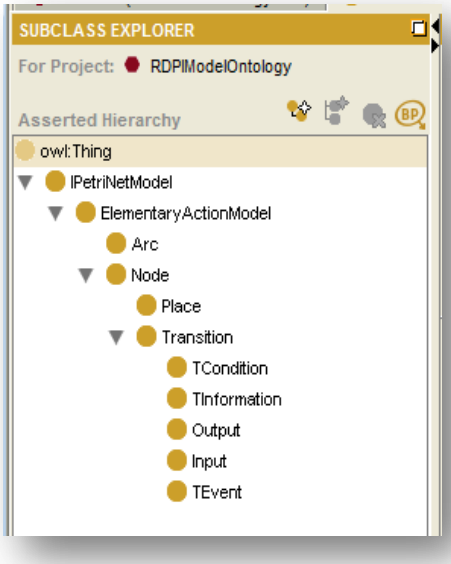


Figure 4. 18 Exemple de l'interface de saisie de facettes des propriétés dans Protégé

```
<owl:DatatypeProperty rdf:ID="hasModelName">
  <rdfs:domain rdf:resource="#IPetriNetModel"/>
  <rdfs:range>
   <owl:DataRange>
    <owl:oneOf rdf:parseType="Resource">
     <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
     >consciousPatientInterventionActivity</rdf:first>
     <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
    </owl:oneOf>
   </owl:DataRange>
  </rdfs:range>
  <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >A datatype properties that represents a model name of a composite activity</rdfs:comment>
 </owl:DatatypeProperty>
```

Figure 4. 19 (a) Extrait de l'ontologie d'un modèle « *IPetriNetModel* » - Datatype « *hasModelName* »

```
<owl:InverseFunctionalProperty rdf:about="#isComposedOf">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  <owl:inverseOf rdf:resource="#isElementOf"/>
  <rdfs:domain rdf:resource="#IPetriNetModel"/>
  <rdfs:range rdf:resource="#ElementaryActionModel"/>
 </owl:InverseFunctionalProperty>
...
<owl:ObjectProperty rdf:ID="isElementOf">
  <owl:inverseOf> <owl:InverseFunctionalProperty rdf:ID="isComposedOf"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#ElementaryActionModel"/>
  <rdfs:range rdf:resource="#IPetriNetModel"/>
 </owl:ObjectProperty>
...
```

Figure 4.19 (b) Extrait de l'ontologie d'un modèle « *IPetriNetModel* » - ObjectProperty « *isComposedof* »

A l'aide de l'outil OWL-S Editor, plug-in de Protégé, nous avons développé les descriptions OWL-S des actions élémentaires. Précisément, nous avons créé les instances des classes *Service*, *Profile*, *Process* et *Grounding*.

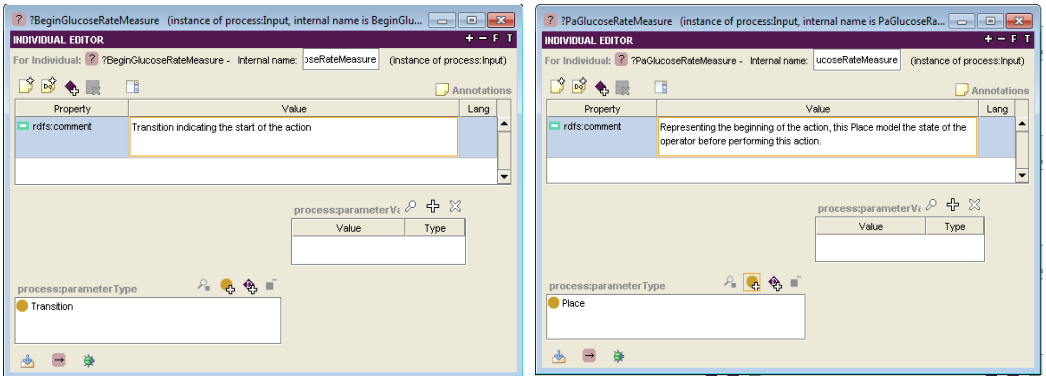


Figure 4. 20 Définition des types de paramètres qui servent de base pour la composition dynamique des actions élémentaires

Nous forçons l'instance *process:Input* à avoir comme type : *process:parameterType* une instance de classe Transition (Figure 4.20). De même, nous forçons le deuxième input à être de classe Place (Figure 4.21).

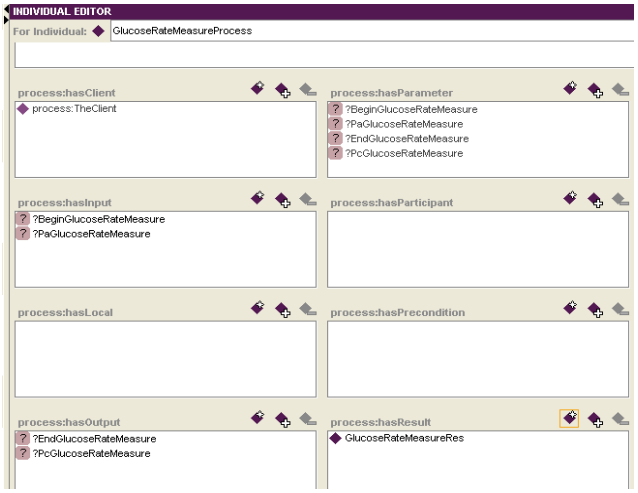


Figure 4. 21Définition des Input/Output du processus « *GlucoseRateMeasureProcess* »

L'étape suivante nous conduit à créer toutes les instances du service atomique (Figure 4.22). De même pour les autres services de l'étude de cas (Figure 4.23). Cette manipulation peut être illustrée à travers un graphe parallèement créé au fur et à mesure que la spécification des services atomiques (Figure 4.24 et 4.25).

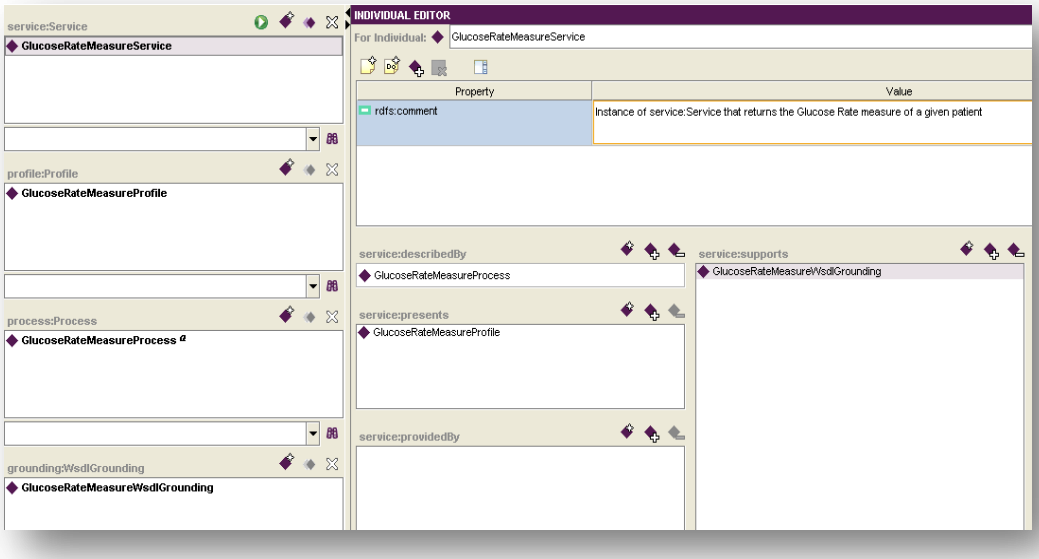


Figure 4. 22 Définition des instances de l'action « *GlucoseRateMeasure* »

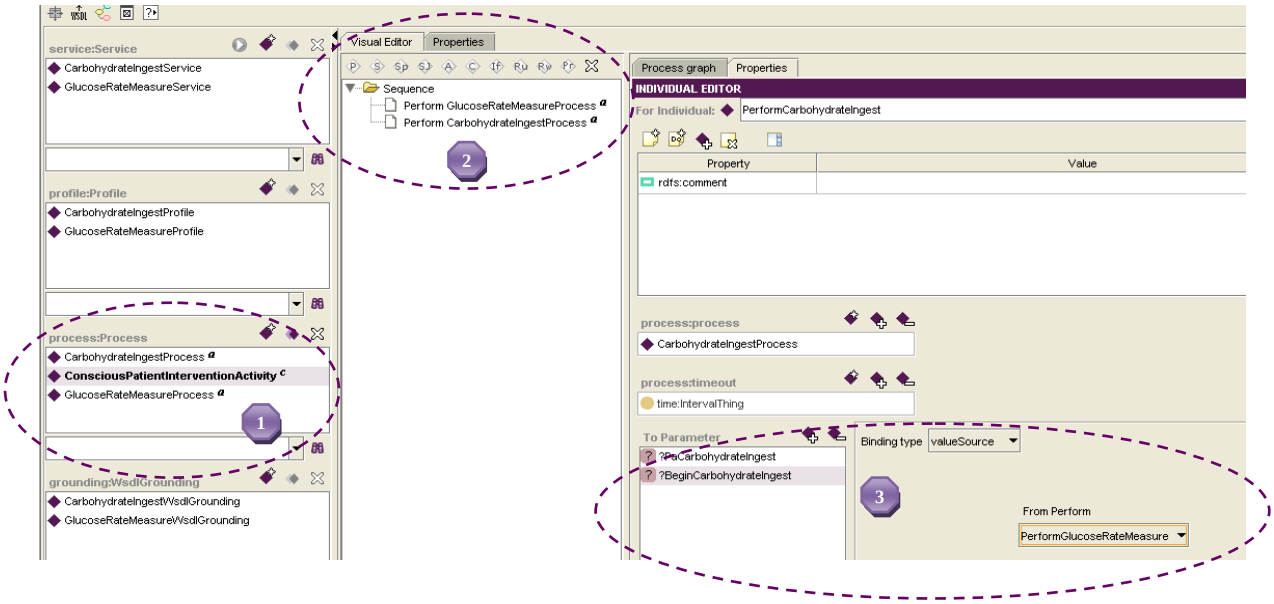


Figure 4. 23 Définition du type d'exécution du processus « *ConsciousPatientInterventionActivityProcess* »

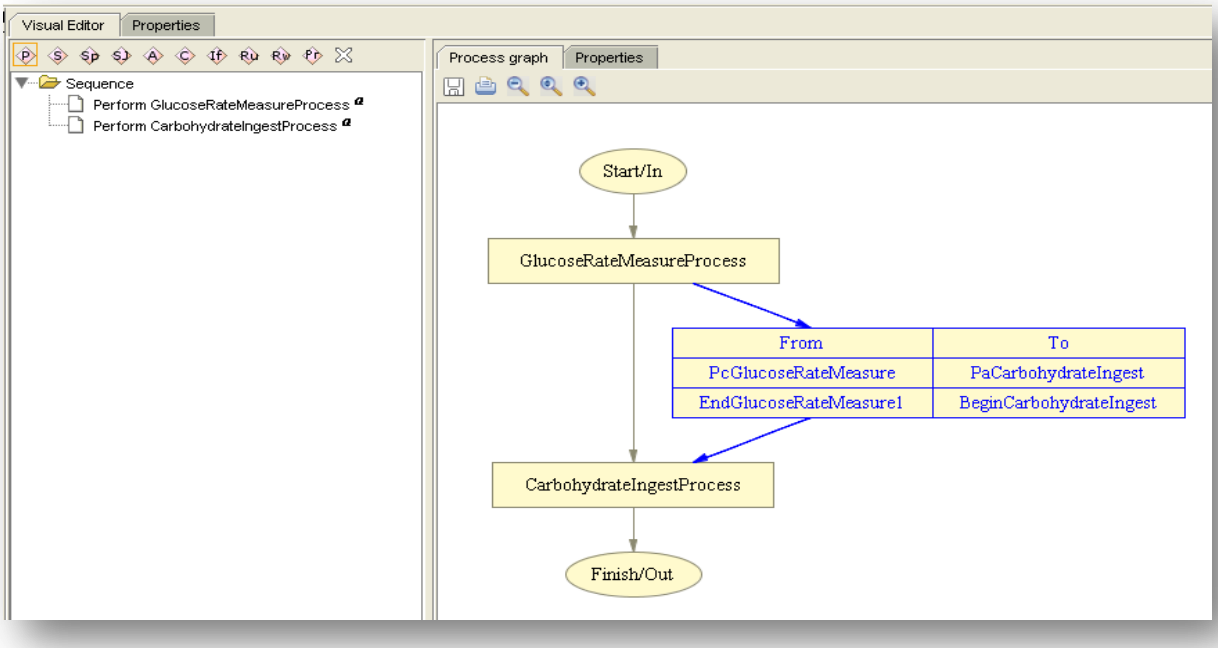


Figure 4. 24 Représentation graphique du processus « *ConsciousPatientInterventionActivityProcess* »

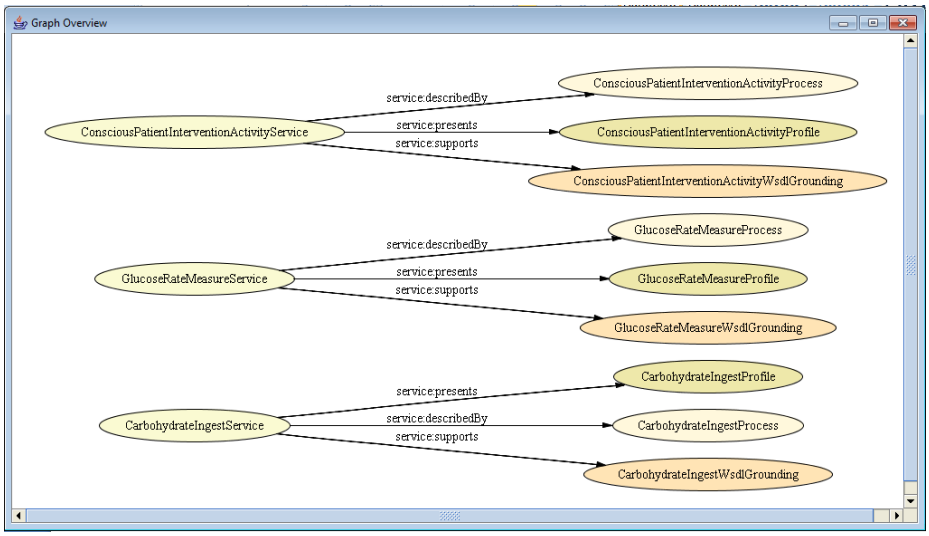


Figure 4. 25 Représentation graphique de l'implémentation générale des trois processus du test

4.3 Mise en œuvre des actions élémentaires (Grounding)

Cette étape exige la création d'une instance *wsdlAtomicPrrocessGrounding* pour chaque processus atomique précédemment décrit (Figure 4.26). Ceci revient à attribuer la propriété *grounding*:*owlsProcess* pour chaque instance. Ensuite, il faut définir la mise en correspondance (Mapping) pour les entrées et les sorties des processus atomiques par rapport aux « Message Parts » de l'implémentation WSDL. La figure 4.27 montre la mise en correspondance de ces paramètres. Dans cette figure, nous discernons comment les opérations WSDL sont mises en correspondance avec le processus atomique ainsi les entrées/sorties du message coïncident avec les entrées/sorties du processus. Ces

correspondances sont définies par *grounding:WSDLMessageMaps* spécifié dans OWL-S. Ceci représente le mécanisme basé sur WSDL pour invoquer le service OWL-S. Autrement exprimé, les Inputs/Outputs du processus atomiques OWL-S correspondront aux messages dits WSDL. Rappelons que cette correspondance se base principalement sur l'identification des éventuelles similitudes entre les descriptions des services disponibles avec la requête système. Elle est fondée sur la correspondance entre « Input/output », les préconditions ainsi que les effets et finalement elle peut tirer profit de la catégorie du service.

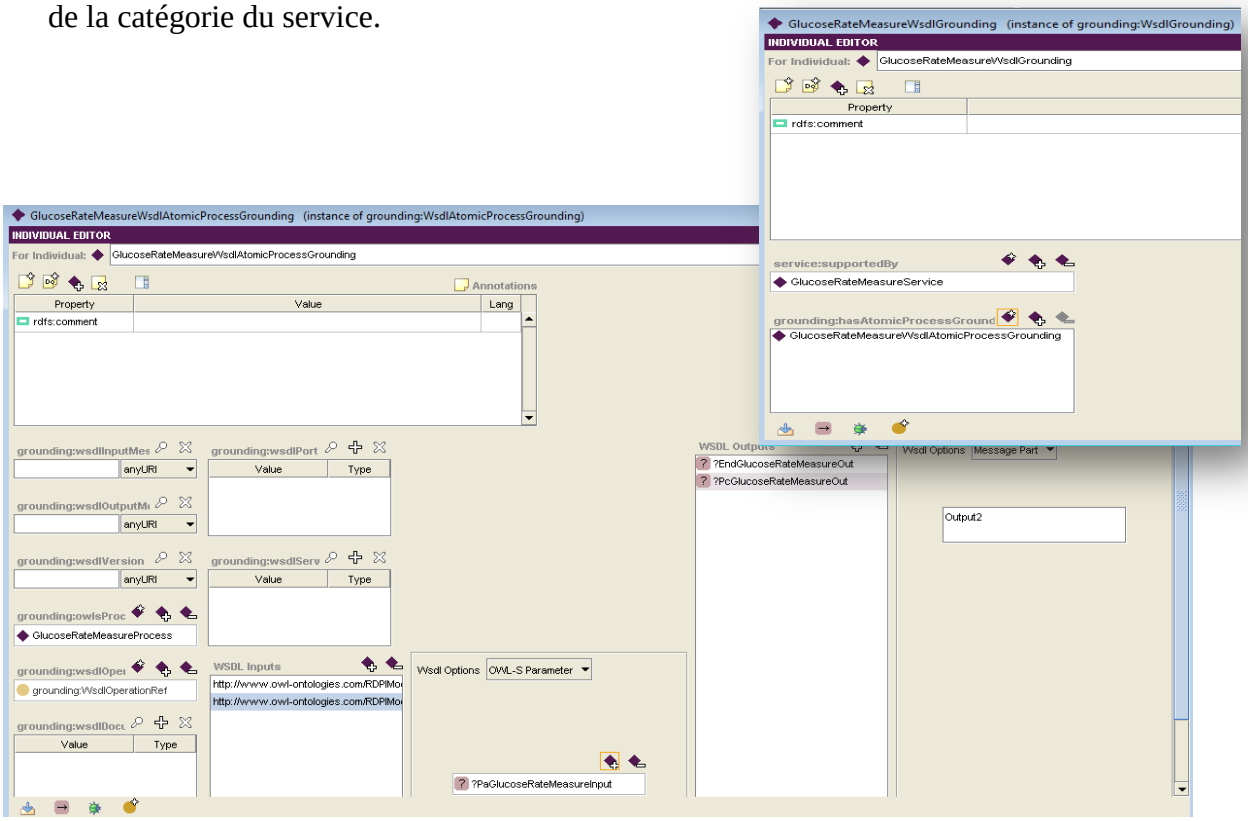


Figure 4. 26 Création de l'instance de grounding :WsdlAtomicprocessGrounding de l'action GlucoseRateMeasure

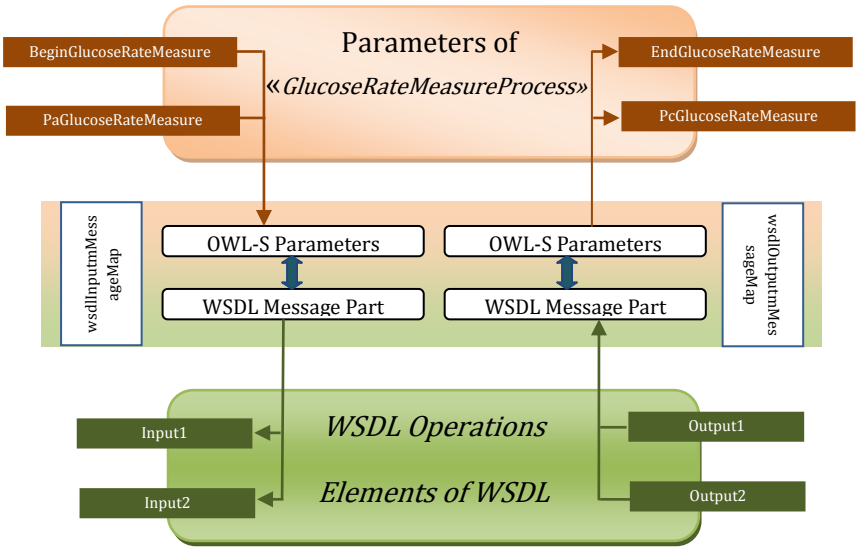


Figure 4. 27 Mise en correspondance entre les paramètres d'une description OWL-S et les éléments d'une description WSDL

5. Implémentation de la partie système (Prototype)

Dans une troisième phase de développement, nous avons envisagé un prototype de notre modèle conceptuel du système proposé afin de montrer la faisabilité et la validité de nos propositions. L'approche proposée a été implémentée en JAVA. Nous décrivons un prototype dont le but est de récupérer l'information appropriée à l'utilisateur suivant son contexte d'utilisation (Figure 4.28). D'après cette figure, le prototype intègre les fonctionnalités suivantes :

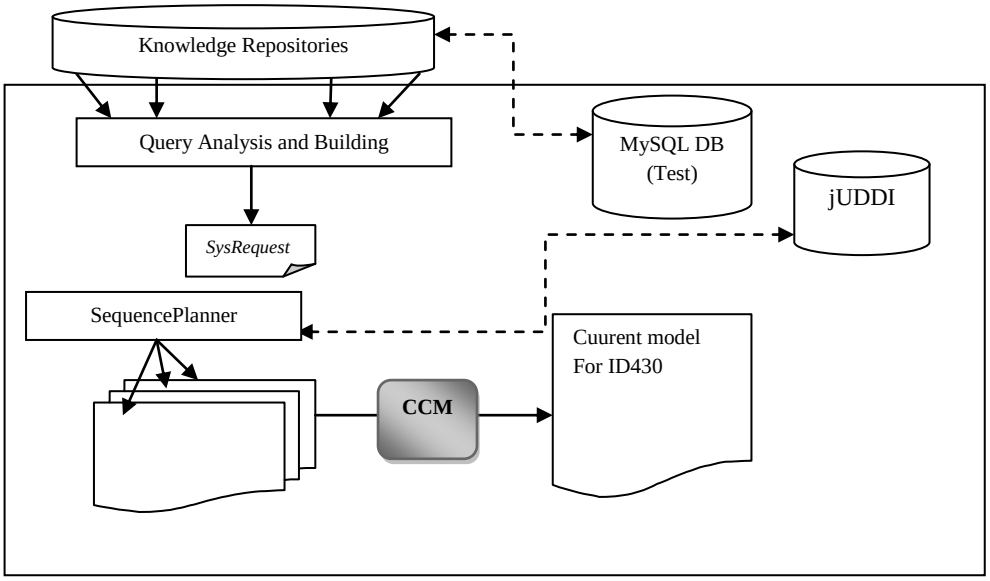


Figure 4. 28 Etapes de construction d'un modèle et les modules intervenant

- Un moteur de décision qui doit analyser les données contextuelles et doit récupérer les éléments nécessaires à la construction de sa requête (*SysRequest*). Son output est un fichier contenant la situation actuelle pour un utilisateur donné. Ces données représentent les mots clés pour identifier la tâche à réaliser. Un exemple est fourni (Figure 4.29) :

```
....
#USER NURSE <ID:430>
#PATIENT<ID:202>
#SITUATION<HYPOGLYCEMIA>
#SITUATIONC<CONSCIOUS*PATIENT>
...
```

Figure 4. 29 Exemple de contenu constituant une requête *SysRequest*

(1) Formulation de la requête :

Système : « *Je veux le modèle de tâche qui correspond à la situation actuelle*» :

- ✓ Les carcatéristiques de l'utilisateur : infirmier
- ✓ « ConsciousPatient » : Situation de l'activité.
- ✓ « Intervention » : Type de l'activité.

⇨ Requête : « ***ConsciousPatientInterventionActivity*** »

(2) Générer le modèle de tâche associé à cette requête :

Phase1 : identifier la spécification owl-s Process Model correspondante (Figure 4.30)

```
<process:CompositeProcess rdf:ID="ConsciousPatientInterventionActivityProcess"/>
<profile:Profile rdf:ID="GlucoseRateMeasureProfile"/>
<process:AtomicProcess rdf:ID="CarbohydrateIngestProcess"/>
<service:Service rdf:ID="CarbohydrateIngestService"/>
<process:AtomicProcess rdf:ID="GlucoseRateMeasureProcess"/>
<service:Service rdf:ID="GlucoseRateMeasureService"/>
<profile:Profile rdf:ID="ConsciousPatientInterventionActivityProfile">
<profile:hasInput>
<process:Input rdf:ID="IPaGlucoseRateMeasure">
<process:parameterValue rdf:parseType="Literal"></process:parameterValue>
<process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
>PLACE</process:parameterType>
</process:Input>
</profile:hasInput>
</profile:Profile>
<profile:Profile rdf:ID="CarbohydrateIngestProfile"/>
<service:Service rdf:ID="ConsciousPatientInterventionActivityService"/>
</rdf:RDF>
<!-- Created with Protege (with OWL Plugin 3.4.4, Build 579)  http://protege.stanford.edu
-->
```

Figure 4. 30 Spécification owl-s du processus *ConsciousPatientInterventionActivityProcess*

Phase2 : Découpage et identification des actions élémentaires à partir du modèle (Figure 4.31)

- Une base de données (MySQL) s'est avérée indispensable pour le stockage des contenus et des données des **tests**.
- EARegistry : jUDDI a été sélectionné comme outils de développement de ce composant afin de stocker les données relatives aux EASW déjà développés.
- Une application : U-RDST (Ubiquitous - RealTime Diabete Supervision and Treatment).

5.1 Tests et Résultats

Nous achevons ce chapitre par une présentation et interprétation des résultats correspondant au cas d'étude préalablement décrit. Les premiers tests ont montré des résultats satisfaisants et acceptables répondant au mieux à nos attentes.

En effet, en se basant sur l'entrepôt des ontologies et des données avec les descriptions sémantiques ainsi que tous les composants développés, le système est en mesure de déduire l'information requise. Cette information représente le résultat escompté du service web qui encapsule l'action élémentaire en question. Outre le retour

des résultats escomptés, il est important de noter que seuls les services d'intérêt seront invoqués par le système pour formuler le modèle associé (ceci est assuré grâce aux paramètres typés des services à savoir les types Place, Transition, TCondition, TInformation, etc.).

```
<!— Service description -->
<service:Service rdf:ID="UpdatePatientChartService">
<service:presents rdf:resource="#UpdatePatientChartProfile"/>
<service:describedBy rdf:resource="#UpdatePatientChart"/>
<service:supports rdf:resource ="#UpdatePatientChartGrounding"/>
</service:Service>
<process:AtomicProcess rdf:ID="UpdatePatientChart">
       <process:hasInput>
              <process:Input rdf:ID="PaUpdatePatientChart"/>
       </process:hasInput>
       <process:hasInput>
              <process:Input rdf:ID="BeginUpdatePatientChart"/>
       </process:hasInput>
       <process:hasOutput>
              <process:Output rdf:ID="EndUpdatePatientChart"/>
       </process:hasOutput>
       <process:hasOutput>
              <process:Output rdf:ID="PcUpdatePatientChart"/>
       </process:hasOutput>
<process:hasPrecondition>
       <expr:SWRL-Condition rdf:about="#SWRL-IsPatient()">
              <expr:expressionObject>
              <swrl:AtomList rdf:ID="AtomListID2"/>
              </expr:expressionObject>
       </expr:SWRL-Condition>
</process:hasPrecondition>
<process:hasLocal>
       <process:Local rdf:ID="PbUpdatePatientChart"/>
</process:hasLocal>
<process:hasResult>
       <process:Result rdf:ID="OperationRoom">
              <process:hasResultVar>
                     <process:ResultVar rdf:ID="VarOperationRoom">
                            <process:parameterValue rdf:parseType="Literal">
                            </process:parameterValue>
                     </process:ResultVar>
              </process:hasResultVar>
       </process:Result>
</process:hasResult>
(…)
</process:AtomicProcess>
```

Figure 4. 31 Extrait du modèle généré

5.1.1 Déduction en temps-réel des besoins utilisateur (en termes d'information)

La déduction de l'information requise lors de l'exécution de cette tâche n'est autre que la récupération des résultats retournés par le fichier WSDL de la description de l'action élémentaire identifiée dont l'adresse est la suivante : <wsdlsoap:address

location="http://localhost:8080/ProjetUbiquitousAdaptation/EAElementaryActions/GlucoseRateMeasure"/>.

Durant cette étape de fonctionnement, l'interface de l'utilisateur a l'allure suivante : il s'agit d'une *Widget* informationnelle qui affiche sur l'écran de l'appareil cible la valeur UGR du patient (Figure 4.32).

Figure 4. 32 Déduction des besoins utilisateur dans un environnement ubiquitaire

5.1.2 Adaptation du contenu selon les spécificités des utilisateurs

Pour une évaluation heuristique, l'adaptation est considérée inutile si elle ne prend pas les spécificités des utilisateurs en compte. En fait, l'information adaptée dépend inévitablement de certaines caractéristiques de l'utilisateur. A titre d'exemple, si l'infirmier est spécialiste en endocrinologie, le poids du patient est une information considérée comme indispensable pour savoir la quantité du glucagon à administrer au patient. Dans le cas contraire, c'est-à-dire un infirmier non-spécialiste, la quantité du glucagon représente bien l'information requise à cette étape (Figure 4.33).

Figure 4. 33 Déduction des besoins utilisateur selon ses spécificités

5.1.3 Adaptation du contenu selon le changement du contexte

L'exemple suivant (Figure 4.34) illustre la façon dont l'information adaptée change selon le contexte d'utilisation. Supposons que le patient a perdu la conscience pendant que l'infirmier s'occupait de lui. Le modèle en cours de fonctionnement n'est plus adapté à la situation actuelle. Le système doit régénérer le modèle approprié afin de mieux s'adapter au nouveau contexte de l'environnement de l'utilisateur. Rappelons notre étude

de cas, les informations requises selon l'étape du traitement d'un patient conscient (Tableau 3.1). Considérons, en particulier, la cinquième étape où l'infirmier a besoin de connaître la valeur usuelle du niveau de glucose du patient afin d'accomplir la tâche correspondante à cette étape. Cependant, si durant l'exécution de cette action, le patient devient inconscient, l'infirmier doit connaître en urgence le poids du patient ainsi que la quantité du glucagon à administrer afin d'accomplir la nouvelle tâche en cours.

Figure 4. 34 Déduction des besoins utilisateur selon changement du contexte

Nous concluons à travers ces exemples qu'une fois la situation de l'utilisateur change donc l'information dont il besoin change également. En conséquence, nous avons montré que l'interface finale de l'utilisateur permet d'adapter les informations qu'elle fournit de façon implicite découlant des besoins instantanés de l'utilisateur ainsi que de son contexte d'utilisation.

6. Conclusion

Ce chapitre a été consacré à la mise en œuvre des aspects conceptuels proposés au travers cette thèse. Via cette étape d'implémentation et de simulation de l'étude de cas, nous avons veillé à élucider davantage la stratégie proposée. Cette dernière a été appliquée et validée tout en suivant les étapes de construction du modèle d'interaction et au fur et à mesure que le contexte change. Par conséquent, nous avons démontré l'adaptation temps-réel des informations fournies à l'utilisateur à ses spécificités, à ses préférences, à son activité en cours et au changement de son contexte d'usage. L'utilisateur se situant dans un environnement ubiquitaire arrivera nécessairement à recevoir sur son interface l'information dont il a besoin. En procédant de cette manière, nous nous sommes parvenu implicitement à aider l'utilisateur à accomplir sa tâche en cours. Il s'agit, en fait, de la principale motivation de l'Informatique Ubiquitaire que notre stratégie nous a permise, entre autres, de la respecter.

CONCLUSION GENERALE
&
PERSPECTIVES

1. Introduction

L'**informatique ubiquitaire** est « une » informatique présente en tout lieu, à tout instant et en toutes choses. En conséquence, les environnements ubiquitaires tendent à mettre en jeu de nombreux dispositifs communicants et mobiles, en conséquence un réseau gigantesque d'information prend naissance : **un réseau ubiquitaire d'information**. Ainsi, un utilisateur mobile pourrait voir sur son interface diverses informations provenant des différents dispositifs disséminés partout dans l'environnement dans lequel il évolue. Toutefois, à un moment donné et dans un contexte bien déterminé, l'utilisateur aura grandement besoin de certaines informations tout en trouvant d'autres complètements inutiles. Ce choix dépend de ses préférences et de son contexte d'utilisation. En conséquence, l'utilisateur mobile doit disposer de l'information la plus appropriée afin de trouver utile son dispositif et afin que ses activités en soient facilitées. Autrement exprimé, le but ultime de l'informatique ubiquitaire est de faciliter les activités de l'utilisateur – notamment celles pénibles et fastidieuses – ainsi que de remplacer son intervention dans toute tâche pouvant lui faire perdre du temps sans qu'il perçoive cette assistance. Notre contribution est ainsi la conception et la réalisation d'un système ubiquitaire permettant une adaptation dynamique et temps-réel du contenu informationnel de l'interface utilisateur en procurant à l'utilisateur l'information qui répond au mieux à ces exigences. Ce fonctionnement doit donc être régi par une stratégie d'adaptation se basant sur des modèles évolutifs et dynamiques.

2. Conclusion sur les travaux effectués

Partant de ce postulat, nous avons cherché à déterminer dans une première phase de ce travail les insuffisances et la problématique relatives à ce paradigme à travers une étude de l'état de l'art (*Chapitre 1*). Nous avons entamé la démarche de la résolution en tentant de proposer un cadre conceptuel prenant la forme d'une plateforme générique pour un système d'adaptation (*Chapitre 2*). Nous avons en même temps souligné l'importance de la couche stratégie d'adaptation pour ces systèmes. A cet effet, nous nous sommes tournés vers l'étude d'une stratégie d'adaptation (*Chapitre 3*). Nous avons rappelé à ce niveau qu'un modèle de fonctionnement fixe est loin d'améliorer les performances de ces systèmes, en revanche il est susceptible d'en risquer l'utilité et la finalité. Ceci laisse à penser que le modèle d'interaction doit être généré au fur et à mesure que l'utilisateur évolue dans ces environnements dynamiques. Une analyse plus poussée dans ce cadre nous a cependant permis de montrer la faisabilité de cette idée via la technologie des services web sémantiques et d'en améliorer la performance en se basant sur les ontologies. Enfin, nous avons développé un prototype du système proposé afin de montrer la faisabilité et la validité de nos propositions (*Chapitre 4*).

3. Contribution

Notre approche s'intègre dans les méthodes de conception centrées sur l'utilisateur (ou encore anthropocentrées, qui s'opposent à celles considérées comme technocentrées).

La motivation était de fournir à l'utilisateur final la meilleure information possible dans un contexte fortement dynamique avec des contraintes, des caractéristiques et des préférences personnalisables. Le tissage de comportements de l'utilisateur est décrit par les aspects de composition dynamique de modèles élémentaires. Il s'agit de créer un modèle d'interaction adapté à un modèle de l'utilisation particulier et les changements qui peuvent en être induits sur le modèle conceptuel. Dans ce sens, ce présent travail a toutefois le mérite d'avoir fait éclore une étude sur l'adaptation. Nous y ont proposé de nouvelles manières de promouvoir le développement des modèles d'interaction en exécution. Notre approche constitue une solution ouverte et générique pour le problème d'adaptation de contenu portant essentiellement sur la déduction des besoins informationnels de l'utilisateur en temps réel et qui permet ainsi d'être déployée dans le cadre de différentes applications.

Concernant la modélisation du contexte qui représente la principale source des données du système, il existe de nombreux modèles et implémentations pour représenter le contexte. La principale hypothèse du « monde ouvert » nous a conduits à une étape de notre conceptualisation à adopter les ontologies. A travers cette approche, nous pouvons dire que tout un modèle de fonctionnement sera partagé et non pas uniquement des données. Par l'intermédiaire de cette recherche, nous avons cherché à améliorer le processus d'adaptation dans un environnement ubiquitaire. Le but est de créer des interfaces mobiles plus riches et orientées utilisateurs. Enfin, nous pouvons faire en sorte que le résultat soit un environnement avec des fonctionnalités d'adaptation fiables qui améliore la qualité de vie pour ses utilisateurs.

4. Perspectives et futurs travaux

Cette thèse a induit différentes perspectives de recherche. L'une d'elles consiste à étudier la découverte « Peer To Peer » des actions élémentaires, par la suite se dispenser des services du proxy. Subséquemment, la découverte sera uniquement basée sur la structure des messages (SOAP) qui s'échangent entre utilisateurs (Interfaces Utilisateurs) et services web sémantiques. La deuxième perspective consiste à intégrer une technique d'apprentissage qui permet de continuer à développer la base de connaissance du système à l'exécution afin d'améliorer l'utilité de l'adaptation, notamment lorsque l'exécution du modèle échoue. En effet, ça permettra d'apprendre de nouvelles connaissances de cette insuffisance qui seront utilisées dans le prochain cycle de génération de modèle.

Troisièmement, déduire l'information idoine en temps opportun n'est pas suffisant pour affirmer avoir bien satisfait les besoins utilisateurs. En effet, cette information est également en forte relation avec l'état psycho-cognitif de l'utilisateur. Par exemple un personnel médical peut parfois être surchargé de travail et psycho-cognitivement non-disponible. D'où l'information à inférer demande beaucoup plus de précision et d'exactitude, notamment lorsqu'il s'agit de la santé humaine en question. Néanmoins, dans un système ubiquitaire, nous ne pouvons pas garantir que les informations éparpillées de part et d'autre dans l'environnement soient toujours correctes ou au moins

mises à jour. Par conséquent, nous ne pouvons assurer que les informations fournies à l'utilisateur soient également correctes. Ce problème peut être considéré comme un état de dysfonctionnement du système de gestion du contexte, un problème très peu étudié et mis en avant dans nombreuses recherches pourtant il s'avère indispensable dans ce domaine.

La progression de la science est exponentielle. Elle se nourrit d'elle-même, comme un virus. Chaque nouvelle avancée en annonce une autre. Il a fallut des milliers d'années pour progresser de la roue à l'automobile. Mais, en quelques décennies, elle est passée de la voiture à la fusée spatiale. Le rythme du progrès scientifique se mesure aujourd'hui en semaines. L'homme est incapable de le maîtriser.

« Anges et démons », Dan Brown

REFERENCES BIBLIOGRAPHIQUES
&
WEBOGRAPHIQUES

A

(Abed, 2001)

Abed M., Ezzedine H., Kolski C. (2001). Modélisation des tâches dans la conception et l'évaluation des systèmes interactifs : la méthode SADT/Petri. In Kolski C. (Ed.),Analyse et conception de l'IHM, interaction Homme-machine pour les SI, vol.1, Hermes, Paris, pp. 145-174.

(Abouzaid, 2010)

Fayçal Abouzaid. Analyse Formelle d'Orchestrations de Services Web. Thèse de doctorat en informatique. Département Génie informatique, Ecole polytechnique de Montréal. Université de Montréal. Décembre (2010).

(Abrams, 2004)

Marc Abrams, James Helms. User Interface Markup Language (UIML) Specification. Working Draft 3.1, 11 March 2004. Available at https://www.oasis-open.org (Dernière consultation Novembre 2012).

(Ahmed, 2007)

N. Ould ahmed, S. Tata. How to consider requester's preferences to enhance web service discovery. Proceedings of the second International Conference on Internet and Web Applications and Services, Morne, Mauritius, pp 59-64 (2007).

(Akkiraju, 2011)

R. Akkiraju, J. Farrell, J.Miller, M. Nagarajan, M. Schmidt, A. Sheth, K. Verma. Web Service Semantics - WSDL-S. UGA-IBM Technical Note, version 1.0, April 18, 2005. Available at: http://lsdis.cs.uga.edu/projects/ METEOR-S/WSDL-S. Last consultation May (2011).

(Akman, 1997)

Akman, V., Suray, M. The use of situation theory in context modeling. Computational Intelligence 13, 3, 427–438 (1997).

(Alonso, 2004)

Alonso, G.: Web services: concepts, architectures and applications. Springer Verlag (2004).

(Alvarez, 2007)

Alvarez-Cortes, V., Zayas-pere, B.E., Zarate-Silva, V.H., Uresti, J.A.R. (2007): Current Trends in Adaptive User Interfaces: Challenges and Applications. In Proceedings of Electronics, Robotics and Automotive Mechanics Conference (CERMA), Los Alamitos, CA, USA. 312-317, IEEE Computer Society.

(Andresen, 2005)

Andresen K., Gronau N. Seeking Optimal IT Strategies by the Determination of Adaptability in Domain-Specific Software Applications. Managing Modern Organizations with Information Technology: Proceedings of the 2005. Information Recources Management Association International Conference, San Diego, USA (2005).

(Ankolekar, 2003)

Ankolekar, A., Burstein, M., Hobbs, J., Lassila, O., Martin, D., McIlraith, S., Narayanan, S., Paolucci, M., Payne, T., Sycara, K., Zheng, H. Daml-s semantic markup for web services. In Proceedings of International Semantic Web Conference (ISWC). (Sardinia, Italy, pp. 348–364 (2003).

(Antona, 2006)

Antona, M., Savidis, A., Stephanidis, C. A Process–Oriented Interactive Design Environment for Automatic User Interface Adaptation. International Journal of Human Computer Interaction, 20 (2), 79-116 (2006).

B

(Baldauf, 2007a)

Bauldauf, M., Dustdar, S., Rosenberg, F.x. International Journal of Ad Hoc and Ubiquitous Computing 2(4), 263-277 (2007).

(Baldauf, 2007b)

Matthias Baldauf, Schahram Dustdar and Florian Rosenberg. A survey on context-aware systems. Int. J. Ad Hoc and Ubiquitous Computing, Vol. 2, No. 4, (2007).

(Bardram, 2005)

Jakob E. Bardram. The java context awareness framework (JCAF) – a service infrastructure and programming framework for context-aware applications. In *Proceedings of the Third international conference on Pervasive Computing* (PERVASIVE'05), Hans-W. Gellersen, Roy Want, and Albrecht Schmidt (Eds.). Springer-Verlag, Berlin, Heidelberg, 98-115 (2005).

(Bastien, 2001)

Bastien, J. M. C., & Scapin, D. L. Évaluation des systèmes d'information et Critères Ergonomiques. In C. Kolski (Ed.), Systèmes d'information et interactions homme-machine. Environnements évolués et évaluation de l'IHM. Interaction homme-machine pour les SI (Vol. 2, pp. 53-79). Paris : Hermes (2001).

(Bazargan, 2006)

Kaveh Bazargan. Adaptation des interfaces hommes-machine aux besoins des utilisateurs: un état de l'art. Thèse de doctorat en informatique de l'université de Genève, Janvier (2006).

(Benaboud, 2013)

Rohallah Benaboud, Ramdane Maamri, Zaidi Sahnoun. Agents And OWL-S Based Semantic Web Service Discovery With User Preference Support. International Journal of Web & Semantic Technology (IJWesT) Vol.4, No.2, April (2013).

(Benaboud, 2012)

Rohallah Benaboud, Ramdane Maamri, Zaidi Sahnoun. Semantic Web Service Discovery Based on Agents and Ontologies. International Journal of Innovation, Management and Technology, Vol. 3, No. 4, August (2012).

(Berdjouh, 2009)

Berdjouh Chafik, and Kazar Okba. Une Approche basée agent pour la Découverte de Services Web. CIIA, volume 547 of CEUR Workshop Proceedings, CEUR-WS.org, (2009)

(Booth, 2004)

Booth, D., Haas, H., McCabe, F., NewCorner, E., Champion, M., Ferris, C., Orchard, D. W3c working group note - web services architecture, February (2004).

(Brdiczka, 2005)

Oliver Brdiczka, Patrick Reignier, James L. Crowley. Supervised Learning of an Abstract Context Model for an Intelligent Environment. Smart Object and Ambient Intelligence (sOc-EUSAI '05), Grenoble, October (2005).

(Briand, 2008)

Henri Briand, Fabien Gandon et Fabien Picarougne, Atelier « Modélisation des connaissances », conférence Extraction et Gestion des Connaissances, 8èmes Journées Francophones, Sophia Antipolis, 29 janvier 2008

(Brogi, 2009)

Antonio Brogi, Sara Corfini and Stefano Iardella: From OWL-S Description to Petri Nets. E. Di Nitto and M. Ripeanu (Eds.): ICSOC Workshops, pp. 427-438. Springer-Verlag (2009).

(Brown, 1997)

P.J. Brown, J.D. Bovey, X. Chen. Context-Aware applications: From the Laboratory to the Marketplace. IEEE Personal Communications, Vol. 4, N°5, pp. 58-64 (1997).

(Brut, 2008)

Mihaela Brut, Florence Sèdes, Romulus Grigoras, Vincent Charvillat. An Ontology-based Approach for Providing Multimedia Personalized Recommendations. Dans : International Journal of Web and Grid Services, Inderscience Publishers, Numéro spécial Semantic Web, Vol. 4 N. 3, p. 314-329, (2008).

(Buttussi, 2008)

F. Buttussi, A user-adaptive and context-aware architecture for mobile and desktop training applications, in Proc. Mobile HCI, (2008).

C

(Calvary, 2003)

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J., A Unifying Reference Framework for Multi-Target User Interfaces, Interacting with Computers, Vol. 15, No. 3, pp. 289-308 (2003).

(Canoe, 2012)

Santé Canoe, MediResource. Available at: http://sante.canoe.ca/condition_info details.asp ?disease_id=73, Last update: January, 2013. Last consultation: December, (2012).

(Caon, 2013)

Caon, M., Khaled, O.A., Mugellini, E., Lalanne, D., Angelini, L. Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference. IEEE (ISSN : 2156-8103), Geneva. Pp. 717 – 718 (2013).

(Chaari, 2006)

Tarek Chaari, Frederique Laforest. Adaptation in Context-Aware Pervasive Information Systems: The SECAS Project. Journal of Pervasive Computing and Communications, Vol. 2, No. 2, June (2006).

(Chaari, 2007)

Tarak Chaari. Adaptation d'applications pervasives dans des environnements multi-contextes. Thèse de doctorat en informatique. Institut National des Sciences Appliquées de Lyon, (2007).

(Chen, 2004)

H Chen, T Finin, A Joshi. An Intelligent Broker Architecture for Context-Aware Systems. Adjunct Proceedings of Ubicomp, (2004).

(Cheung, 2006)

Daniel Cheung, J-Y Tigli, Stéphane Lavirotte, Michel Riveill. Wcomp: a Multi-Design Approach for Prototyping Applications using Heterogeneous Resources. Proceedings of the Seventeenth IEEE International Workshop on Rapid System Prototyping. pp. 119 – 125, June (2006).

(Crowley, 2006)

J.L Crowley, O. Brdiczka, P Reignier. Learning Situation Models for Understanding Activity. The 5th International Conference on Development and Learning (2006).

D

(Davenport, 2000)

Davenport, T. H., Prusak, L. Working knowledge: How organizations manage what they know. Boston, MA: Harvard Business School Press (2000).

(Decker, 2011)

Decker, G., Weske, M. Interaction-centric modelling of process choreographies. In Information Systems Volume 36, Issue 2, (Special Issue: Semantic Integration of Data, Multimedia, and Services), pp. 292–312, (2011).

(Dey, 1998)

Dey, A.K. Context-aware computing: The CyberDesk project. The Proceedings of the Spring Symposium on Intelligent Environments (AAAI Technical Report SS-98-02), pp. 51-54 (1998)..

(Dey, 1999)

Anind K. Dey, Daniel Salber, Gregory D. Abowd, Masayasu Futakawa. The Conference Assistant: Combining Context Awareness with Wearable Computing. 3rd International Symposium on Wearable Computers, San Francisco, California, pp. 21, (October 1999).

(Dey, 2000)

Anind K. Dey, Gregory D. Abowd. Towards a Better Understanding of Context and Context-Awareness. Proceedings of CHIA'00 workshop on Context-Awareness, (2000).

(Dey, 2001.a)

A. Dey, D. Salber, G. Abowd. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Special issue on Context-Aware Computing in the Human-Computer Interaction (HCI) Journal, vol. 16 (2-4), pp. 97-166, (2001).

(Dey, 2001.b)

Anind K. Dey. Understanding and Using Context. Personal and ubiquitous computing, Volume 5, pp 4-7 (February 2001).

(Tremblay, 2006)

Louise Tremblay. Lignes directrices de pratique clinique pour la prévention et le traitement de l'hypoglycémie liée au diabète de l'Association canadienne du diabète. (Mars 2006). Last Update: March, 2006. Last consultation: January, 2013, Available at: http://www.diabete.qc.ca/html/vivre_avec_diabete/hypo_severe_incon.html

(Duy, 2009)

Duy-Ngan Le, Van-Quoc Nguyen, Goh, A. Matching WSDL and OWL-S Web Services. IEEE International Conference on Semantic Computing. ICSC '09. 14-16 Berkeley, CA. Sept (2009).

E

(Elenius, 2005)

Daniel Elenius, Grit Denker, David Martin, Fred Gilham, John Khouri, Shahin Sadaati, Rukman Senanayake. The OWL-S Editor - A Development Tool for Semantic Web Services. In ESWC, pp 78-92, 2005.

F

(Fuhrer, 2006)

Patrik Fuhrer, Dominique Guinard. Building a Smart Hospital using RFID Technologies. 01/2006. In proceeding of: European Conference on eHealth 2006, Proceedings of the ECEH'06, Fribourg, Switzerland, October (2006).

G

(Grammenos, 2007)

Grammenos, D., Savidis, A., Stephanidis, C. Unified Design of Universally Accessible Games. In C. Stephanidis, (Ed.) Universal Access in Human-Computer Interaction: Intelligent and Ubiquitous Interaction Environments. Volume 6 of the Proceedings of the 12th International Conference on Human-Computer Interaction (HCI International 2007), Beijing, P.R. China, pp. 607-616. Berlin Heidelberg: Springer (LNCS). July (2007).

(Greenfield, 2006)

Adam Greenfield. Everyware: The Dawning Age of Ubiquitous Computing. New Riders. 1249 Eighth Street Berkeley, CA 94710. Published in association with AIGA. ISBN 0-321-38401-6) (2006).

(Gu, 2005)

Gu, T., Pung, H.K., Zhang, D.Q. A service-oriented middleware for building context-aware services. Journal of Network and Computer Applications, Volume: 28, Issue: 1, Publisher: Elsevier, Pp: 1-18 (2005).

(Guitton, 2006)

Julien Guitton. Planification multi-agent pour la composition dynamique de services Web. Mémoire de master, Université Joseph Fourier, Informatique et Mathématique Appliquée, Intelligence Interaction Information. Grenoble, le 12 Juin (2006).

H

(Halpin, 2001)

Halpin, T. A. Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design. Morgan Kaufman Publishers, San Francisco, 2001.

(Hanmansetty, 2004)

R. Hanmansetty. Model based approach for context aware and adaptive user interface generation. Phd. Thesis. Faculty of the Virginia Polytechnic Institute and State University. July 2004.

(Harel, 2003)

Harel, D., Marelly R.: Specifying and Executing Behavioral Requirements: The Play In/Play-Out Approach. Software and System Modelling 2, 82–107 (2003).

(Held, 2002)

Held, A., Buchholz, S., and Schill, A. Modeling of context information for pervasive computing applications. In Proceedings of SCI, ISAS (2002).

(Henricksen, 2003)

Henricksen, K., Indulska, J., and Rakotonirainy, A. Generating Context Management Infrastructure from High-Level Context Models. In Industrial Track Proceedings of the 4th International Conference on Mobile Data Management (MDM2003) pp. 1–6.

(Hernandez, 2006)

N. Hernandez, J. Mothe, B. Ralalason, and P. Stolf. Modèle de représentation sémantique des documents électroniques pour leur réutilisabilité dans l'apprentissage en ligne". In Proc. CIDE, pp.181-198 (2006).

(Hervas, 2011)

Ramón Hervás, , José Bravo. Towards the ubiquitous visualization: Adaptive user-interfaces based on the Semantic Web. Interacting with Computers Volume 23, Issue 1, Pages 40–56, January (2011).

(Hocine, 2011)

Nadia Hocine, Abdelkader Gouaïch, Ines Di Loreto, Lylia Abrouk. Etat de l'art des techniques d'adaptation dans les jeux ludiques et sérieux. RIA jeux sérieux. Vol.1, pp.30. DOI:10.3166. Lavoisier, Paris (2011).

(Hussein, 2011)

Mahmoud Hussein, Jun Han, Alan Colman. An Architecture-based Approach to Context-aware Adaptive Software Systems. Technical Report, Swinburne University of Technology, Australia. January (2011).

I

(Indulska, 2003)

Indulska, J., Robinsona, R., Rakotonirainy, A., and Henricksen, K. Experiences in using cc/pp in context-aware systems. In LNCS 2574:Proceedings of the 4th International Conference on Mobile Data Management (MDM) - Lecture Notes in Computer Science (LNCS), Springer, pp. 247–261 (2003).

(Ikeda, 1998)

Ikeda, M. Ontology as Theoretical Foundations of Knowledge Engineering. J. Jpn. Soc. for Artificial intelligence, 13, pp. 11-13 (1998).

J

(Jamal, 2005)

Jamal, S. Environnement de procédé extensible pour l'orchestration - Application au service web. PhD thesis, Université Grenoble UJF, (2005).

K

(Kaldeli, 2010)

Eirini Kaldeli, Ehsan Ullah Warriach, Jaap Bresser, Alexander LazoviketMarco Aiello. Interoperation, Composition and Simulation of Services at Home. Service-oriented Computing Lecture Notes in Computer Science, Volume 6470, pp. 167-181. (2010).

(Keller, 2007)

Uwe Keller, Ruben Lara, Holger Lausen, Dieter Fensel: Semantic Web Service Discovery in the WSMO Framework. In Semantic Web Services: Theory, Tools and Applications, Chapter XII pp, 281-324, (2007).

(Kellert, 2004)

Patrick Kellert, Farouk Toumani. Les Web services sémantiques. In Interaction Intelligence Information I3 International Journal, Janvier (2004).

(Khadka, 2010)

Khadka, R., Sapkota, B. An Evaluation of Dynamic Web Service Composition Approaches. In: 4th International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing - ACT4SOC, Athens, Greece (2010).

(Khelil, 2001)

N. Khelil. Man-Machine Interface Modeling. Mémoire de master, université de Tunis, October, (2001).

(Kiciman, 2000)

Kiciman, E., Fox, A. Using dynamic mediation to integrate COTS entities in a ubiquitous computing environment. Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing (HUC2K). Heidelberg, Germany: Springer-Verlag. (2000).

(Kiss, 2007)

Cédric Kiss, Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0 - W3C Working Draft, April (2007).

Available at: http://www.w3.org/TR/2007/WD-CCPP-struct-vocab2-20070430/ - last consultation October (2009).

(Klush, 2012a)

Matthias Klusch, Patrick Kapahnke: Adaptive signature-based semantic selection of services with OWLS-MX3. Multiagent and Grid Systems 8(1): pp. 69-82 (2012).

(Klush, 2012b)

Matthias Klusch, Patrick Kapahnke: The iSeM matchmaker: A flexible approach for adaptive hybrid semantic service selection. J. Web Sem. 15: 1-14 (2012).

(Korpipaa, 2003)

Korpipaa, P., Matyjarvi, J., Kela, J., Keranen, H., Malm, E.J. Managing Context Information in Mobile Devices. IEEE Pervasive Computing, Mobile and Ubiquitous Systems pp. 42–51 (2003).

(Kranz, 2010)

Kranz, M., Holleis, P., Schmidt, A.: Embedded Interaction: Interacting with the Internet of Things. Internet Computing, IEEE, vol.14, no.2, pp.46-53 (2010).

(Kubski, 2013)

kubski S., Lepreux S., Kolski C. Distributed UI on interactive tablets ps: issues and context model. M.D. Lozano, J.A. Gallud, R. tesoriers, V.M.R Peschet (Eds.), Distributed user interfaces: collaboration and usability, Springer, pp. 27-38 (2013).

L

(Lee, 2001)

Berners-Lee, T., Hendler, J., Lassila, O. The Semantic Web. ScientificAmericain, (2001).

(Leonidis, 2012)

A. Leonidis, M. Antona, C. Stephanidis. Rapid Prototyping of Adaptable User Interfaces. Presented at Int. J. Hum. Comput. Interaction, pp.213-235, (2012).

(Lima, 2011)

Lima, J.C.D., Rocha, C.C., Augustin, I. Dantas, M.A.R. A Context-Aware Recommendation System to Behavioral Based Authentication in Mobile and Pervasive Environments. Embedded and Ubiquitous Computing (EUC), vol., pp.312-319. Oct. (2011).

M

(Maesano, 2003)

L. Maesano, C. Bernard, X. Le Balles. Services web avec J2EE et .net Conception et implémentation. ISBN: 2-212-11067-7. Edition eyrolles, pp 1021,1022, Sep (2003).

(Marinesco, 2008)

S. Marinesco, P. Pernot Biocapteurs implantables in vivo. In Nanotechnologies et biotechnologies pour la santé. 12 – (2008) Editions T.I. Available at: http://www.techniquesingenieur.fr/basedocumentaire/biomedicalharmath15/nanotechnologies et-biotechnologies-pour-la-sante-42608210/biocapteurs-implantables-re108/

(Martin, 2004)

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sicara, K. Owl-s: Semantic markup for webservices. Tech. rep., France Telecom, MINDL Maryland, NIST, Nokia, (2004).

(Masmoudi, 2013)

Masmoudi, N., Conan D. Contrats de contexte pour la gestion de contexte répartie. Actes 9è journées francophones Mobilité et Ubiquité, Nancy, (2013).

(Matej, 2010)

Matej, P., Matej, K., Janez, P., Gašper, M., Goran, V., Stanislav, K.: Analysis of multi-agent activity using Petri nets. Pattern Recognition, 43 (4), p.1491-1501, (2010).

(Maxidico, 1997)

Dictionnaire encyclopédique de la langue française. Classification Dewey. Editions de la connaissance. (1997).

(McCarthy, 1993)

McCarthy, J. Notes on formalizing contexts. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (San Mateo, California, R. Bajcsy, Ed., Morgan Kaufmann, pp. 555–560 (1993).

(McCarthy, 1997)

McCarthy, J., Buvac. Formalizing context (expanded notes). In Working Papers of the AAAI Fall Symposium on Context in Knowledge Representation and Natural Language (Menlo Park, California), S. Buvac and L. Iwanska, (Eds.). American Association for Artificial Intelligence, pp. 99–135 (1997).

(McIlraith, 2001)

S. McIlraith, T.C. Son, H. Zeng. Semantic Web Services. IEEE Intelligent Systems. Special Issue on the Semantic Web, 16(2):46–53 (2001).

(McIlraith, 2003)

McIlraith, S., Martin D. Bringing Semantics to Web Services. IEEE Intelligent Systems. vol.18, n°1, pp.90-93, (2003).

(Moalla, 1985)

M. Moalla, Réseaux de Petri Interprétés et Grafcet, technique et science Informatique, Vol. 14, no. 1, pp. 17-30, (1985).

(Moussa, 2000)

Moussa, F., Kolski, C., Riahi, M. A model based approach to semiautomated user interface generation for process control interactive applications. Interacting with Computers, 12, pp. 279-292, (2000).

(Moussa, 2002)

F. Moussa, M. Riahi, C. Kolski and M. Moalla. Interpreted Petri Nets used for Human-Machine Dialogue Specification in Process Control: principles and application to the Ergo-Conceptor+ tool. Integrated Computer-Aided Engineering, 9, pp. 87-98, (2002).

N

(Najar, 2009)

Salma Najar, Oumaima Saidani, Manuele Kirsch-Pinheiro, Carine Souveyet and Selmin Nurcan. Semantic Representation of Context Models: A Framework for analyzing and understanding. 1st Workshop on Context, Information and Ontologies 6th European Semantic Web Conference, Heraklion: Greece (2009).

(Nagati, 2001)

Pr. Nagati Khmais, Dr. Kammoun Mervet et Pr. Abid Abdelmajid. Programme national de prise en charge des diabétiques dans les structures de 1ère ligne. Editions: direction des soins de santé de base. Janvier (2001).

(Natalya, 2001)

Natalya F. Noy, Deborah L. Ontology Development. A Guide to Creating Your First Ontology. Mcguinness. (2001).

Available at: http://protege.stanford.edu/publications/ontologydevelopment/ontology101-noy-mcguinness.html, Last consultation: March (2013).

(Nevejan, 2010)

Nevejan, C., Brazier, F. M. T., Witnessed Presence in Merging Realities in Healthcare Environments, in: Advanced computational Intelligence Paradigms in Healthcare 5: Intelligent Decision Support Systems, Springer-Verlag, (2010).

(Nezhad, 2010)

Nezhad, Hamid Reza Motahari, Xu, Guang Yuan, Benatallah, Boualem. Protocol-aware matching of web service interfaces for adapter development. Proceedings of the International Conference on the World Wide Web. pp. 731-740 (2010).

(Niels, 2003)

Niels O. Bouvin, Bent G. Christensen, Kaj G. and k, Frank A. HyCon: a framework for context-aware mobile hypermedia. Hansen Hypermedia, Vol. 9, No. 1. (January 2003), pp. 59-88.

O

(Osman, 2005)

Osman, T., Thakker, D., Al-Dabass, D. Bridging the gap between workflow and semantic-based web services composition. Proceedings of WWW Service composition with Semantic Web Services, Compiegne, France, pp. 13–23 (2005).

(Ouali, 2005)

Anis Ouali, Brigitte Kerhervé, Paul Landon: An adaptation framework for new media artworks. ACM Multimedia 586-589 (2005).

(Ouali, 2006)

Anis Ouali, Brigitte Kerhervé, Odile Marcotte, Paul Landon: Adaptation Scenarios for New Media Artworks. ICME : pp. 1933-1936 (2006).

(OWL-S, 2004)

Semantic Markup for Web Services, available at: http://www.w3.org/Submission/OWL-S/. Last update, November (2004). Last consultation May 2011.

P

(Palanque, 1996)

Philippe Palanque, Rémi Bastide. Time modelling in Petri nets for the design of interaction active. Published in ACM SIGCHI BULLETIN (Volume 28 Issue 2). ACM New York, NY, USA. Pp 43-46 (1996).

(Paolucci, 2002)

Paolucci, M., Kawamura, T., Payne, T., Sycara, K.: Semantic matching of web services capabilities. Proceedings of the First International Semantic Web Conference, LNCS 2342, Springer-Verlag. Pp. 333–347 (2002).

(Pascoe, 1998)

J. Pascoe, N. Ryan, and D. Morse. Human-Computer-Giraffe Interaction – HCI in the field. Proceedings of the Workshop on Human Computer Interaction with Mobile Devices, Glasgow, Scotland, (1998).

(Pascoe, 1999)

Jason Pascoe, Nick Ryan, and David Morse. Issues in Developing Context-Aware Computing. In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, Hans-Werner Gellersen (Ed.). Springer-Verlag, London, UK, UK, 208-221 (1999).

(Pascoe, 2001)

J. S. Pascoe, R. J. Loader and V. S. Sunderam. An Election Based Approach to Fault-Tolerant Group MemberShip in Collaborative Envrionment. 25th IEEE Anniversary Annual International Computer Software and Applications Conference (COMPSAC), pages 196-201, Chicago, II. IEEE Press. ISBN: 0-7695-1372-7 (2001).

(Patonnier, 2013)

Jérémie Patonnier , Rudy Rigot. Projet responsive web design. ISBN13 : 978-2-212-13713-2 (2013). Editeur(s) : Eyrolles

(Petri, 1962)

C. A. Petri: Fundamentals of a Theory of Asynchronous Information Flow: 386-390 (1962).

(Picot-Clémente, 2011)

Romain Picot-Clémente. Une architecture générique de Systèmes de recommandation de combinaison d'items. Application au domaine du tourisme. Thèse de Doctorat en Sciences Informatiques, présenté le 07 Decembre 2011. Laboratoire Le2i de l'Université de Bourgogne.

(Phan, 2005)

Phan Quang Trung Tien. Ontologies et Web Services. Activity Report. Institut de la Francophonie pour l'Informatique. Hanoi, Juillet (2005).

(Puerta, 2004)

Angel Puerta and Jacob Eisenstein. XIML: A Universal Language for User Interfaces RedWhale Software. Available at http://www.ximl.org (Dernière consultation Novembre 2012)

Q

(Quade, 2011)

Michael Quade, Marco Blumendorf, Grzegorz Lehmann, Dirk Roscher and Sahin Albayrak. Evaluating User Interface Adaptations at Runtime by Simulating User Interaction. In: 25th BCS Conference on Human Computer Interaction – HCI, (2011).

R

(Ramos, 2005)

Angela Carrillo Ramos, Jérôme Gensel, Marlène Villanova-oliver, Hervé Martin. PUMAS: Un Framework que Adapta la Información en Ambientes Ubicuos. Colombian Journal of Computation - RCC , vol. 6, no. 2, pp. 28-47, (2005).

(Rasch, 2011)

Rasch, Katharina. Li, Fei. Sehic, Sanjin, Ayani, Rassul; Dustdar, Schahram. Context-driven personalized service discovery in pervasive environments. In World Wide Web, Volume: 14, Issue: 4, 2011-07-01. Pp 295-319. Springer Netherlands, Issn: 1386-145X.

(Razmerita, 2003)

Razmerita, L. V. Modèle Utilisateur et Modélisation Utilisateur dans les Systèmes de Gestion des Connaissances: une Approche fondée sur les Ontologies (2003).

(Riahi, 2001)

Riahi, M., & Moussa, F. Contribution of the Petri Nets and the multi Agent system in HCI Specification. 9th International (2001).

(Richards, 2003)

Richards, D., Sabou, M., van Splunter, S., Brazier, F. M. T., Artificial Intelligence: A Promised Land for Web Services. The Proceedings of The 8th Australian and New Zealand Intelligent Information Systems Conference (ANZIIS2003), Macquarie University, Sydney, Australia, pp. 205-210, (2003).

(Roscher, 2011)

Dirk Roscher, Grzegorz Lehmann, Veit Schwartze, Marco Blumendorf, Sahin Albayrak. Dynamic Distribution and Layouting of Model-Based User Interfaces in Smart Environments. In: Springer Berlin / Heidelberg. (2011).

(Rouillard, 2012)

Rouillard José. The Pervasive Fridge. A smart computer system against uneaten food loss, The Seventh International Conference on Systems, ICONS, Saint Gilles, Reunion Island, (2012).

(Ruben, 2008)

Ruben Lara, Miguel Angel Corella, Pablo Castells: A flexible model for service discovery on the Web, Semantic Web Services with WSMO, Special Issue on Semantic Matchmaking and Resource Retrieval, International Journal of Electronic Commerce, Volume 12, Number 2, pp 11-41 (2008).

(Ryan, 1997)

N. Ryan, J. Pascoe, and D. Morse. Enhanced reality fieldwork: the context-aware archaeological assistant. In Gaffney, V. et al. (Eds.) Computer Applications in Archaeology, (1997).

S

(Samaan, 2006)

Kinan Samaan. Thèse de doctorat de l'Ecole Centrale de Lyon- Ecole Doctorale Informatique : Prise en Compte du Modèle d'Interaction dans le Processus de Construction et d'Adaptation d'Applications Interactives (2006)

(Savidis, 2004)

Savidis, A., Leonidis, A., Lilis, I., Moga, L., Gaeta, E., Villalar, J.L., Fioravanti, A. & Fico, G. Self-configurable User Interfaces. ASK-IT Deliverable - D3.2.1 (2004).

(Schilit, 1994)

Schilit, Bill N., Adams, Norman I. and Want, Roy. Context-Aware Computing Applications, in Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, December 1994. IEEE Computer Society (1994).

(Schlegel, 2011)

Thomas Schlegel, Christine Keller. Model-Based Ubiquitous Interaction Concepts and Contexts in Public Systems. In proceeding of: Human-Computer Interaction. Design and Development Approaches - 14th International Conference, HCI International, Orlando, FL, USA, July 9-14 (2011).

(Schmidt, 1999)

A. Schmidt, K. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, and W. Van de Velde, Advanced Interaction in Context, 1th International Symposium on Handheld and Ubiquitous Computing (HUC99), Lecture notes in computer science, vol. 1707, Springer, pp. 89-101, 1999.

(Sowa, 2000)

J. F. Sowa. Ontology, Metadata, and Semiotics. In proceedings of the International Conference on Conceptual Structures, ICCS'2000, Darmstadt, Germany, August (2000).

(Splunter, 2010)

van Splunter, S., van Langen, P. H. G. and Brazier, F. M. T., Using composition knowledge for design adaptations, poster at fourth international conference on design computing and cognition (DCC10), 2010.

(Stephanidis, 2001)

Stephanidis, C., Paramythis, A., Sfyrakis, M., Savidis, A. A Case Study in Unified User Interface Development: The AVANTI Web Browser. C. Stephanidis (Ed.), User Interfaces for All - Concepts, Methods, and Tools (pp. 525-568). Mahwah, NJ: Lawrence Erlbaum Associates (2001).

(Stephanidis, 2004)

Stephanidis, C., Paramythis, A., Zarikas, V., Savidis, A. The PALIO Framework for Adaptive Information Services. In A. Seffah & H. Javahery (Eds.), Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces (pp. 69-92). Chichester, UK: John Wiley & Sons, Ltd (2004).

(Strang, 2004)

Thomas Strang and Claudia Linnhoff-Popien. A Context Modeling Survey. Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp - The Sixth International Conference on Ubiquitous Computing, Nottingham/England (2004).

(Suchman, 1987)

Suchman L. Plans and situated actions: the problem of human/machine communication, Cambridge University Press (1987).

(Syväniemi, 2011)

Susanna Pantsar-Syväniemi, Jarkko KuusijärvietEila Ovaska. Context-Awareness Micro-architecture for Smart Spaces. In Advances In Grid and Pervasive Computing, Lecture Notes in Computer Science. Volume 6646, Pp 148-157 (2011).

T

(Thevenin, 2001)

David Thevenin. Adaptation en Interaction Homme-Machine : le cas de la Plasticité. Thèse de Doctorat en Informatique de l'Université Joseph Fourier, Grenoble. Décembre (2001).

(Timmerer, 2009)

Christian Timmerer, Johannes Jabornig, and Hermann Hellwagner. Delivery Context Descriptions A Comparison and Mapping Model. Proceedings of the 9th Workshop on Multimedia Metadata held in conjunction with the 13th French Multimedia Conference on Compression and Representation of Audiovisual Signals (2009).

(Tran, 2009)

Tran, M.H., Han, J., Colman, A.: Social context: Supporting interaction awareness in ubiquitous environments. In Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous, pp 1-10 (2009).

V

(Vidal, 2011)

Vidal, J.C., Lama, M., Bugarín, A. : Towards the use of Petri nets for the formalization of OWL-S choreographies. In Knowledge and Information Systems. Springer-Ed (2011).

W

(Weiser, 1993)

Mark Weiser, some computer science issues in ubiquitous computing. Commun. ACM. 36(7): 74-84. (1993).

(Williem, 1988)

R. Williem and V. Biljon. Extending Petri Nets for specifying Man-Machine dialogues, International journal of Man-Machine Studies, vol. 28, pp. 437-445 (1988).

(Winograd, 2001)

Winograd, T. Architectures for context, Hum.-Comput. Interact. 16(2): 401–419 (2001).

(WSDL, 2001)

Erik Christensen, Francisco Curbera, Greg Meredith and Sanjiva Weerawarana Heidelberg. Web Services Description Language (WSDL) 1.1 W3C Note. Latest version: http://www.w3.org/TR/wsdl, March (2001).

Y

(Ye, 2005)

A Ye, Jing-Hua, A Herbert, A John. Human-Computer Interaction - INTERACT 2005- V 3585 - Lecture Notes in Computer Science - Adaptive User Interfaces Development Platform Springer Berlin Heidelberg, Pp 1034-1037 (2005).

(Ye, 2012)

Juan Ye, Simon Dobson, Susan McKeever, Situation identification techniques in pervasive computing: A review, Pervasive and Mobile Computing, Volume 8, Issue 1, Pages 36-66, ISSN 1574-1192, 10.1016/j.pmcj.2011.01.004, February (2012).

(Yu, 2010)

Yu, B., Coll, S.: Verifying web services of OWL-S with Petri net. In: 5th IEEE International Conference on Computer Science and Education (ICCSE), pp. 891- 896 (2010).

Z

(Zhanga, 2010)

Daqiang Zhanga, Minyi Guoa, Jingyu Zhoua, Dazhou Kangb, Jiannong Caoc. Context reasoning using extended evidence theory in pervasive computing environments. Future Generation Computer Systems. Volume 26, Issue 2, Pages 207–216, February (2010).

PUBLICATIONS

(Ismail, 2010)

Ismail I., Moussa F. User Requirements Deduction in a Pervasive Environment. NGMAST: IEEE International Conference on Next Generation Mobile Application, Services and Technologies. July (2010).

(Ismail, 2011)

Ismail I., Moussa F. Towards a Runtime Evolutionary Models of User Interface Adaptation in a Ubiquitous Environment. The Fifth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies: UBICOMM. Lisbon November (2011).

(Ismail, 2012a)

Ismail I. Contextual Data Management in Ubiquitous Environment. International Conference on Computing and Information Technology. Al-Madinah Al-Munawwarah, Saudi Arabia. March (2012).

(Ismail, 2012b)

Ismail I., Moussa F. A Pervasive System Architecture For Smart Environments. The International Journal of Artificial Intelligence and Applications (IJAIA). Septembre (2012).

(Ismail, 2012c)

Imen Ismail, Faouzi Moussa. Dynamic and Realtime Modelling of Ubiquitous Interaction. The International Conference on Control, Modelling, Computing and Applications (CMCA-2012). December (2012).

(Ismail, 2013)

Imen Ismail, Faouzi Moussa. Realtime User Interface Generation In Ubiquitous Environment. International Journal of Innovative Research in Computer and Communication Engineering. Vol. 1, Issue 9, November (2013).

(Moussa, 2014)

Faouzi Moussa, Imen Ismail. Interaction Homme Machine dans Les Environnements Ubiquitaires. IHM de l'Association Francophone d'Interaction Homme-Machine. Lille. octobre (2014).