Vers une approche de spécification et de génération d'interfaces sensibles au contexte : Application au domaine médical

Ines Riahi

RÉPUBLIQUE TUNISIENNE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ DE LA MANOUBA

ÉCOLE NATIONALE DES SCIENCES DE L'INFORMATIQUE

THÈSE EN INFORMATIQUE

En vue de l'obtention du diplôme de

Docteur en Informatique

Présentée par :

Ines Riahi Maiza

Vers une approche de spécification et de génération d'interfaces sensibles au contexte : Application au domaine médical

Réalisée au sein du
Laboratoire CRISTAL

Soutenue le 18/04/2016 devant le jury composé de :

Président : Faouzi GHORBEL (Professeur, Ecole Nationale des Sciences de l'Informatique)

Rapporteur : Sadok BEN YAHIA (Professeur, Faculté des Sciences de Tunis)

Rappoteur : Christophe KOLSKI (Professeur, Université de Valenciennes)

Examinateur : Wided CHAARI (Maître de conférences, Ecole Nationale des Sciences de l'Informatique)

Directeur de Thèse : Faouzi MOUSSA (Professeur, Faculté des Sciences de Tunis)

Dédicaces

Toutes les lettres ne sauraient trouver les mots qu'il faut. Tous les mots ne sauraient exprimer la gratitude, l'amour, le respect, la reconnaissance. Aussi, c'est tout simplement je dédie ce travail à :

A mon papa chéri

Je ne trouverai de mots assez forts pour t'exprimer mon affection, mon amour, mon estime et mon dévouement pour ta patience, ta compréhension, tes innombrables encouragements. Puisse dieu t'accorder une bonne santé et une longue vie.

A ma maman chérie

Tu représentes pour moi le symbole de la bonté par excellence, la source de tendresse et l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi. Merci tout simplement d'être ma mère.

A mon mari chéri

Merci d'être toujours là pour moi. Merci pour ta patience, ta tolérance, ton amour et tes encouragements.

A ma jumelle Onna et à mon frère Samsoumi

Puisse dieu, le tout puissant, vous procurer santé et longue vie et vous réserver un avenir plein de succès comme vous le souhaitez.

A ma grand-mère, mes tantes, mes oncles, mes cousins ainsi qu'à mes cousines pour leur soutien moral et leurs encouragements. A mes très chères amies : Ghaya Tej el Molk, Oumaya, Abir et Ines pour leur présence, leur soutien moral et leurs encouragements. A mes beaux parents, à Arwa, Safa, Adel et Wassim pour leur soutien moral et leurs encouragements. A tous ceux qui me sont trop chers et que j'ai omis involontairement de citer. A tous ceux qui m'ont aidé de près ou de loin dans la réalisation de ce travail. A tous mes professeurs et maîtres, avec tous mes respects et mon éternelle reconnaissance.

Remerciements

Il m'est très agréable de réserver cette page comme un témoin de reconnaissance à toutes les personnes qui m'ont soutenue et encadrée pour réaliser ce travail.

Je tiens à exprimer ma profonde gratitude à mon très cher encadrant Monsieur ***Faouzi Moussa****, Professeur à la Faculté des Sciences de Tunis, qui, en plus de son soutien amical constant, m'a encadrée, guidée et conseillée avec enthousiasme et rigueur tout au long de cette thèse. Je lui suis en outre très reconnaissante pour m'avoir fourni les moyens de mener mes recherches dans d'excellentes conditions humaines et matérielles. Je tiens également à remercier Madame* ***Meriem Riahi****, Maître assistante à l'ENSIT, pour ses remarques précieuses. Pour les longues heures de discussion, qui étaient une source de repos et loisirs, grande est ma dette envers elle.*

Je suis très reconnaissante envers Monsieur ***Sadok Ben Yahia****, Professeur à la Faculté des Sciences de Tunis, et Monsieur* ***Christophe Kolski****, Professeur à l'Université de Valenciennes, de m'avoir fait l'honneur d'examiner ce travail et d'en être les rapporteurs.*

Je remercie également Monsieur ***Faouzi Ghorbel****, Professeur à l'Ecole Nationale des Sciences de l'Informatique, pour l'honneur qu'il me fait d'accepter de présider le jury de soutenance de ce mémoire.*

Je tiens aussi à remercier Madame ***Wided Chaari****, Maître de conférences à l'Ecole Nationale des Sciences de l'Informatique pour avoir accepté d'examiner mon travail.*

Mes remerciements s'étendent également à mes amis et collègues de la Faculté des Sciences de Tunis : Hmida, Rami, Nidhal, Néjib, Mohamed, Aymen, Sawssen, Khawla, Hajer et Mouna. Je les remercie tous pour cette opportunité et cette atmosphère de recherche conviviale.

Table des matières

Liste des figures

Liste des tableaux

Acronymes

ACC/PP	Augmented Composite Capabilities/Preferences Profile
ADO	Anti Diabétique Orale
BIO	Besoins Informationnels de l'Opérateur
CC/PP	Composite Capabilities/Preferences Profile
CCTT	Collaborative Concur Task Trees
CFG	Context-Free Grammars (les grammaires hors-contexte)
CML	Context Modeling Language (Langage de Modélisation du Contexte)
CSCP	Comprehensive Structured Context Profiles
CTML	Collaborative Task Modeling Language
CTT	Concur Task Trees
DiaMon	Diabete Monitoring
ECG	Electrocardiogramme
GL	Glucose Level
GOMS	Goals, Operators, Method, Selection
GPS	Global Positioning System
Hba1c	Hémoglobine glyquée
HMM	Hidden Markov Model (Modèle caché de Markov)
IHM	Interaction Homme-Machine
IU	Interface Utilisateur
JSON	JavaScript Object Notation
MMS	Multimedia Messaging Service
OMA	Open Mobile Alliance
ORM	Object-Role Modeling (Modélisation Objet-Rôle)
OWL	Ontology Web Language
OWL/DL	Ontology Web Language Description Logics
PDA	Personal Digital Assistant

PNML	Petri Net Markup Language
RDF	Resource Description Framework
RDF/S	Resource Description Framework Schema
RdP	Réseau de Petri
RdPC	Réseau de Petri Coloré
RdPI	Réseau de Petri Interprété
SCFG	Stochastic Context-Free Grammars (Les grammaires stochastiques hors-contexte)
SHM	Système Homme-Machine
SWRL	Semantic Web Rule Language
TEA	Time, Events and Architectures
UAProf	User Agent Profile
UML	Unified Modeling Language
W3C	World Wide Web Consortium
WGS84	World Geodesic System 1984
XML	Extensible Markup Language

Introduction générale

L'étude de l'Interaction Homme-Machines (IHM) représente un domaine multidisciplinaire. En effet, il englobe plusieurs domaines, entre autres l'informatique, la sociologie, l'ergonomie, la psychologie et l'ingénierie. Cependant, l'informatique demeure l'axe central de l'IHM.

Afin de concevoir un logiciel tout en prenant en compte un cycle de vie bien particulier, il est impératif de prendre en considération les exigences d'un système interactif. La conception, la mise en œuvre et l'évaluation du système sont réalisées tout en tenant compte de la tâche de l'utilisateur ainsi que de l'environnement dans lequel il évolue. La création d'un bon système interactif repose en particulier sur le modèle de la tâche. L'analyse et la modélisation de la tâche représentent, en outre, des concepts fondamentaux de l'interaction Homme-Machine puisqu'elles permettent d'intégrer les tâches dans le processus de développement du système interactif.

Le processus de développement doit être structuré et bien organisé d'où le terme d'ingénierie. L'ingénierie décrit la manière de concevoir un objet par rapport à un certain nombre de critères tels que la qualité, la maintenabilité et la traçabilité. De ce fait, l'ingénierie de l'Interaction Homme-Machine consiste à appliquer des techniques de développement structurées tout en respectant certaines règles importantes comme la facilité d'utilisation et l'ergonomie des interfaces, ce qui garantit des interfaces moins sujettes aux risques d'erreurs. Pour ce faire, plusieurs outils ont été proposés ces dernières années pour créer, éditer, manipuler, implémenter, visualiser, valider et vérifier des IHM [Giulio *et al.*, 2002] [Reichart *et al.*, 2008] [Vanderdonckt, 2011].

Les Interfaces utilisateur évoluent de jour en jour et de nouvelles problématiques sont apparues. En effet, les environnements dans lesquels interagissent les utilisateurs via leurs IHM, sont devenus de plus en plus compliqués. Les environnements actuels représentent des environnements ubiquitaires, intelligents et pervasifs, puisqu'ils englobent plusieurs sortes de capteurs qui fournissent différentes informations aux utilisateurs. L'IHM a, de ce fait, évolué

et est devenue une IHM qui interagit directement avec l'environnement ubiquitaire où elle se trouve.

L'informatique ubiquitaire intègre donc, de manière transparente, des ordinateurs dans la vie quotidienne en réponse à des informations fournies par des capteurs dans l'environnement. Elle implique un certain nombre d'entités informatiques qui interagissent à la fois avec les utilisateurs et l'environnement dans lequel elles opèrent. A l'aide de ces entités, un système informatique omniprésent est capable de fournir des services personnalisés aux utilisateurs d'une manière sensible au contexte quand ils sont en interaction et en échange d'informations avec l'environnement [Ye *et al.*, 2012].

L'informatique ubiquitaire ou diffuse offre donc la capacité de construire des systèmes qui peuvent adapter l'information délivrée, aux besoins des utilisateurs. Ce nouveau type d'interaction soulève la question de l'interface sensible au contexte. En effet, nous sommes passés d'une conception des interfaces Homme-Machine centrée utilisateur vers une conception centrée contexte. Les Systèmes Homme-Machine (SHM) deviennent sensibles au contexte. Le SHM passe du triplet (Système, Tâche, utilisateur) au triplet (Système, Tâche, Contexte). La conception, la mise en œuvre et l'évaluation du SHM sensible au contexte doit être régie par la tâche de l'utilisateur tout en prenant en considération le contexte dans lequel il évolue.

La prise en compte de la tâche de l'utilisateur représente un critère primordial dans un environnement où le contexte possède un impact direct sur la tâche. L'interface de l'utilisateur doit changer, simultanément, selon le contexte et la tâche. La modélisation de l'environnement ubiquitaire ainsi que la tâche de l'utilisateur présentent plusieurs problématiques. En effet, les environnements intelligents présentent une dynamique assez importante et une quantité d'information conséquente. De plus, la tâche de l'utilisateur peut varier d'un instant à un autre selon le changement du contexte et des intentions de l'utilisateur.

Dans les domaines critiques tels que le domaine médical, le nucléaire ou les transports, la modélisation du SHM sensible au contexte doit être assez rigoureuse afin de diminuer au maximum le risque d'erreur. Si l'interface de l'utilisateur indique des données erronées, le risque de conséquences éventuellement désastreuses sur les vies humaines et sur les équipements augmente. Il faut donc trouver un formalisme de modélisation assez rigoureux pour décrire ces deux concepts et proposer par la suite un lien entre la tâche et l'environnement ubiquitaire afin de les intégrer dans une même modélisation.

Positionnement de la problématique

L'adaptation de l'interface mobile à son contexte d'usage représente un axe de recherche en plein essor. Cela est justifié par l'expansion constante des dispositifs électroniques. Le progrès des appareils mobiles, des réseaux sans fils et des capteurs ont permis l'émergence de plusieurs approches d'adaptation au contexte.

Suite à une étude approfondie de l'état de l'art, nous avons constaté que la plupart des approches proposaient des systèmes d'adaptation assez performants. Nous avons également remarqué le manque d'engouement autour de la prise en considération de la tâche de l'utilisateur et de la validation de l'interface générée. Nous avons noté la rareté des travaux qui prenaient la tâche de l'utilisateur comme axe fondamental dans la génération d'interfaces adaptables au contexte. Or, cette phase est primordiale lors de l'analyse du SHM sensible au contexte puisque la tâche varie selon le contexte. Nous nous sommes également attardés sur l'étude des différentes approches de modélisation du contexte et nous avons remarqué que la majorité d'entre elles n'utilisait pas de méthode formelle lors de la modélisation du contexte. L'utilisation d'une approche formelle dès la phase de modélisation permet la génération d'une interface valide avant la phase d'implémentation et de validation.

Une interface contextualisée doit être valide, fiable et ne présenter aucune défaillance. Elle doit être consciente de son environnement, du profil de l'utilisateur connecté et de la plateforme sur laquelle elle est déployée.

Finalement, nous nous sommes rendus compte, qu'il n'existait pas de travaux concernant l'évaluation des interfaces contextualisées. La majorité des travaux ont été proposés par la communauté des ergonomes et se sont plutôt penchés sur la proposition de critères ergonomiques destinés aux interfaces statiques. Aucun travail n'a comparé les interfaces statiques par rapport aux interfaces adaptables au contexte.

Nos travaux de recherche se situent dans cette optique. En effet, nous avons essayé de proposer une approche formelle de spécification d'interfaces adaptables au contexte.

Contribution

La contribution de cette thèse consiste en la proposition d'une approche formelle de spécification d'interfaces sensibles au contexte qui couvre toutes les phases, de la phase d'analyse du SHM jusqu'à la génération et l'évaluation de l'interface générée. Cette contribution a été réalisée en suivant plusieurs étapes. En effet, nous nous sommes intéressés au début de nos

travaux de recherche, à la manière de modéliser le contexte. Nous avons proposé à cet effet une modélisation basée sur un langage formel à savoir les réseaux de Petri (RdP).

Par la suite, nous nous sommes intéressés à l'éventuelle relation entre le contexte et la tâche. Nous avons ainsi analysé et modélisé la tâche de l'utilisateur dans un environnement ubiquitaire pour déduire cette relation. Ceci a fait l'objet de notre deuxième contribution.

Après avoir modélisé le contexte et la tâche, l'objectif était de spécifier les besoins de l'utilisateur en termes d'information. La question qui s'est posée à ce stade est comment traduire nos modèles en un langage compréhensible par la machine. Petri Net Markup Language (PNML) a été proposé à cet effet. Cette étape ne représentait en réalité, qu'une transition pour la spécification des objets graphiques de l'interface contextualisée.

Nous avons également proposé, comme dernière contribution, une plateforme expérimentale pour l'évaluation des interfaces contextualisées. Nous avons réalisé une étude afin de comparer les interfaces statiques avec celles contextualisées selon des critères ergonomiques bien choisis.

Application

Afin de démontrer la force des approches formelles dans la modélisation des SHM sensibles au contexte, nous avons choisi de travailler sur un domaine critique à savoir le domaine médical. Nous avons repris l'exemple proposé par [Ben Ismail, 2014] qui traite le suivi des patients diabétiques. En effet, le diabète est une maladie très répandue dans le monde et le nombre de décès liés à une hypoglycémie ne cesse d'augmenter. Afin de faciliter la vie des personnes diabétiques, plusieurs applications ont été proposées. Nous avons remarqué au niveau de nos études la rareté des applications destinées à l'équipe médicale. En effet, la majorité des applications proposées sont destinées aux patients. Elles s'intéressent à l'autorégulation du taux d'insuline.

De ce fait, nous avons proposé une application nommée DiaMon (Diabete monitoring) destinée aux professionnels afin des les aider à mieux accomplir leurs tâches. DiaMon est une application qui a pour objectif :

- D'identifier l'état du patient ;
- De solliciter les personnes adéquates en cas d'intervention urgente ;
- De s'adapter au profil et aux connaissances de l'utilisateur connecté ;
- De fournir à l'utilisateur les informations adéquates pour son intervention ;
- De s'adapter selon le contexte de l'utilisateur et selon l'état du patient.

La mise en place de notre plateforme expérimentale pour la comparaison entre les interfaces statiques et celles contextualisées a été basée sur cette application.

Organisation de la thèse

Ce mémoire présente les résultats de nos travaux de recherche et se compose de cinq chapitres.

Le premier commence par introduire les notions de base de notre thématique à savoir : les notions du contexte, de la sensibilité au contexte et de l'adaptation. Nous avons ensuite présenté les principales architectures de systèmes sensibles au contexte proposées dans la littérature pour finir avec une étude comparative.

Le deuxième chapitre présente une étude bibliographique portant sur les différentes approches de modélisation du contexte. Nous avons commencé par exposer ces différentes approches, puis nous avons définit les exigences d'un système sensible au contexte dans un domaine critique pour pouvoir comparer entre ces différentes approches.

La présentation détaillée de la solution proposée fait l'objet du troisième chapitre.

Le quatrième chapitre s'intéresse à l'expérimentation de cette solution sur une application médicale.

Le cinquième chapitre décrit la mise en place d'un protocol expérimental d'évaluation des interfaces sensibles au contexte. Nous avons établi au niveau de ce chapitre une étude comparative entre les interfaces statiques et les interfaces contextualisées.

Finalement, dans le chapitre « Conclusion générale et perspectives », nous avons repris les éléments clés de nos travaux et nous avons proposé des perspectives à court terme comme la proposition de règles ergonomiques pour l'évalaution des interfaces contextualisées.

chapitre 1

Architectures des systèmes sensibles au contexte

1.1 Introduction

Tout au long de ces dernières décennies, diverses applications provenant de dispositifs mobiles ont progressivement infiltré notre vie quotidienne. L'informatique est devenue invisible comme l'avait prédit Weiser. Sa vision était de pouvoir travailler avec des environnements intelligents qui anticipent, s'adaptent et subviennent aux besoins des utilisateurs [Weiser, 1991]. Le concept de la sensibilité au contexte a pris de l'ampleur dans le domaine des systèmes distribués depuis les années 90, car il semblait être une solution prometteuse pour un grand nombre de problèmes qui ont été impliqués par l'utilisation de terminaux mobiles dans des environnements en constante évolution.

Ce chapitre présentera, dans la première partie, les notions du contexte, de la sensibilité au contexte et de l'adaptation. Un état de l'art des différentes architectures d'adaptation des systèmes sensibles au contexte sera exposé en deuxième partie. Finalement, une étude comparative portant sur ces différentes architectures sera présenté.

1.2 Le contexte et la sensibilité au contexte

1.2.1 La notion de contexte

De nos jours, la notion de contexte est largement utilisée dans plusieurs domaines surtout dans le domaine de l'informatique. Le contexte doit être pris en compte dès le début de la conception du futur système afin d'établir les règles d'adaptation du système aux différents

contextes [Kubicki *et al.*, 2013].

Le dictionnaire Larousse définit le contexte comme étant un : « ensemble de circonstances dans lesquelles se produit un évènement, se situe une action ».

Concernant les définitions proposées par la communauté scientifique, il est clair qu'il n'existe pas une seule et unique définition du contexte et du terme sensibilité au contexte. Nous allons suivre un ordre chronologique des définitions apparues lors des deux dernières décennies. L'une des premières définitions du terme sensibilité au contexte a été introduite par [Schilit *et al.*, 1994] qui associe à un système mobile la localisation et l'identité de l'utilisateur ainsi que les ressources se trouvant dans son environnement.

En 1996, Brown considère le contexte comme étant les objets de l'environnement et ajoute à la localisation de l'utilisateur la notion de temps, de température et de saison [Brown, 1996]. Ryan *et al.* se baseront sur ces travaux pour proposer des ordinateurs qui s'adaptent en fonction des informations de l'environnement, de la position, du temps, de la température ou encore de l'identité des utilisateurs [Ryan *et al.*, 1997].

Par la suite, Ward *et al.* interprètent le contexte en se basant sur la localisation de l'utilisateur et l'état de son environnement. Ils présentent une application qui s'adapte selon la position de l'objet ou de l'utilisateur [Ward *et al.*, 1997].

Dey *et al.* définissent le contexte comme étant un ensemble d'informations qui caractérise la situation d'une entité. Une entité peut être une personne, un lieu ou un objet [Dey *et al.*, 1999]. Au même moment, Thevenin *et al.* approuvent la définition proposée par Dey et présentent l'environnement sous la forme d'un triplet d'entités à savoir *<Objet, Personne, Evènement>*. Ils présentent ainsi la notion de plasticité et d'adaptation des interfaces aux utilisateurs [Thevenin et Coutaz, 1999].

En 2001, Dey complète sa définition en divisant le contexte en un contexte primaire et un autre secondaire. Le premier décrit la situation d'une entité particulaire (temps, lieu, identité...) et le second présente les propriétés et les caractéristiques de ce dernier [Dey *et al.*, 2001].

En 2004, Calvary *et al.* substituent le terme contexte par contexte d'usage et présentent l'adaptation des interfaces utilisateur à leur contexte d'usage. Ils proposent donc un nouveau triplet à savoir *<utilisateur, plateforme, environnement>* [Calvary *et al.*, 2004]. Rey définit ensuite le contexte comme un ensemble d'entités, de relations entre les entités et de rôles [Rey, 2006].

Plus récemment, toutes les définitions du contexte tournent autour de la définition de Dey [Dey *et al.*, 2006]. Toutefois, Pascoe *et al.* ajouteront la notion de contexte social et prendront

en compte les émotions ainsi que l'humeur de l'utilisateur [Pascoe *et al.*, 2007]. Dalmau *et al.* se basent sur la définition de Dey et considèrent le contexte comme « l'ensemble des paramètres externes à l'application pouvant influer sur le comportement d'une application en définissant de nouvelles vues sur ses données et ses services » [Dalmau *et al.*, 2009]. Ye *et al.* rejoignent l'idée de Dey [Dey *et al.*, 2001] et divisent le contexte en un contexte primaire, dont les données découlent directement des capteurs, et un contexte secondaire dont les données sont déduites et proviennent de plusieurs flux de données. Ils considèrent que les données récupérées à partir de capteurs physiques ou virtuels sont regroupées pour former le contexte [Ye *et al.*, 2012].

Les chercheurs dans le domaine de l'adaptation au contexte n'ont pas encore abouti à une définition à la fois générique et pragmatique de la notion du contexte, et plus précisément des paramètres constituant le contexte. Toutefois, après avoir exploré les principales définitions proposées dans la littérature, nous avons remarqué que la majorité tourne autour de la définition proposée par Dey. Nous retiendrons donc pour nos travaux de recherche cette définition ainsi que celle de Calvary *et al.* qui considèrent le contexte comme étant le triplet *<utilisateur, plateforme, environnement>* [Calvary *et al.*, 2004].

1.2.2 La sensibilité au contexte

L'informatique sensible au contexte permet au système d'interagir avec les utilisateurs par le biais de dispositifs sensibles au contexte. Son principal objectif est de maximiser l'efficacité et la convivialité des services [Park *et al.*, 2007], [Han *et al.*, 2008], [Song *et al.*, 2007]. Un système est considéré comme sensible au contexte lorsqu'il *(i)* se base sur les différentes informations contextuelles détectées ou déduites de l'environnement, *(ii)* traite, *(iii)* évalue ces informations et *(iv)* éventuellement décide du meilleur service à fournir à l'utilisateur. La notion d'adaptation des interfaces au profil des utilisateurs et à leurs contextes d'usage est alors apparue. En effet, cette dernière est devenue une nécessité dans un monde où les technologies et les techniques d'adaptations des interfaces à l'environnement courant prennent de plus en plus d'importance.

Dans la section qui suit, nous présentons, en premier lieu, la notion d'adaptation puis nous nous attardons sur l'état de l'art relatif aux mécanismes et architectures d'adaptation.

1.3 L'adaptation : définition et classification

1.3.1 Définition

Concernant les définitions académiques du terme « adaptation », le dictionnaire Larousse stipule que l'adaptation est « l'action d'adapter ou de s'adapter à quelque chose : adaptation aux circonstances ».

Les définitions présentées par la communauté scientifique rejoignent cet esprit. En effet, plusieurs travaux de recherche se focalisent sur la manière d'adapter une interface aux besoins des utilisateurs et à leur environnement et ce, à un moment donné. Capra *et al.* définissent l'adaptation comme étant la capacité d'une application à *(i)* s'auto-configurer selon les changements du contexte, et *(ii)* à fournir le même service sous différentes formes, à des moments et des contextes différents [Capra *et al.*, 2001].

Plusieurs auteurs ont tenté à travers leurs travaux de recherche de fournir une classification au terme « adaptation ». La section qui suit présente les classifications les plus répandues dans la littérature.

1.3.2 Classification du terme « adaptation »

La majorité des travaux [Dalmau *et al.*, 2009] [Da *et al.*, 2011] divisent l'adaptation en deux grandes familles à savoir l'auto-adaptation et l'adaptation supervisée.

L'auto-adaptation : C'est l'application qui gère l'adaptation (Figure 1.1). Ces applications autonomes s'adaptent sans aide extérieure. Elles sont censées s'auto-adapter dynamiquement aux changements des ressources, des besoins des utilisateurs et aux failles des systèmes. Ce genre d'adaptation pose plusieurs problèmes. En effet, c'est assez difficile pour le développeur de l'application auto-adaptative de réutiliser facilement son code ultérieurement. Chaque application propose sa propre solution d'adaptation. L'application ne peut interagir qu'avec son contexte local. Afin d'obtenir ou de modifier à distance des informations contextuelles, le concepteur de l'application doit mettre en place des services spécifiques sur les différents sites de son application. Il devient nécessaire de mettre en place de nombreux mécanismes non fonctionnels qui augmentent fortement la complexité de la demande et sont difficiles à maintenir à jour. Pour toutes ces raisons, la plupart des systèmes ont tendance à résoudre ces problèmes en utilisant l'adaptation supervisée.

L'adaptation supervisée : La plateforme gère l'adaptation et permet l'accès aux informa-

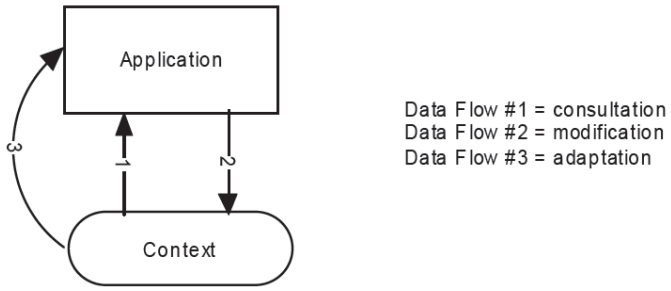


Figure 1.1 — Application auto-adaptable [Dalmau *et al.*, 2009]

tions lointaines (Figure 1.2). L'adaptation est transparente pour les applications. Ces derniers réagissent aux changements du contexte au moyen de la couche plateforme. Dans ce type d'adaptation, les développeurs ne sont pas concernés par les mécanismes d'adaptations, ils s'intéressent seulement à la logique métier de l'application. Le défi de ce type de plateforme est de savoir comment prendre les bonnes décisions pour les différentes applications dans différentes situations.

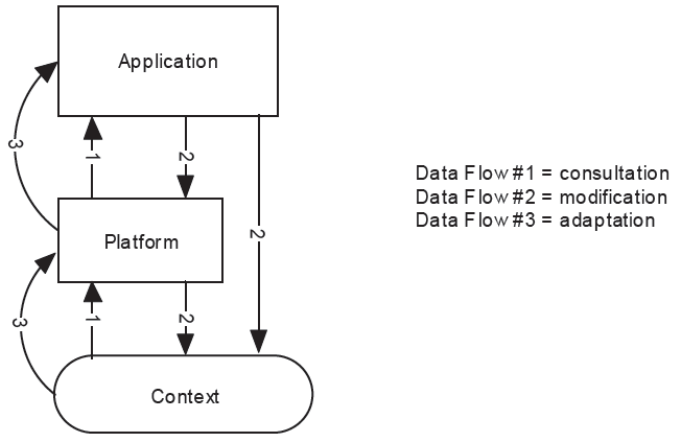


Figure 1.2 — Adaptation supervisée [Dalmau *et al.*, 2009]

Da *et al.* affirment qu'il existe trois raisons pour vouloir adapter une application. Chaque raison définit un type d'adaptation : l'adaptation réactive, l'adaptation évaluative et l'adaptation à l'intégration. *L'adaptation réactive* concerne les applications qui changent en fonction de l'environnement en raison du contexte ou des changements des préférences de l'utilisateur. *L'adaptation évaluative* vise à étendre les fonctionnalités d'une application, afin de corriger ses erreurs ou d'accroître sa performance. *L'adaptation à l'intégration* traite le problème d'in-

tégration des services ou des composants incompatibles (c.-à-d. un matériel différent ou des interfaces logicielles ou des protocoles différents). Ce type d'adaptation est souvent lancé après une décision d'adaptation qui a été faite. Par exemple, la première adaptation est lancée par les changements du contexte. En raison de l'exécution de cette adaptation, un composant ne peut plus communiquer avec un autre car ils ont des protocoles de sécurité différents. Dans ce cas, le système va lancer une adaptation d'intégration [Da *et al.*, 2011].

Ketfi *et al.* proposent de classer l'adaptation en *(i)* adaptation fonctionnelle (modification ou correction de la fonctionnalité du système), *(ii)* adaptation extra-fonctionnelle (agit sur la performance, la précision des applications), *(iii)* adaptation corrective, qui se produit lorsqu'une anomalie est détectée, *(iv)* adaptation adaptative, qui traite l'effet du contexte sur le comportement du système, *(v)* adaptation étendue, qui traite de nouvelles fonctionnalités qui peuvent être ajoutées au système et qui n'ont pas été considérées lors du développement du système et enfin *(vi)* adaptation perfective qui vise à améliorer l'application même si elle tourne correctement [Ketfi *et al.*, 2002] [Kakousis *et al.*, 2010].

Un an plus tard, Jameson présente une classification plus étendue. En effet, il considère deux types d'adaptation : l'adaptation au profil de l'utilisateur et l'adaptation à l'environnement où se situe l'interaction : la plasticité [Jameson, 2003].

La plasticité vise à résoudre les problèmes de dispositifs mobiles utilisés par les usagers lors de leurs interactions. En effet, les appareils mobiles utilisés de nos jours (PDA, ordinateurs, téléphones troisième génération...) portent des caractéristiques qui diffèrent d'un dispositif à un autre ; l'objectif étant d'adapter les interfaces aux ressources limitées offertes par les différents équipements et aux contraintes qu'impose la mobilité sur les capacités motrices de l'utilisateur [Calvary *et al.*, 2005].

L'adaptation selon le comportement de l'utilisateur, appelée aussi *personnalisation*, peut s'effectuer de deux manières : adaptation aux préférences des utilisateurs, considérée comme étant une adaptation *statique*, l'interface est alors dite « configurable » ou « adaptable », et l'adaptation du système aux besoins et aux préférences de l'utilisateur, considérée comme étant une adaptation *dynamique*, l'interface est dite adaptative [Simonin et Carbonell, 2007]. Nous remarquons donc deux nouveaux types d'adaptation à savoir, l'adaptation statique et l'adaptation dynamique.

Une adaptation est dite « *statique* » lorsqu'elle intervient avant le déploiement de l'application [Fox et Clarke, 2009]. Ce type d'adaptation nécessite l'arrêt de l'application et son redéploiement. Elle requiert donc que les contraintes d'adaptation varient peu. Dans un environnement ubiquitaire, la fréquence des changements des situations est généralement assez élevée, il est donc impossible d'envisager l'arrêt et le redéploiement d'une application à chaque

adaptation.

Une adaptation *dynamique* permet la modification du comportement d'un logiciel au cours de son exécution. Pour ce faire, il faudra que l'application soit construite sur une plateforme qui assure ce genre d'adaptation. Lors d'une adaptation dynamique, l'application est plus ou moins bloquée. Dans le cas le plus extrême, l'application est totalement indisponible lors de la phase d'adaptation. Ceci est bien évidemment très contraignant pour le bon déroulement du système puisque l'application est totalement inutilisable alors que des parties de celle-ci ne sont pas affectées par les modifications. Il serait donc plus judicieux de bloquer uniquement les sous-parties ciblées par l'adaptation.

Yang *et al.* proposent de combiner l'adaptation dynamique avec l'adaptation statique en adaptant statiquement l'application en un premier temps, puis en ajoutant ou en supprimant des bouts de code dynamiquement au fur et à mesure des changements survenus dans l'environnement [Yang *et al.*, 2002]. Ce type d'approche est difficilement envisageable dans un environnement ubiquitaire en raison de son imprévisibilité. En effet, il est impossible de prédire les changements qui auront lieu dans un environnement ubiquitaire et de les prendre en compte lors de la phase de conception d'une application sensible au contexte [Ferry, 2011].

Deux principales approches d'adaptation dynamique émergent dans la littérature, à savoir, l'adaptation paramétrée et l'adaptation compositionnelle. Afin de changer le comportement de l'application, l'adaptation *paramétrée* modifie certains paramètres. Ces derniers peuvent être remplacés lors de l'exécution de l'application. Les adaptations qui peuvent subvenir à l'application doivent être prédéfinies à l'avance, c'est-à-dire lors de la phase de conception de l'application. Ces approches donnent de très bons résultats dans des environnements « bornés » où les adaptations sont connues à l'avance. Puisqu'il est impossible d'envisager toutes les adaptations (changements potentiels au niveau du contexte) dans un environnement ubiquitaire, l'utilisation de l'adaptation paramétrée reste ambiguë voir même impossible [Ferry, 2011].

L'adaptation *compositionnelle* a pour objectif de reconfigurer une application en ajoutant, retirant ou échangeant des modules qui la composent. Elle offre la possibilité d'intégrer de nouveaux algorithmes qui n'avaient pas été prévus lors de la phase de conception. Ce type d'adaptation devra prendre en considération l'apparition et la disparition de dispositifs mobiles [Tigli *et al.*, 2009][Geihs *et al.*, 2009]. Il est à noter que pour pouvoir réaliser une telle adaptation, il faut que le logiciel en question soit basé sur une architecture modulaire.

Après avoir défini et présenté les différents types d'adaptation, nous allons, à présent, exposer les différentes architectures d'un système sensible au contexte.

1.4 Architectures des systèmes sensibles au contexte

1.4.1 Structure générale d'une application adaptable au contexte

Dey est l'un des premiers chercheurs à avoir généralisé la notion de contexte en spécifiant trois étapes nécessaires à savoir : (i) la capture du contexte, (ii) son interprétation qui permet de passer à une représentation de haut niveau plus exploitable pour l'application et finalement (iii) fournir ces informations à l'application. Le Context Toolkit de Dey est l'une des premières architectures qui prend en considération ces différentes étapes [Dey *et al.*, 2001].

En règle générale, le cycle d'adaptation d'un système sensible au contexte se base sur les travaux de Dey. Il inclut les observations de l'environnement, la sélection des adaptations et leurs exécutions.

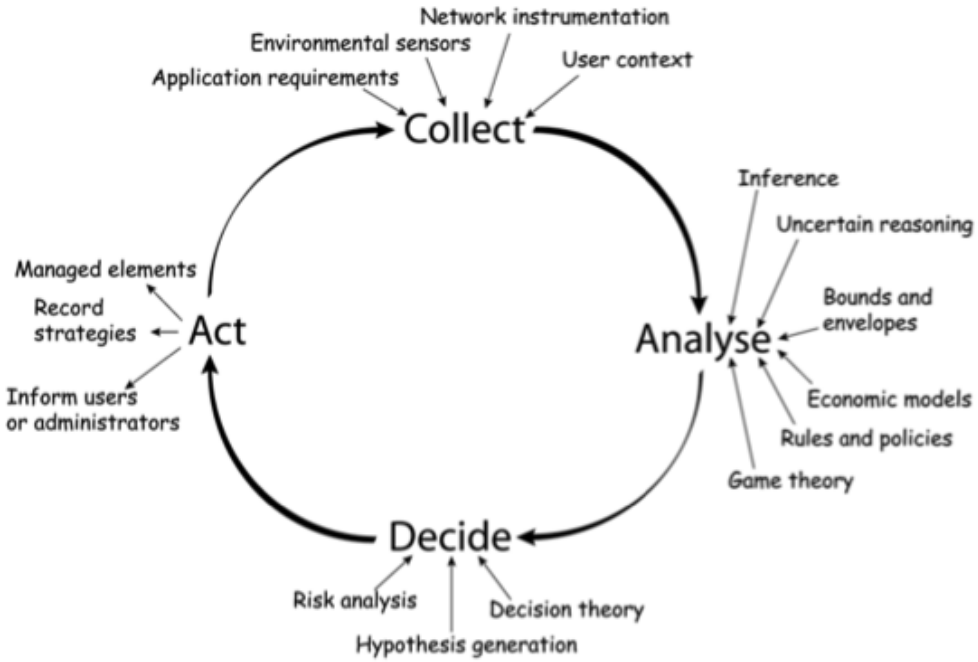


Figure 1.3 — Le cycle d'adaptation [Da *et al.*, 2011]

Par ailleurs, plusieurs auteurs considèrent le même cycle d'adaptation. Il est schématisé par la Figure 1.3 et contient quatre étapes à savoir la Collecte d'information, l'Analyse, la Décision et l'Action (CADA). Premièrement, le collecteur collecte les informations de l'en-

vironnement d'exécution et du contexte de l'utilisateur. Ces informations sont appelées les méta informations du contexte. Puis, le collecteur exploite ces informations pour produire un modèle abstrait de haut niveau afin de décrire les informations du contexte. L'analyseur évalue ce modèle et produit des plans d'adaptations. Le décideur analyse le risque d'erreur pour chaque plan et choisit le meilleur plan d'exécution. Finalement, l'acteur exécute le plan d'adaptation choisi [Da *et al.*, 2011] [Cheng *et al.*, 2009].

Inspiré de la boucle de MAPE-K [Dobson *et al.*, 2006], [Kakousis *et al.*, 2010] rejoignent l'esprit de [Da *et al.*, 2011] et définissent l'adaptation dans l'informatique ubiquitaire comme une boucle fermée (Figure 1.4) qui comprend les phases suivantes :

- Détection et traitement du contexte : Durant cette phase, toutes les données à savoir le contexte de l'utilisateur ou celui du système, sont collectées et traduites en un langage de haut niveau qui pourrait entraîner une adaptation du système ;
- Raisonnement et planification de l'adaptation : Dans cette phase, le système auto-adaptatif est appelé à raisonner sur le nouveau contexte et à décider sur ce qui doit être changé, dans le but d'atteindre l'objectif d'adaptation globale ;
- Action pour l'adaptation : Dans cette phase, les mécanismes d'adaptation appropriée sont utilisés pour mettre en œuvre les décisions d'adaptation prises par le processus de raisonnement.

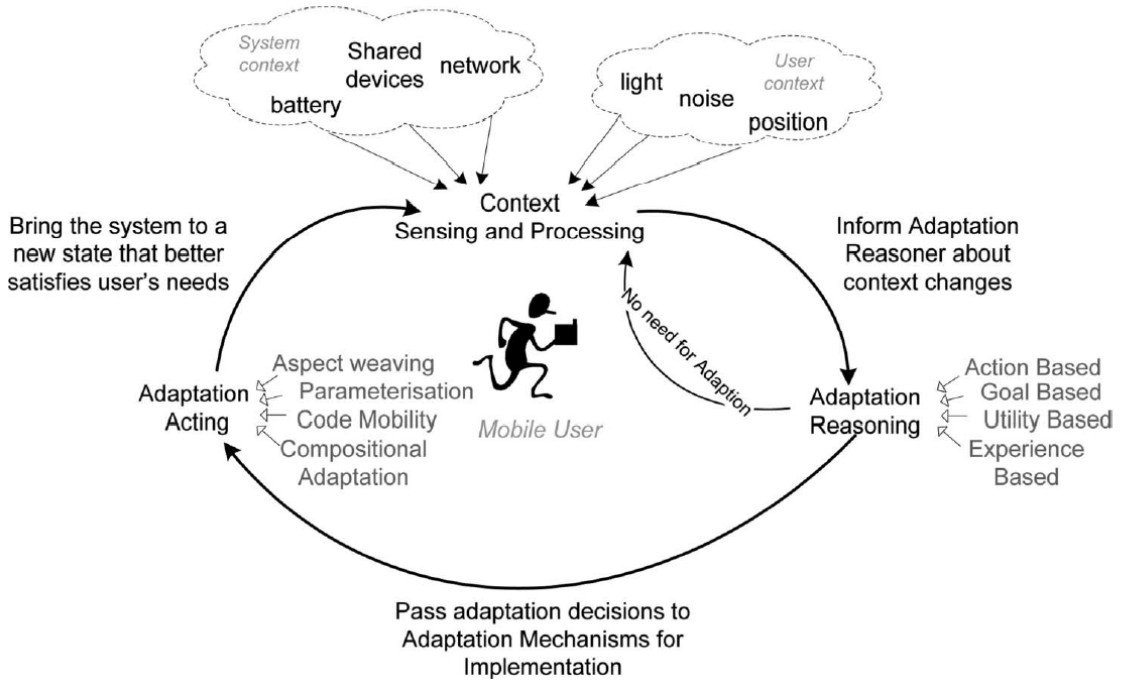


Figure 1.4 — Boucle d'adaptation selon [Kakousis *et al.*, 2010]

Nous remarquons que ces deux approches se ressemblent énormément puisqu'elles contiennent les mêmes phases. Nous pouvons donc déduire qu'un système d'adaptation dans un environnement ubiquitaire possède quatre rôles : gestionnaire du contexte, planificateur, décisionnaire et middleware. Le rôle de planificateur et du décisionnaire étant les plus importants.

- Le gestionnaire du contexte possède le rôle de collecteur : Le système doit recueillir des informations du contexte et construire les modèles du contexte de haut niveau. Les informations du contexte peuvent provenir de différents appareils ;
- Le planificateur : Le système doit produire dynamiquement des plans d'adaptation lors de l'exécution. Une heuristique de planification est utilisée pour déterminer la meilleure configuration d'une application. Le raisonnement est basé sur des informations du contexte ;
- Le décisionnaire : Il analyse le risque de chaque plan, prend une décision et demande à l'acteur l'exécution de ces changements ;
- Le Middleware : Il prend en charge les modèles de coordination entre les entités, et rend la distribution aussi transparente que possible [Da *et al.*, 2011]

A ce stade, nous pouvons affirmer que l'objectif de la majorité des travaux se rejoignent sur le fait qu'il est impératif de séparer l'acquisition et la modélisation du contexte de son utilisation dans les applications. Tous ces travaux convergent vers une architecture qui se compose de cinq couches où chacune d'elle est responsable d'une fonction bien déterminée : (1) la couche capture du contexte, (2) interprétation ou modélisation du contexte, (3) la couche stockage du contexte, (4) la couche diffusion du contexte et enfin (5) la couche application. Nous avons également remarqué que les architectures diffèrent principalement en nom et emplacement des couches mais nous retrouvons forcément les couches proposées dans la Figure 1.5.

1.4.1.1 La couche capture du contexte

La capture du contexte constitue la base de chaque architecture sensible au contexte. En effet, l'acquisition du contexte représente la première étape dans le processus de développement des applications sensibles au contexte.

Pour pouvoir intercepter les données du contexte, la couche capture présente une collection de capteurs. Un capteur est une source matérielle ou logicielle capable d'intercepter ou de générer des informations contextuelles. Il existe trois types de capteurs :

- Les capteurs physiques : dispositifs matériels capables d'intercepter les données du

Figure 1.5 — Structure générale d'un sytème sensible au contexte

contexte (capteurs infrarouge, GPS, microphones, capteurs biométriques, thermomètres...) ;

- Les capteurs virtuels : ils fournissent l'information à partir d'applications ou services logiciels (détecter l'activité d'un utilisateur sur un PC en analysant les évènements de la souris) ;
- Les capteurs logiques : ils utilisent plusieurs sources d'informations contextuelles pour déduire une autre information plus précise.

Chaque type de capteur doit être rattaché à un composant logiciel permettant l'accès aux données contextuelles interceptées.

1.4.1.2 La couche interprétation du contexte

Cette couche permet l'analyse et la transformation des données brutes fournies par la couche capture du contexte afin de convertir les données interceptées en des données de plus haut niveau plus faciles à comprendre et à manipuler. En effet, généralement, les données contextuelles fournies par les capteurs sont des données techniques spécifiques aux types de

capteurs utilisés. Elles sont inappropriées pour une utilisation directe par les applications et doivent être transformées (par exemple : transformer les coordonnées GPS en une adresse compréhensible avec le nom de la rue, de la ville,...).

La complexité des interprétations faites sur le contexte peut varier d'une simple agrégation de valeurs provenant de différents capteurs à des raisonnements ou des analyses statistiques complexes. Par exemple, la localisation de plusieurs personnes à un moment donné et dans un même endroit peut induire le fait qu'ils soient en réunion. Il faudra alors vérifier s'ils sont en réunion de travail ou de loisir, par exemple en interceptant le niveau de bruit dans la salle [Chaari, 2007].

L'utilisation de plusieurs sources provenant du contexte peut amener à des situations de conflit et des résultats contradictoires qui aboutissent à des situations imprécises, voire même totalement incorrectes. De ce fait, cette couche doit avoir une certaine forme d'intelligence afin d'intercepter et de résoudre ces conflits.

1.4.1.3 La couche stockage du contexte

Cette couche modélise et archive les données fournies par la couche interprétation pour une utilisation ultérieure des données captées et interprétées. Pour ce faire, un modèle approprié doit être défini afin de décrire ces informations. Ce modèle du contexte sera ensuite utilisé dans la couche adaptation. Ainsi cette phase présente une étape primordiale dans le développement d'applications sensibles au contexte.

Il existe dans la littérature plusieurs approches de modélisation du contexte à savoir : le modèle clé/valeur, le modèle de marquage, le modèle orienté objet, le modèle basé sur la logique, le modèle spatiale et temporel, le modèle probabiliste, les modèles graphiques, les ontologies et les modèles hybrides. Le chapitre suivant décrira ces différentes approches de modélisation du contexte.

1.4.1.4 La couche diffusion du contexte

Cette couche a pour objectif de transmettre et d'adapter la couche application selon les informations transmises par la couche stockage. Vu la diversité des données interceptées par les différents capteurs, la majorité des chercheurs ont trouvé primordial de dédier une couche pour la transmission des données à la couche application. Cette couche possède des techniques de communication pour avertir la couche application d'éventuels changements survenus au contexte. En d'autres termes, cette couche possède deux principales fonctionnalités : *(i)* fournir à la couche application les services et fonctions nécessaires aux développements d'application

sensible au contexte, et *(ii)* notifier la couche application des changements du contexte au moyen d'une technique de communication.

1.4.1.5 La couche application

La couche application contient le moteur d'adaptation responsable de la définition des différents mécanismes d'adaptation ainsi que les différentes réactions que le système doit effectuer suite à un changement du contexte.

Les applications doivent s'abonner à la couche diffusion du contexte afin d'être informées de tout changement du contexte et pour accéder aux différentes données contextuelles. L'accès à ces données peut s'effectuer de deux manières : *(i)* synchrone : c'est l'application qui demande à la couche diffusion de lui fournir l'information dont elle a besoin et *(ii)* asynchrone : l'application s'abonne à des évènements spécifiques qui aboutissent à un changement du contexte.

Après avoir défini la structure générale d'un système sensible au contexte, nous allons décrire dans la section qui suit, quelques architectures présentes dans la littérature.

1.4.2 Quelques architectures sensibles au contexte

1.4.2.1 Simple Environment for Context-aware Systems : SECAS

L'architecture SECAS représentée par la Figure 1.6, est basée sur trois composants : la couche de gestion du contexte, la couche adaptation et la couche application [Chaari *et al.*, 2007].

La couche gestion du contexte se compose de quatre modules : *(i)* le « Context Provider » c'est-à-dire le fournisseur de contexte qui a pour objectif de capturer les paramètres du système. Ensuite, il est tout à fait logique de transformer les données captées en des données de plus haut niveau beaucoup plus significatives (adresse postale à la place de coordonnés GPS). C'est le rôle du *(ii)* « context interpreter », *(iii)* le module « context repository » utilise des représentations XML pour stocker et échanger les valeurs des paramètres contextuels. Pour qu'elle soit sensible au contexte, l'application doit consommer une partie du contexte. Elle doit s'inscrire au *(iv)* « context broker » (courtier) qui transmet les paramètres contextuels pertinents pour chaque service de l'application.

La couche d'adaptation au contexte : Cette adaptation peut concerner trois niveaux différents de l'application : le flux de données entre les services et l'utilisateur (adaptation de

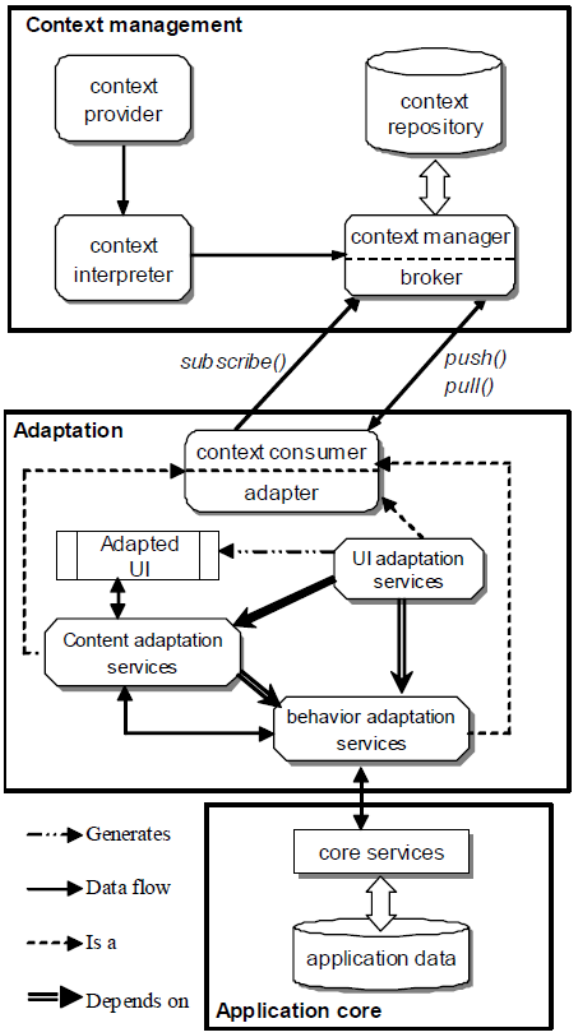


Figure 1.6 — L'architecture SECAS [Chaari *et al.*, 2007].

contenu), l'interface utilisateur (adaptation de présentation) et le fonctionnement des services offerts à l'utilisateur (adaptation de comportement).

1.4.2.2 Context Toolkit

Le context Toolkit, proposé par [Dey *et al.*, 2001], propose une boîte à outils pour le développement d'applications sensibles au contexte. Il offre un modèle d'exécution et des composants de base réutilisables pour les applications sensibles au contexte. Son objectif est de séparer l'acquisition du contexte de la manière dont il est présenté et utilisé par les autres composants de l'architecture.

Comme le montre la Figure 1.7, cette architecture est basée sur :

- « Les widgets » : Ils représentent des composants logiciels qui encapsulent un capteur physique. Ils communiquent les informations perçues aux interpréteurs. Ils ont aussi pour rôle d'enregistrer les données acquises et de les rendre disponibles pour les applications ;
- Les interpréteurs : Ils ont pour objectif de donner une sémantique aux signaux qui leur sont transmis par les « Widgets ». Ils les transforment en des données ayant un niveau d'abstraction plus élevé pour être mieux comprises et traitées ;
- Les serveurs : Ils représentent un intermédiaire entre les applications et les « Widgets ». Ils collectent les différents signaux émis par les autres composants et font le lien entre les applications et les « Widgets ». Les serveurs ont également la charge de synthétiser les informations contextuelles dans un niveau d'abstraction supérieur.

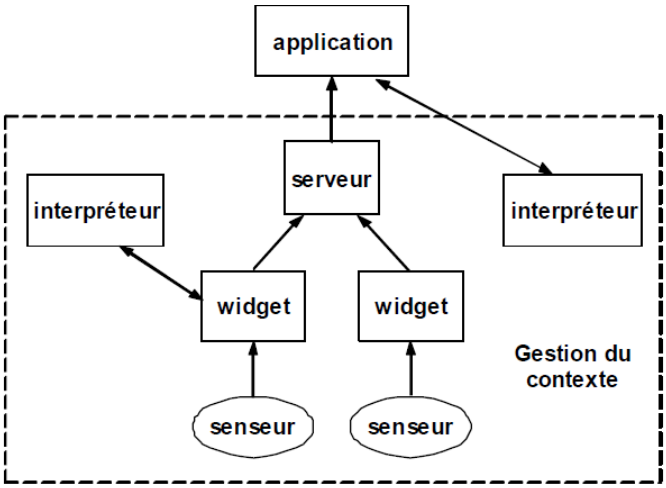


Figure 1.7 — Context Toolkit [Dey *et al.*, 2001].

1.4.2.3 Context Broker Architecture : CoBrA

CoBrA [Chen, 2004], représentée par la Figure 1.8, est une architecture orientée agents pour les systèmes sensibles au contexte dans des environnements intelligents. Elle se base sur un agent intelligent appelé le « Context Broker » qui maintient et gère un modèle du contexte. Il assure aussi la protection et la confidentialité des données échangées avec les utilisateurs.

Le « context broker » est composé de quatre modules :

- Context Knowledge base : Le « broker » stocke et contrôle les données contextuelles afin d'assurer la cohérence de la base de connaissance à tout instant ;

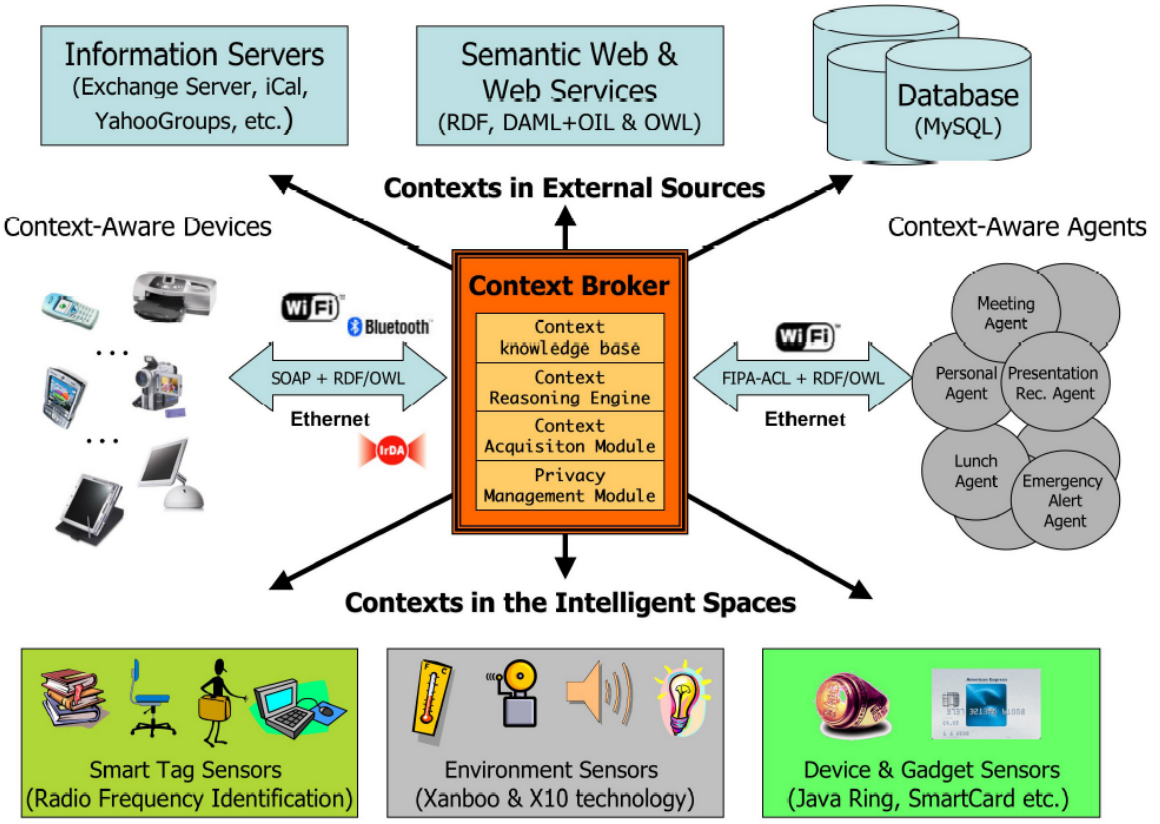


Figure 1.8 — Architecture générale du système CoBrA [Chen, 2004].

- Context Reasoning Engine : Il permet le raisonnement et le partage des connaissances contextuelles ;
- Le module d'acquisition du contexte « Context Acquisition Module » : Il capture les données provenant des différents capteurs ;
- Privacy Management Module : Le Broker partage les données contextuelles selon des politiques de confidentialités assurées par ce module afin de protéger la confidentialité des utilisateurs.

1.4.2.4 Context Management Framework : CMF

Le CMF [Korpipaa *et al.*, 2003] (Figure 1.9) est une plateforme analogue au Context Toolkit de Dey. Elle est composée de quatre entités principales : *(i)* Resource Server (capture du contexte), *(ii)* Context recognition Service (interprétation du conexte), *(iii)* Context Manager (dissémination du contexte) et *(iv)* application. Comme la plateforme CoBrA, le context management framework se base sur un gestionnaire de contexte centralisé qui communique avec

tous les autres modules de la plateforme. En effet, dans CMF, le gestionnaire de contexte récupère les informations contextuelles à l'aide du module Resource Server. Ensuite, il les interprète en utilisant le module Context Recognition Service. Enfin, il les diffuse à l'application. CMF utilise la logique floue pour fournir les informations contextuelles complexes.

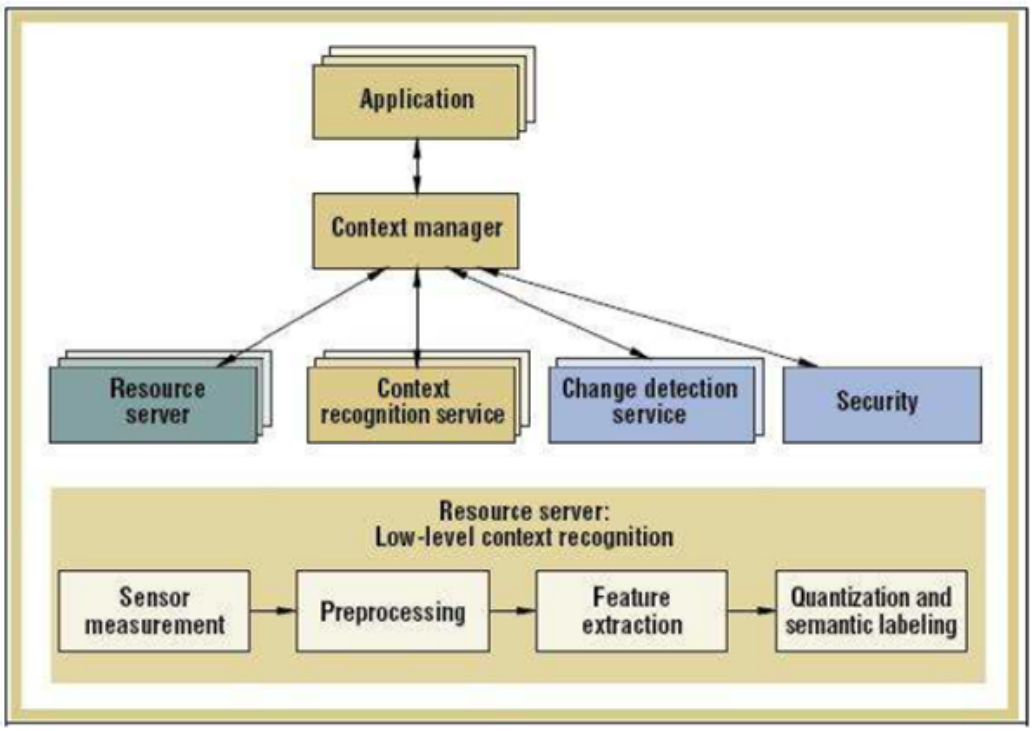


Figure 1.9 — CMF : architecture générale [Korpipaa *et al.*, 2003].

Les autres plateformes assument que le contexte (même de haut niveau) est bien spécifié et représenté alors que CMF considère que l'obtention du contexte de haut niveau n'est ni simple ni directe. Les auteurs de CMF proposent donc d'utiliser la logique floue pour obtenir des informations de haut niveau et ainsi enrichir le contexte et la description de l'environnement de l'utilisateur.

1.4.2.5 Service Oriented Context-Aware Middleware : SOCAM

SOCAM [Gu *et al.*, 2005] est une architecture qui vise à assurer le développement et le prototypage de services sensibles au contexte dans des environnements intelligents. Elle est basée sur un middleware qui convertit les données contextuelles captées en des données sémantiques qui peuvent être partagées et utilisées par les services sensibles au contexte. Ce

middleware se base sur des ontologies pour modéliser le contexte. En première étape, SOCAM utilise une première ontologie « pervasive » de haut niveau qui décrit les informations caractérisant l'environnement d'exécution de l'application d'une façon générale. Ensuite, une ontologie spécifique au domaine de l'application doit être définie (par exemple ontologie décrivant un utilisateur dans une maison intelligente). L'architecture de SOCAM, présentée dans la Figure 1.10, comprend les composants suivants (qui évoluent en tant que services indépendants les uns des autres) :

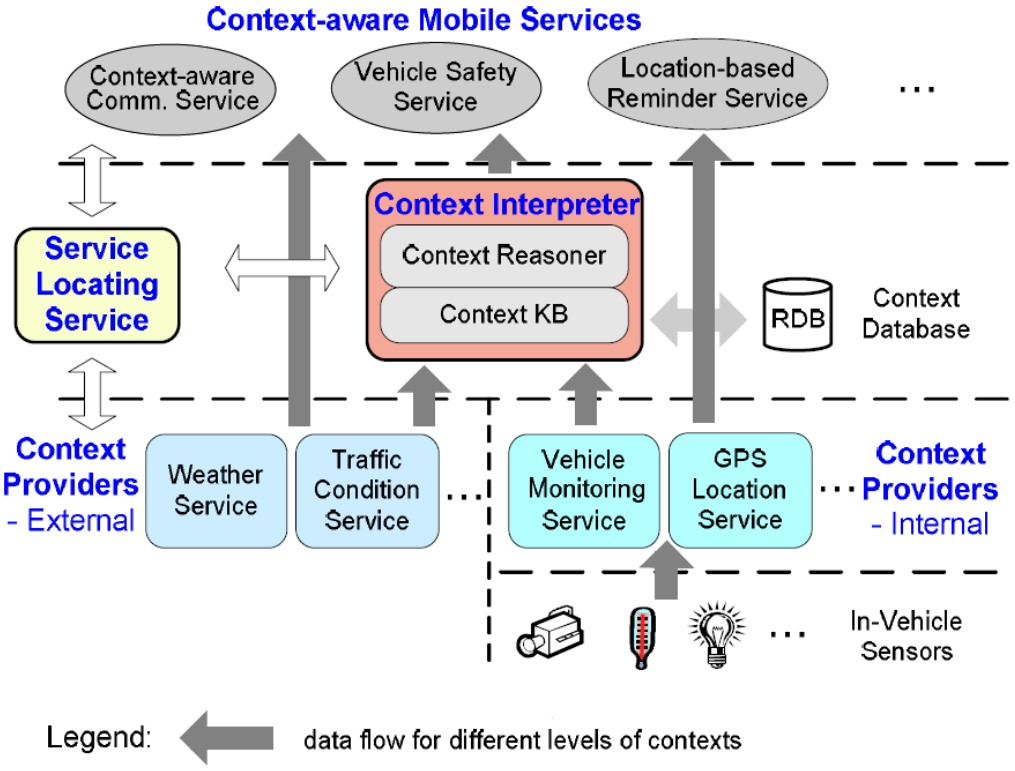


Figure 1.10 — Architecture de la plateforme SOCAM [Gu *et al.*, 2005].

- Context Providers (fournisseurs de contexte) : 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 (interprète de contexte) : 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 ;

– Context Database (base de données de contexte) : Elle stocke les différents éléments de 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 ;
– Context-aware Services (services sensibles au contexte) : Ces services utilisent les différentes informations stockées dans la base de données de contexte (Context Database) pour modifier leur comportement 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 de contexte.

1.4.2.6 Architecture based Approach to Context-aware Adaptive Software Systems : CAASS

L'architecture CAASS, représentée par la Figure 1.11, est composée de trois couches [Hussein *et al.*, 2011] :

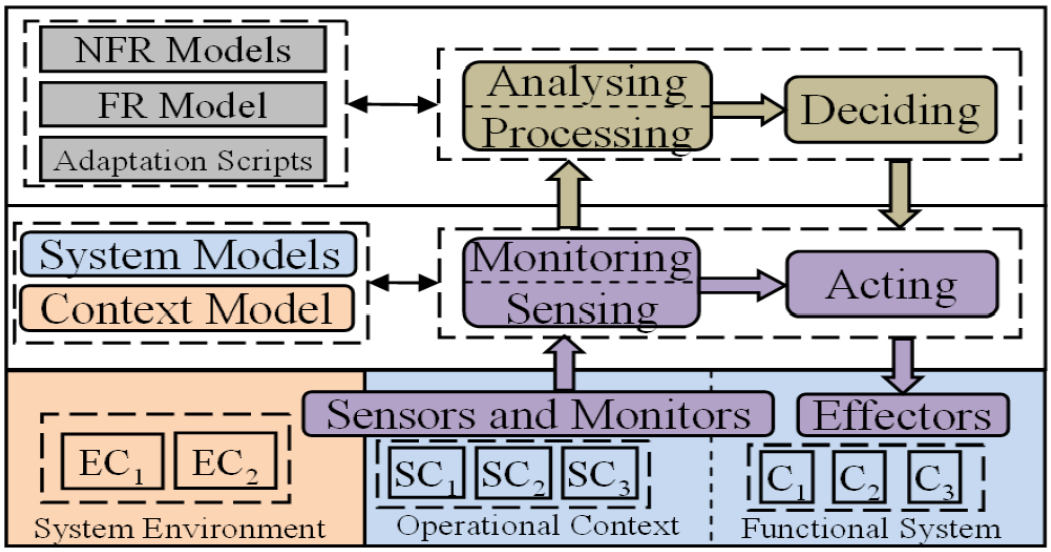


Figure 1.11 — Architecture CAASS [Hussein *et al.*, 2011].

– La couche système fonctionnel et son contexte : Cette couche comporte trois éléments. Tout d'abord, « le système fonctionnel » qui est composé d'un ensemble de composants qui sont utilisés pour réaliser la fonctionnalité de base du système. Au cours de l'exécution, ces composants peuvent être en cours d'exécution (C1 et C2) ou inactifs (contexte

opérationnel) qui peuvent être utilisés par le système fonctionnel lorsque cela est nécessaire (à savoir SC1, SC2, SC3). Ensuite, le contexte contient un ensemble d'entités qui appartiennent à l'environnement et qui affectent le fonctionnement du système et nécessitent des adaptations (EC1 et EC2). Finalement, on trouve les interfaces avec la couche de gestion. Ces interfaces sont : *(i)* un ensemble de capteurs/moniteurs pour détecter les changements dans le système ou dans le contexte et informer les couches supérieures de prendre les mesures d'adaptation requises, et *(ii)* un ensemble d'effecteurs qui appliquent les mesures d'adaptation qui ont été prises par les couches de gestion sur le système fonctionnel ;

– La représentation du système et de son contexte : Cette couche fonctionne selon deux opérations. Tout d'abord, l'opération de « surveillance/détection » responsable de la mise à jour des modèles de contexte et des modèles du système. Ensuite, l'opération « action » qui a pour objectif l'exécution des actions spécifiées par la couche précédente ;
– La gestion du changement : Cette couche gère les exigences fonctionnelles du système (FR modèle) et les exigences non fonctionnelles telles que la performance et la sécurité (NFR modèles). En outre, elle dispose de trois opérations : *(i)* l'analyse : elle consiste à vérifier la cohérence entre l'état du système en cours d'exécution, l'état de l'environnement et les exigences du système afin de lancer l'opération de « décision » en cas d'absence de cohérence. *(ii)* Le traitement qui est utilisé pour déduire les informations de haut niveau. Enfin, l'opération *(iii)* décision afin de prendre les mesures d'adaptation nécessaires en réponse aux changements de contexte et/ou des conditions et de la configuration du système actuel. « Les scripts d'adaptation » proposent un ensemble de mesures d'adaptation pré-définies en réponse à des changements connus à l'avance. Toutefois, si aucune de ces adaptations n'est adéquate à ce changement, l'opération de décision tente de générer un nouveau script à la volée. Si la génération de ce script n'est pas possible, l'administrateur système est informé pour effectuer les modifications requises (définir manuellement les modifications nécessaires).

1.4.2.7 Hycon

Cette architecture est divisée en quatre couches comme le montre la Figure 1.12, à savoir : la couche de stockage, la couche serveur, la couche terminale et la couche des capteurs. Elle fournit une plateforme favorable pour les simulations des systèmes hypermédias dans les environnements ubiquitaire mobiles. La couche stockage est responsable de l'enregistrement des données contextuelles. Elle comprend une interface qui facilite l'échange d'information avec la couche supérieur à savoir la couche serveur. L'objectif de la couche serveur est de fournir l'in-

formation adéquate et les fonctionnalités nécessaires à la couche terminale. La division entre les composants réutilisables et ceux spécifiques aux applications se fait au niveau de la couche terminale. Les composants de cette couche comprennent des interfaces permettant la communication avec la couche serveur. La plateforme matérielle de la couche terminale peut inclure toutes sortes d'appareils mobiles : ordinateurs portables, tablettes, PDA et téléphones cellulaires. Ces dispositifs peuvent avoir accès aux informations de la couche capteur. Les capteurs peuvent être intégrés dans le terminal (un appareil photo dans un téléphone) ou être externes (GPS). Le fonctionnement de la couche capteur est très similaire au mécanisme mis en œuvre dans le contexte Toolkit (l'utilisation des widgets et des interpréteurs) [Bouvin *et al.*, 2003].

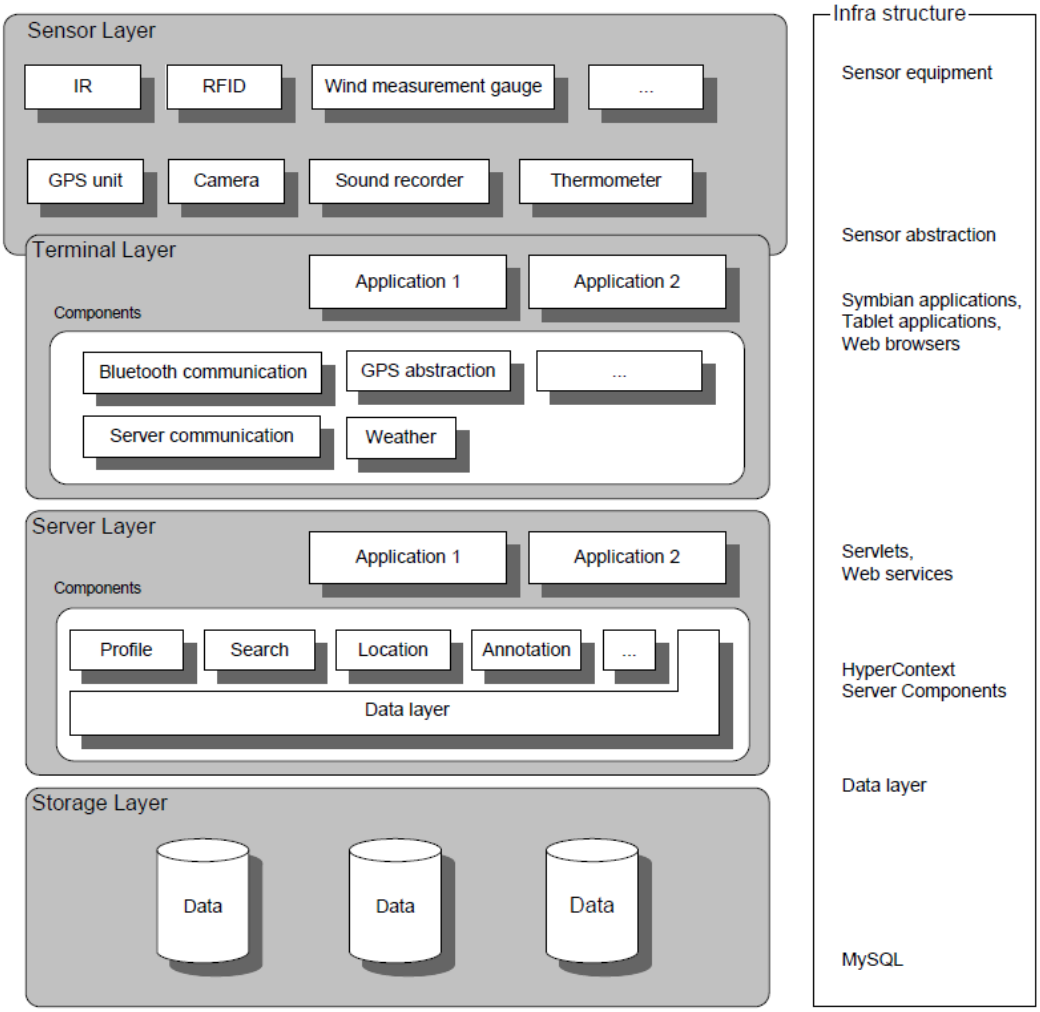


Figure 1.12 — Architecture HyCon [Bouvin *et al.*, 2003].

1.4.2.8 Java Context-Aware Framework : JCAF

JCAF est une architecture orientée services, basée sur la technologie JAVA et suit le modèle MVC (Modèle-Vue-Contrôleur). Cette plateforme, représentée par la Figure 1.13.a, se compose comme suit :

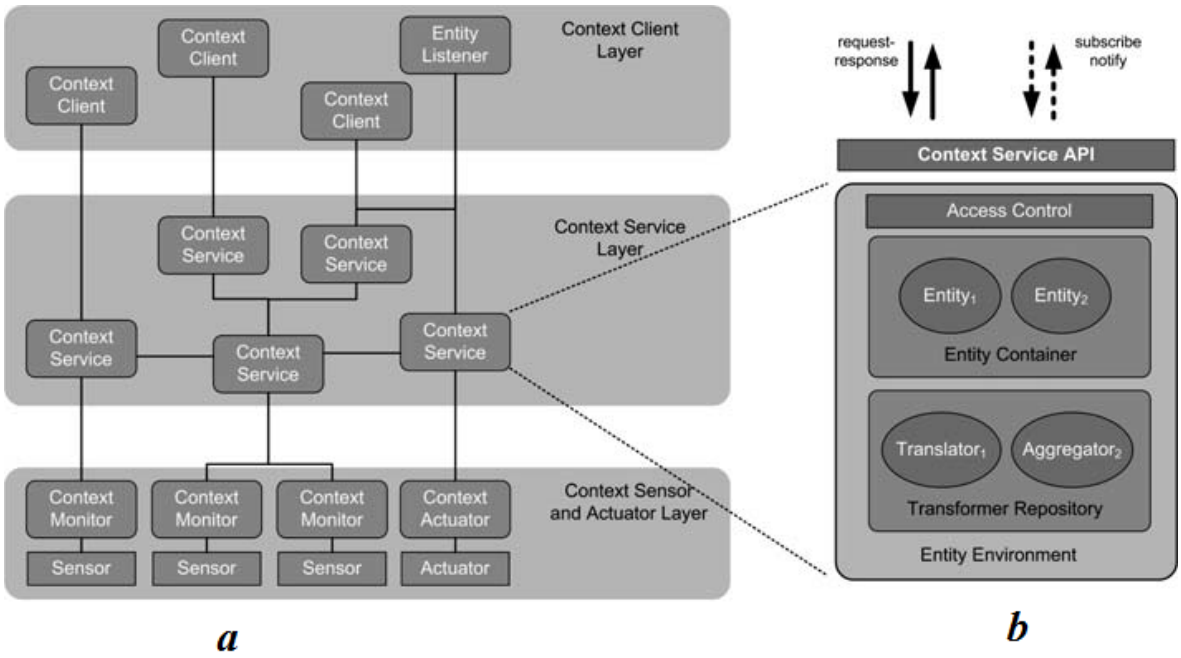


Figure 1.13 — Architecture JCAF [Bardram, 2005].

- La couche « Context Client » : Elle représente les applications sensibles au contexte. Les clients peuvent accéder à des entités du contexte. Ils peuvent ajouter ou supprimer des informations du contexte. Les clients peuvent accéder à des entités et à leurs informations de deux manières : soit à la suite d'un schéma de « requête-réponse », ou en s'inscrivant en tant qu'entité auditrice à l'écoute des changements survenus sur des entités spécifiques ;
- La couche « context Service » : La Figure 1.13.b illustre les détails d'un service de contexte. Cette couche est composée d'un ensemble d'entités. Une entité possède des informations sur le contexte et elle est gérée par un conteneur d'entités du service. Une entité est un programme Java qui s'exécute dans le contexte de service et répond aux changements du contexte. Le cycle de vie d'une entité est commandé par le conteneur dans lequel elle a été ajoutée. Le conteneur d'entités informe les clients sur les modifications pertinentes qui ont eu lieu. L'entité environnement permet la collaboration

des entités avec d'autres composants pour accomplir leurs tâches. Cette entité permet l'accès aux ressources générales et aux ressources spécifiques de l'utilisateur. En outre, l'entité environnement détient des transformateurs de contexte, qui sont des petites applications Java que les développeurs peuvent écrire et ajouter au « référentiel transformateur ». L'accès à un service de contexte est contrôlé par le composant de contrôle d'accès, ce qui assure une authentification correcte des demandes des clients. Ce module se compose essentiellement de deux parties, à savoir une liste de contrôle d'accès et des mécanismes pour l'authentification du client ;

– La couche « context Monitor/Actuator » : Un moniteur est un client spécialement conçu pour l'acquisition des informations du contexte dans l'environnement en coopération avec un certain type de capteur. Un actionneur est un client conçu pour fonctionner conjointement avec un ou plusieurs actionneurs pour détecter les changements du contexte [Bardram, 2005].

Les différentes entités qui composent cette architecture permettent de capter tout changement survenu au contexte vu que JCAF est basé sur les évènements. L'entité environnement, qui appartient à la couche service, permet l'accès aux ressources générales et aux ressources spécifiques utilisées par l'utilisateur.

1.4.2.9 User-Adaptive and Context-Aware Architecture for Mobile and Desktop Training Applications : CAAMDTA

CAAMDTA représentée par la Figure 1.14, est composée comme suit [Buttussi, 2008] :

L'analyseur du contexte : Ce module capte les données brutes des différents capteurs disponibles, les analyse, pour en tirer des informations de niveau supérieur sur l'état physiologique de l'utilisateur ainsi que les différents mouvements. Ensuite, l'analyseur du contexte fournit toutes ces informations à « l'UM/CM service ».

UM/CM service : Il reçoit les informations concernant l'utilisateur à partir du moteur d'adaptation, de l'interface utilisateur et de l'analyseur du contexte. Ce module stocke ces informations dans la base de données « Context Model » et « User Model » et alimente le moteur d'adaptation ainsi que l'analyseur du contexte des informations dont ils ont besoin. En outre, ce service est conçu pour le partage des informations de l'utilisateur et du contexte entre les différentes instances de l'architecture.

La BD du modèle de l'utilisateur : Elle conserve les renseignements personnels explicitement fournis par l'utilisateur (âge, sexe, poids, taille) ainsi que les informations implicites acquises lors du fonctionnement du système (le volume d'oxygène que l'utilisateur consomme

en une minute, la durée passée pour accomplir un but particulier ...). Ces informations sont à la disposition des différents modules par le biais du « UM/CM service ».

La BD du modèle de contexte : Elle stocke les informations acquises ou provenant des différents capteurs. Ces informations sont aussi disponibles via le « UM/CM service ».

Le moteur d'adaptation : Il prend les informations fournies par le « UM/CM service » et applique des règles d'inférence pour décider ou conseiller l'interface utilisateur ou l'humain virtuel des éventuelles adaptations à faire.

L'interface utilisateur : Elle applique les demandes d'adaptation du moteur en offrant à l'utilisateur les éléments d'interaction appropriés (textuels, graphiques, sonores...) et envoie les entrées de l'utilisateur au « UM/CM service ».

L'humain virtuel : Il est chargé de fournir à l'utilisateur des conseils et des démonstrations à l'aide d'animations virtuelles 3D. En suivant les instructions du moteur d'adaptation, les animations sont adaptées à la fois à l'utilisateur et au contexte.

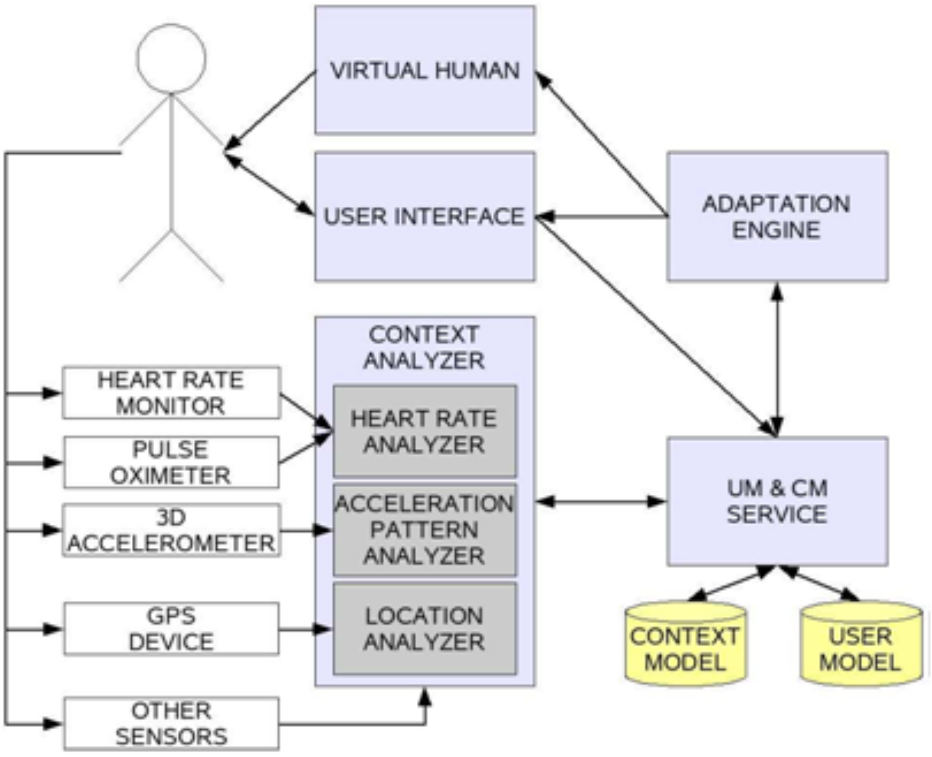


Figure 1.14 — Architecture CAAMDTA [Buttussi, 2008].

1.4.2.10 Generic and Extensible Context Aware Framework : GECAF

GECAF est une architecture basée sur le concept de tubes et filtre (Figure 1.15). Dans ce type d'architecture, les processus sont modélisés comme étant des filtres et l'information circule via des tubes (Pipes). Cette architecture est basée sur un modèle de contexte nécessaire pour la spécification des primitives du contexte. Les informations contextuelles sont décrites moyennant le Context-aware Description Language (CDL) écrites en XML. Cette architecture contient également un module pour la gestion de l'historique du contexte et un autre pour la capture des données contextuelles. Ce module représente les moyens d'acquisition de l'information à partir d'un environnement intelligent. Il déclenche également le fonctionnement du système développé [Sabagh et Al-Yasiri, 2015].

Le fonctionnement du système est contrôlé par le « System Manager and Scheduler », qui est construit pour planifier les événements déclenchés par les capteurs logiciels et matériels. Il gère également le déploiement du système sensible au contexte.

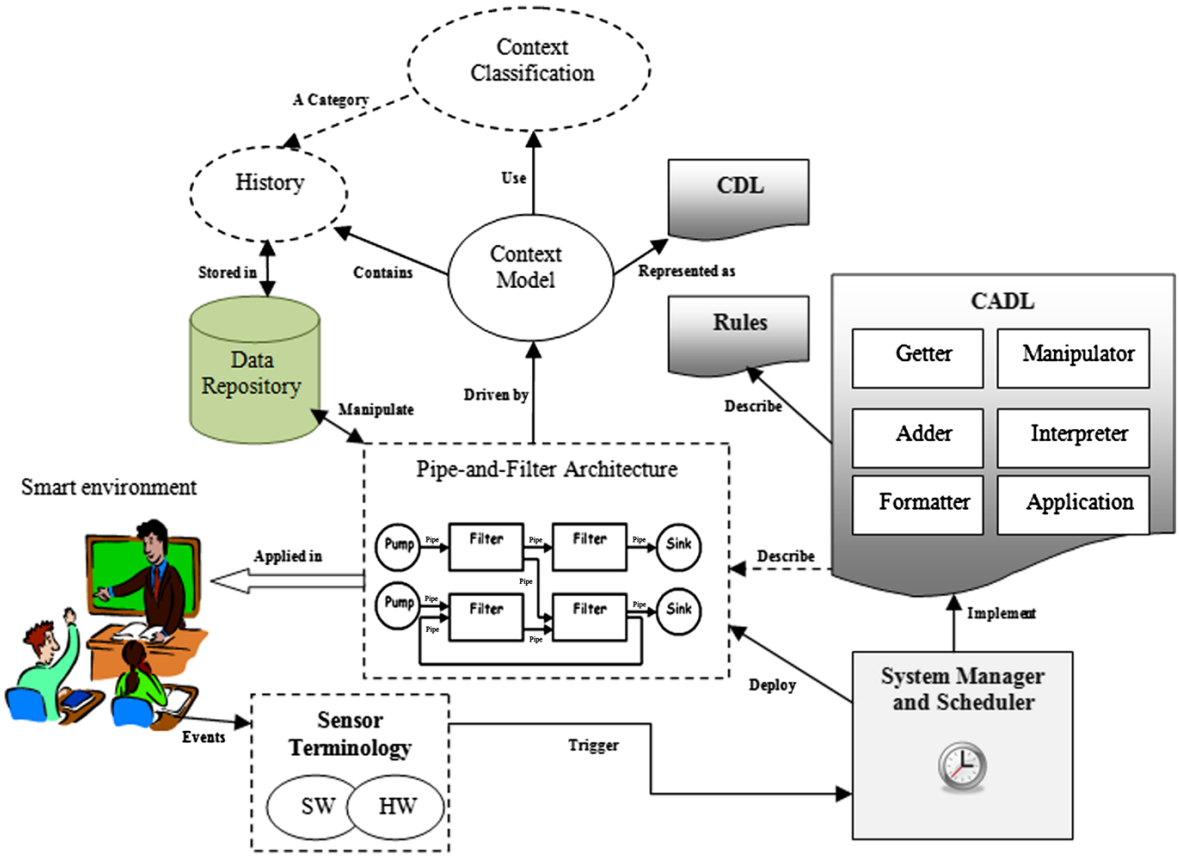


Figure 1.15 — Architecture GECAF [Sabagh et Al-Yasiri, 2015].

GECAF se base sur plusieurs filtres possédant chacun un rôle particulier :
- « Getter » : Est responsable de l'acquisition des informations contextuelles ;
- « Formatter » : Est responsable de la modélisation du contexte ;
- « Adder » : Enrichie les informations contextuelles par des relations entre les différentes informations ;
- « Interpreter » : Interprète les données contextuelles pour produire un modèle abstrait du contexte ;
- « Manipulator » : Stocke toutes les données contextuelles ;
- « Outputer » : Représente le module d'adaptation.

1.4.2.11 Adaptive Human-Computer Interface of a PT Operation Platform : AHCI

AHCI (interface homme-machine adaptative) représentée par la Figure 1.16, est une interface capable d'adapter son contenu aux données contextuelles afin de répondre automatiquement aux exigences de l'utilisateur tout en tenant compte de ses préférences, de ses capacités intellectuelles et de sa tâche. Cette architecture est composée principalement de quatre couches [Xue *et al.*, 2014] :

- Couche capteur : Elle a pour objectif de percevoir les informations atomiques du contexte à partir de sources externes et internes ;
- Couche contrôle d'adaptation : Cette couche a pour objectif le traitement de l'information du contexte et le raisonnement sur celle-ci afin de générer les règles d'adaptation ;
- Couche de configuration de l'interface : Selon les décisions d'adaptation, cette couche extrait les éléments de base pour la génération de l'interface ;
- Couche génération de l'interface : Elle a pour objectif la visualisation de l'interface finale adaptée au contexte de l'utilisateur.

1.5 Discussion

Après avoir présenté quelques architectures de systèmes sensibles au contexte, il devient nécessaire à ce stade, de pouvoir les comparer. Tout d'abord, nous remarquons que ces plateformes utilisent généralement l'architecture en cinq couches, définie dans la section 1.4.1. Ces couches diffèrent en noms mais leurs fonctions restent la même. Par exemple dans l'architecture SECAS, le « context Provider » assure la capture des données du contexte, « le Context Interpreter » assure son interprétation, le stockage des données est réalisé par le module

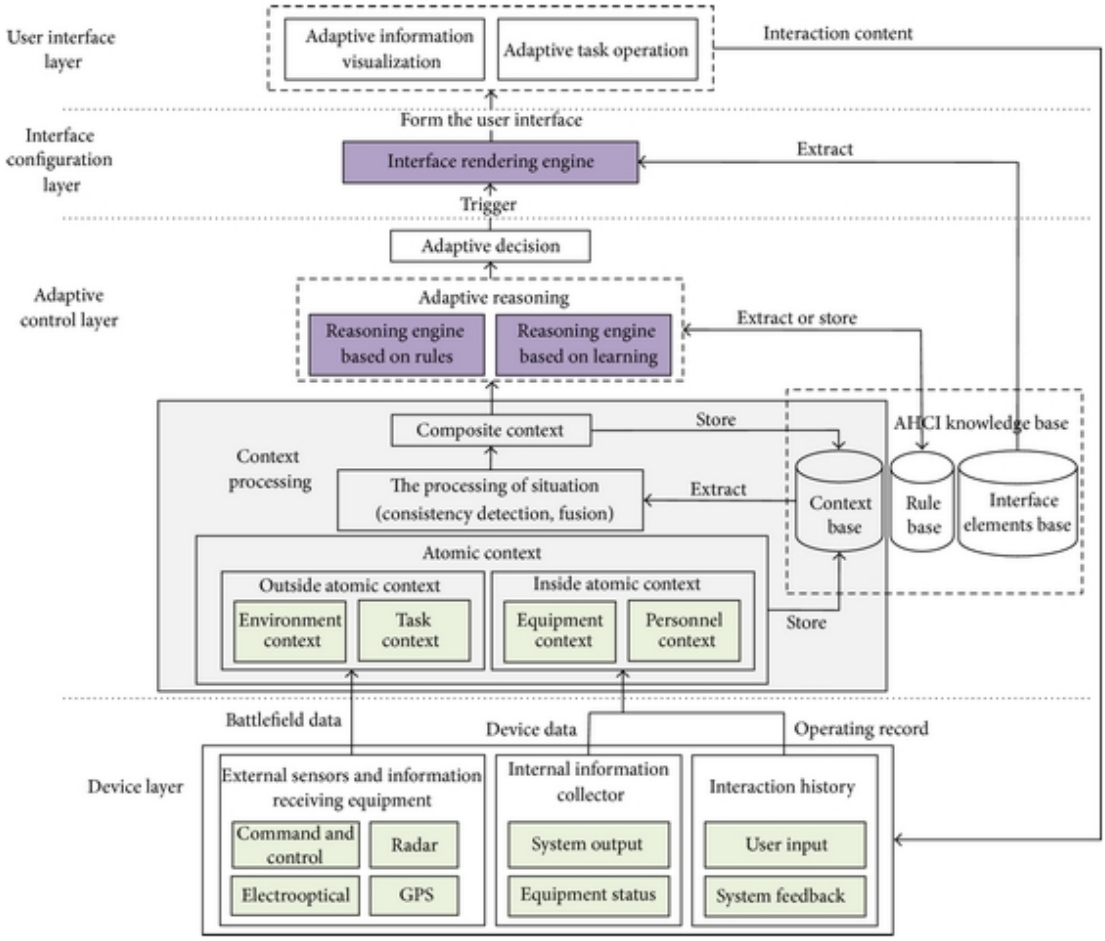


Figure 1.16 — Architecture AHCI [Xue *et al.*, 2014].

« Context Repository » et « le context Broker » assure la diffusion des données contextuelles. En outre, nous pouvons assurer qu'il y a comme un accord commun sur la séparation de l'acquisition du contexte de son utilisation dans les applications. Le tableau 1.1 synthétise les aspects principaux des architectures étudiées.

A ce stade, il devient primordial de définir des critères de comparaison pour pouvoir comparer les différentes architectures et en déduire les atouts et les inconvénients des différentes plateformes.

Tout d'abord, *le modèle de représentation du contexte* est un critère très important. En effet, l'utilisation d'un modèle inapproprié dans la représentation du contexte peut aboutir à une interprétation fausse des données contextuelles qui peut compromettre tout le fonction-

Tableau 1.1 — Récapitulatif des différentes architectures

	Couche capture	Couche interprétation	Couche stockage	Couche diffusion	Couche application	Remarques
SECAS	Gestion du Conexte				Couche application	Ajout de la couche adaptation entre la couche gestion du contexte et application
	Fournisseur du contexte	Interprèteur du contexte	Contexte Référentiel (CC/PP)	Agent de rupture du Contexte		
Context Toolkit	Widget	Interpréteur	Serveur (clé/valeur)		Application	
CoBra	Module d'acquisition	Moteur de raisonnement du contexte (moteur d inférence + base de connaissances	Base de connaissances (OWL)	Agent de rupture du Contexte	Application distribuée	Architecture basée sur les agents. Le Module Privacy Management : gestion des confidentialités selon le type d utilisateur
CMF	Serveur de Ressources	Service de reconnaissance du contexte		Gestionnaire du contexte	Application	
SOCAM	Fournisseur du Contexte	Interpréteur	BD du Contexte (ontologies)	Service sensible au contexte	Service de localisation	
CAASS	Capteurs	Processus d'analyse	Modèle du contexte		Script d'adaptation	Ajout de la couche gestion changement contenant des scripts d adaptation. Prise en compte relative de la gestion de l imprévu
HyCon	Couche capteur	Couche serveur	Couche stockage	Couche terminale		
JCAF	Context Moniteur / actionneur	Couche service			Couche client	L'accès à un service du contexte est contrôlé par Access Control ce qui assure l'authentification des clients
		Traducteur		Agrégateurs		
CAAMDTA	Capteur	Analyse du contexte	Modèle du contexte (ontologie) & et de l'utilisateur (UserMl : XML)	service de UM et CM	Moteur d adaptation, interface utilisateur	Ajout de « l'homme virtuel » pour la prise en charge de la 3D
GECAF	Terminologie du capteur (accesseurs)	Interpréteur	Manipulateur (CDL : XML)	Additionneur	Outputer	
AHCI	Couche capteur	Couche contrôle d'adaptation	BD contexte (XML)	Couche configuration de interface	Couche génération de l'interface	

nement de l'architecture et fournir des adaptations inadéquates à l'utilisateur. Les différentes architectures présentées dans la section 1.4.2, se distinguent par leur modèle de représentation du contexte. L'utilisation d'un modèle trop simpliste comme le modèle attribut/valeur peut être source de conflits d'interprétation et de description du contexte actuel de l'application et de l'utilisateur. Les modèles de marquage comme CC/PP donnent généralement de mauvais résultats dans la représentation de données complexes en raison des contraintes imposées par la sérialisation XML et la représentation RDF. A travers une étude détaillée des différentes approches de modélisation du contexte, nous avons convenu que les approches basées sur les réseaux de Petri sont les mieux appropriées pour la représentation des données contextuelles. Le chapitre suivant présentera cette étude.

La gestion de l'historique du contexte est aussi un critère important à réaliser lors de la conception d'architectures sensibles au contexte. En effet, l'historique permet d'implémenter et de stocker des algorithmes d'apprentissage pour fournir des services hautement adaptables au contexte. De plus, avec ce genre d'algorithmes, des actions prédéfinies peuvent être automatiquement déclenchées pour un certain nombre de services à l'utilisateur sans qu'il formule une demande explicite [Chaari, 2007]. Les informations du contexte doivent être stockées pour des utilisations ultérieures ; ce qui permet un gain de temps énorme dans la spécification des stratégies d'adaptation et dans la génération automatique d'interfaces utilisateur.

Les applications sensibles au contexte doivent s'adapter aux besoins des utilisateurs. Les adaptations doivent être centrées sur *les préférences des utilisateurs*. La génération des interfaces sensibles au contexte doit être faite en fonction du profil de l'utilisateur : capacités intellectuelles, état physique, âge, sexe, etc.

L'objectif de l'informatique ubiquitaire est de rendre invisible les ordinateurs pour les utilisateurs et les aider à accomplir leurs tâches en leurs fournissant l'information adéquate sans qu'ils se rendent compte. De ce fait, l'architecture doit être capable d'identifier *l'activité en cours de l'utilisateur* pour pouvoir l'assister et lui fournir les données contextuelles dont il a besoin à un moment donné.

Une architecture sensible au contexte doit avoir un module relatif à la *gestion des situations imprévues*. Cette partie est très importante puisqu'un environnement ubiquitaire est en constante évolution. Il faut donc envisager toutes les adaptations qui peuvent avoir lieu en ajoutant un composant portant sur la gestion des situations imprévues relatives au contexte ou à l'utilisateur.

La question qui se pose à ce stade est de savoir si une architecture qui ne répond pas à ces différents critères peut mener à bien son rôle et générer des interfaces adaptables au contexte à savoir à l'utilisateur, à son environnement et à la plateforme utilisée.

Le tableau 2.1 présente et décrit la prise en compte de ces critères au niveau des architectures vues dans la section précédente. Nous constatons tout d'abord, que la plateforme SECAS assure l'adaptation des applications au contexte d'utilisation. Cette architecture garantit la sensibilité des applications au contexte d'utilisation selon trois dimensions : l'adaptation du contenu, du comportement et de la présentation. En outre, cette architecture prend en compte les préférences de l'utilisateur vu que les données stockées dans le « context repository » sont écrites en CC/PP. Cependant, rien ne prouve que cette architecture prenne en considération l'activité en cours de l'utilisateur et la gestion de l'imprévu et des droits d'accès.

Tableau 1.2 — Tableau comparatif des différentes architectures

	Modèlisation du contexte	Gestion de l'historique	Considération des préférences de l'utilisateur	Considération de l'activité courante de l'utilisateur	Gestion de l'imprévu
SECAS	XML	Oui (context Repository)	Oui (utilisation de CC/PP)	Non	Non
Context Toolkit	Attribut/valeur	Disponible sur un serveur	Non	Non	Non
CoBra	Ontologie OWL	Oui (Context Knowledge base)	Non	Non	Non
CMF	Ontologie RDF	Non	Oui (utilisation de la logique floue)	Non	Non
SOCAM	Ontologie OWL	Oui (BD du contexte)	Oui	Oui	Non
CAASS	XML	Non	Non	Non	Oui via les scripts d'adaptation, il est possible d'identifier les situations imprévues
HyCon	XML	Oui (BD MySql)	Non	Non	Non
JCAF	Approche orientée objet	Non	Fournir les ressources dont l'utilisateur a besoin	Non	Non
CAAMDTA	Ontologie (UbisWorld)	BD du contexte	Oui (BD user avec UserML basé sur XML contenant les préférences de l'utilisateur	Oui	Non
GECAF	CDL (XML)	Oui	Non	Non	Non
AHCI	XML	Oui (couche capteur et BD contexte)	Oui	Oui	Non

Nous remarquons aussi à travers l'architecture Context Toolkit, qu'elle est plutôt orientée « application ». Ses composants ne prennent pas en considération les préférences de l'utilisateur ainsi que son activité en cours. Aucun de ces composants ne prend en charge la gestion de l'imprévu.

L'architecture CoBra propose un raisonnement sur les composants physiques du contexte à savoir les capteurs qui se trouvent dans les environnements intelligents. Elle ne prend en aucun cas en considération l'activité en cours de l'utilisateur.

L'architecture CMF prend en charge l'utilisateur ainsi que ses préférences vu l'utilisation de la logique floue dans la description des données contextuelles. Cependant, l'activité en cours de l'utilisateur n'est pas spécifiée par cette architecture.

La plateforme SOCAM possède beaucoup de points positifs à comparer aux autres architectures puisqu'elle prend en considération l'activité courante de l'utilisateur ainsi que ses préférences.

Le principal avantage de l'architecture CAASS, c'est qu'elle tente de remédier au problème de gestion de l'imprévu au sein d'un système sensible au contexte. En effet, à l'aide des « scripts d'adaptation » on peut identifier si la situation est courante ou si elle présente un imprévu. Si c'est le cas, c'est la partie « décision » qui va tenter de produire un script. Et si la génération de ce dernier n'est pas possible, alors l'administrateur doit le faire manuellement. Cependant, cette architecture ne prend pas en compte le type de l'utilisateur, ses préférences ainsi que son activité en cours.

HyCon fournit une adaptation selon le dispositif physique et non selon le contexte courant de l'utilisateur ou encore selon son profil, son comportement et ses préférences. HyCon prend en charge plusieurs type de capteurs à savoir le GPS, les thermomètres, les antennes RFID, etc.

JCAF prend relativement en charge les préférences de l'utilisateur (en lui fournissant les ressources dont il a besoin). L'accès à ces entités est contrôlé par l'entité « contrôle d'accès », ce qui assure une authentification correcte des clients. Cependant, la gestion de l'imprévue n'est pas assurée par JCAF.

Ce qui différencie CAAMDTA des autres c'est la présence du moteur d'adaptation. Cette couche en coopération avec l'interface utilisateur et l'utilisateur virtuel assure la phase d'adaptation proprement dite en appliquant des règles d'inférences à l'information reçue et en déduisant les adaptations adéquates pour l'utilisateur. CAAMDTA est une architecture centrée utilisateur puisqu'elle adapte l'interface utilisateur selon son activité courante ainsi que ses préférences. Cependant, on remarque l'absence de la gestion de l'imprévue dans CAAMDTA.

GECAF est plus concentré sur le stockage et la gestion de l'historique du contexte. La modélisation du contexte est faite via XML. Toutefois, les préférences et le profil de l'utilisateur n'ont pas été pris en charge et la gestion de l'imprévu a été omise.

AHCI est la seule architecture qui répond plus au moins à nos critères. En revanche le modèle utilisé pour la représentation et la modélisation du contexte, représente son point négatif. En effet, pour modéliser un domaine critique, il faut absolument avoir un modèle de représentation formel et rapide.

1.6 Conclusion

Nous avons défini dans ce chapitre les notions de contexte, de sensibilité au contexte et d'adaptation. Nous avons présenté ensuite l'architecture générale d'un système sensible au contexte composée de cinq couches à savoir : la couche capture du contexte, l'interprétation du contexte, le stockage du contexte, la diffusion du contexte et la couche application.

Nous avons aussi présenté les principales architectures des systèmes sensibles au contexte qui existent dans la littérature. Nous avons également proposé des critères de comparaison afin de déduire les avantages et les inconvénients de chaque architecture.

L'un des critères les plus importants est la modélisation du contexte. En effet, comment modéliser le contexte afin de fournir à l'utilisateur une interface qui soit adaptable à ses préférences et à son contexte d'usage ?

Le chapitre suivant présentera, dans ce sens, un état de l'art sur les approches de modélisation du contexte.

chapitre 2

Approches de modélisation du contexte

2.1 Introduction

A partir de l'état de l'art portant sur les différentes architectures des systèmes sensibles au contexte, nous avons pu déterminer les caractéristiques fondamentales d'une application sensible au contexte.

L'architecture doit être conforme aux spécificités faites par la communauté scientifique qui stipule la nécessité de séparer, dans les applications, l'acquisition et la modélisation du contexte de son utilisation. Cette architecture doit être formelle et doit prendre en considération le profil et le niveau de connaissance de l'utilisateur.

L'interface doit fournir à l'utilisateur l'information adéquate à ses besoins à un moment donné. La première phase de toute approche d'implémentation d'applications sensibles au contexte, est la phase de capture et de modélisation du contexte.

En effet, un certain nombre d'approches de modélisation du contexte ont été développées au fil des années. Afin de faciliter notre choix, notre approche de modélisation doit répondre à certains critères relatifs aux propriétés d'un environnement ubiquitaire.

Tout d'abord, nous allons présenter dans la première partie de ce chapitre, l'état de l'art relatif aux différentes approches de modélisation du contexte. Nous allons ensuite, énumérer les exigences relatives à un environnement ubiquitaire. Nous conclurons avec une comparaison entre ces différentes approches.

2.2 Différents modèles de contexte

Les approches de modélisation du contexte ont été ces dernières années, la cible de plusieurs travaux de recherches [Ye *et al.*, 2012], [Bettini *et al.*, 2010], [Strang et Linnhoff−Popien, 2004][Ben Ismail, 2014]. En effet, les données contextuelles captées doivent être représentées sous une forme appropriée afin d'être analysées et implémentées. Dans cette partie, nous présentons les techniques de modélisation les plus fréquemment utilisées (Figure 2.1).

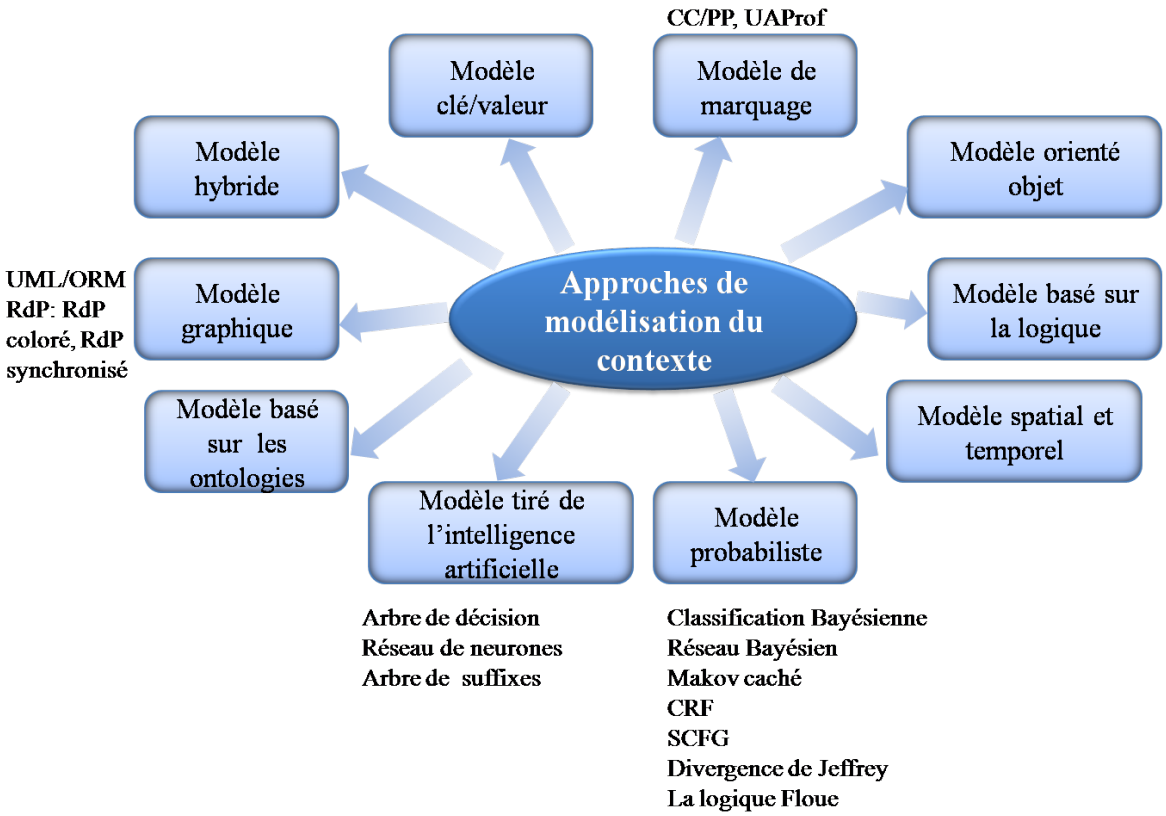


Figure 2.1 — Approches de modélisation du contexte.

2.2.1 Le modèle clé/valeur

Le modèle clé/valeur est la structure la plus simple pour la modélisation des données de l'information contextuelle [Strang et Linnhoff−Popien, 2004] [Samulowitz *et al.*, 2002]. En effet, cette technique de modélisation utilise des paires clé/valeur pour définir la liste des

attributs et leurs valeurs décrivant les informations du contexte utilisées par les applications sensibles au contexte (Figure 2.2).
Schilit *et al.* ont utilisé cette approche pour modéliser le contexte en fournissant la valeur d'une information du contexte à une application en tant que variable de l'environnement [Schilit *et al.*, 1994].

Category	Example
Date and time	`after April 15`
	`between 10 and 12noon`
Location	`in room 35-2200`
Co-location	`with {User Adams}`
	`with {Type Display} having {Features Color}`

Figure 2.2 — Exemple de modèle clé/valeur [Schilit *et al.*, 1994].

L'approche de modélisation clé/valeur est fréquemment utilisée dans les services distribués. Cette approche est facile à gérer, mais elle manque de structuration pour permettre à des algorithmes efficaces de récupérer les données du contexte.

2.2.2 Les modèles de marquage

Ce type d'approche possède une structure de données hiérarchique constituée de balises avec des attributs et du contenu. Elle est fondée sur la norme générique SGML. Parmi ces méthodes nous pouvons citer CC/PP et UAProf.

2.2.2.1 Composite Capabilities/Preferences Profile CC/PP

CC/PP [Klyne *et al.*, 2004] est probablement la première approche de modélisation du contexte basée sur Resource Description Framework (RDF) et qui inclut des contraintes élémentaires et des relations entre les différents types des données du contexte. CC/PP est un standard W3C conçu pour la description des caractéristiques du dispositif et des préférences de l'utilisateur. Généralement, dans ce genre d'approches de modélisation, les auteurs étendent ces modèles avec leurs propres vocabulaires afin de les adapter à leurs objectifs. Par exemple, nous pouvons citer :

- Le modèle Augmented-CC/PP (ACC/PP) [Hanmansetty, 2004]. Cet exemple ajoute au modèle CC/PP d'autres données contextuelles telles que le rôle de l'utilisateur et son comportement ;

– Comprehensive Structured Context Profiles (CSCP) par [Held *et al.*, 2002]. Elle soutient la flexibilité complète de RDF/S pour exprimer les structures des informations contextuelles. Les noms des attributs sont interprétés avec la sensibilité au contexte en fonction de leur position dans la structure de profil. Cette approche, comme le montre la Figure 2.3, tente de remédier à quelques lacunes du CC/PP, comme la hiérarchie fixe des données
– Le modèle CC/PP Context Extension [Indulska *et al.*, 2003], qui étend CC/PP par certaines propriétés (Composant-Attribut) relatives aux informations contextuelles.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:cscp="context-aware.org/CSCP/CSCPProfileSyntax#"
  xmlns:dev="context-aware.org/CSCP/DeviceProfileSyntax#"
  xmlns:net="context-aware.org/CSCP/NetworkProfileSyntax#"
  xmlns="context-aware.org/CSCP/SessionProfileSyntax#"
  <SessionProfile rdf:ID="Session">
    <cscp:defaults rdf:resource=
      "http://localContext/CSCPProfile/previous#Session"/>
    <device><dev:DeviceProfile>
      <dev:hardware><dev:Hardware>
        <dev:memory>9216</dev:memory>
      </dev:Hardware></dev:hardware></dev:DeviceProfile>
    </device>
  </SessionProfile>
</rdf:RDF>
```

Figure 2.3 — Exemple de CSCP [Strang et Linnhoff−Popien, 2004].

2.2.2.2 User Agent Profile (UAProf)

UAProf est l'extension la plus importante de CC/PP. Elle est basée sur la technique RDF et est supportée par l'organisme Open Mobile Alliance (OMA). Ce modèle permet de capturer des données selon les capacités (le fournisseur du dispositif, le modèle, la taille de l'écran, les capacités multimédia du dispositif, le jeu de caractères adapté, les caractéristiques du vidéo, streaming ainsi que les capacités MMS (Multimedia Messaging Service)) et les caractéristiques des dispositifs mobiles à savoir les caractéristiques matérielles (capacité de sorti et d'interaction, l'affichage du clavier, etc.), logicielles (système d'exploitation, types de multimédia supporté) et les caractéristiques du réseau (option de sécurité ou les détails sur le bluetooth) (Figure 2.4) [Timmerer *et al.*, 2010].

```
<?xml version="1.0"?>
<! DOCTYPE rdf:RDF [
  <!ENTITY ns-rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <!ENTITY ns-prf "http://www.openmobilealliance.org/tech/profiles/UAPROF/ccppschema
                   -20030226#">
  <!ENTITY prf-dt "http://www.openmobilealliance.org/tech/profiles/UAPROF/xmlschema
                   -20030226#">
]>
<rdf:RDF xmlns:rdf="&ns-rdf;"
         xmlns:prf="&ns-prf;">
 <rdf:Description rdf:ID="MyDeviceProfile">
  <prf:component>
   <rdf:Description rdf:ID ="HardwarePlatform">
    <rdf:type rdf:resource="&ns-prf;HardwarePlatform"/>
    <prf:ScreenSizeChar rdf:datatype="&prf-dt;Dimension">15x6</prf:ScreenSizeChar>
    <prf:BitsPerPixel rdf:datatype="&prf-dt;Number">2</prf:BitsPerPixel>
    <prf:ColorCapable rdf:datatype="&prf-dt;Boolean">No</prf:ColorCapable>
    <prf:TextInputCapable rdf:datatype="&prf-dt;Boolean">Yes</prf:TextInputCapable>
    <prf:ImageCapable rdf:datatype="&prf-dt;Boolean">Yes</prf:ImageCapable>
    <prf:Keyboard rdf:datatype="&prf-dt;Literal">PhoneKeypad</prf:Keyboard>
    <prf:NumberOfSoftKeys rdf:datatype="&prf-dt;Number">0</prf:NumberOfSoftKeys>
   </rdf:Description>
  </prf:component>

  <prf:component>
   <rdf:Description rdf:ID="SoftwarePlatform">
    <rdf:type rdf:resource="&ns-prf;SoftwarePlatform"/>
    <prf:AcceptDownloadableSoftware
         rdf:datatype="&prf-dt;Boolean">No</prf:AcceptDownloadableSoftware>
    <prf:CcppAccept-Charset>
     <rdf:Bag>
      <rdf:li rdf:datatype="&prf-dt;Literal">US-ASCII</rdf:li>
      <rdf:li rdf:datatype="&prf-dt;Literal">ISO-8859-1</rdf:li>
      <rdf:li rdf:datatype="&prf-dt;Literal">UTF-8</rdf:li>
      <rdf:li rdf:datatype="&prf-dt;Literal">ISO-10646-UCS-2</rdf:li>
     </rdf:Bag >
    </prf:CcppAccept-Charset>
   </rdf:Description>
  </prf:component>
 </rdf:Description>
</rdf:RDF>
```

Figure 2.4 — Exemple de UAProf [Timmerer *et al.*, 2010].

Il existe plusieurs autres approches de modélisation du contexte dans la catégorie des modèles de balisage. Ils sont souvent soit exclusifs ou limités à un petit ensemble d'aspects contextuels par exemple : Context Configuration [Capra *et al.*, 2001], ConteXtML [Ryan, 1999] et Pervasive Profile Description Language (PPDL) [Chtcherbina et Franz, 2003].

Les principaux inconvénients de ces approches résident dans le fait que ces approches ont du mal *(i)* à capter les différents type de données du contexte, les relations, les dépendances et la qualité de l'information du contexte, *(ii)* à permettre le contrôle de la cohérence des données, et *(iii)* à supporter le raisonnement sur le contexte, sur l'incertitude et sur les abstractions de haut niveau du contexte.

2.2.3 Les modèles orientés objets

L'objectif de cette modélisation est d'employer les principaux avantages des approches orientées objet à savoir l'encapsulation et la réutilisation pour couvrir une partie des problèmes liés à la dynamique du contexte dans les environnements ubiquitaires. Les détails de la transformation du contexte sont encapsulés au niveau de l'objet et, par conséquent, cachés aux autres composants. L'accès à l'information contextuelle est fourni à travers des interfaces spécifiques. Parmi ces approches nous pouvons citer « les indices » développés au sein du projet TEA [Schmidt et Van Laerhoven, 2001]. Le concept d'indice fournit une abstraction à partir de capteurs physiques et logiques. Un repère est considéré comme une fonction prenant la valeur d'un seul capteur physique ou logique jusqu'à un certain temps en entrée et fournissant en sortie des symboles. Un ensemble fini ou infini de valeurs possibles est défini pour chaque repère. La sortie de chaque repère dépend d'un seul capteur, mais différents indices peuvent être basés sur les mêmes capteurs. Le contexte est modélisé comme un niveau d'abstraction au-dessus des indices disponibles. Ainsi, les indices sont des objets qui fournissent des informations contextuelles grâce à leurs interfaces, cachant les détails de la détermination des valeurs de sortie.

Une autre approche dans cette catégorie est le modèle actif d'objet du projet GUIDE. Cette modélisation a été principalement alimentée par l'exigence d'être en mesure de gérer une grande variété d'informations contextuelles personnelles et environnementales, tout en maintenant l'évolutivité. Tous les détails de collecte de données et de fusion sont encapsulés dans les objets actifs et donc confidentiels à d'autres composants du système.

Bouzy *et al.* proposent d'utiliser des mécanismes généraux orientés objets pour représenter les connaissances contextuelles de point de vue temporel, objectif et contexte spatial. Ils ont justifié leur approche avec les notions d'héritage et les capacités de réutilisation, ce qui permet de définir le plus petit nombre de propriétés, de fonctions et de règles afin de simplifier la représentation des connaissances dans des domaines très complexes [Bouzy et Cazenave, 1997].

Henricksen *et al.* proposent le langage de modélisation du contexte (CML), basé sur la modélisation objet-rôle (ORM) qui a été développé pour la modélisation conceptuelle des bases de données [Henricksen *et al.*, 2002]. CML fournit une notation graphique conçue pour soutenir l'ingénierie logicielle dans l'analyse et la spécification des exigences du contexte dans une application sensible au contexte [Henricksen et Indulska, 2004].

CML présente une extension d'ORM qui soutient :
- La capture des différentes classes et sources du contexte ;
- La capture des informations imparfaites du contexte ;

– La capture des différentes relations entre les informations du contexte [Henricksen et Indulska, 2006].

La Figure 2.5 illustre un exemple de notation graphique pour la modélisation du contexte. Le modèle capture les activités de l'utilisateur sous la forme d'une entité temporelle qui couvre les activités passées, présentes et futures, les relations qui existent entre les utilisateurs, les canaux de communications et les dispositifs mobiles. Les cercles représentent les objets avec entre parenthèses leurs valeurs décrivant la représentation du type de l'objet en question, les cadres décrivent le rôle joué par chaque objet et la clé résume le reste de la notation utilisée dans la figure.

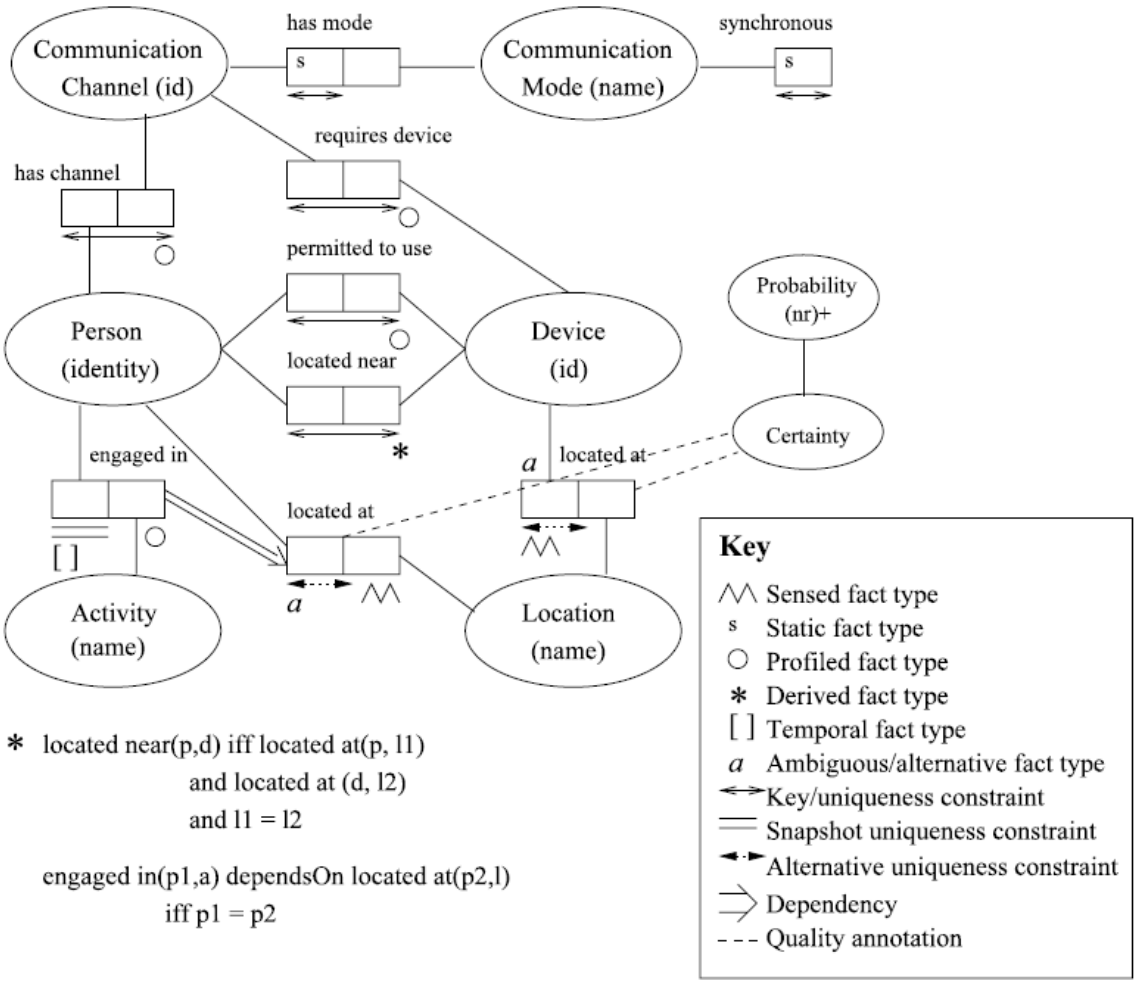


Figure 2.5 — Exemple d'un modèle CML [Bettini *et al.*, 2010].

2.2.4 Les modèles basés sur la logique

Une logique définit les conditions dans lesquelles un fait peut être dérivé à partir d'un ensemble d'expressions ou d'autres faits. Un modèle de cadre logique basé sur le contexte est par conséquent défini comme un ensemble de faits, d'expressions et de règles. Fréquemment les informations contextuelles sont ajoutées, mises à jour ou supprimées.

Une des premières approches appelée Formalizing Context fondée sur la logique a été étudiée et publiée en 1993 par McCarthy et son groupe à Stanford. McCarthy a présenté les contextes comme étant des entités mathématiques abstraites ayant des propriétés tirées de l'intelligence artificielle. Il a essayé de fournir une méthode permettant la formalisation des axiomes simples pour des phénomènes de sens commun. La relation de base de cette approche est is (c, p), qui affirme que la proposition p est vrai dans le contexte c. Le modèle de McCarthy prend également en charge le concept d'héritage [McCarthy et Buvac, 1997]. Par exemple la figure 2.6 traduit une situation dans laquelle une personne ne peut pas être interrompue (en réunion ou au téléphone).

$$\begin{aligned} Occupied(person) = \exists t_1, t_2, event | engagedIn(person, event, t_1, t_2) \\ \cdot((t_1 \leq timenow() \wedge (timenow() \leq t_2 \vee isnull(t_2))) \\ \vee((t_1 \leq timenow() \vee isnull(t_1)) \wedge timenow() \leq t_2)) \\ \wedge(event = \text{`}inmeeting\text{'} \vee event = \text{`}takingcall\text{'}) \end{aligned}$$

Figure 2.6 — Exemple d'un modèle basé sur la logique [Ye *et al.*, 2012].

L'approche Giunchiglia, appelée systèmes multi contextes, est orientée sur le raisonnement du contexte [Ghidini et Giunchiglia, 2001]. Il a considéré le contexte comme étant un sous-ensemble spécifique de l'état complet d'une personne physique qui est utilisé pour raisonner sur un objectif donné.

L'approche d'Akman et Surav [Akman et Surav, 1997] appelée la théorie des situations étendues, étend la théorie des situations proposée par Barwise et Perry [Barwise et Perry, 1980]. Ils ont tenté de définir un modèle théorique sémantique du langage naturel dans un système de logique formelle. Akman et Surav ont utilisé ce système pour modéliser le contexte avec des situations ordinaires. La variation du contexte est traitée sous forme de règles.

Une approche similaire à cette dernière a été également proposée par Gray et Salber [Gray et Salber, 2001]. Ils utilisent la logique des prédicats de premier ordre comme une re-

présentation formelle des propositions et des relations contextuelles.

L'objectif de ces approches est de fournir un fondement théorique pour la construction d'un système intègre tout en *(i)* représentant officiellement les spécifications logiques des situations, *(ii)* vérifiant l'intégrité et la cohérence des spécifications de la situation dans une base de règles, et *(iii)* étendant les installations existantes pour être en mesure d'intégrer les données des capteurs des nouvelles situations.

Cette approche facilite le développement de systèmes sensibles au contexte. Elle permet aux développeurs de déterminer la situation en cours tout en précisant les relations qui existent entre les situations (telles que la composition et la concurrence) et d'évaluer la solidité et l'exhaustivité des programmes de la situation.

2.2.5 Les modèles spatiaux tomporels

L'espace représente un point primordial dans la modélisation des applications sensibles au contexte. La plupart des définitions du contexte définissent l'espace comme étant un facteur vital, par exemple, Schilit et Adams considèrent trois aspects importants du contexte à savoir : Où êtes-vous ?, avec qui êtes vous ? Et quelles sont les ressources à proximité ? [Schilit *et al.*, 1994]. En outre, la définition du contexte la plus utilisée est celle proposée par Dey [Dey *et al.*, 2001] qui considère l'espace comme étant l'aspect central des entités du contexte : Une entité est une personne, un lieu ou un objet qui est considéré comme pertinent pour l'interaction entre un utilisateur et une application.

La majorité des modèles spatiaux du traitement du contexte sont des modèles basés sur les faits qui organisent leurs informations du contexte par emplacements physiques. Cela pourrait être l'emplacement des entités du monde réel qui est décrit dans les informations du contexte ou l'emplacement du capteur qui mesure les informations du contexte. Cette information de localisation est soit prédéfinie (si les entités sont statiques), ou obtenue par les systèmes de positionnement qui permettent de suivre les objets mobiles par rapport à leur position.

Fondamentalement, deux types de systèmes de coordination sont pris en charge par les systèmes de positionnement :

- Les coordonnées géométriques : elles représentent des points ou des zones dans un espace métrique, telles que les coordonnées WGS84 du GPS (latitude, longitude et altitude). L'utilisation des fonctions géométriques telles que la distance euclidienne permet de calculer la distance du plus proche voisin ;
- Les coordonnées symboliques : elles sont représentées par un identifiant, tel qu'un numéro de chambre ou l'ID d'un point d'accès dans la téléphonie mobile. Contrairement

aux coordonnées géométriques il n'y a pas de relation spatiale offerte par les coordonnées symboliques. Afin de permettre le raisonnement spatial sur l'inclusion et les distances, des informations explicites sur les relations spatiales entre des paires de coordonnées symboliques doivent être fournies.

Augusto *et al.* introduisent les opérateurs temporels ANDlater ANDsim dans les règles « Événement-Condition-Action » sur lesquelles la connaissance temporelle sur les activités humaines peut être spécifiée [Augusto *et al.*, 2008]. Gottfried *et al.* appliquent des techniques qualitatives d'intelligence artificielle dans le traitement de la connaissance spatiale et temporelle dans les maisons intelligentes [Gottfried *et al.*, 2006].

De toute évidence, les modèles contextuels spatiaux temporels sont bien adaptés pour les applications sensibles au contexte, qui sont principalement basées sur la localisation. Cependant, même si l'emplacement n'est pas un contexte primaire pour une application sensible au contexte, une organisation spatiale des informations du contexte peut être bénéfique : si la quantité d'information gérée du contexte est importante, le cloisonnement spatial peut être utilisé pour faire face à la complexité. Un autre inconvénient du modèle spatial est l'effort rencontré pour rassembler les données de localisation de l'information du contexte et les maintenir à jour.

2.2.6 Les modèles probabilistes

La spécification et l'identification des situations du contexte peuvent avoir une grande variabilité en fonction de certains facteurs tels que le temps, l'emplacement, les utilisateurs et l'environnement de travail. Les techniques probabilistes ont été largement appliquées sur des associations complexes entre les situations et les données des capteurs.

2.2.6.1 La classification Bayésienne

La classification Bayésienne [Lowd et Domingos, 2005] est un modèle de classification simple qui applique le théorème de Bayes avec l'hypothèse d'indépendance conditionnelle, c'est-à-dire que, compte tenu de la classification de sortie, tous les attributs d'entrée ou les paramètres qui caractérisent les éléments de preuve sont conditionnellement indépendants les uns des autres. Cette méthode repose sur la formule suivante :

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

Où X est considéré comme preuve de l'hypothèse H, $P(X)$ et $P(H)$ sont les probabilités a

priori de X et H, et $P(X|H)$ est la probabilité *a posteriori* (conditionnelle) de X conditionnée par H [van Kasteren et Krose, 2007]. En théorie, la classification Bayésienne possède un taux d'erreur minimal par rapport aux autres classificateurs. Dans la pratique, sa performance peut se dégrader lorsque l'hypothèse d'indépendance d'un attribut n'est pas valide ou qu'il y a un manque de données sur les probabilités.

2.2.6.2 Les réseaux Bayésiens

Lorsqu'il existe des dépendances entre les attributs, les réseaux bayésiens peuvent être appliqués. Un réseau bayésien est un graphe acyclique dirigé, dans lequel chaque nœud représente une variable qui peut être discrète ou continue, et chaque arc est un lien de causalité entre les nœuds. S'il y a un arc qui relie le nœud A au nœud B, alors A est appelé un parent de B, ce qui implique que la variable B est considérée comme dépendante directement de A [Naïm *et al.*, 2011].

Dans un réseau bayésien, si un nœud n'a pas de parent, il est alors appelé la racine. Chaque nœud racine est associé à une probabilité *a priori*. Chaque nœud non racine est associée à une distribution de probabilité conditionnelle (CPD). Si les variables sont discrètes, la CPD est représentée avec une table de probabilité conditionnelle qui contient toutes les combinaisons possibles de leurs nœuds parents. Les réseaux bayésiens ont été utilisés dans de nombreux systèmes sensibles au contexte [Ye *et al.*, 2007b]. La figure 2.7 présente un exemple de ce type d'approche.

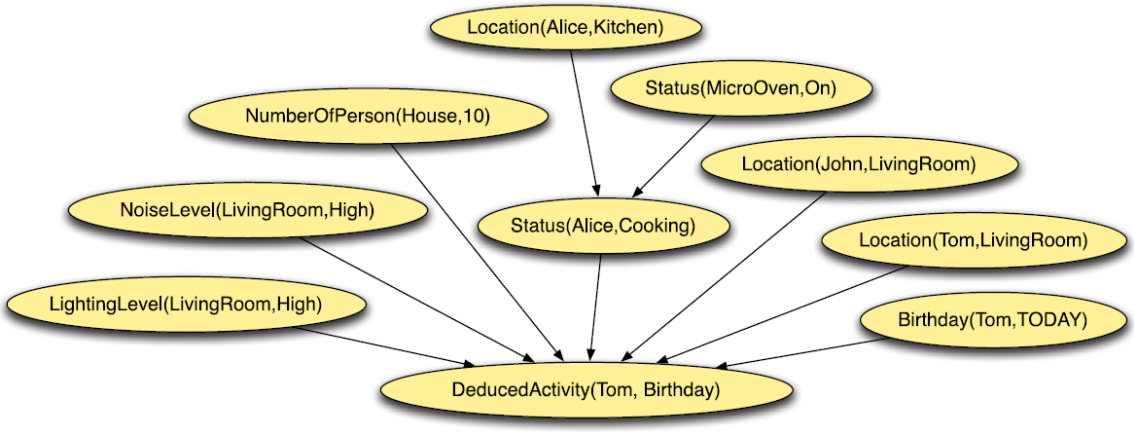


Figure 2.7 — Exemple d'un réseau Bayésien [Ye *et al.*, 2012].

Tel que présenté, les réseaux bayésiens fournissent une méthode claire et compréhensible pour incorporer la façon dont la probabilité d'une possibilité d'un événement est conditionnée

à un autre événement. Ils sont les mieux adaptés aux applications où il n'est pas nécessaire de représenter l'ignorance, où le conditionnement est facile à extraire par la représentation probabiliste et où les probabilités *a priori* sont disponibles. Cependant, ils peuvent manquer de crédibilité en raison de l'indisponibilité des estimations.

2.2.6.3 Les modèles cachés de Markov

Un modèle caché de Markov (Hidden Markov Models HMM) [Atallah et Yang, 2009] est un modèle statistique, où un système en cours de modélisation est supposé être une chaîne de Markov qui est une séquence d'événements. Un HMM est composé d'un ensemble fini d'états cachés (par exemple, st-1, st, et st+1) et des observations (par exemple, ot-1, ot, et ot +1) qui sont générés à partir de ces états. Un HMM repose sur trois hypothèses : *(1)* chaque état ne dépend que de son prédécesseur immédiat, *(2)* chaque variable d'observation ne dépend que de l'état actuel, et *(3)* les observations sont indépendantes les unes des autres. Au sein d'un HMM, il existe trois types de distributions de probabilité : *(1)* les probabilités *a priori* des états initiaux p (si), $1 \leq i \leq n$, où n est le nombre d'états ; *(2)* les probabilités de transition d'état p (st | st- 1), et *(3)* probabilités d'émission d'observation p (ot | st).

Les HMM ont été un modèle populaire dans le domaine de la reconnaissance de l'activité [van Kasteren *et al.*, 2008]. En effet, ils représentent les observations des données du capteur selon trois types dans chaque activité, à savoir : les données brutes du capteur, le changement des données de capteurs, les dernières données observées du capteur, et la combinaison entre eux. Le HMM permet l'obtention de trois paramètres de probabilité : *(i)* la probabilité *a priori* d'une activité représente la probabilité que l'utilisateur fasse cette activité, *(ii)* la probabilité de transition d'état représente la probabilité que l'utilisateur change d'une activité à une autre, et *(iii)* la probabilité d'émission d'observation représente la probabilité d'observation du capteur lorsque l'utilisateur effectue une certaine activité.

2.2.6.4 Les grammaires stochastiques hors-contexte (SCFG)

Les grammaires hors-contexte (CFG) fournissent une base théorique solide pour la modélisation des processus structurés. Une CFG conduit naturellement à la représentation des concepts de façon récursive, ce qui permet à l'action d'être définie sur la base des sous-événements. Les grammaires Stochastiques hors-contexte (SCFG) sont une extension probabiliste des CFG, où les règles de production sont complétées avec des probabilités qui fournissent une base quantitative pour le classement et l'élagage des dépendances. Elles ont été appliquées pour modéliser la sémantique des activités dont la structure est supposée être

connue à l'avance. Les résultats de l'évaluation montrent que les SCFG présentent des atouts dans la détection des activités de niveau supérieur et ainsi que dans le traitement des erreurs dans les tâches de bas niveau tels que les erreurs de suivi et les observations manquantes. Les deux approches fonctionnent bien lorsque la structure des activités d'intérêt n'est pas très complexe et inconnue par les développeurs. Dans des situations complexes impliquant plusieurs agents et nécessitant des relations temporelles plus complexes comme le parallélisme, le chevauchement, ou le synchronisme, il s'avère difficile de formuler des règles grammaticales manuellement [Turaga *et al.*, 2008].

2.2.7 Les modèles issus de l'intelligence artificielle

2.2.7.1 Les approches basées sur la logique floue

Dans l'informatique omniprésente, la logique floue est utilisée pour cartographier les données des capteurs en des variables linguistiques et évaluer le degré de similitude des concepts [Hong *et al.*, 2009]. Une valeur de probabilité reflète la fiabilité des données, qui peuvent être fournies par l'expert du domaine, ou obtenues à partir de l'expérience de l'utilisateur. La logique floue prend en charge plusieurs types d'opérations, à savoir l'intersection, l'union et la modification des ensembles flous.

[Haghighi *et al.*, 2008] ont introduit des fonctions floues pour caractériser les données des capteurs et pour montrer dans quelle mesure les données du capteur d'entrée (par exemple, la pression artérielle systolique ou diastolique) correspondent à des variables linguistiques (par exemple, faible, normal ou élevé) qui sont utilisées dans les définitions de la situation.

2.2.7.2 Les arbres de décision

Un arbre de décision est un modèle prédictif, où chaque feuille représente une classification et chaque branche représente une conjonction d'éléments qui conduit à des classifications cibles. Un arbre de décision repose sur des informations d'entropie en ce qui concerne sa construction. Il fonctionne en choisissant une variable à chaque étape qui est la meilleur variable à utiliser dans le fractionnement de l'ensemble des éléments. Par rapport aux modèles bayésiens, un arbre de décision ne prend pas en compte l'hypothèse de dépendance ou la séquence temporelle entre les classifications. Un des principaux avantages d'un arbre de décision, c'est qu'il peut générer des règles de classification qui sont faciles à comprendre et à expliquer. Ces règles sont utiles dans l'analyse des performances des capteurs et l'extraction de caractéristiques. Les arbres de décision sont efficaces dans le raisonnement pour les ensembles de

données relativement petits. Cependant, l'efficacité de leur construction est liée à la taille des données. De plus, la construction de ces arbres exige que les données d'apprentissage résident dans la mémoire [Karantonis *et al.*, 2006].

2.2.7.3 Les réseaux de neurones

Les réseaux de neurones sont une technique sous-symbolique, inspirée par les réseaux de neurones biologiques. Ils fournissent une technique alternative pour la reconnaissance d'activité. Ils peuvent apprendre automatiquement les mappages complexes et extraire une combinaison non linéaire de fonctions. Un réseau de neurones est composé de nombreux neurones artificiels qui sont reliés entre eux selon une architecture spécifique [Yang *et al.*, 2008]. La Figure 2.8, représente une structure d'un classificateur neuronal, qui consiste en une couche d'entrée (u1, u2, . . . , ur), une couche cachée et une couche de sortie (y1, . . . , yh). Les mappages entre l'entrée et la sortie sont représentés dans la composition de l'activation des fonctions f qui peuvent être appris par un processus de formation.

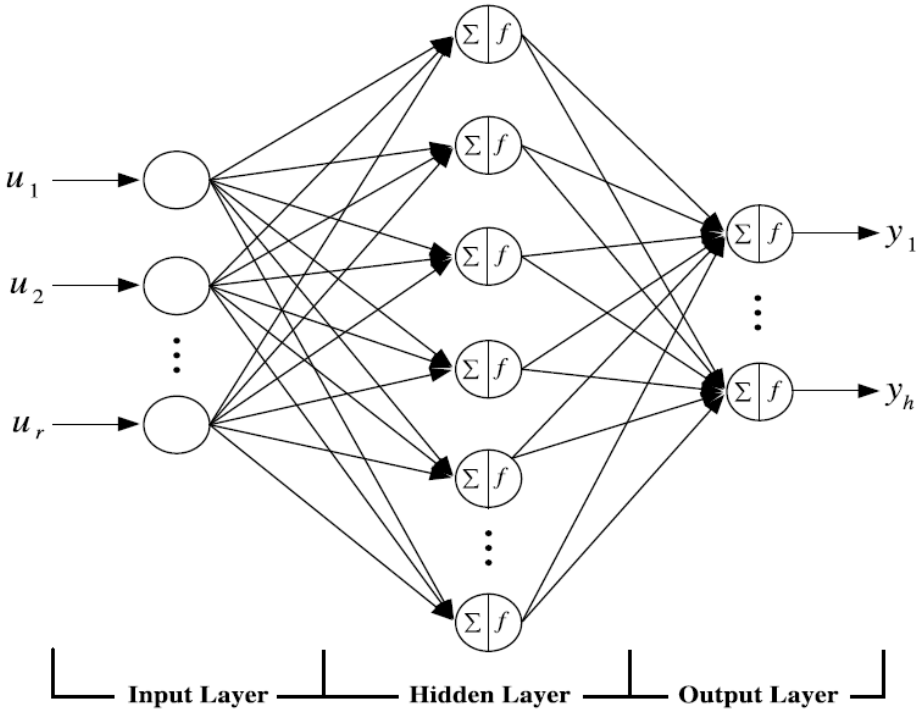


Figure 2.8 — Structure d'un réseau de neurones [Ye *et al.*, 2012].

La performance des réseaux de neurones est relative à la quantité des données de formation. Un réseau de neurones est considéré comme un bon choix s'il y a beaucoup de données sur

la formation et si le domaine du problème est mal compris. Cependant, si les données de formation ne couvrent qu'une partie importante des conditions de fonctionnement ou si les données sont bruyantes, les réseaux de neurones ne donneront pas de bons résultats. Les données contextuelles sont par définition, assez complexes et trop bruyantes. Par conséquent, cette méthode est considérée comme étant inappropriée pour la modélisation du contexte.

2.2.7.4 Les arbres de suffixes

Hamid *et al.* présentent un cadre non supervisé pour découvrir les plus petits grains d'un événement dans la vie quotidienne pour les activités humaines. Une activité est définie comme une séquence finie d'événements, alors qu'un événement est défini comme une interaction particulière entre un sous-ensemble d'objets clés sur une durée de temps limitée. Un objet clé est un objet présent dans un environnement qui offre des fonctionnalités qui peuvent être nécessaires pour l'exécution d'une activité. L'idée est d'extraire les événements à partir des interactions entre les utilisateurs humains et les objets clés, d'organiser ces événements dans un arbre de suffixes, et d'extraire des modèles d'événements pour chaque activité [Hamid *et al.*, 2009].

La principale contribution de ce travail est la représentation des connaissances et leur manipulation qui peuvent faciliter l'apprentissage des activités humaines dans la vie quotidienne. L'utilisation des arbres de suffixes fournit une représentation efficace des activités capable d'analyser les séquences d'événements. Cependant cette approche est défaillante sur la modélisation de séquence d'activité complexe et incapable de gérer les activités qui présentent des chevauchements au niveau des évènements. De ce fait, ce modèle est également défaillant lors de la modélisation du contexte. En effet, les connaissances contextuelles sont assez complexes et sont généralement définies par une séquence infinie d'évènements.

2.2.8 Les modèles basés sur les ontologies

Le contexte peut être considéré comme un type spécifique de la connaissance. Ainsi, il est naturel de rechercher si un cadre connu pour la représentation des connaissances et le raisonnement peut être approprié pour le traitement du contexte. Les ontologies possèdent une forme générique, formelle et explicite pour « capturer et préciser les connaissances d'un domaine avec une sémantique intrinsèque à travers des axiomes consensuels et une sémantique formelle des contraintes » [Ye *et al.*, 2007a]. Elles fournissent un moyen formel pour représenter les données du capteur, du contexte et des situations dans une terminologie bien structurée, ce qui les rend compréhensibles, partageables et réutilisables par les humains et les machines [Chen *et al.*, 2009].

Les ontologies fournissent un ensemble de primitives de modélisation pour définir les classes, les individus, les propriétés des attributs et les propriétés d'objets (c.-à-d. les relations entre les objets). Par exemple, la propriété « est-un » est l'une des propriétés les plus utiles dans la modélisation de deux concepts du domaine : « Salle à manger » est-un « espace d'alimentation et d'activité » [Yamada *et al.*, 2007]. La figure ci-dessous présente un exemple de modélisation du contexte.

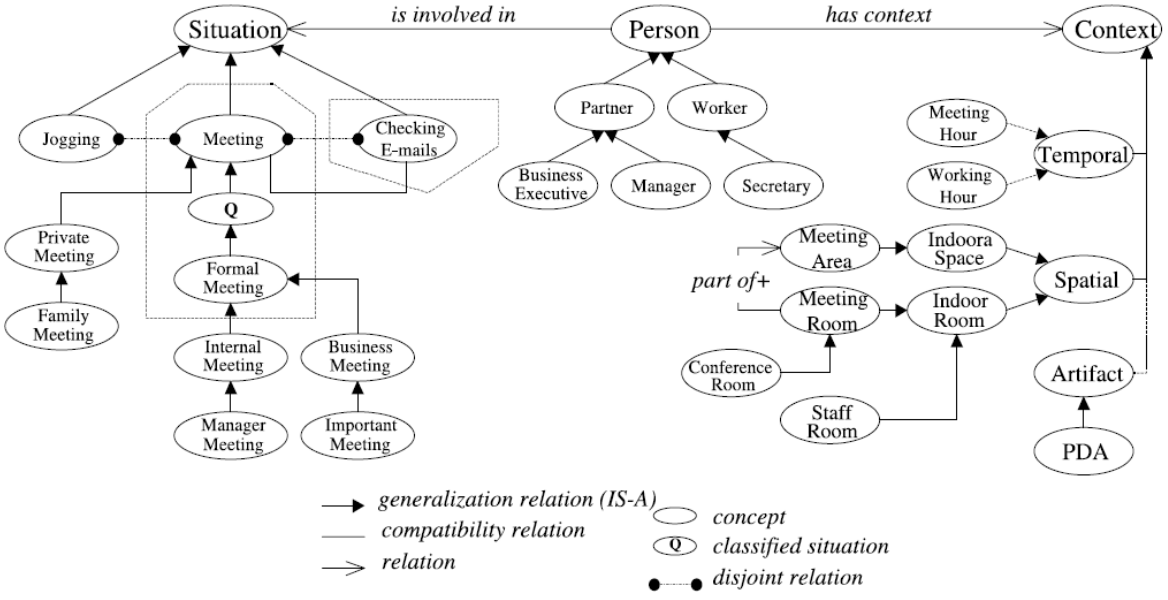


Figure 2.9 — Exemple de modélisation du contexte [Ye *et al.*, 2012].

La plupart des modèles basés sur les ontologies utilisent le langage OWL-DL [Horrocks *et al.*, 2003] ou certaines de ses extensions, car il est devenu un standard de facto dans divers domaines d'application et il est soutenu par un certain nombre de services de raisonnement. Par le biais du langage OWL-DL, il est possible de modéliser un domaine particulier en définissant les classes, les individus, les caractéristiques des individus (propriétés de type de données), et les relations entre les individus (propriétés de l'objet). Les descriptions complexes des classes et des propriétés peuvent être construites par la composition des descriptions élémentaires via des opérateurs spécifiques prévus par le langage.

Des définitions plus complexes peuvent être obtenues en utilisant des opérateurs tels que les restrictions. Par conséquent, les données contextuelles complexes peuvent être induites par des modèles de tâches de raisonnement sur la base de données brutes directement acquises par des capteurs et représentées par des expressions structurées OWL-DL. Ces données comprennent généralement des informations concernant l'environnement socioculturel des utilisateurs ainsi

que leurs préférences. Les ontologies fournissent un formalisme expressif pour représenter les données complexes du contexte, et sont bien adaptées pour le partage des connaissances car ils fournissent une spécification formelle de la sémantique des données du contexte. À cette fin, diverses ontologies OWL ont été proposées pour représenter les descriptions communes des données contextuelles. Parmi les propositions les plus importantes nous pouvons citer : Soupa [Chen, 2004] une ontologie dédiée à la modélisation du contexte dans un environnement ubiquitaire, et Conon [Zhang *et al.*, 2005] une ontologie dédiée aux environnements domestiques intelligents.

Les ontologies ont été également utilisées dans la modélisation de concepts complexes. Nevatia *et al.* introduisent un langage formel pour décrire une ontologie de l'événement, appelé le langage de représentation vidéo de l'événement, VERL. Cette représentation d'événements vidéo vise à soutenir les tâches comme la surveillance vidéo, la navigation vidéo, et l'indexation vidéo basée sur le contenu [Nevatia *et al.*, 2004]. Yau *et al.* ont développé une ontologie de modélisation d'une situation, qui intègre une sémantique riche (y compris les contraintes de temps) dans l'écriture des spécifications logiques de situations. Au-delà de l'enrichissement sémantique au niveau conceptuel, les ontologies peuvent aussi servir comme une base efficace pour le développement de la connaissance centrée logiciel [Yau et Liu, 2006]. Chen *et al.* proposent une architecture sémantique conceptuelle en couches pour les maisons intelligentes. La nouveauté de cette architecture réside dans le fait qu'il existe une couche sémantique et une couche de service entre les données et les couches applicatives. La couche sémantique réalise l'interopérabilité des données et la compréhensibilité par la machine en utilisant des ontologies pour fournir une vue homogène sur des données hétérogènes, tandis que la couche de service offre la capacité de l'interopérabilité et de l'automatisation de haut niveau en exploitant la sémantique et les connaissances descriptives des ontologies [Chen, 2004].

Les modèles basés sur les ontologies présentent des avantages en termes d'hétérogénéité et d'interopérabilité par rapport aux autres approches [De Oliveira *et al.*, 2013]. En ce qui concerne la notion d'utilisabilité, plusieurs outils conviviaux graphiques existent tel que Protégé, offrent la possibilité de concevoir des modèles ontologiques. Cependant, il y a très peu de soutien pour la modélisation des aspects temporels. En outre, en dépit de sa capacité à exprimer les relations et les dépendances entre les données du contexte, le modèle ontologique représente une solution satisfaisante pour un large éventail d'applications sensibles au contexte. Cependant, il existe quelques modèles basés sur les ontologies qui montrent que les opérateurs fournis par OWL-DL sont parfois insuffisants pour définir une description complexe du contexte [Agostini *et al.*, 2006]. Ce problème est dû au fait que les constructeurs inclus dans le langage OWL-DL ont été choisis afin de garantir des procédures de raisonnement décidables. Pour cette raison, OWL-DL ne comprend pas les constructeurs très expressifs qui seraient

utiles pour la modélisation des domaines complexes, tels que les activités des utilisateurs, la composition des relations ou ceux qui restreignent l'appartenance à une classe.

Par conséquent, la possibilité d'augmenter l'expressivité des langages ontologiques par une extension des règles a été récemment étudiée par la communauté du Web sémantique, et amenée à la définition de langages logiques tels que SWRL [Horrocks *et al.*, 2004b].

En plus des restrictions mentionnées ci-dessus, le raisonnement ontologique avec OWL-DL pose également des problèmes de performance. En effet, une solution naturelle pour dériver des données contextuelles complexes à travers le raisonnement ontologique est d'effectuer la réalisation des intérêts de l'utilisateur afin de trouver la classe la plus adéquate. Par conséquent, l'exécution du raisonnement ontologique pose des problèmes d'évolutivité, en particulier lorsque l'ontologie est peuplée par un grand nombre d'individus. Afin d'améliorer l'efficacité du raisonnement avec OWL-DL, diverses optimisations basées sur l'utilisation des techniques de bases de données relationnelles ont été récemment proposées. Une proposition bien connue dans ce sens est le système Instance Store [Horrocks *et al.*, 2004a]. Toutefois, au moment de l'écriture, Instance Store comporte certaines limites. En effet, il ne permet pas l'instanciation des relations entre les individus.

2.2.9 Les modèles graphiques

Plusieurs approches graphiques de modélisation des services sensibles au contexte ont été proposées afin de mettre en œuvre un modèle efficace qui spécifie l'acquisition et la gestion des différents composants du contexte. Dans cette section, nous allons nous intéresser à trois d'entre eux, à savoir UML, ORM et les réseaux de Petri (RdP).

2.2.9.1 UML/ORM

Le langage de modélisation le plus connu est Unified Modeling Language (UML). Grâce à sa structure générique, UML est approprié pour la modélisation du contexte. Ceci est illustré par (*i*) Bauer dans [Bauer *et al.*, 2003], où les aspects contextuels pertinents à la gestion du trafic aérien sont modélisés comme des extensions UML et par (*ii*) Kubicki et ses collègues qui exploitent le diagramme de classe d'UML pour la modélisation du contexte afin de décrire l'interaction sur les tables interactives avec les objets tangibles [Kubicki *et al.*, 2013]. Baas et ses collègues utilisent également la même technique pour la modélisation du contexte dans des situations de saisie de données médicales (figure 2.10) [Baas *et al.*, 2014a][Baas *et al.*, 2014b].

Nous pouvons citer comme autre méthode de conception la modélisation objet-rôle (ORM).

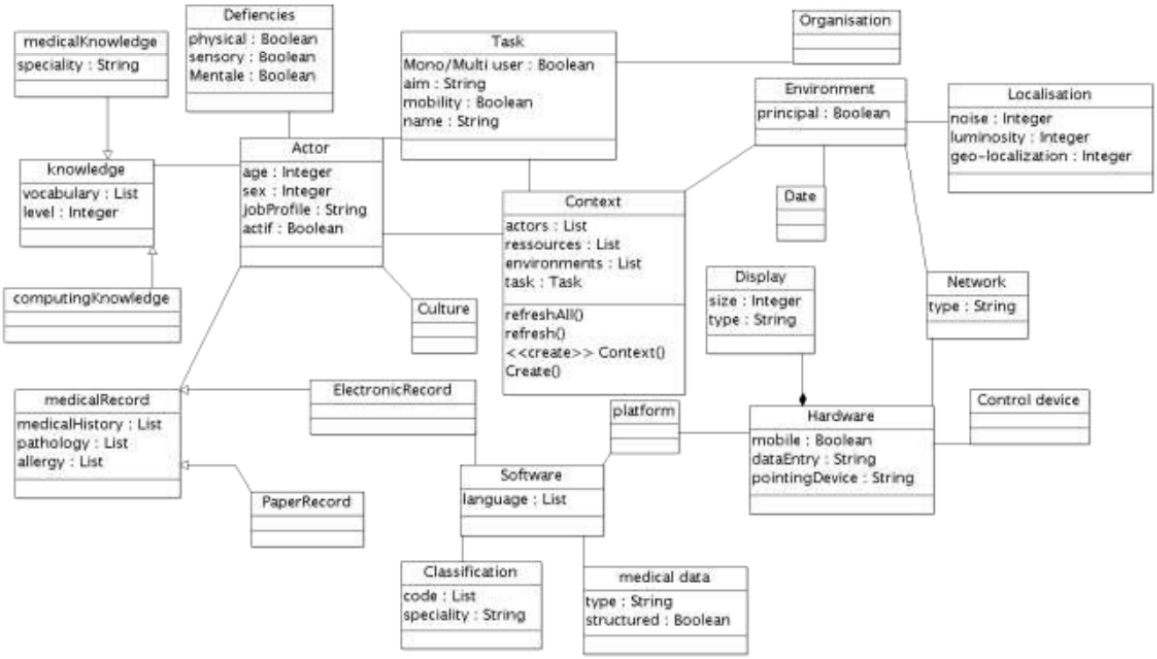


Figure 2.10 — Exemple de modélisation du contexte avec le diagramme de classe [Baas *et al.*, 2014b].

En ORM, le concept de modélisation de base est le fait. Henricksen a étendu ORM pour classer plusieurs types d'information selon leur persistance et leurs sources, de manière statique ou dynamique. La dernière extension d'ORM faite par Henricksen couvre l'aspect temporel des informations du contexte et les dépendances entre les informations [Henricksen *et al.*, 2003].

2.2.9.2 Les réseaux de Petri

Les approches basées sur les RdP ont été employées en raison de la sémantique formelle, de la description du système basé sur l'état et des techniques d'analyses abondantes qu'offre ce formalisme. Récemment, des approches de modélisation des systèmes sensibles au contexte basées sur les RdP ont été proposées par plusieurs chercheurs et elles ont été reconnues comme des modèles prometteurs et efficaces pour la représentation du contexte [Han et Youn, 2012] [Sun *et al.*, 2010]. Avant de présenter les approches de modélisation basées sur les RdP, nous allons les définir.

2.2.9.2.1 Définition Un réseau de Petri est une méthode de modélisation graphique utilisé pour la représentation des systèmes dynamiques, discrets ayant des entrées, des sor-

ties et des états. La sémantique formelle à caractère graphique, l'expressivité, l'analyse des propriétés des RdP permettent une modélisation et une analyse efficace des processus dynamiques [Murata, 1989]. Un RdP est un graphe orienté, bipartie, défini comme un tuple <C, P, T, F, B> :

- C : représente la structure du réseau ;
- P : l'ensemble de place ;
- T : l'ensemble de transition ;
- F : P * T $\longrightarrow$ N, la fonction d'incidence ascendante ;
- B : P * T $\longrightarrow$ N, la fonction d'incidence descendante.

Sun *et al.* représentent les données du contexte (selon le degré d'abstraction et de la sémantique) en trois couches : capteur, atomes du contexte et la couche de situation contextuelle. La couche capteur représente la source des données contextuelles. La couche des atomes du contexte récupère les données de la couche capteur. Elle joue donc le rôle de médiateur entre le monde physique et le monde sémantique. Elle utilise les ontologies pour définir les attributs et les classes. Cependant, elle est incapable de représenter des connaissances complexes. Par conséquent, une couche de situation contextuelle est construite sur le sommet de la couche d'atome afin de fusionner les atomes individuels du contexte. Les places représentent les situations du contexte et les transitions indiquent une modification de la situation et représentent la corrélation entre les situations à savoir la concurrence, le conflit et la causalité [Sun *et al.*, 2010].

Cette approche fournit une représentation intuitive des relations entre les informations du contexte. L'analyse de la synchronisation permet d'évaluer la tolérance du retard du système. Cependant, cette technique ne représente pas les contraintes de temps. Par conséquent, il devient difficile de manipuler l'apparition séquentielle des contextes avec des contraintes temporelles.

Il existe plusieurs types de réseaux de Petri dont les RdP colorés capables de représenter les informations complexes dans les jetons, et les RdP synchronisés qui prennent en charge la notion du temps pour exprimer les retards. Toutes ces extensions de RdP ont été utilisées à des fins diverses pour la modélisation du contexte.

2.2.9.2.2 Les RdP synchronisés Reignier *et al.* introduisent une approche pour la représentation du contexte, du comportement de l'application et l'évaluation des disponibilités en utilisant les RdP synchronisés. Ils ont employé le concept de situation sur les places et les évènements sur les transitions. Un événement indique un changement d'état de l'activité, du rôle ou d'une relation décrivant la situation. Le contrôle sur la transition est géré avec

l'utilisation des événements de transition des RdP synchronisés [Reignier *et al.*, 2007].

Un RdP synchronisé fournit un modèle composite sur l'utilisateur et le comportement de l'application. Les entrées de l'application sensible au contexte sont les événements décrivant le changement de l'état, tandis que les sorties sont les situations actuelles et les actions associées. Cette approche permet la manipulation des informations du contexte dans les conditions concrètes de situations générales. Cependant, si une des informations du contexte est ajoutée, les dépendances de la situation doivent être reprécisées. Cette faiblesse est corrigée par les RdP colorés en fournissant une définition plus souple des événements.

2.2.9.2.3 Les RdP colorés Silva *et al.* ont proposé de combiner des outils de modélisation 3D avec les RdP colorés pour modéliser les environnements 3D. Dans ce modèle, la place est utilisée pour indiquer l'état actuel de l'utilisateur et les composants tels que les PDA et l'écran. Le jeton identifie l'utilisateur et sa position, et la transition est utilisée pour déduire le mouvement de l'utilisateur et le comportement des composants. La combinaison de l'outil de modélisation 3D avec les RdPC est efficace dans la modélisation des systèmes sensible au contexte car elle permet la simulation et l'évaluation de l'application. Cependant, ce modèle ne prend pas en charge les contraintes relationnelles et les contraintes de temps nécessaires à la modélisation dynamique du contexte [Silva *et al.*, 2009].

Kwon a étendu les RdPC pour décrire les systèmes sensibles au contexte. Il décompose le système en plusieurs sous-systèmes significatifs. Il sépare les contextes à partir du modèle global pour réaliser des modèles de contexte indépendants (figure 2.11). En outre, le modèle peut être converti en code java. Les systèmes sensibles au contexte sont classés comme des routines de systèmes récurrents ou des routines de systèmes rares. Les systèmes de routines peuvent être automatisés ou programmés. Une méthodologie de modélisation appelée « Amended CPN » a été proposée pour représenter et analyser les systèmes sensibles au contexte. « Amended CPN » se compose de plusieurs RdPC [Kwon, 2004].

Un RdPC modifié possède les propriétés avancées suivantes :

- Les prestataires du contexte sont ajoutés au réseau du contexte pour spécifier explicitement la source du contexte ;
- Le réseau du contexte et le modèle sont liés par des arcs. Ces arcs seront modifiés à chaque changement du contexte.

Les approches de modélisation du contexte via les RdP diffèrent selon le but de la modélisation. Certains auteurs s'intéressent principalement à la modélisation du comportement de l'application sensible au contexte, d'autres tentent de résoudre le problème du temps et des ressources dans les applications.

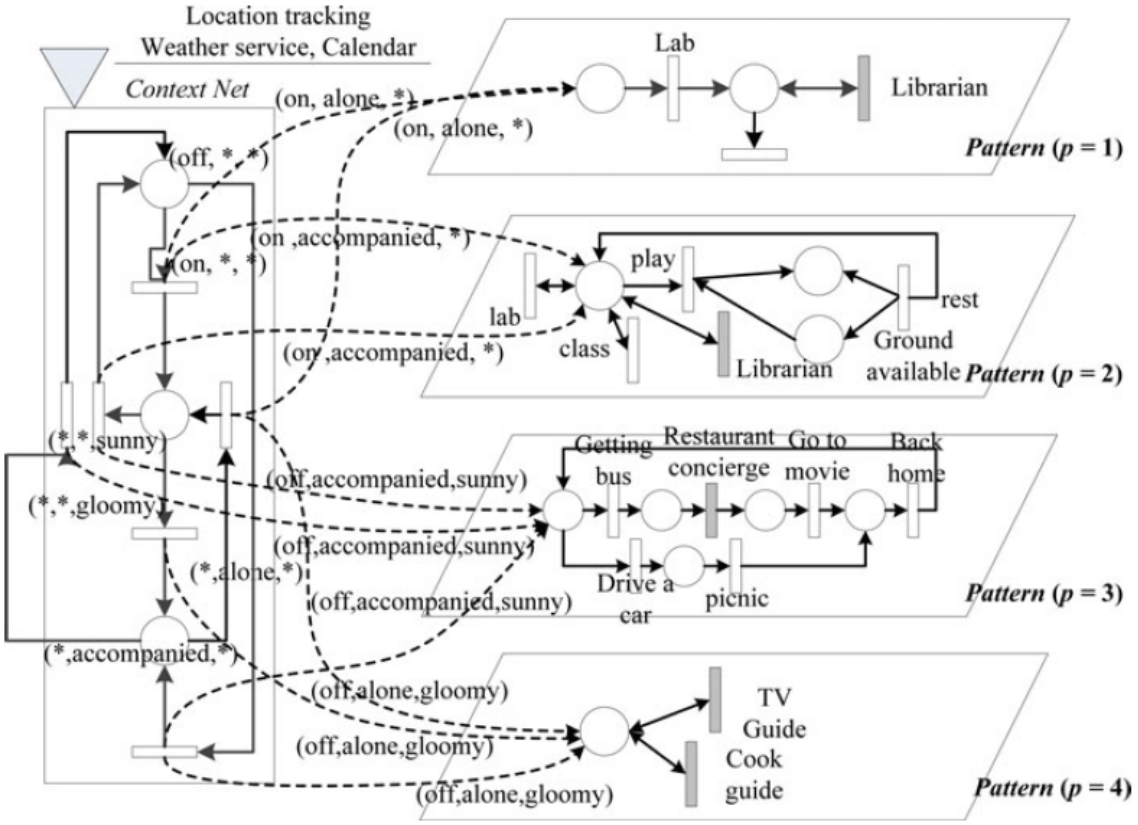


Figure 2.11 — Exemple de modélisation du contexte avec RdPC étendu [Han et Youn, 2012].

2.2.10 Les modèles hybrides

Plusieurs auteurs affirment qu'aucune des approches citées ci-dessus ne satisfait pleinement les exigences des applications sensibles au contexte.

Henricksen *et al.* proposent une approche hybride de la modélisation du contexte, en combinant les approches basées sur les ontologies avec les modèles basés sur les faits CML (langage de modélisation du contexte)[Bettini *et al.*, 2010]. L'objectif est de combiner les avantages du modèle CML (le traitement des informations du contexte ambigu et imparfait) avec le soutien de l'interopérabilité et des divers types de raisonnement fournis par les modèles ontologiques. Cette approche est basée sur une cartographie de modélisation CML pour OWL-DL. Il est intéressant de noter qu'en raison de certaines limites d'expressivité d'OWL-DL, une cartographie complète entre CML et OWL-DL ne peut pas être obtenue. En ce qui concerne les questions d'interopérabilité, les avantages procurés par une représentation ontologique du

modèle de contexte sont clairement reconnaissables [Henricksen *et al.*, 2004].

Agostini *et al.* proposent une combinaison des approches basées sur les langages de balisage avec les modèles basés sur les ontologies. En effet, les auteurs intègrent la représentation des données du contexte (profils CC/PP) avec les classes et les relations OWL-DL [Agostini *et al.*, 2009] .

Les approches hybrides peuvent être également étendues pour soutenir la modélisation des données hiérarchiques du contexte. L'une des premières approches a été de combiner les modèles spatiales et ontologiques. Le modèle présenté ici est destiné à fournir une solution plus complète, à la fois en termes d'intégration des différentes formes de raisonnement, et en termes d'expressivité. Le modèle proposé inclut un formalisme de représentation pour représenter les données directement recueillies par les capteurs. Afin de soutenir les exigences d'évolutivité des services informatiques omniprésents, ce formalisme de représentation devrait permettre l'exécution de techniques de raisonnement efficaces pour obtenir des données du contexte de haut niveau.

Même si le projet de modèle hiérarchique hybride détermine à première vue des avantages évidents, il faut savoir que l'intégration de diverses techniques de raisonnement pose encore des problèmes, par exemple, comment intégrer la sémantique ouverte des ontologies avec la sémantique fermée de la programmation logique.

Afin de bien choisir le modèle le plus approprié pour la modélisation du contexte, plusieurs critères de comparaison ont été mis en place. La section qui suit présente les exigences d'un environnement ubiquitaire qui nous ont permis de choisir notre modèle.

2.3 Les exigences d'un système sensible au contexte dans un environnement ubiquitaire

Afin de mener à bien notre étude comparative, des critères de comparaison ont été fixés sur la base : *(i)* des exigences d'un environnement ubiquitaire qui existe dans la littérature (essentiellement ceux fixées par [Bettini *et al.*, 2010] et [Strang et Linnhoff−Popien, 2004], pour les modèles contextuels) et *(ii)* des besoins indispensables, selon nous, pour la modélisation des domaines critiques. Dix critères sont considérés :

L'hétérogénéité et la mobilité : Les modèles de description des informations du contexte doivent traiter une grande variété de sources d'information qui diffèrent dans leurs taux de mise à jour et leurs niveaux de sémantique. Les capteurs doivent observer certains états du

monde physique et fournir rapidement et en temps quasi réel l'accès à ces informations, tout en fournissant des données plutôt brutes qui doivent être interprétées avant d'être utilisées par les applications. Les données du contexte peuvent également être obtenues à partir des informations du contexte existant. Un modèle du contexte devrait être en mesure d'exprimer ces différents types de données contextuelles. De nombreuses applications sont mobiles et sensibles au contexte. Cela ajoute au problème de l'hétérogénéité, le problème de l'approvisionnement de l'information du contexte aux différentes applications. En outre, l'emplacement et la disposition spatiale des informations du contexte jouent un rôle important dans un système ubiquitaire.

Validation partielle : Il est préférable de pouvoir valider partiellement les connaissances contextuelles d'un système ubiquitaire vu la complexité des relations qui existent entre les données du contexte. Dans un domaine critique, à savoir le domaine médical, une erreur au niveau de la modélisation du contexte n'est pas permise. A cet effet, il est préférable de décomposer les données contextuelles en des granularités plus fines afin de faciliter leurs vérifications.

La qualité de l'information : La qualité et le flux d'informations fournie par un capteur varie au fil du temps. Ainsi, un modèle de contexte approprié dans un système présent dans un environnement ubiquitaire devrait intrinsèquement soutenir la qualité et la richesse des données du contexte.

L'incomplétude : En raison de leurs natures dynamiques et hétérogènes, les informations contextuelles peuvent être de qualités variables et incorrectes. La plupart des capteurs disposent d'une imprécision inhérente. En outre, les informations du contexte peuvent être incomplètes ou contradictoires. Ainsi, une approche de modélisation du contexte doit comporter la modélisation de la qualité des informations du contexte pour soutenir un raisonnement correct.

La convivialité des formalismes de modélisation : Les modèles de description des informations du contexte sont créés par les concepteurs des applications sensibles au contexte et sont également utilisés par les systèmes de gestion du contexte afin de manipuler les informations du contexte. Par conséquent, les caractéristiques importantes des formalismes de modélisation sont la facilité avec laquelle les concepteurs peuvent traduire les concepts du monde réel en des concepts de modélisation et de les utiliser lors de l'exécution et de la manipulation des informations du contexte.

Application aux environnements existants : Du point de vue mise en œuvre, il est important qu'un modèle de contexte soit applicable au sein de l'infrastructure existante des environnements informatiques omniprésents.

Les dépendances : Il existe diverses relations entre les informations du contexte qui ont été capturées afin d'assurer un comportement correct des applications. Les informations du contexte dépendent les unes des autres. En effet, un changement de la valeur d'une propriété peut affecter les valeurs des autres propriétés.

Le raisonnement : Les applications sensibles au contexte utilisent les informations du contexte pour évaluer s'il y a un changement au niveau de l'utilisateur et/ou au niveau du contexte de l'environnement informatique afin de prendre une décision (si une adaptation à ce changement est nécessaire ou pas). Il est donc important que les techniques de modélisation du contexte soient en mesure de soutenir la vérification de la cohérence du modèle et des techniques de raisonnement sur le contexte. Ces dernières peuvent être utilisées pour dériver de nouveaux faits à partir de faits contextuels existants.

L'utilisation d'un langage formel : La méthode de modélisation choisie doit avoir une sémantique formelle. Les méthodes formelles comprennent une spécification basée sur les mathématiques afin de spécifier les propriétés désirées du système omniprésent dans un domaine critique. Une spécification formelle est exprimée dans un langage dont la syntaxe et la sémantique sont officiellement définies. Les méthodes formelles sont avantageuses pour faciliter la représentation des données. En outre, le développement d'une spécification formelle donne un aperçu et une compréhension des exigences du système, de l'utilisateur et de la conception de l'interface.

La vérification formelle : Le modèle doit être vérifié par des règles ou des propriétés mathématiques. La vérification des propriétés importantes de l'interface dans un domaine critique, doit être effectuée dès la phase de modélisation du contexte pour garantir une spécification fiable des interfaces. La vérification formelle est indispensable pour prouver l'exactitude des systèmes pervasifs ; ce qui assure la validation des interfaces utilisateur avant leur mise en œuvre.

Nous pouvons donc estimer que les exigences mentionnées ci-dessus sont extrêmement importantes dans la modélisation d'un système sensible au contexte dans un environnement critique. Il est primordial de pouvoir représenter de manière efficace les relations et les dépendances entre les entités d'un contexte via des méthodologies claires et bien formalisées.

2.4 Discussion

Plusieurs approches de modélisation du contexte ont été proposées au fil des années. A ce stade, il devient nécessaire de pouvoir choisir la technique de modélisation qui convient

le mieux à un environnement ubiquitaire. En outre, cette section présente une étude comparative entre les différents modèles vus dans la première section basée sur les exigences d'un environnement ubiquitaire.

Les modèles clé/valeur présentent une grande faiblesse sur les cinq premières exigences et sont inefficaces dans la description des informations contextuelles complexes. La composition distribuée et la manipulation de l'incomplétude n'est possible qu'au niveau de l'instance. La validation partielle d'une tâche est très difficile. La simplicité de cette approche est un avantage sur la gestion des risques d'erreur, mais c'est un inconvénient si la qualité de l'information ou l'ambiguïté doivent être traitées. Le seul point positif de ce type d'approche porte uniquement sur l'applicabilité aux environnements ubiquitaires existants.

Dans les modèles de marquage, il existe généralement une définition de schéma et un ensemble d'outils de validation qui peuvent être utilisés pour la vérification du type d'information. Par conséquent, la validation partielle est assurée. Il est à noter qu'Indulska *et al.* ont obtenu de mauvais résultats avec CC/PP et UAProf en raison des contraintes imposées par la sérialisation XML et la représentation RDF [Indulska *et al.*, 2003].

Les approches orientées objet assurent la composition distribuée et la validation partielle. Les nouveaux types d'informations contextuelles ainsi que les nouvelles instances peuvent être facilement traitées. L'approche TEA garantit la qualité de l'information. Ceci est primordial pour gérer de manière correcte l'incomplétude et l'ambiguïté. Cependant, l'encapsulation des données peut être un inconvénient dans la déduction des propriétés importantes de l'environnement et des relations qui existent entre les différentes entités du contexte.

Les modèles basés sur la logique peuvent assurer une composition distribuée, mais la validation partielle est difficile à maintenir. Leur niveau de formalité est extrêmement élevé. Cependant, sans validation partielle des connaissances contextuelles, le modèle devient très sensible aux erreurs. Aucun des modèles logiques ne semble répondre aux exigences de la qualité de l'information, de l'incomplétude et de l'ambiguïté.

Les techniques supervisées basées sur l'apprentissage à savoir HMM, les réseaux bayésiens, etc., nécessitent l'intervention d'un être humain pour l'étiquetage des données du capteur aux différentes situations. Cependant, si le nombre de situations est élevé, l'étiquetage manuel des données de formation pourrait être un fardeau pour les développeurs impliqués dans la collecte des données. Par conséquent, ces techniques peuvent avoir des limites dans l'extensibilité, l'applicabilité et l'adaptabilité. Pour résoudre ce problème, les chercheurs ont eu recours à des approches d'apprentissage non supervisées tel que l'arbre-suffixe (section 2.2.7.4) qui peuvent extraire les données à partir des observations du capteur.

Les modèles spatiaux temporels fournissent des indices importants sur les types de situa-

tions actuelles. Il existe trois questions de recherche sur les séquences temporelles à savoir : *(1)* comment extraire les motifs séquentiels ; *(2)* comment utiliser les fonctions temporelles, et *(3)* comment détecter la frontière où les situations changent en localisant chaque paire d'événements qui sont temporellement adjacente. La détection d'une frontière où un changement de situation a eu lieu est également un enjeu de recherche important.

En raison des similitudes entre les approches de modélisation basées sur les ontologies (concepts, faits) et les objets (classes, instances), les approches basées sur les ontologies sont également fortes en ce qui concerne l'exigence de la composition distribuée. La validation partielle est possible, et un ensemble complet d'outils de validation existent. L'approche CoBrA semble ne pas satisfaire la qualité d'information et l'ambiguité. L'incomplétude ainsi que l'applicabilité aux environnements existants sont couvertes par toutes les approches.

Les points forts des modèles graphiques résident dans la représentation du contexte et la vérification formelle du modèle. Ils sont principalement utilisés pour décrire la structure des connaissances contextuelles et d'en déduire les relations qui existent entre elles. Les techniques de modélisation basées sur les RdP répondent aux exigences. En effet, les relations entre les contextes peuvent être facilement comprises en raison de la nature graphique et formelle des RdP. L'exactitude et la fiabilité des RdP peuvent être facilement vérifiées. Cependant, elles présentent un inconvénient majeur au niveau de la dynamique de l'environnement ubiquitaire. En effet, si un nouveau contexte est ajouté ou des modifications sont apportées au système sensible au contexte, le modèle existant doit être entièrement réévalué. Il est donc difficile d'évaluer les mises à jour de l'information du contexte de manière dynamique. Toutes les approches basées sur les RdP soutiennent les propriétés des relations et des dépendances. L'identification du contexte a rarement été traitée dans les modèles précédents. Cette exigence est importante dans les systèmes sensibles au contexte. Une méthodologie de modélisation plus élaborée est donc nécessaire pour couvrir toutes ces questions relatives aux systèmes sensibles au contexte. Le tableau 2.1 résume notre discussion sur la pertinence des différentes approches de modélisation du contexte pour l'informatique ubiquitaire. Pour plus de clarté, nous avons utilisé un codage numérique à savoir : 0 : mauvais, 1 : bien et 2 : très bien.

En conclusion et d'après cette étude, les approches de modélisation du contexte basées sur les ontologies et sur les réseaux de Petri répondent le mieux aux exigences d'un environnement ubiquitaire. En effet, ces deux approches cumulent respectivement 18 et 17 points. L'écart qui existe entre ces deux approches vient principalement du critère de la vérification formelle du modèle. En effet, les ontologies sont utilisées principalement pour le raisonnement du contexte et il n'existe, à nos connaissances, aucun outil mathématique capable de vérifier l'exactitude et la fiabilité de l'ontologie.

Tableau 2.1 — Tableau comparatif des différentes approches de modélisation

	Clé/valeur	Marquage	Orienté objet	Logique	Probabiliste	Graphique (RdP)	ontologie
Hétérogénéité et mobilité	0	0	1	2	0	1	2
Validation partielle	0	2	1	0	1	2	2
Qualité de l'information	0	0	1	0	0	2	2
Incomplétude	0	0	1	0	1	2	2
Convivialité des formalismes	0	1	1	1	1	2	2
Applicabilité aux environnements existants	1	2	0	1	0	1	1
Relation et dépendance	0	0	0	0	0	2	2
raisonnement	0	1	0	1	1	2	2
Utilisation langage formel	0	0	0	0	0	2	2
Vérification formelle	0	0	0	0	2	2	0
Total	1	6	5	5	6	18	17

Rappelons que notre objectif de recherche est de savoir comment décrire le contexte et modéliser la tâche de l'utilisateur dans un environnement ubiquitaire critique pour aboutir à une interface valide. A cette fin, les approches fondées sur les RdP présentent toutes les caractéristiques requises. L'utilisation des réseaux de Petri pour la modélisation vise à préparer le terrain pour la vérification et la validation des critères de l'interface. La modélisation des systèmes pervasifs dans un domaine critique nécessite une validation rigoureuse de l'interface afin de présenter la meilleure solution pour faire face à une situation d'urgence. Cela permet d'économiser une quantité considérable de temps dans le cycle de développement, en particulier pendant la phase de validation. En effet, les réseaux de Petri ont une définition formelle ; ils sont capables d'exprimer des aspects tels que le temps et la concurrence. Ils possèdent de nombreuses techniques pour une vérification automatique des propriétés. Ils fournissent également une représentation graphique très vaste.

2.5 Conclusion

Nous avons exposé dans ce chapitre un état de l'art portant sur les différentes approches de modélisation du contexte à savoir le modèle clé/valeur, les modèles de marquages, les modèles orientés objets, les modèles basés sur la logique, les modèles probabiliste, les modèles tirés de l'intelligence artificielle, les ontologies, les modèles graphiques et les modèles hybrides.

Nous avons, par la suite, défini les exigences d'un système sensible au contexte dans un domaine critique. Nous nous sommes principalement focalisés sur les critères de la validité partielle du modèle, de l'utilisation d'outils formels et de la vérification formelle du modèle.

Finalement, en fonction de nos critères de comparaison, nous avons choisi le modèle le plus approprié pour la modélisation du contexte, à savoir les réseaux de Petri. Ce choix a été régi par un critère très important qui est la vérification formelle du modèle de contexte.

Nous avons signalé au cours de notre étude critique que les RdP avaient une petite défaillance au niveau de la modélisation dynamique du système ubiquitaire. Le chapitre suivant présentera l'approche adoptée pour répondre à cette problématique.

chapitre 3

Approche formelle pour la spécification et la génération d'interfaces sensibles au contexte dans un environnement ubiquitaire

3.1 Introduction

L'objectif de ce travail de recherche est l'élaboration d'une approche globale pour la spécification et la génération des interfaces sensibles au contexte dans un domaine critique. Notre méthodologie s'appuie sur l'utilisation d'outils formels pour couvrir de façon progressive et cohérente l'ensemble des étapes, depuis l'analyse du SHM et la modélisation de l'interaction Homme-Machine jusqu'à la génération des différentes vues graphiques. Nous commençons dans ce chapitre par exposer la structure générale de l'approche préconisée. Nous présentons, par la suite, les détails des différentes étapes de notre approche.

3.2 Approche proposée : vue d'ensemble

Comme nous l'avons expliqué dans le premier chapitre, les chercheurs dans le domaine de l'adaptation au contexte n'ont pas encore abouti à une définition à la fois générique et prag-

matique de la notion du contexte, et plus précisément des paramètres constituant le contexte. Toutefois, après avoir étudié les principales définitions proposées dans la littérature, nous avons remarqué que la majorité tournait autour des définitions proposées par [Dey *et al.*, 1999] et [Calvary *et al.*, 2004]. Ces travaux nous ont conduit à définir le contexte comme étant le triplet : <*utilisateur, plateforme, environnement*>.

Ces deux définitions permettent d'éclaircir un peu plus la notion de contexte dans les Systèmes Homme-Machine et d'identifier d'une manière nette et précise les caractéristiques contextuelles qui peuvent être prises en considération pour supporter efficacement les actions de l'utilisateur. Les questions qui se posent à ce niveau *(i)* est comment modéliser les informations du contexte et *(ii)* comment fournir une architecture adéquate pour la génération d'interfaces sensibles au contexte ?

Rappelons que l'objectif global de nos travaux de recherche, est d'aboutir à une interface utilisateur adaptable au contexte courant de l'utilisateur. Pour ce faire, il faut tout d'abord procéder à une analyse du Système Homme-Machine pour ensuite le modéliser. La modélisation du système Homme-Machine doit prendre en considération la modélisation du contexte. En effet, la modélisation du contexte dans lequel évolue l'utilisateur, représente la pièce maitresse de notre approche puisque la tâche de l'utilisateur dépendra de son contexte actuel.

L'étude critique de l'état de l'art relative aux différentes architectures de systèmes sensibles au contexte, établie dans le premier chapitre, nous a permis de définir notre propre architecture représentée par la Figure 3.1 [Riahi et Moussa, 2015][Riahi et Moussa, 2014].

La première étape de notre approche concerne la détection des informations contextuelles. Une fois que les données sont captées à partir de la couche capteur, elles seront modélisées et décomposées en un modèle utilisateur, un modèle de la plateforme et un modèle de l'environnement. Cette décomposition est basée sur la définition du contexte proposée par [Calvary *et al.*, 2004]. En parallèle, une analyse du Système Homme-Machine est nécessaire.

La seconde étape consiste à analyser la tâche de l'utilisateur en précisant l'enchaînement des actions qu'il doit accomplir. Cette analyse conduira à la modélisation de la tâche et à l'identification des Besoins Informationnels de l'Opérateur (BIO).

La troisième étape assure la modélisation du contexte ainsi que de la tâche. Celle-ci sera réalisée via la base de données des RdP qui contient des structures et des compositions élémentaires de RdP. Une modélisation formelle de l'Interaction Homme-Machine, moyennant les réseaux de Petri Interprétés, permet, par la suite, de situer en chaque point d'interaction, l'ensemble des BIO nécessaires.

La quatrième étape assure la détection de la situation courante de l'utilisateur. Cette

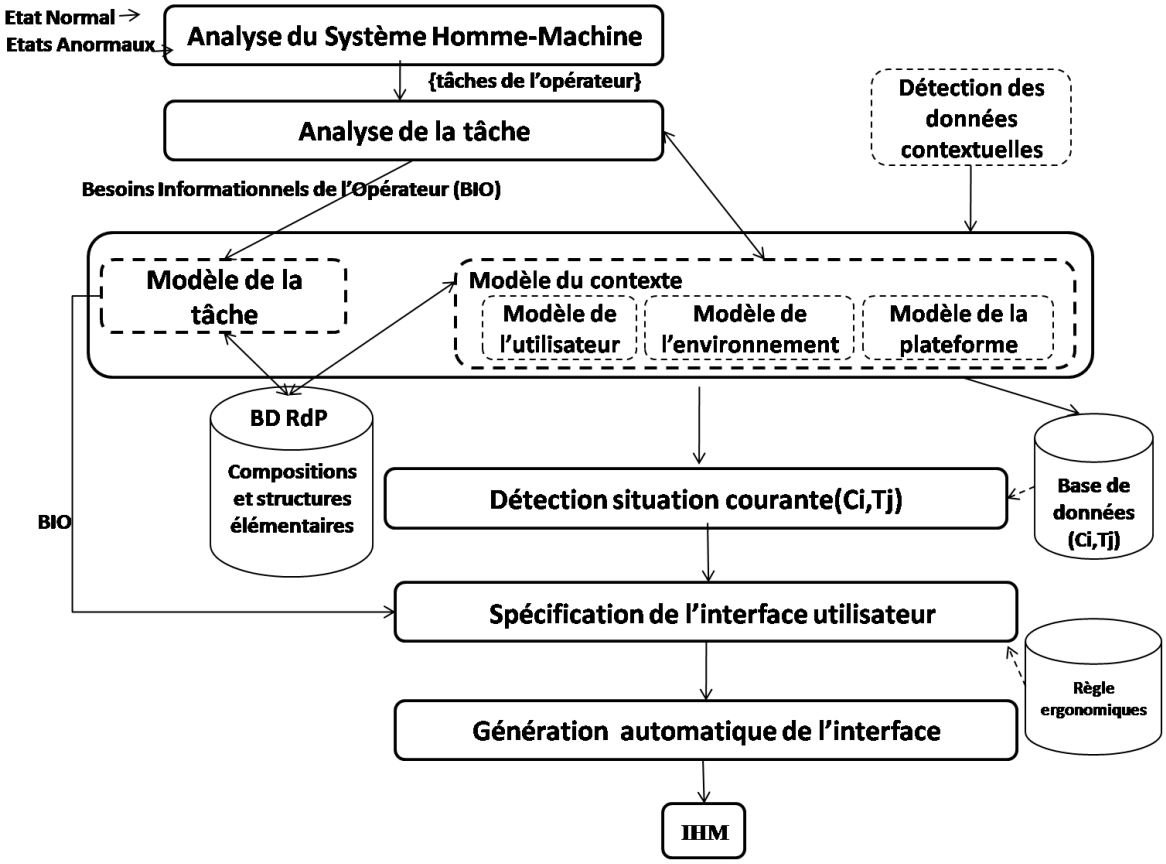


Figure 3.1 — Approche proposée.

dernière consiste à identifier le couple (Contexte, Tâche) à un moment donné. L'idée d'associer une tâche utilisateur à son contexte d'usage, a été constatée à la lumière de notre étude critique. En effet, aucune des approches citées dans le premier chapitre, ne prend pleinement en considération l'activité de l'utilisateur ou de la tâche qu'il doit accomplir. Or, la tâche de l'utilisateur dépend pleinement de son contexte d'utilisation à un instant donnée. Par exemple, prenons le cas d'un citadin qui veut déjeuner et qui effectue une recherche sur son téléphone mobile. L'interface devra alors lui afficher des adresses de restaurants se trouvant en ville. Si à un moment donné, le temps devient pluvieux, alors son interface devra « éliminer » les adresses de restaurants en « plein air ».

Tous les couples *« contexte, tâche »* seront donc stockés auparavant dans une base de données. A un moment donné, l'état des différents modèles du contexte déterminera l'état actuel Ci de l'utilisateur. Pour connaître la tâche Ti adéquate à Ci, il faudra parcourir la base de données *« contexte, tâche »* .

La cinquième étape assure la spécification de l'interface utilisateur. Dans cette étape les BIO sont identifiés en termes d'objets de l'interface tout en prenant en compte des critères ergonomiques de présentation d'une interface mobile.

La dernière étape de cette approche est dédiée à la génération de l'interface.

3.3 Présentation détaillée de l'approche

Dans ce qui suit, nous allons détailler et illustrer les différentes phases de notre approche.

3.3.1 Détection des données contextuelles

Cette phase constitue la première étape de toute architecture sensible au contexte. Avant le développement d'une application sensible au contexte, il est primordial de capter les données contextuelles. Cette étape est basée sur un ensemble de capteurs dispersés dans l'environnement ubiquitaire. Ces capteurs représentent le fournisseur de notre système en termes d'informations contextuelles. Un capteur peut être soit un capteur physique, virtuel ou logique. Les données interceptées doivent être analysées et modélisées.

3.3.2 Analyse du SHM et de la tâche

L'analyse du SHM représente l'identification des sous-systèmes importants afin de bien définir leurs comportements. Cette étape a pour objectif d'identifier les différentes vues graphiques appropriées à chaque sous-système. Le découpage du SHM en sous-système sera influencé par les différentes vues souhaitées. Ces vues reflèteront à un instant T les besoins informationnels de l'opérateur [Moussa *et al.*, 2006].

La décomposition hiérarchique du SHM en des sous-systèmes doit être régie par un modèle bien défini. De nombreuses méthodes d'analyse existent dans la littérature. Elles peuvent être regroupées en trois classes :

- Des méthodes procédant à une découpe cartésienne du système : Elles procèdent à une décomposition hiérarchique du système telles que SA et SADT [IGL, 1989] ;
- Des méthodes relevant de l'approche systémique : Elles procèdent à une analyse des données et une autre pour les traitements telles que Merise [Tardieu *et al.*, 1984] ;
- Des méthodes orientées objet : Elles procèdent à une structuration du système en termes d'objets encapsulant données et traitements. Les relations entre ces ob-

jets sont également illustrées par de nombreux langages unifiés désormais via UML [Jacobson *et al.*, 2000].

Notre choix pour la méthode d'analyse a été régi par les caractéristiques d'un environnement ubiquitaire. La dynamique et la quantité des données contextuelles doivent être analysés via une approche regroupant ces propriétés.

Les approches orientées objets regroupent l'analyse :

- Des aspects structurels : description des objets, leurs propriétés et leurs relations ;
- Des aspects dynamiques : description du comportement des objets et de leurs états ;
- Des aspects fonctionnels d'un système : description des fonctions et traitements relatifs à chaque objet.

L'approche objet permet une vérification et une validation des modèles par la simulation, ce qui garantit la faisabilité de l'analyse. Elle autorise plusieurs niveaux de réutilisation : les objets eux-mêmes, mais aussi des groupes d'objets cohérents, sous-systèmes, modules ou catégories de classes. Elle offre une facilité de transition vers la phase de conception, soit par enrichissement progressif des modèles, soit par application des règles de passage tenant compte de l'architecture de la réalisation retenue.

Nous avons choisi d'adopter une approche orientée objet, basée sur UML comme langage de modélisation. UML a été normalisé en 1997. Il facilite la représentation et la compréhension des données. Sa notation graphique, son aspect semi-formel et son indépendance par rapport aux langages de programmation en font un langage universel. Une autre caractéristique importante d'UML, est qu'il cadre l'analyse. UML permet de représenter un système selon différentes vues complémentaires : les diagrammes. Chaque type de diagramme possède une structure (les types des éléments de modélisation qui le composent sont prédéfinis) et véhicule une sémantique précise (il offre toujours la même vue d'un système). Combinés, les différents types de diagrammes UML offrent une vue complète des aspects statiques et dynamiques d'un système.

L'utilisation de cette approche nous permettra d'identifier pour chaque sous-système, dans un contexte bien défini, les tâches nécessaires de l'utilisateur. Grâce à cette analyse, nous saurons pour chaque instance du contexte, l'ensemble des tâches de l'utilisateur relatif à chaque sous-système. Le résultat de cette analyse permet de définir les actions relatives aux tâches de l'utilisateur. L'identification des paramètres de ces actions, à savoir, l'ensemble des variables d'entrée/sorties et les variables de contrôle constitueront les besoins informationnels de l'opérateur (BIO). Soulignons à ce propos que les BIO diffèrent d'un contexte à un autre. En effet, à chaque instance du contexte, l'utilisateur aura besoin d'une interface adaptée à ces nouveaux besoins.

Ainsi, cette analyse du SHM a pour objectif l'identification *(i)* des sous-systèmes qui évoluent dans des instances différentes du contexte et *(ii)* les besoins de l'utilisateur relatifs à ces instances. L'étape qui suit consiste à modéliser l'interaction homme-machine dans un environnement ubiquitaire. Les réseaux de Petri ont été proposés à cet effet (section 2.2.9).

3.3.3 Modélisation du contexte et de la tâche de l'utilisateur

Le résultat de l'analyse du SHM devrait être un artefact précisant les sous-systèmes ainsi que les tâches actuelles des différents utilisateurs. Cependant, la tâche évolue. En effet, de nouvelles tâches peuvent être introduites et d'autres peuvent être obsolètes. Ainsi, il existe un écart entre les tâches effectuées par les utilisateurs évoluant dans leurs environnements et les nouvelles tâches survenues lors de la création d'un nouveau système. Ce problème est très connu dans le domaine de l'IHM et a été pris en considération par les méthodes d'analyse des tâches [van der Veer et van Welie, 2000] [Wilson *et al.*, 1993].

Après l'étape d'analyse, nous trouvons l'étape de modélisation du SHM. Cette étape possède un impact direct et assez fort sur la conception de l'interface utilisateur. La modélisation de l'interaction homme-machine dans un environnement ubiquitaire consiste à modéliser dans un premier temps le contexte, puis la tâche de l'utilisateur.

Concernant la modélisation de la tâche, plusieurs approches ont émergé tout au long de ces années [Diaper et Stanton, 2003]. Plusieurs notations ont ainsi vu le jour (CTT, CTML, GOMS, RdP...). Ces formalismes diffèrent en termes de présentation, d'expressivité et de granularité. Cependant, ils reposent généralement sur le même principe : les tâches sont organisées hiérarchiquement. Elles sont exécutées afin de répondre à un but particulier. La décomposition des tâches est stoppée lorsqu'on atteint une tâche atomique appelée action.

L'approche la plus utilisée dans la communauté scientifique est ConcurTaskTrees (CTT)[Paterno, 2000]. Elle prend en charge le concept de relations temporelles qui limite les séquences valides de tâches pour atteindre un certain objectif. La coopération de la modélisation des utilisateurs en termes de modèles de tâches a été adressée par le CCTT (Collaborative ConcurTaskTrees) [Giulio *et al.*, 2002]. CCTT utilise une approche basée sur les rôles. Une spécification CCTT se compose de plusieurs arbres de tâches. Un arbre de tâche pour chaque rôle d'utilisateur impliqué et l'autre comme un « coordinateur » qui spécifie la collaboration et l'interaction globale entre les rôles des utilisateurs concernés. Dernièrement, une nouvelle variante de CTT a vu le jour : CTML : the Collaborative Task Modeling Language [Wurdel *et al.*, 2008]. Elle est basée sur l'idée que le comportement de l'utilisateur peut être défini par son rôle et que le comportement de chaque rôle peut être exprimé de ma-

nière adéquate par une expression de la tâche collaborative associée. Le modèle de la tâche collaborative est défini par un tuple < *acteurs, rôle, dispositifs, expression de la tâche de collaboration*>. Chaque expression de tâche collaborative a la forme d'un arbre de tâches, où les nœuds sont soit des tâches ou des opérateurs temporels. Chaque tâche est attribuée à un identifiant (unique), une condition préalable et un effet. Intuitivement, la condition définit un état de l'environnement de collaboration nécessaire pour l'exécution de la tâche, alors que l'effet dénote de l'état résultant après avoir exécuté la tâche. Les opérateurs temporels limitent les séquences possibles de l'exécution des tâches.

D'autres auteurs ont préféré travailler avec GOMS (Goals, Operators, Method, Selection). GOMS a été développé par Card [Card *et al.*, 1983]. Il affirme qu'un utilisateur interagit avec un ordinateur afin d'accomplir un but en sélectionnant des méthodes composées d'une séquence d'opérations. GOMS permet à un concepteur de simuler la séquence d'actions d'un utilisateur qui réalise une tâche en la décomposant en des objectifs et des sous objectifs. Il existe plusieurs variantes du modèle GOMS : Le modèle KLM [Card *et al.*, 1983], NGOMSL [Kieras, 1994] (Natural GOMS Language) : les méthodes y sont représentées via des règles basées sur le modèle CTT et CPM GOMS [John et Kieras, 1996] proposé pour décrire le parallélisme entre les actions de l'utilisateur.

Outre les approches citées ci-dessus, les réseaux de Petri (RdP) ont été également proposés et considérés comme étant des outils prometteurs pour la modélisation de la tâche. Ils ont été proposés par Carl Adam Petri en 1962. Ils représentent un formalisme mathématiquement fondé dédié à la modélisation du parallélisme et de la synchronisation dans les systèmes discrets, prenant en considération les aspects événementiels qui ne peuvent pas être traités par les automates. Les RdP sont en expansion continue et représentent un outil adéquat pour la modélisation de l'interface utilisateur. Au début, ils n'ont été utilisés qu'à des fins de description des tâches à informatiser. Mais par la suite, et surtout avec l'apparition des Réseaux de Petri de Haut Niveau, on a commencé à profiter de leur puissance pour modéliser le dialogue Homme-Machine. Nous pouvons citer, principalement, les travaux de Willem et Biljon qui ont été les premiers à avoir pensé à adapter les RdP à la modélisation du dialogue Homme-Machine [Van Biljon, 1988] ; ceux de Palanque *et al.* avec le formalisme ICO [Palanque et Bastide, 1995] ; ceux de Tabary *et al.* avec TOOD [Tabary et Abed, 2002](faisant suite aux travaux sur SADT/RdP de [Abed et Ezzedine, 1998]) et ceux de De Rosis avec XDM [Riahi, 2004].

Que ce soit pour la modélisation du contexte ou de la tâche, nous avons donc convenu de travailler avec la technique formelle des réseaux de Petri. En effet, on considère dans la littérature que cette technique est assez efficace pour l'expression des aspects de synchro-

nisme, de concurrence et de parallélisme. Les RdP proposés ici sont les RdPI (Réseaux de Petri interprétés) [Moalla, 1985]. Ce type de réseaux introduit les notions d'événements et de conditions ainsi que la notion d'actions. En effet, une condition de passage (Cj), un événement de déclenchement (Evj) et une action éventuelle (Aj) sont associés à chaque transition Tj d'un RdPI (Figure 3.2).

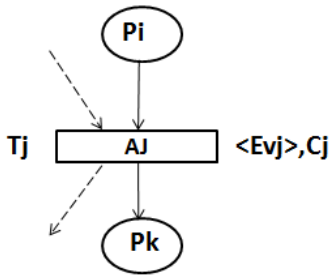


Figure 3.2 — Réseau de Petri Interprété (RdPI).

L'étape de modélisation (tâche, contexte) doit être réalisée avec rigueur. En effet, cette modélisation influe directement sur la génération de l'interface utilisateur. Ces modèles doivent donc être valides et vérifier un certain nombre de propriétés relatives à l'interface. Dans notre approche, nous avons choisi de vérifier ces propriétés dès la première étape de notre approche, plutôt que de les vérifier dans la phase de génération de l'interface. Pour ce faire, nous avons présenté dans nos travaux antérieurs des actions et des compositions élémentaires qui garantissaient des réseaux de Petri valides et qui vérifiaient toutes les propriétés importantes de l'interface utilisateur [Riahi, 2004][Ben Ismail, 2014][Moussa *et al.*, 2015][Moussa, 2005].

Pour modéliser le contexte ainsi que la tâche de l'utilisateur moyennant les RdP, nous avons convenu d'utiliser *(i)* les places pour représenter l'état du contexte/comportement de l'utilisateur vis-à-vis de l'évolution de l'environnement et *(ii)* les transitions pour décrire l'évolution entre ces différents états. Il est à noter que chaque changement au niveau du contexte ou au niveau de la tâche doit affecter l'interface utilisateur afin de l'informer de la situation courante.

Nous considérons que le contexte (utilisateur, plateforme et environnement) et la tâche de l'utilisateur sont composés d'un ensemble organisé d'actions élémentaires. La structure modélisant une action élémentaire est donnée par la figure 3.3.

Concernant la modélisation de la tâche, la validation de la condition i (transition T1) 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 T2) exprime le fait que

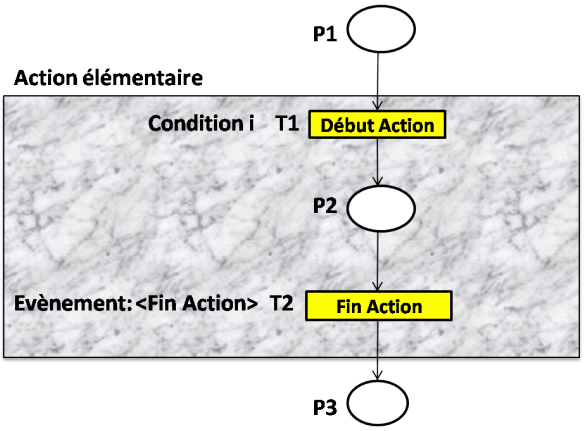


Figure 3.3 — Structure d'une action élémentaire.

l'action de l'utilisateur a été exécutée et a pris fin. La place P2 exprime un état d'attente de la fin de l'exécution de l'action, alors que les places P1 et P3 modélisent l'état de l'utilisateur avant et après l'exécution de son action. Par exemple, P1 modélise l'intention mentale de l'utilisateur pour agir. La place P3 exprime son état à la fin de l'accomplissement de l'action [Rasmussen, 1986][Moussa *et al.*, 2002].

Concernant la modélisation du contexte, la validation de la condition i (transition T1) modélise le fait que le contexte va passer d'un état à un autre. L'apparition, par la suite, de l'événement « fin action » (transition T2) exprime le fait que l'état du contexte a été exécuté et a pris fin. La place P2 exprime un état d'attente, alors que les places P1 et P3 modélisent l'état du contexte avant et après changement de son état.

Toutes les actions élémentaires sont ordonnées selon des compositions typiques : séquentielle, parallèle, alternative, de choix, itérative ou de fermeture [Riahi et Moussa, 2015]. Nous présentons dans ce qui suit le principe de chacune d'entre elles.

3.3.3.1 Composition séquentielle

La composition séquentielle de n actions traduit le séquencement de leur exécution. La composition séquentielle de n actions est assurée en fusionnant la place fin du réseau modélisant l'action i, avec la place début de l'action i+1 (Figure 3.4).

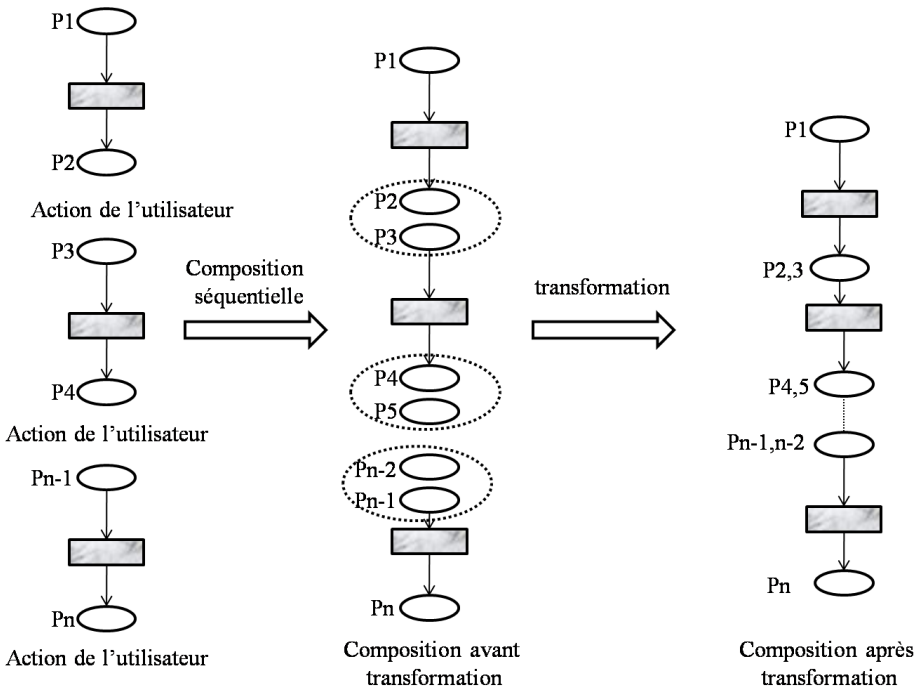


Figure 3.4 — Composition séquentielle.

3.3.3.2 Composition parallèle

La composition parallèle exprime la possibilité de leur exécution simultanée. Le parallélisme est assuré par le fait qu'une place de synchronisation d'entrée, active en même temps toutes les places d'initialisation des actions parallèles à exécuter. Cependant, le parallélisme effectif ne peut s'effectuer que si les actions à exécuter ne sollicitent pas les mêmes ressources. Sinon, un séquencement partiel ou total serait nécessaire. La composition parallèle de n réseaux relatifs à n actions fait intervenir deux structures de composition PAR1 et PAR2.

- La structure PAR1 modélise une synchronisation de départ de n réseaux (Figure 3.5.a) ;
- La structure PAR2 modélise une synchronisation à l'arrivée des n réseaux (Figure 3.5.b).

Evidemment, le nombre de places Pn dans les deux structures PAR1 et PAR2 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).

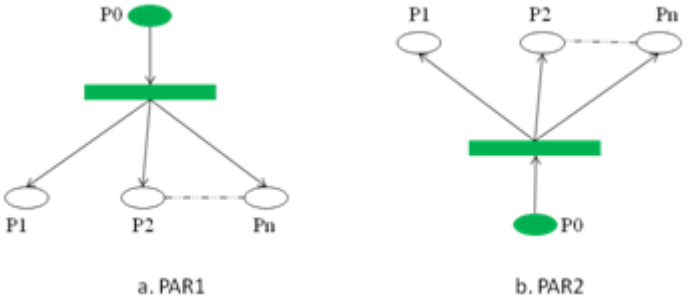


Figure 3.5 — Structure PAR1, PAR2.

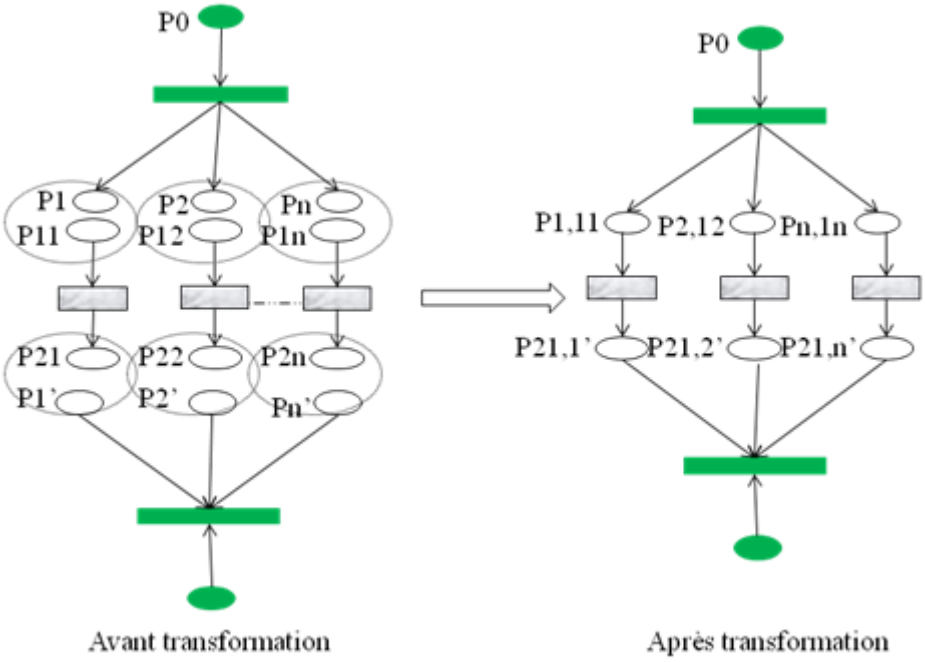


Figure 3.6 — Composition parallèle.

3.3.3.3 Composition alternative

La composition alternative de n actions traduit une exécution toujours exclusive de ces actions. Afin d'éviter un conflit effectif, des conditions sont associées aux transitions pour déterminer sans ambiguïté laquelle des actions doit être exécutée. La composition alternative de n réseaux est réalisée en les composant en séquentiel avec une structure ALT et en fusionnant toutes les places fin de ces réseaux. La structure ALT permet la validation d'une seule condition à la fois (Figure 3.7). La structure ALT comporte un ensemble de transitions égal

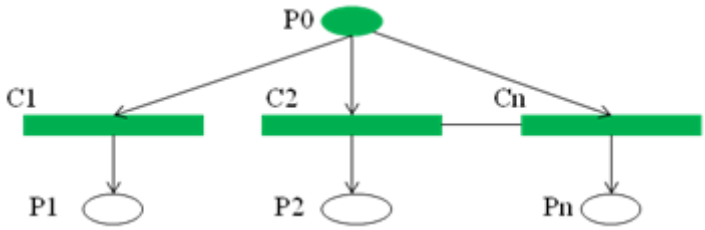


Figure 3.7 — Structure ALT.

au nombre des réseaux à composer en alternatif. Ces transitions sont issues d'une même place d'entrée P0. Elles permettent, grâce aux conditions qui leur sont associées, d'initialiser sans ambiguïté un réseau unique parmi les n modélisés, ce qui garantit l'absence de conflit effectif (Figure 3.8). Les conditions Ci, i variant de 1 à n, dépendent de l'état courant interne au système.

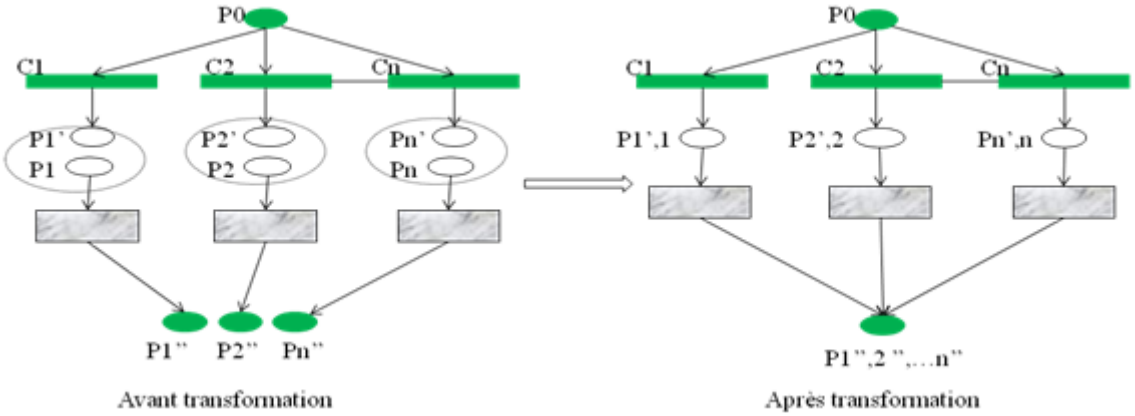


Figure 3.8 — Composition Alternative.

3.3.3.4 Composition choix opérateur

Il y a deux compositions choix opérateur possibles : le choix exclusif et le choix inclusif.

Composition choix exclusif : Pour la composition choix exclusif, elle rappelle la composition alternative, à la différence que la décision de l'action à exécuter n'est pas déterminée par l'état courant interne du système mais par un choix de l'opérateur : l'opérateur intervient à travers l'exécution d'une action élémentaire pour décider laquelle des actions parmi les n est à exécuter. La composition de n réseaux permettant un choix exclusif par l'opérateur (Figure

3.9) peut être faite en composant en séquentiel une structure élémentaire de décision du choix opérateur avec la structure obtenue par composition alternative des n réseaux.

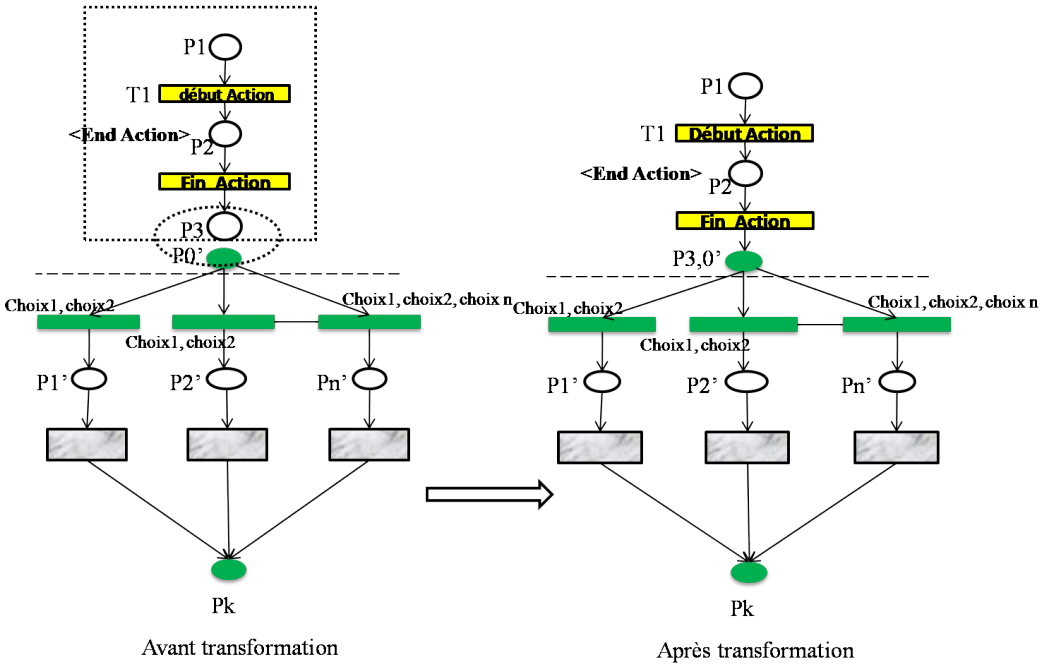


Figure 3.9 — Composition de choix opérateur exclusif.

Composition choix inclusif : A la différence de la composition choix exclusif pour laquelle l'opérateur décide du choix d'une seule action à exécuter parmi n, dans la composition choix inclusif, l'opérateur peut décider du choix d'un sous-ensemble d'actions à exécuter parmi les n (0 à n actions). Ainsi, pour un choix inclusif, l'opérateur intervient à travers l'exécution d'une action élémentaire pour décider lesquelles parmi les n actions sont à exécuter, les autres seront « court-circuitées ».

3.3.3.5 Composition itérative

La composition itérative de n actions exprime leur exécution successive avec itération possible. L'itération est soumise à une condition « Itération » calculée au cours de l'exécution séquentielle des n actions. La composition itérative d'un réseau est réalisée en englobant le réseau dans une structure **I** d'itération (Figure 3.11). La structure d'itération comporte deux transitions Ti et Tni. Ces transitions sont issues d'une même place d'entrée. Elles permettent, grâce aux conditions qui leur sont associées, d'être franchies sans ambiguïté ce qui garantit l'absence de conflit effectif (Figure 3.10).

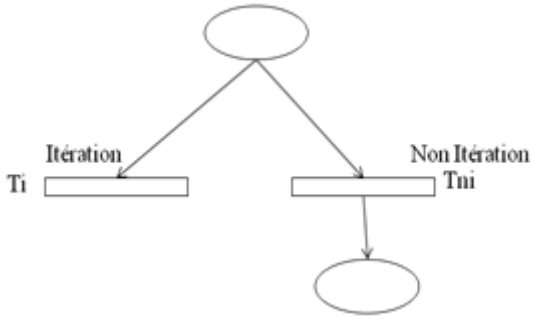


Figure 3.10 — Réseau I.

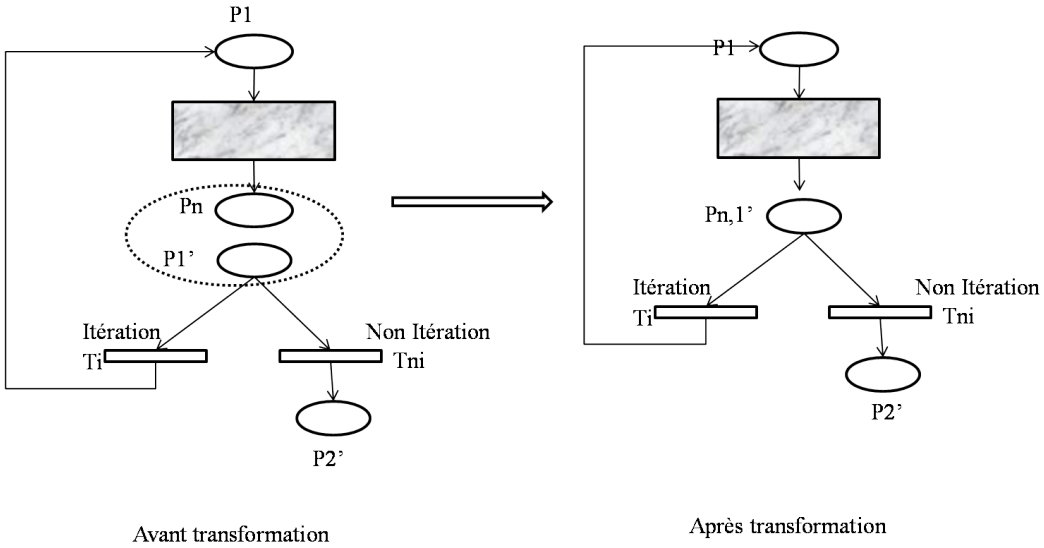


Figure 3.11 — Composition itérative.

3.3.3.6 Composition de fermeture

La composition de fermeture d'un réseau traduit la mise en boucle de ce réseau. La composition de fermeture d'un réseau est réalisée en englobant le réseau dans une structure F de fermeture (Figures 3.12, 3.13).

La composition de fermeture sera utilisée pour construire le modèle global du contexte ou de la tâche permettant de revenir à la situation initiale d'un état donné.

La construction du modèle global du contexte ou d'une tâche utilisateur s'effectue à partir des structures de base modélisant les différentes actions élémentaires. Elle se base sur l'application des différentes opérations de composition explicitées ci-dessus et son principe réside

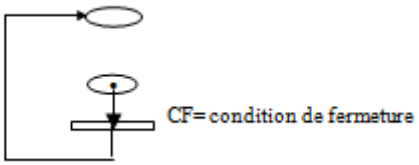


Figure 3.12 — Structure de fermeture.

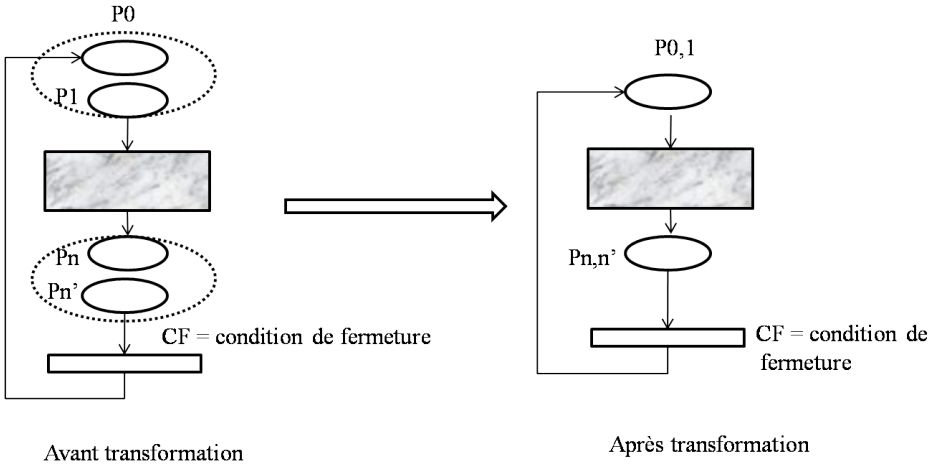


Figure 3.13 — Composition de fermeture.

dans le fait que la construction du modèle 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 ces actions et structures élémentaires seront stockées dans une base de données afin de faciliter la construction de notre modèle. Notre base doit prendre en considération toutes les compositions définies ci-dessus.

Après avoir analysé et modélisé le SHM dans un environnement ubiquitaire, l'étape qui suit représente la détection de la situation courante de l'utilisateur.

3.3.4 Détection de la situation courante (Ci, Ti)

L'objectif global de nos travaux de recherche, est d'aboutir à une interface utilisateur adaptable au contexte courant de l'utilisateur dans un environnement critique. Pour ce faire,

il faut tout d'abord procéder à une analyse du SHM pour ensuite le modéliser. La modélisation du SHM doit prendre en considération la modélisation du contexte. Rappelons que le contexte est défini par le triplet « utilisateur, plateforme, environnement », de ce fait, chaque composante sera modélisée par son propre RdP :

- RdP utilisateur : Ce réseau de Petri aura pour objectif de modéliser les différents utilisateurs qui peuvent utiliser l'application ;
- RdP plateforme : Ce réseau présentera les différentes plateformes qui peuvent être hébergées sur notre application et qui peuvent être utilisées par les différents utilisateurs ;
- RdP environnement : Ce réseau aura pour fonction de décrire les différentes caractéristiques de l'environnement (localisation géographiques, horaire, etc.).

Etant donné que chaque composante du contexte est modélisée par son propre RdP, le marquage du réseau à un moment donné, déterminera l'état actuel du contexte. Les jetons des places dans les trois réseaux simultanément, détermineront l'état du contexte à un instant donné, c'est-à-dire la valeur du triplet « utilisateur, plateforme, environnement ». Ce modèle englobera toutes les valeurs possibles et imaginables pour le contexte.

La tâche de l'utilisateur dans le Système Homme-Machine adaptable au contexte, doit être reliée à ce dernier. En effet, une tâche peut varier selon le contexte où évolue l'utilisateur. Chaque tâche utilisateur est spécifique à un contexte donné. Notre Système Homme-Machine global sera donc composé d'un ensemble de couples « contexte, tâche ».

Le modèle de la tâche est utilisé pour obtenir toutes les exigences du système dès la première phase de développement en décrivant comment les utilisateurs peuvent atteindre leurs objectifs en effectuant un ensemble de tâches. Cependant, ces dernières années, des modèles de la tâche ont également été utilisés pour la conception de systèmes dans des environnements ubiquitaire. Nous pouvons citer par exemple le travail de [Gharsellaoui *et al.*, 2014] qui ont proposé d'utiliser le modèle de tâches au moment de l'exécution, afin de suivre les actions de l'utilisateur, vérifier qu'il n'a pas fait d'erreurs et lui procurer de l'aide en cas de besoin. Ils ont présenté un modèle de tâches adapté aux environnements ambiants qui attribue dynamiquement des états aux tâches au moment de l'exécution.

En revanche, nous avons noté la râreté de ce type de travaux. Cette constatation est assez surprenante car les environnements intelligents comprennent une vaste complexité en termes de performances de la tâche des utilisateurs. Une compréhension approfondie des tâches utilisateurs s'exécutant dans de tels environnements est une condition préalable pour fournir une assistance appropriée.

Dans un environnement ubiquitaire, c'est le modèle du contexte qui va déclencher le modèle de la tâche adéquat. En effet, la tâche varie selon le contexte et la tâche de l'utilisateur n'est

pas un modèle fixe. En outre, en fonction des valeurs du contexte à un instant donné, on pourra déduire la tâche adéquate pour l'utilisateur.

Les tâches des utilisateurs seront aussi modélisées via les réseaux de Petri. Chaque tâche sera décomposée en des tâches élémentaires qui seront modélisées via des RdP élémentaires.

A un instant donné, le marquage des trois RdP constituant le contexte plus le RdP modélisant la tâche, donnera l'état de notre SHM dans notre environnement ubiquitaire. Les valeurs de ces marquages sont stockées auparavant dans une base de données. Cette BD stockera ainsi les couples « contexte, tâche ». Si à un moment donné, les valeurs du marquage des RdP décrivant le contexte courant ainsi que la tâche de l'utilisateur, figurent dans cette BD, alors cette situation sera considérée comme étant une situation normale, sinon elle sera considérée comme étant un état de dysfonctionnement ou un évènement imprévu. En d'autres termes, lors de l'exécution de notre architecture, le marquage des réseaux constituant le contexte déterminera son état actuel Ci. Pour connaitre la tâche Ti adéquate à Ci, il faudra parcourir la base de données " contexte, tâche ". Une fois la détection de la situation actuelle faite, la valeur de ce couple sera transmise afin de spécifier l'interface adéquate pour le couple (Ci,Ti).

De point de vue implémentation, tous ces modèles doivent être exploités et convertis en un langage compréhensible par la machine. De nos jours, un grand nombre d'outils pour la spécification et la génération d'interface sont basés sur XML. De ce fait, nous avons cherché à convertir tous nos modèles (tâche et contexte) en XML. La section qui suit présente un état de l'art des réseaux de Petri codés en XML.

3.3.4.1 Les réseaux de Petri codés en XML

La combinaison des réseaux de Petri avec le langage XML fournit deux genres de formalismes utilisés jusqu'à présent et dans la majorité des travaux dans la gestion des processus métier. En effet, durant ces dernières années, et en raison de la croissance rapide d'internet, des médias mobiles, des langages pour la modélisation, l'analyse et l'exécution des processus distribuées, une nouvelle variante des RdP de haut niveau a vu le jour : les XML Nets [Lenz et Oberweis, 2003]. Ces derniers assurent l'échange de documents électroniques entre plateformes hétérogènes. En plus de la croissance permanente des variantes des RdP, la standardisation de la technique a été perçue comme une opportunité d'obtenir une meilleure organisation du travail dans la communauté des réseaux de Petri et ce, pour remédier à plusieurs problèmes :

- Permettre aux chercheurs et aux ingénieurs d'utiliser la même terminologie ;
- Développer des extensions futures sur une base commune stable ;

– Fournir une implémentation de référence qui facilitera l'échange de données entre les différents outils de RdP grâce à un format commun.

Pour ce faire, Matthias Jûngel a présenté les concepts et la terminologie de PNML (Petri Net Markup Language), et a ainsi fourni un point de départ pour l'élaboration d'un format d'échange standard pour les RdP [Jüngel *et al.*, 2000].

Nous présentons, ci-dessous, les XML Nets, une variante des RdP de haut niveau, puis le Petri Net Markup Language (PNML), format d'échange entre les RdP.

3.3.4.1.1 XML Nets XML Nets représente une nouvelle variante de réseau de Petri de haut niveau. XML Nets est un langage formel de modélisation graphique qui modélise les flux de documents XML, les flux de contrôle et de gestion des processus. XML Nets identifie et décompose le processus en un ensemble de fragments XML et les affecte à l'unité organisationnelle adéquate [Lenz et Oberweis, 2003].

Les composants statiques de XML Nets à savoir les places dans les RdP sont modélisées par des diagrammes de schéma XML. Les places peuvent être interprétées comme des conteneurs pour les documents XML valides.

Cependant, XML Nets n'a aucun moyen de vérifier la validité des réseaux de Petri générés. La représentation des systèmes complexes conduit à des schémas XML complexes qui sont difficiles à exploiter. De plus, ce formalisme ne peut être considéré comme étant un standard de normalisation vu le manque de documentation. Par conséquent, cette technique de modélisation ne peut atteindre nos objectifs. Voilà pourquoi nous nous sommes orientés vers PNML.

3.3.4.1.2 Petri Net Markup Language PNML Dès les années 2000, plusieurs applications et théories d'échanges entre les réseaux de Petri basés sur XML ont vu le jour dont PNML (Petri Net Markup Language) standardisé selon la norme internationale ISO/IEC 15909 [15909, 2007]. Cette norme est structurée en trois parties :

– La première partie concerne les définitions du formalisme, donnant ainsi une sémantique formelle à plusieurs types usuels de réseaux de Petri : les réseaux places/transitions et les réseaux de haut niveau (colorés). Cette partie a été publiée comme norme internationale en décembre 2004.
– La seconde partie se concentre sur la syntaxe et l'élaboration d'un langage d'échange de réseaux de Petri. Ce langage, Petri Net Markup Language (PNML), s'appuie sur les concepts introduits dans la première partie du standard. Sa conception repose sur des méta-modèles en Unified Modeling Language (UML), puis une transformation en

eXtensible Markup Language (XML). Cette partie a été normalisée en 2011.
- La troisième partie porte sur le modèle définissant la hiérarchie, la modularité, et le cadre des nombreuses extensions de types de RdP, dont les travaux ont démarré en 2012.

La conception de PNML est régie par les principes suivants :
- La lisibilité : Le format doit être lisible par l'homme et modifiable par n'importe quel éditeur de texte ;
- La mutualité : PNML doit permettre d'extraire autant d'informations que possible à partir d'un réseau de Petri même si parfois son format est inconnu, donc le format doit extraire les principes communs à tous types de réseau de Petri ;
- La flexibilité, l'universalité : PNML devrait être en mesure de représenter tout type de RdP avec ses éventuels extensions et fonctionnalités spécifiques. Afin d'atteindre cette fonctionnalité, PNML considère le RdP comme étant un graphe étiqueté où toutes les informations peuvent être stockées sous forme d'étiquettes assignées à l'arc, au nœud et/ou le réseau lui-même ;
- La compatibilité : Plusieurs types de réseaux de Petri peuvent échanger plusieurs renseignements. Pour assurer ceci, PNML donne des conventions sur la façon de définir une étiquette avec un sens particulier. Dans le document des conventions, la syntaxe ainsi que le sens voulu de toutes sortes d'extensions sont prédéfinies. Lorsqu'on effectue la définition d'un nouveau type de RdP, les étiquettes peuvent être choisies à partir de ce document des conventions [Billington *et al.*, 2003].

L'utilisation d'XML garantit la lisibilité. La notion d'universalité est garantie avec les principes d'étiquetages que nous allons voir par la suite. Enfin, la mutualité est garantie avec l'utilisation des conventions qui représente un ensemble d'étiquettes normalisées. XML présente plusieurs avantages à savoir sa popularité, l'abondance de ces outils pour la lecture, l'écriture et la validation des documents XML et son indépendance par rapport à la plateforme utilisée.

Chaque classe du méta-modèle PNML est traduite en un élément XML comme le montre le tableau 3.1.

Un document qui répond aux exigences de PNML est appelé un document de RdP (PetriNetDoc). Il contient un ou plusieurs RdP (PetriNet). Chaque RdP possède un identifiant unique et un type. Un RdP se compose d'une ou plusieurs pages qui à leur tour se composent d'objets.

Chaque RdP se compose d'objets qui au fond représentent la structure graphique du réseau. Chaque objet possède un identifiant unique qui peut être utilisé pour faire référence

Tableau 3.1 — Syntaxe XML pour PNML

Classe	**Elément XML**	**Les attributs XML**
Un fichier de RdP	<PNML>	
RdP	<net>	Id : ID, Type : anyURI
Place	<place>	Id : ID
Transition	<transition>	Id : ID
Arc	<arc>	Id : ID, Source :IDRef (nœud), Cible : IDRef (nœud)
Page	<page>	Id : ID
Place de référence	<referencePlace>	Id : ID, Ref : IDRef (place ou place de référence)
Transition de référence	<referenceTransition>	Id : ID, Ref : IDRef (transition ou transition de référence)
Outils d'information spécifique	<toolspecific>	outil : String
Informations graphiques	<graphics>	

à une place, une transition ou un arc.

Les étiquettes : elles ont été conçues afin d'attribuer un sens à chaque objet. Une étiquette représente le nom d'un nœud, le marquage initial d'une place, l'état d'une transition ou d'un arc. Il existe deux sortes d'étiquettes : les annotations et les attributs. Une annotation est affichée sous la forme d'un texte à côté de l'objet contrairement à l'attribut qui a un effet sur la forme ou la couleur de l'objet correspondant. Il n'existe pas d'éléments PNML qui définissent les étiquettes. Elles sont définies par le type des réseaux de Petri. Tous les éléments XML ne figurant pas dans ce tableau sont considérés comme étant des éléments PNML. La balise <initialMarking> pourrait être affectée à une place pour représenter son marquage initial. La balise <inscription> correspond à l'annotation d'un arc. <text> représente la valeur de l'étiquette comme étant une chaîne de caractère. <Structure> peut être utilisé pour représenter la valeur d'une étiquette comme étant un arbre de syntaxe abstraite XML. La balise <graphics> définit l'aspect graphique de l'objet ou l'étiquette PNML. Enfin, la balise <toolSpecific> est utilisée afin d'ajouter des informations d'outils spécifiques relatives à l'objet.

Les informations graphiques : elles peuvent être associées à chaque objet et à chaque annotation. Elles concernent la taille, la couleur, la police, la position d'un nœud ou d'un arc.

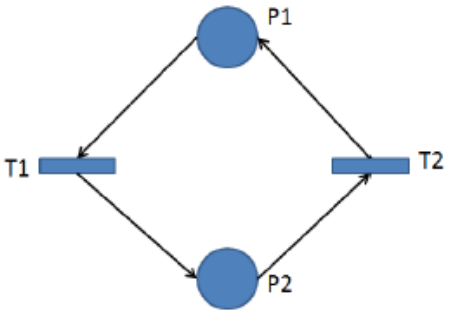
Les outils d'informations spécifiques : ils peuvent être associés à chaque objet et à chaque étiquette. Ils ne sont pas spécifiés par PNML, il suffit juste de mettre le nom de l'outil et sa version. Un même objet peut être étiqueté par différents outils ; donc un même document peut être utilisé par différents outils et ce en même temps.

Les pages et les nœuds de référence : une page peut contenir plusieurs pages ou plusieurs objets. PNML exige qu'un arc doit connecter des nœuds sur une même page. Si on a besoin de connecter deux nœuds présents sur deux pages, alors l'un des deux migre vers l'autre page. On l'appelle alors, nœud de référence puisqu' il fait référence à l'autre nœud de l'autre page. Les places de référence et les transitions de référence doivent se référer respectivement à une place/place de référence et à une transition / transition de référence [Hillah *et al.*, 2009].

Pour être interprétée sans ambigüité et pour assurer la propriété de compatibilité, chaque définition d'un type de réseau de Petri doit avoir une sémantique formelle qui doit être connue par chaque outil qu'il utilise. La description de la sémantique et des fonctions des RdP et de leurs types n'est pas définie par ce document. Les notions de PNML fournissent un point de départ à cette description. Si on a besoin d'importer un RdP dont la sémantique est inconnue c'est-à-dire que notre outil ne le prend pas en charge, une entrée manuelle peut être effectuée ou encore l'outil peut extraire un maximum d'informations devinant leurs sens à partir des noms de leurs étiquettes ce qui peut impliquer une mauvaise interprétation. Afin de remédier à ce problème, PNML offre des fonctions de définition d'interface. Chaque élément fixe la syntaxe ainsi que la sémantique de son étiquette. Toutes ces informations sont stockées dans le document des conventions. En consultant ce dernier, un outil peut alors connaitre la signification des étiquettes et identifier ainsi le RdP inconnu pour le convertir en un RdP standard. Cependant, le risque de perte d'informations du réseau d'origine est fort probable [Billington *et al.*, 2003]. La figure 3.14 illustre un exemple de RdP avec son code PNML.

PNML est considéré comme étant le format le plus souple et le plus développé pour l'échange entre les réseaux de Petri. Il a été conçu pour fournir un format d'échange commun pour tous les types de réseaux de Petri quelque soit l'outil employé [Kindler, 2006]. L'une de ces plus grandes forces est le développement d'un processus de normalisation de RdP de haut niveau.

En regard des objectifs que nous nous sommes fixés, il apparaît que les XML Nets ne peuvent en aucun cas nous convenir. En effet le résultat des XML Nets est un RdP de haut niveau, hors notre modélisation doit aboutir à un fichier XML pour pouvoir déduire la situation courante de l'utilisateur. Nous optons donc pour le standard PNML. En effet, celui-ci offre un canevas assez riche et ouvert pour répondre aux contraintes de modélisation des SHM sensibles au contexte sous forme de RdP. PNML s'avère être le formalisme le plus adéquat




```
<pnml>
<net>
<place><name><text>P1</text></name>
<initialMarking><text>1</text>
</initialMarking>
</place>
<place><name><text>P2</text></name>
<initialMarking><text>0</text>
</initialMarking>
</place>
<transition><name><text>T1</text></name>
</transition>
<transition><name><text>T2</text></name>
</transition>
<arc id="A1" source="p1" target="T1"/>
<arc id="A2" source="T1" target="P2"/>
<arc id="A3" source="P2" target="T2"/>
<arc id="A4" source="T2" target="p1"/>
</net>
</pnml>
```

Figure 3.14 — Exemple d'un fichier PNML

pour la conversion de nos modèles en du XML [Riahi *et al.*, 2011b][Riahi *et al.*, 2011a].

Il existe dans la littérature plusieurs solutions qui permettent la conversion de RdP en PNML, cependant, aucune d'entres elle n'est basée sur les structures élémentaires. Par conséquent, nous avons conçu et développé une solution logicielle nommée PetriNetEditor capable de générer le code PNML d'un RdP basé sur les structures élémentaires. La présentation de cette solution sera exposée dans le prochain chapitre.

Après avoir analysé, modélisé le SHM, traduit tous les modèles en PNML et déduit la situation courante de l'utilisateur, il est primordial à ce stade d'identifier les différents BIO pour pouvoir spécifier l'interface utilisateur visée.

3.3.5 Spécification de l'interface utilisateur

Afin de spécifier les composantes graphiques de l'interface, il faut tout d'abord déduire les Besoins Informationnels de l'Opérateur (BIO). En effet, les composantes graphiques de

l'interface sont étroitement liées avec les BIO. Nous ne pouvons pas générer une interface adaptable au contexte sans avoir déduit les BIO.

L'utilisateur a besoin d'identifier, dans la mesure du possible, instantanément, les tâches à accomplir dans un contexte bien déterminé. Ces informations lui sont transmises à travers différents objets au niveau de l'interface (messages, valeurs, graphiques, alarmes, etc.). Ces objets sont, en fait, reliés à des variables qui varient selon le changement de l'interface, donc selon le changement du contexte. Il faut identifier l'ensemble des variables d'information appropriées à chaque interface pour chaque instance du contexte. Ces variables d'information découlent, en fait, de l'analyse du SHM sensible au contexte menée auparavant.

De plus, et dans le but d'accomplir ses tâches, l'utilisateur aura à intervenir et commander différentes variables lors de l'apparition de situations normales, de recherche, de validation ou critiques. L'interface doit donc présenter dans de telles situations, l'ensemble des objets dits « de commande » qui permettent à l'utilisateur d'exécuter ses actions (appuyer sur un bouton, écrire dans un champ de texte, etc.). L'ensemble de ces variables d'information et de commande constitue les BIO.

Nous associons aux transitions « début action » dans chacun de nos modèles, les variables d'information et de commande reflétant l'état du système et respectant les besoins des utilisateurs en matière d'information (ces BIO).

Une fois les BIO identifiés (c'est-à-dire l'ensemble des variables d'information et de commande formé), l'étape suivante consiste à spécifier les objets graphiques relatifs à ces BIO :

- à chaque variable d'information, on associe un objet graphique d'information ;
- à chaque variable de commande, on associe un objet graphique de commande.

La spécification de l'interface consiste à identifier les composantes de présentation et de dialogue.

La spécification des composantes de présentation est faite grâce à l'identification des BIO. Le positionnement des composantes au niveau de l'interface doit être régi par des règles ergonomiques relatives aux interfaces mobiles. Ces règles sont stockées dans la base des règles ergonomiques.

Le processus d'identification et de modélisation des règles ergonomiques ne concerne pas directement nos travaux. En effet, nous avons utilisé des règles élémentaires de spécification de présentation des interfaces mobiles. Ces règles ont été formalisées sous forme clausale et stockées dans notre base. Les questions qui se posent à ce stade sont les suivantes : comment associer les vues graphiques en fonction des BIO et comment afficher les vues appropriées (informationnelles ou de commande) ?

Afin de répondre à ces questions, quelques règles ergonomiques (Figure 3.15), formalisées sous forme clausale et basées sur les travaux de [Riahi, 2004], ont été utilisées. Un exemple sera proposé au quatrième chapitre.

```
Si{type_variable(V)= "informationnel"
Et type_vue(Sous-sys)= "informationnel"
Et type_representation(Sous-sys)= "consultation"}
Alors {select_tableau(V)}
Si{type_variable(V)= "informationnel"
Et type_vue(Sous-sys)= "informationnel"
Et type_representation(Sous-sys)= "intervention"}
Alors {select_barre-graphe(V)}
Si{type_variable(V)= "commande"
Et type_vue(Sous-sys)= "commande"
Et type_representation(Sous-sys)= "intervention"}
Alors {select_bouton_commande(V)}
```

Figure 3.15 — Exemples de règles ergonomiques pour la sélection des objets de l'interface.

Ainsi, pour le choix de la représentation graphique d'un objet donné, certaines règles ergonomiques sont proposées. Par exemple, pour le cas de la représentation graphique d'une variable d'information dans une vue informationnelle avec un type de représentation « consultation », nous recommandons l'utilisation d'un tableau. Si par contre, le type de représentation associé à cette vue est « intervention », nous conseillons d'utiliser les barre-graphes. Pour le cas d'une vue de commande, s'il s'agit d'une variable de commande, des modes de représentation « bouton de commande » sont prévus.

Une fois l'aspect graphique de l'interface réalisé, la deuxième étape préconise la spécification du dialogue homme-machine. Cette étape est définie par la description du comportement des différents objets de l'interface. Chaque objet de notre interface, informationnel ou de contrôle, passe principalement par deux états : affiché ou masqué. Le changement de l'état de la variable dépend du changement du contexte, de la tâche et du type de la variable. En effet, si la situation du contexte représente une situation critique alors l'utilisateur doit intervenir. De ce fait, le type de variable sera une variable de commande et le bouton associé passera de l'état « masqué » à l'état « affiché ». Ce processus sera régi par des algorithmes de parcours de fichiers PNML du contexte et de la tâche et des structures conditionnelles qui assureront le changement d'état selon les valeurs du couple (contexte, tâche) fournies.

Après avoir spécifié notre interface, la dernière étape consiste à générer l'interface adaptable au contexte.

3.3.6 Génération de l'interface utilisateur

Lors de cette étape, nous avons préconisé l'utilisation d'un ensemble de web services interagissant ensemble afin de répondre à une requête précise. Lors de l'envoi de la requête, le premier service récupère le fichier PNML et réalise le parcours du fichier grâce au DOM Parser afin de retrouver les valeurs du contexte à savoir l'environnement, l'utilisateur et la plateforme. Ces informations seront par la suite intégrées dans un service assurant l'extraction des informations de la tâche à partir des données contextuelles et de la base de données réalisée en JSON. Un autre service de parcours est mis en place dans le but de récupérer l'ensemble de variables d'information et de contrôle qui, à l'aide de la base des règles ergonomiques, assureront la spécification et ainsi la génération de l'interface en question. Les services mis en œuvre sont indépendants afin de garantir la réutilisabilité et l'interopérabilité. Cette étape ne concerne pas directement notre travail. En effet, nous nous sommes arrêtés à l'étape de spécification et nous avons préparé, en fonction, des maquettes d'interfaces pour la phase d'évaluation.

3.4 Discussion

Nous pouvons affirmer à ce stade, que notre approche suit la structure générale d'un système adaptable au contexte (Chapitre 1, section 1.4.1). En effet, nous retrouvons :

- La couche capture du contexte, nommée dans la Figure 3.1, capture des données ;
- La couche interprétation du contexte, représentée dans notre architecture par le modèle du contexte (modèle utilisateur, modèle plateforme et modèle environnement) ;
- La couche stockage du contexte, définit par la BD « Contexte i, Tâche j » ;
- La couche diffusion du contexte décrite par l'étape de spécification de l'IU ;
- La couche application décrite par la phase de génération de l'IU.

Notre architecture sépare l'acquisition et la modélisation du contexte de son utilisation. Chaque composante de notre architecture remplit une tâche particulière. En raison de la complexité des données du contexte, nous avons choisi de décomposer ces données en trois modèles et de prendre en considération la tâche de l'utilisateur dès la première étape de notre approche à savoir l'analyse du SHM sensible au contexte.

L'originalité de notre approche réside dans la proposition du couple (contexte, tâche). Le SHM est devenu sensible au contexte c'est-à-dire que la tâche de l'utilisateur varie selon le contexte dans lequel l'utilisateur opère. En effet, la tâche de chaque utilisateur est spécifique à un contexte donné. Voilà pourquoi un ensemble de paires (contexte, tâches) ont été définies pour composer le modèle global du SHM.

Notre modèle décrit le comportement des données contextuelles. Il décompose les composantes du contexte en des granularités fines afin d'assurer la validité de notre modèle. Pour ce faire, nous utilisons un ensemble bien organisé de structures élémentaires de réseaux de Petri.

En règle générale, les applications contextualisées présentent un niveau élevé de dynamisme. De ce fait, de nombreuses actions doivent être effectuées en parallèle. Nous estimons que l'aspect de parallélisme est très important surtout dans les domaines critiques. Certaines situations nécessitent l'intervention de deux ou plusieurs utilisateurs en même temps, pour répondre à une situation particulière. L'utilisation d'une méthode formelle pour décrire le comportement d'un système sensible au contexte nous permet de déduire les propriétés du système et les besoins des utilisateurs afin de générer l'interface appropriée à un moment donné.

Les approches sensibles au contexte ont encore de sérieuses difficultés d'adaptation et de génération d'interfaces dynamiques afin de répondre aux besoins des utilisateurs. Ces approches ne sont pas formelles et ne couvrent pas la validation de l'interface utilisateur. En comparaison avec les approches proposées, vues dans le premier chapitre, notre architecture est centrée sur la modélisation du contexte et la spécification de l'interface générée. Le choix de la méthode à utiliser pour la modélisation du contexte est très important. En effet, les informations contextuelles ont un impact direct sur l'interface générée notamment dans les domaines critiques, ce qui justifie l'utilisation de RdP dans notre approche.

3.5 Conclusion

Ce chapitre a été consacré à la présentation de notre approche pour la spécification formelle et la génération d'interfaces sensibles au contexte dans un environnement ubiquitaire. L'approche proposée est composée de six parties à savoir : la détection des données contextuelles, l'analyse du SHM et de la tâche de l'utilisateur, la modélisation de la tâche et de l'utilisateur, la détection de la situation courante, la spécification de l'interface utilisateur et la génération de l'interface.

Cette approche vise à satisfaire les exigences d'un environnement ubiquitaire présentées

au niveau du premier chapitre, et à couvrir toutes les étapes nécessaires depuis l'analyse du SHM sensible au contexte jusqu'à la génération de l'interface mobile.

Les outils et méthodes adoptés dans chaque phase ont été également présentés. En effet, pour l'étape d'analyse du SHM, nous avons utilisé UML comme langage de modélisation. Pour la phase de modélisation, nous avons convenu de travailler avec les réseaux de Petri et nous avons détaillé le méta-modèle utilisé. Tous nos modèles ont été par la suite traduits en PNML afin de déduire la situation courante de l'utilisateur et de spécifier les composantes graphiques de l'inerface.

La faisabilité de cette démarche fait l'objet du chapitre suivant à travers une étude de cas relative à une application médicale.

chapitre 4

Etude de cas : Application DiaMon pour le suivi des patients diabétiques

4.1 Introduction

Dans le chapitre précédent, nous avons présenté et détaillé les différentes étapes de l'approche préconisée, pour la spécification et la génération d'interfaces sensibles au contexte. Nous proposons de montrer dans ce chapitre, son applicabilité sur un cas d'application médicale.

Pour ce faire, nous commençons, dans une première partie, par la présentation de l'application médicale en question et sa spécification fonctionnelle et technique. Nous passons, par la suite, à l'application de toutes les étapes de la démarche dans l'ordre proposé dans le chapitre précédent et nous présentons les résultats obtenus.

4.2 Présentation du domaine d'application

Afin de mettre en évidence notre approche, nous avons choisi de travailler sur le domaine médical et plus particulièrement sur la maladie du diabète. En effet, le domaine médical est assez critique et fortement dynamique. Une erreur au niveau de l'interface utilisateur peut aboutir à des résultats catastrophiques allant jusqu'au décès du patient.

Le diabète est une maladie qui prend de l'ampleur depuis quelques années. En effet, on estime que 347 millions de personnes sont diabétiques dans le monde. En 2004, on estimait que 3,4 millions de personnes étaient décédées des conséquences d'une glycémie élevée à jeun

[World, 2009]. En 2010, le nombre de décès a été comparable.

Le diabète est une maladie chronique qui apparaît lorsque le pancréas ne produit pas suffisamment d'insuline ou que l'organisme n'utilise pas correctement l'insuline qu'il produit. L'insuline est une hormone qui régule la concentration de sucre dans le sang. L'hypoglycémie, ou concentration sanguine basse de sucre, est un effet fréquent du diabète non contrôlé qui conduit avec le temps à des atteintes graves de nombreux systèmes organiques et plus particulièrement des nerfs et des vaisseaux sanguins [Danaei *et al.*, 2011].

Il existe 3 types de diabètes à savoir : *(i)* le diabète de type 1 (précédemment connu sous le nom de diabète insulinodépendant ou juvénile) est caractérisé par une production insuffisante d'insuline et exige une administration quotidienne de cette dernière, *(ii)* le diabète de type 2 : Le diabète de type 2 (précédemment appelé diabète non insulinodépendant ou diabète de la maturité) résulte d'une mauvaise utilisation de l'insuline par l'organisme et *(iii)* le diabète gestationnel qui représente une hyperglycémie apparue ou décelée pour la première fois pendant la grossesse.

Afin de réduire le nombre de décès liés à une hypoglycémie et avec l'avènement de l'informatique mobile, plusieurs travaux de recherche ont vu le jour. En effet, l'objectif global de ces travaux est de bien exploiter la technologie mobile afin de proposer d'éventuelles solutions pour le suivi des patients diabétiques. Nous supposons que ces derniers se trouvent dans un environnement ubiquitaire contenant plusieurs types de capteurs qui émettent sans cesse une quantité énorme d'information. Ces travaux de recherche peuvent être décomposés en deux catégories : des applications destinées aux patients et d'autres destinées aux personnels médicaux (médecins/infirmiers).

L'objectif des applications destinées aux patients, est d'amener progressivement les diabétiques vers une auto-régulation de leur glycémie sans avoir besoin d'une intervention externe. En outre, plusieurs auteurs ont proposé des approches pour l'auto-surveillance du taux de glucose chez le patient. Ces approches ont été considérées comme étant des thérapies modernes pour les diabétiques [Liebl *et al.*, 2013]. Muni de capteurs biologiques, le système surveille continuellement la glycémie du patient.

D'autres auteurs se sont penchés sur la quantité d'insuline injectée par le patient en proposant des systèmes d'auto-ajustement du taux d'insuline dans le corps du patient en fonction de sa glycémie à un instant donné [Reichel *et al.*, 2013]. Partant du même principe, un système nommé « pancréas artificiel » a vu le jour. En effet, plusieurs chercheurs dans le domaine de la médecine et de l'informatique ont collaboré pour proposer un système informatique pour la gestion du taux de glucose dans le sang contrôlé par le patient. Parmi ces applications, nous pouvons citer « l'interface graphique d'assistance des diabétiques », une interface mobile

dédiée au « pancréas artificiel ». Cette application est connectée à un serveur web qui assure la réception, le stockage et l'affichage des données. Le taux de glucose dans le sang ainsi que la quantité d'insuline injectée sont présentés à l'aide de diagramme afin de faciliter la lecture et l'interprétation des données enregistrées [Keith-Hynes *et al.*, 2013] [Place *et al.*, 2013].

Afin de tirer profit de la portabilité que peuvent offrir les plateformes mobiles, certaines applications destinées aux corps médical ont vu le jour. Ces applications ont été spécialement conçues pour aider les gens du domaine à bien accomplir leurs tâches. Elles sont considérées comme étant des outils ayant pour but de renforcer et de faciliter les prestations de soins offertes aux patients. Le nombre d'applications parues ces dernières années est rédhibitoire par rapport à l'ampleur de cette maladie dans le monde. Nous avons tout de même trouvé quelques exemples d'applications destinées aux diabétologues.

« Diabetes Diagnostics » est une application mobile qui détecte et diagnostique le type de diabète chez le patient. Basée sur des calculs statistiques de prédiction et sur l'avis de divers experts, l'application génère un cahier des charges pour la classification des patients diabétiques [International, 2016] [Genes, 2010].

« DiAppBetes » est une application mobile destinée aux non-spécialistes qui offre des informations d'ordre générale sur la diabétologie. Elle a pour objectif de *(i)* calculer les doses d'insuline à administrer et d'ajuster la quantité selon le patient, *(ii)* de gérer les cas d'hypoglycémie quand le patient est hospitalisé [HSPL, 2012].

« AAP Essentials Type 2 Diabetes application » : cette application a été spécialement conçue pour les patients atteints de diabète de type 2. Elle a pour objectif de déterminer les doses exactes d'insuline à administrer. Elle offre deux grandes fonctionnalités à savoir « le suivi du patient » et la « détermination du traitement adéquat ». La partie relative au suivi comprend des recommandations sur la façon de vérifier la glycémie au doigt et/ou le taux de HbA1c en fonction du degré de résistance à l'insuline [AAP, 2013].

De nos jours, les médecins utilisent, de plus en plus, les applications médicales mobiles dans leurs pratiques, pour être plus efficaces et efficients. Les patients les utilisent pour surveiller des aspects spécifiques de leur santé afin de combler quelques lacunes dans les soins médicaux. Les médecins et les patients trouvent que les applications mobiles peuvent fournir un moyen rapide et efficace pour l'échange d'informations. Toutefois, nous remarquons que ces applications sont faites afin de gérer statiquement l'état de santé des patients sans tenir compte de leur statut en temps réel. Pour cette raison, nous avons proposé une nouvelle application, nommée DiaMon, destinée aux corps médical pour le suivi des patients diabétiques.

4.3 DiaMon : Système de suivi de l'état des patients diabétiques

Etant donné la rareté des travaux de recherche concernant les systèmes d'aide destinés au corps médical, nous avons choisi de travailler sur la deuxième catégorie des applications destinées aux diabétiques et d'imaginer un système capable d'aider les médecins et infirmiers dans le suivi des patients diabétiques.

4.3.1 Cadre général de l'application

Comme son nom l'indique, l'objectif général de notre application est le suivi des patients diabétiques dans un « Smart Hospital ». Muni de capteurs biologiques implantés sous la peau des patients, le système vérifie périodiquement le taux de glucose chez les patients. Le système doit observer régulièrement l'état du patient afin d'avertir le personnel médical en cas d'intervention.

Notre application, nommée DiaMon (Diabetes Monitoring) représentée par la Figure 4.1, est destinée aux médecins et aux infirmiers. Chaque membre du personnel de santé doit disposer d'un terminal mobile (téléphone, tablette, etc.). L'application doit être continuellement connectée à notre système. A un moment donné, lors du recueil des informations par notre serveur, si le système remarque une valeur inhabituelle du taux de glucose chez le patient, une alerte doit s'afficher sur les terminaux mobiles du corps médical. L'utilisateur accède alors aux informations relatives au patient pour une meilleure intervention.

Figure 4.1 — Logo de DiaMon.

Plusieurs problèmes peuvent découler d'une telle application. En effet, en cas d'intervention urgente et immédiate, comment avertir l'équipe médicale et de quelle manière pouvons-nous générer des interfaces valides et significatives pour guider le médecin/infirmier à bien

accomplir sa tâche ? En cas de coma hypoglycémique comment pouvons-nous notifier les personnes présentes instantanément pour garantir une intervention urgente. Et si le médecin traitant n'est pas présent à cet instant, comment guider l'utilisateur à bien accomplir sa tâche et de quelle manière l'interface de notre application va-t-elle s'adapter au profil de l'utilisateur et à ses connaissances ?

Etant donné que notre environnement est un environnement ubiquitaire, les interfaces utilisateurs générées doivent être contextualisées. L'interface doit s'adapter au contexte tout en restant valide, ergonomique et simple d'utilisation. Rappelons que le contexte est constitué du triplet *<utilisateur, plateforme et environnement>*. Par conséquent, l'application doit en premier lieu, identifier le profil de l'utilisateur tout en tenant compte de ses préférences ainsi que de son activité courante. En second lieu, le système devra détecter la plateforme utilisée afin d'ajuster et d'afficher les composantes graphiques de l'interface, et, en dernier lieu, le système devra prendre en considération l'environnement dans lequel évolue l'utilisateur de l'interface, à savoir, sa géo-localisation ainsi que le temps. De plus, les composantes graphiques de l'interface devront s'adapter en fonction de la valeur du taux de glucose.

Les interfaces générées doivent être valides et doivent présenter des informations fiables et pertinentes. Leur adaptation doit être instantanée. Aucune erreur n'est tolérée au niveau de l'interface. En effet, les erreurs au niveau de la spécification et de la modélisation des domaines critiques peuvent, dans certain cas, causer la mort du patient.

Afin de démontrer la faisabilité de notre approche, nous allons tester notre application sur un échantillon comportant une dizaine de pharmaciens et de médecins (internes et résidents). Nous allons proposer quelques scénarios, poser plusieurs hypothèses et définir des questions auxquelles notre échantillon devra répondre afin d'établir une étude comparative qui porte sur la fiabilité de nos interfaces ainsi que sur l'avantage des interfaces contextualisées par rapport aux interfaces statiques. Mais avant de commencer cette étude, nous allons présenter quelques connaissances médicales afin de mieux interpréter les résultats par la suite.

4.3.2 Diabète type 2 : signes et traitement curatif

Le diabète de type 2 est une maladie qui atteint généralement les personnes à partir de l'âge de 40 ans. Mais de nos jours, nous voyons de plus en plus de personnes atteintes de diabètes type 2 à partir de l'âge de 25 ans. Ceci est dû aux mauvaises règles hygiéno-diététiques [World, 2009].

Les signes précurseurs du diabète de types 2 réside dans le fait que l'insuline produite n'est pas suffisante pour le besoin de la personne. Le pancréas fonctionne normalement mais

le sucre reste dans le sang et ne pénètre pas dans les cellules sanguines. Les signes cliniques du diabète sont décrits par un taux de glucose dans le sang <2g/l pour n'importe quelle heure dans la journée.

Pour le suivi d'un patient diabétique, plusieurs analyses sont réalisées. Certaines analyses sont demandées tous les ans et d'autres tous les trois mois. Les analyses demandées sont :

- Bilan lipidique : Contient essentiellement la valeur du : *(i)* LDL-cholestérol (« mauvais » cholestérol) qui doit être <1.5g/l pour un diabétique. Le cholestérol est transporté vers les cellules. S'il est en excès, il se dépose sur les artères. *(ii)* : HDL-cholestérol (« bon » cholestérol) qui doit être >0.35g/l pour une personne normale. Pour un diabétique, cette valeur diminue. Le HDL-cholestérol collecte le cholestérol en excès dans le sang et le transporte au foie pour son élimination. Et la valeur de la *(iii)* triglycéride qui doit être <2g/l pour une personne normale. Cette valeur augmente pour une personne diabétique ;
- Créatinine : Elle représente le déchet métabolique produit par l'organisme. Elle traduit l'état fonctionnel des reins. Le dosage de la créatinine se fait dans le sang et dans les urines. Les valeurs normales de créatinine dans les urines sont : pour un homme de 1200-2000 mg/ 24h et pour une femme de 900-1800 mg/24h. Ces valeurs augmentent chez une personne diabétique ;
- Clairance de la créatinine : C'est le rapport entre le débit urinaire de la créatinine et sa concentration dans le sang. Les valeurs normales de la clairance chez un homme est de 7-13 mg/l et chez une femme de 6-11mg/l. Ces valeurs augmentent pour une personne diabétique ;
- Electrocardiogramme ECG : Il sert à enregistrer l'activité électrique du cœur. Il est représenté sous forme de lignes tracées sur une feuille avec des pics et des creux qui montrent le passage de l'activité électrique dans les différentes parties du cœur. L'ECG peut se faire au repos, à l'effort ou sur 24h. Pour un diabétique, l'ECG est pratiqué au repos ;
- Hémoglobine glyquée Hba1c : Elle représente le meilleur indice de suivi des patients diabétiques (tous les 2 à 3 mois). Sa valeur varie selon le type du patient et le type de traitement administré. Pour une personne en bonne santé, sa valeur varie entre 4 et 6%. Pour un diabétique traité par un antidiabétique orale : inférieur à 6.5 ou 7%, traité par insuline : inférieur à 7% et pour un sujet très âgé la valeur doit être inférieure à 8% ;
- Micro-albuminurie : Elle représente les analyses des urines des 24H. Sa valeur ne doit pas dépasser les 300 mg/24H ;
- Fond d'œil : Cet examen permet de voir la rétine (sa partie centrale appelée Macula et les nombreux vaisseaux qui la parcourent). Chez les sujets diabétiques on constate

l'apparition de la rétinopathie diabétique : les vaisseaux sanguins au niveau de la rétine se bouchent. Cet examen est représenté sous forme de photos ;

– Tension artérielle : Elle est composée généralement de 3 chiffres : A/B/C avec : A étant le maximum de la tension au niveau de l'artère (tension systolique). B : le minimum de la tension de l'artère (tension diastolique) et C la fréquence cardiaque. Chez une personne diabétique sa valeur doit être inférieure à 130/80 [Chaker et Mchirgui, 2013].

Une personne atteinte de diabète de type 2 doit suivre un traitement bien précis. Ce traitement se fait selon plusieurs étapes. Tout d'abord, le médecin traitant prescrit des Biguanides avec des règles hygiéno-diététiques. Les Biguanides représentent des insulino-sensibilisateurs qui diminuent la production hépatique de glucose comme la Metformine (Glucophage1000mg). Si la valeur du Hba1c est $> 7\%$ alors nous associons un deuxième Anti Diabétique Orale (ADO) ou nous administrons l'insuline. Le deuxième ADO peut être :

– Un glitazone : augmente l'insulino-sensibilité comme l'Actos 15mg ;
– Un sulfamide : stimule l'insulino-sécrétion des cellules pancréatiques (Sucrazide, Amarel, Diamicron).

Si la bithérapie est un échec (Hb1ac>7%), alors le médecin a le choix entre une trithérapie (ajout d'un troisième médicament) ou une insuline basale. Si la trithérapie représente un échec, alors le médecin prescrit une insulinothérapie intensive. Cette thérapie est basée soit sur l'insuline classique (Astrapid, Insulatard, Mixtard) soit sur les analogues de l'insuline (Lantus, Apidra, Novo-Rapid).

Pour un sujet atteint de diabète de type 2, il y a un grand risque d'hypoglycémie. Cette dernière représente une concentration de sucre dans le sang anormalement basse. L'hypoglycémie peut aboutir à un coma hypoglycémique pouvant générer le décès du patient. Nous pouvons reconnaître une hypoglycémie à travers plusieurs signes [Chaker et Mchirgui, 2013] :

– Signes neurovégétatifs : ils sont représentés sous forme de signes digestifs (nausées, crampes gastriques) et signes cardiovasculaires (palpitation, pâleur, sueurs froides...),
– Signes neuroglycopéniques : ils sont liés au système nerveux. Ils peuvent être sous forme de fatigue physique, troubles neurologiques (vertige, mal de tête, tremblements, troubles de la vision, ralentissement intellectuel...), troubles psychiatriques, troubles du comportement, paralysie faciale ou locale.

Une hypoglycémie peut entraîner un coma hypoglycémique. Cet état survient suite à une installation brutale d'un état d'agitation, d'une contracture musculaire, de réflexes vifs et de sueurs profuses.

Le traitement curatif d'une hypoglycémie dépend de l'état du patient et de son taux de glycémie. Deux cas de figure se présentent : patient conscient ou inconscient (Tableau 4.1)[Chaker et Mchirgui, 2013].

Tableau 4.1 — Traitement curatif d'une Hypoglycémie

Hypoglycémie avec trouble de la conscience	**Hypoglycémie modérée**
Condition : Taux de glycémie entre 0.3 et 0.4 g/l Patient inconscient ou peu coopérant	Condition : Taux de glycémie entre 0.4 et 0.7 g/l Patient conscient
Traitement : il existe deux types de traitements : 1. Perfusion en intraveineuse lente de 30 à 50 ml de sérum glucosé à 30% relayé par une perfusion lente de sérum glucosé à 10% jusqu'au réveil du patient. Un apport en sucre lent (pain) est alors préconisé. 2. Injection d'une ampoule de 1 mg de glucagon en intramusculaire ou sous-cutanée. Apport en glucides par voie orale ou voie veineuse : perfusion lente de sérum glucosé à 10%.	Traitement : Resucrage par voie orale : 15g de sucre rapide (3 morceaux de sucre dilués dans l'eau ou du jus industriel), suivie d'un apport en sucre lent (20g de pain), Attendre 15 minutes et refaire le resucrage.

Le traitement des deux types d'hypoglycémie présente un apport en sucre lent. Dans ce cas, il faut prendre en considération les diabétiques allergiques au gluten.

Le gluten est une masse protéique élastique et visqueuse qui se trouve dans les graines de plusieurs céréales comme le blé. Les symptômes de cette allergie sont définis par des signes gastriques (douleurs abdominales, ballonnements, diarrhées, nausées, vomissements...), des crampes musculaires, des douleurs des os et des articulations ainsi que des crises d'asthme. Il faut alors remplacer le pain par du pain sans gluten à base de farine de riz, de maïs ou de blé noir.

Un patient diabétique peut aussi être allergique au fructose. Cette substance est une composante du sucre. Dans ce cas, il faut éviter les trois morceaux de sucre et le jus de fruit en cas d'hypoglycémie modéré. Les symptômes de cette allergie sont définis par des douleurs abdominale, la constipation, des maux de tête, la fatigue, l'installation d'un état dépressif et des troubles du sommeil. Le sucre doit alors être remplacé par la saccharine, l'aspartame, l'acésulfame, le maltose ou le sirop de Malt [Chaker et Mchirgui, 2013].

Après avoir défini et présenté la maladie du diabète, nous allons à présent démontrer la faisabilité de notre approche sur cet exemple.

4.4 Application de l'approche sur notre exemple

L'approche proposée débute par la détection des données contextuelles pour ensuite procéder à l'analyse du SHM sensible au contexte. Une fois l'analyse faite, il faut modéliser par la suite le contexte ainsi que la tâche de l'utilisateur. L'approche préconise par la suite la détection de la situation courante pour enfin pouvoir spécifier et générer l'interface de l'utilisateur. Toutes ces étapes sont illustrées ci-dessous sur le cas médical étudié.

4.4.1 Détection des données contextuelles

Pour notre exemple, nous avons utilisé plusieurs types de capteurs afin de détecter les données contextuelles. Nous avons essentiellement utilisé des capteurs physiques afin de capter les données géographiques de l'utilisateur et le taux de glucose pour le patient.

Afin de détecter la position exacte de l'utilisateur, nous avons utilisé le service de géolocalisation présent dans n'importe quel Smartphone. Ce service fait appel à deux technologies à savoir le GPS qui représente un repérage par satellite et le GSM (utilisation des ondes radio).

Concernant la détection du taux de glucose chez le patient nous avons utilisé le « e-Health Sensor Shield » [cooking hacks, 2013]. Le choix du capteur biologique était un vrai challenge pour nous. En effet, il existe sur le marché médical plusieurs solutions destinées aux produits d'autorégulation du taux d'insuline (pompe à insuline). Plusieurs laboratoires pharmaceutiques mettent sur le marché des kits dont la majorité utilise des glucomètres comme biocapteur. Nous pouvons citer comme exemple le laboratoire SANOFI ou encore MEDTRONIC [Biotech, 2013]. Le problème avec ces multinationales c'est que ces solutions sont extrêmement gardées puisqu'elles sont directement mises en vente. Donc il était impossible pour nous d'avoir le biocapteur sans acheter le kit tout entier qui coutait énormément cher. Nous nous sommes donc orientés vers des biocapteurs qui étaient utilisés à des fins expérimentales. Le « e-Health Sensor Shield » permet aux utilisateurs de développer des applications médicales. Les données recueillies peuvent être transmises par Wifi, 3G, GPRS ou par Bluetooth. Après la capture des données vient alors l'étape d'analyse du SHM sensible au contexte.

4.4.2 Analyse du SHM et de la tâche de l'utilisateur

4.4.2.1 Analyse du SHM

Nous allons tout d'abord identifier les différents utilisateurs de DiaMon pour pouvoir ensuite décomposer notre système.

Les utilisateurs de DiaMon peuvent être soit :

- Une secrétaire : elle peut saisir les informations personnelles du patient et ses analyses, consulter le dossier personnel du patient ;
- Un infirmier, un urgentiste, un médecin généraliste ou un diabétologue (médecin spécialiste) : ils peuvent consulter ou mettre à jour le dossier du patient et intervenir en cas de complication.

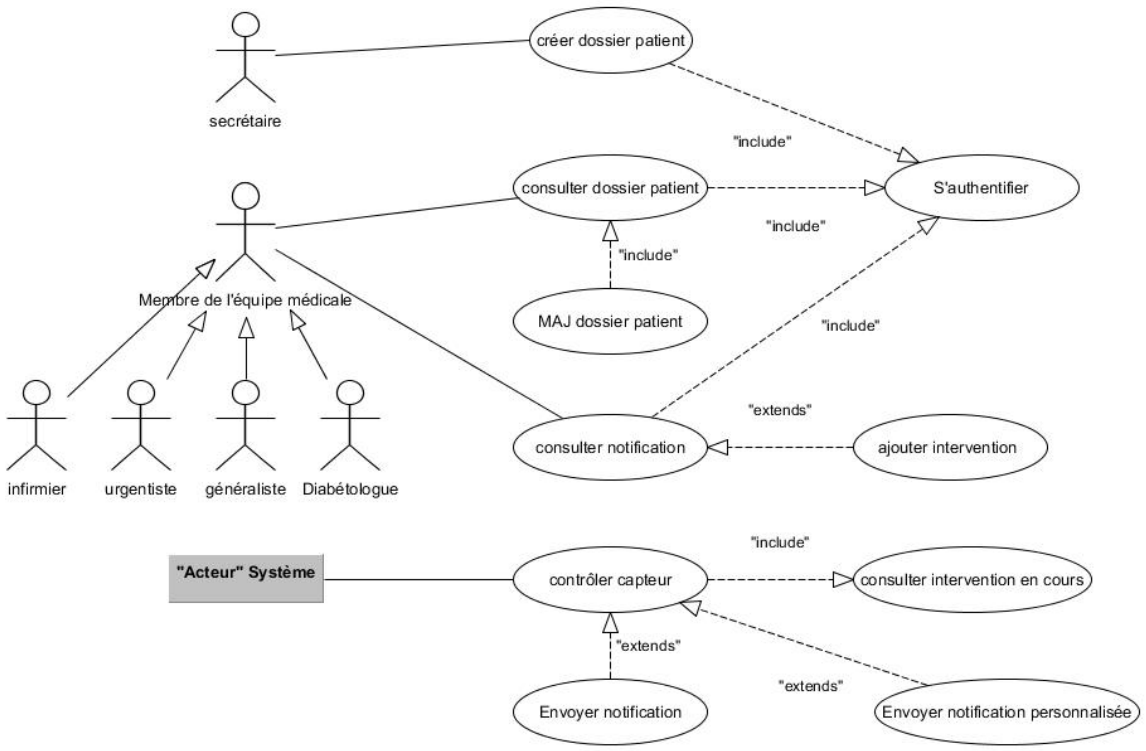


Figure 4.2 — Diagramme de cas d'utilisation général.

En étudiant les différents profils des utilisateurs de notre application, nous avons constaté que la fonctionnalité principale du système proposé consiste en l'intervention en cas de complication et la mise à jour du dossier du patient (consultation, ajout, modification et suppression). De ce fait, nous allons appliquer une analyse UML afin de déceler les principaux

sous-systèmes. Le diagramme de cas d'utilisation proposé en figure 4.2 décrit l'interaction des différents utilisateurs avec les sous-systèmes.

En analysant les différents cas d'utilisation, nous pouvons déduire l'existence de deux fonctionnalités principales à savoir la mise à jour (sous-système 1 SS1) et l'intervention en cas de complication (sous-système 2 SS2). L'intervention est décrite par le cas d'utilisation « ajouter intervention ». Ce dernier est déclenché par le cas d'utilisation « consulter notification ». La notification est envoyée par notre acteur « système » qui contrôle périodiquement le biocapteur implanté chez le patient.

De cette analyse, nous pouvons déduire les variables du système. Ces variables peuvent être soit des variables de commande (***Cmd***), des variables d'information ou des variables de consigne ***(Cs)***. Les variables d'informations peuvent être soit des variables entrées par l'utilisateur ***(In)*** ou des variables de sortie ***(Out)***.

a- Variables d'informations Entrées par l'utilisateur

In.IdP : identifiant du patient.

In.IdU : identifiant de l'utilisateur.

In.PU : mot de passe de l'utilisateur.

b- Variables d'informations de type sorties

Out.Gl : taux de glucose à un instant donné.

Out.P.Nom : nom du patient.

Out.P.Prénom : prénom du patient.

Out.P.Sexe : sexe du patient (H/F).

Out.P.Dn : date de naissance du patient.

Out.P.Poids : poids du patient.

Out.P.Prof : profession du patient.

Out.P.Ts : traitement suivi.

Out.P.Mt : nom et prénom du médecin traitant.

Out.P.LA : Liste des analyses du patient : bilan lipidique, créatinine, glycémie à jeun, ECG, HbA1c, micro-albuminurie, fond d'œil, tension.

Out.P.Patho : liste des pathologies/allergies du patient.

Out.P.Etat : état du patient (conscient ou inconscient).

Out.Type.U : profil de l'utilisateur.

Out.Qs.30 : quantité sérum glucosé 30%.

Out.Qs.10 : quantité sérum glucosé 10%.

Out.Qsr : quantité sucre rapide.

Out.Qsl : quantité sucre lent.

Out.Qsa : quantité saccharine.

Out.QPSG : quantité pain sans gluten.

c- Variables d'informations de type consignes

Cs.Pain : consigne pour donner du pain au patient.

Cs.Resucre : Consigne pour faire un resucrage si besoin.

Cs.TA : consigne pour afficher le type d'allergie.

d- Variable de commandes

Cmd.AMT : commande d'appel du médecin traitant.

Cmd.FI : commande de fin d'intervention.

Après avoir identifié les différents sous-systèmes ainsi que leurs variables, nous allons à présent analyser la tâche de l'utilisateur.

4.4.2.2 Analyse de la tâche

Dans cette phase, nous allons analyser la tâche de l'utilisateur lors du fonctionnement normal de notre système. Tout ce qui est dysfonctionnement et fonctionnement anormal du système n'est pas pris en considération dans ces travaux. De ce fait, en cas de fonctionnement normal du système, la tâche de l'utilisateur varie selon l'état de santé du patient et le niveau d'expertise de l'utilisateur. En effet, le patient peut être soit en état normal, soit en état anormal c'est-à-dire qu'il présente une hypoglycémie ou encore en état critique (coma hypoglycémique). L'utilisateur peut être soit un diabétologue ou encore un non expert du domaine (infirmier, médecin généraliste ou urgentiste).

a- Patient en état normal

Si le taux de glucose (GL) est normal (GL>0.7 g/l) alors l'utilisateur (diabétologue ou pas) peut consulter ou mettre à jour le dossier du patient (SS1). Cet état peut être reflété au niveau de l'interface par des variables décrivant le dossier du patient. L'état normal du patient peut être représenté par les informations suivantes : Out.P.Nom, Out.P.Prénom, Out.P.Sexe, Out.P.Dn, Out.P.Poids, Out.P.Prof, Out.P.Ts, Out.P.Mt, Out.P.LA, Out.P.Patho.

Cet ensemble de variables constitue les besoins informationnels de l'opérateur (BIO).

b- Patient en état anormal : hypoglycémie modérée

Cet état se traduit par le fait que le patient est conscient et qu'il présente un GL compris entre 0.4 et 0.7 g/l. L'utilisateur de DiaMon doit alors intervenir pour réguler le taux de glucose chez le patient (SS2). Dans ce cas, la tâche de l'utilisateur diffère selon son niveau de connaissance (type de l'utilisateur connecté) et selon le type d'allergie du patient.

Si l'utilisateur connecté est un non expert du domaine à savoir : un urgentiste, un infirmier ou un médecin généraliste et si le patient est non allergique alors l'utilisateur doit réaliser :

- Un resucrage rapide par voie orale de 15g de sucre rapide (3 morceaux de sucre dilués dans de l'eau ou du jus industriel) ;
- Un apport en sucre lent à savoir 20g de pain ;
- Attendre 15 minutes et refaire un resucrage si la valeur de GL demeure comprise entre 0.4 et 0.7g/l.

Cet état peut être traduit par les variables suivantes :

- Out.Gl, Out.P.Etat, Out.Type.U afin d'identifier l'état du patient le profil de l'utilisateur connecté ;
- Out.Qsr, Out.Qsl, Cs.Pain, Cs.Resucre afin d'identifier le type d'intervention ;
- Cmd.AMT, Cmd.FI pour contacter le médecin traitant.

Si le patient est allergique au gluten, l'utilisateur accomplit la même tâche en remplaçant le pain par du pain à base de riz, maïs ou du blé noir. La variable relative à la quantité de pain (Out.Qsl) est remplacée par quantité de pain sans gluten (Out.QPSG). Une nouvelle variable de type consigne doit apparaitre afin d'avertir l'utilisateur du type d'allergie (Cs.TA).

Si le patient est allergique au fructose, l'utilisateur accomplit la même tâche en remplaçant le sucre par la saccharine. La variable relative à la quantité de sucre rapide (Out.Qsr) est remplacée par quantité de saccharine (Out.Qsa). La variable de type consigne Cs.TA doit apparaitre afin d'avertir l'utilisateur du type d'allergie.

Si le patient est allergique au gluten et au fructose, l'utilisateur doit remplacer le pain par du pain sans gluten et le sucre par la saccharine.

Si l'utilisateur connecté est un diabétologue, alors son niveau de connaissance lui permet d'accomplir sa tâche sans avoir les détails sur son interface (quantité de sucre rapide/lent). Il doit seulement intervenir en tenant compte du type d'allergie du patient. Cet état peut se traduire par les variables suivantes :

- Out.Gl, Out.P.Etat pour savoir l'état du patient ;
- Cs.TA pour identifier le type d'allergie du patient ;

– Cmd.FI pour signaler la fin de l'intervention.

c- Patient en état critique : coma hypoglycémique

Une hypoglycémie modérée peut aboutir à un coma hypoglycémique si le taux de GL devient égal ou inférieur à 0.4 ou 0.3g/l. Afin de remédier à cet état l'utilisateur devra faire :

– Une perfusion en intraveineuse lente de 30 à 50 ml de sérum glucosé à 30% ;
– Une perfusion lente de sérum glucosé 10% jusqu'au réveil du patient ;
– Apport en sucre lent (pain) pour un patient non allergique et apport en pain sans gluten pour un patient allergique.

Cet état se traduit par les variables suivantes :

– Out.Gl, Out.P.Etat, Out.Type.U afin d'identifier l'état du patient le profil de l'utilisateur connecté ;
– Out.Qs.30, Out.Qs.10 pour déterminer la quantité de sérum à administrer ;
– Out.Qsl ou Out.QPSG pour déterminer le type de pain ;
– Cs.Pain, Cs.TA pour identifier le type d'allergie du patient ;
– Cmd.AMT, Cmd.FI pour contacter le médecin traitant et finir l'intervention.

Si l'utilisateur connecté est un diabétologue, alors les variables relatives aux quantités de sérum disparaissent.

4.4.3 Modélisation du contexte et de la tâche de l'utilisateur

Une fois l'analyse faite, l'étape suivante consiste à élaborer le modèle du contexte et de la tâche afin de déduire les BIO pour chaque état. Cette modélisation est basée sur une composition de structures élémentaires de RdP, comme expliqué dans le chapitre 3.

4.4.3.1 Modélisation du contexte

En accord avec la définition du contexte exposée au premier chapitre, nous avons convenu de considérer le contexte comme étant le triplet *<utilisateur, plateforme, environnement>*. Chaque composante sera modélisée par son propre réseau afin de faciliter la validation partielle du modèle globale [Riahi *et al.*, 2013].

a- RdP utilisateur

Il a pour objectif d'identifier le profil de l'utilisateur connecté (secrétaire, infirmier, médecin généraliste, urgentiste ou diabétologue). Le marquage de ce réseau déterminera le profil de l'utilisateur connecté à un moment donné (Figure 4.3). Nous avons utilisé aprés la place « utilisateur connecté » une composition alternative. Nous avons utilisé dans les transitions

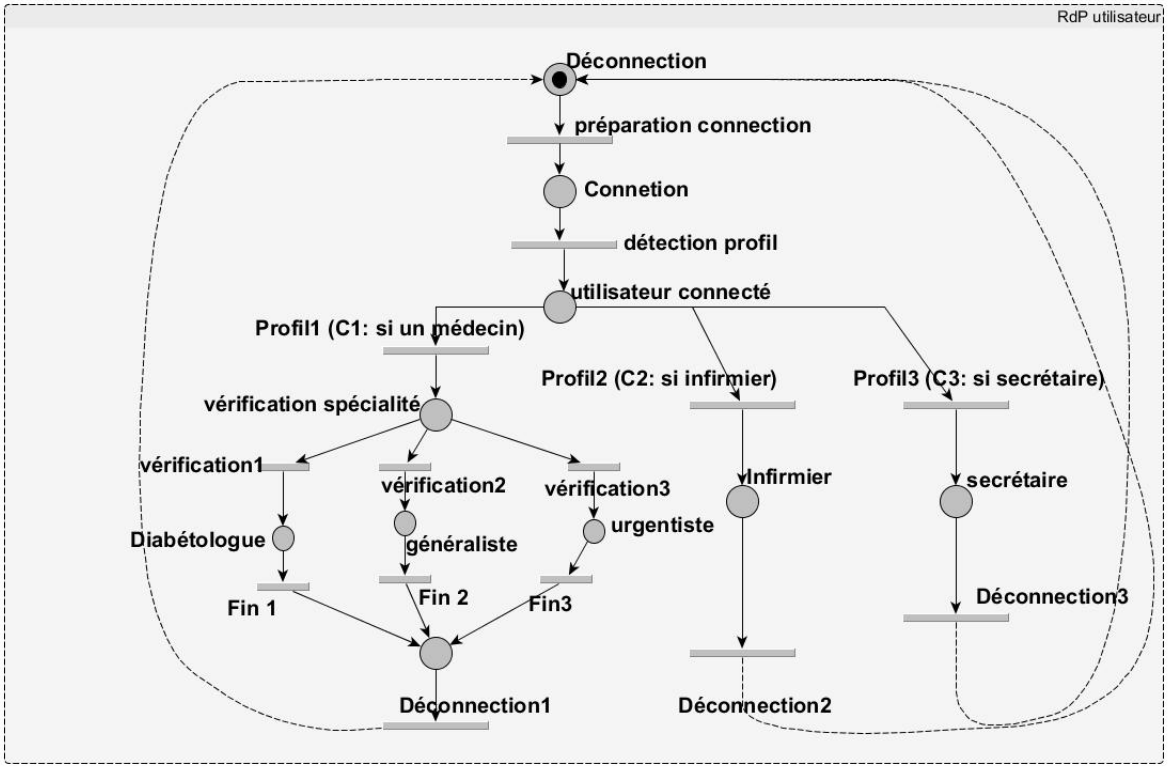


Figure 4.3 — RdP utilisateur.

« profil1, profil2, profil3 » des conditions afin de savoir l'identité de l'utilisateur connecté.

b- RdP plateforme

Ce RdP décrit les différentes plateformes qui peuvent héberger notre application. Nous allons considérer les smartphones, les tablettes et les PC (Figure 4.4).

c- RdP environnement

Ce RdP décrit les différentes valeurs de notre environnement à un instant donné. Lors de l'ouverture de DiaMon, plusieurs capteurs interceptent en parallèle la valeur du taux de glucose chez le patient, les coordonnées géographiques de l'utilisateur et le temps. L'utilisateur peut être soit dans l'hôpital, dans son cabinet, chez lui ou bien dans la rue. Les jetons présents dans les différentes places détermineront à un moment donné la situation courante de l'utilisateur (Figure 4.5).

Ce modèle doit être à l'écoute de tout changement qui peut survenir à l'environnement. Cette action est possible grâce aux transitions « détection changement 1 » et « détection changement 2 ».

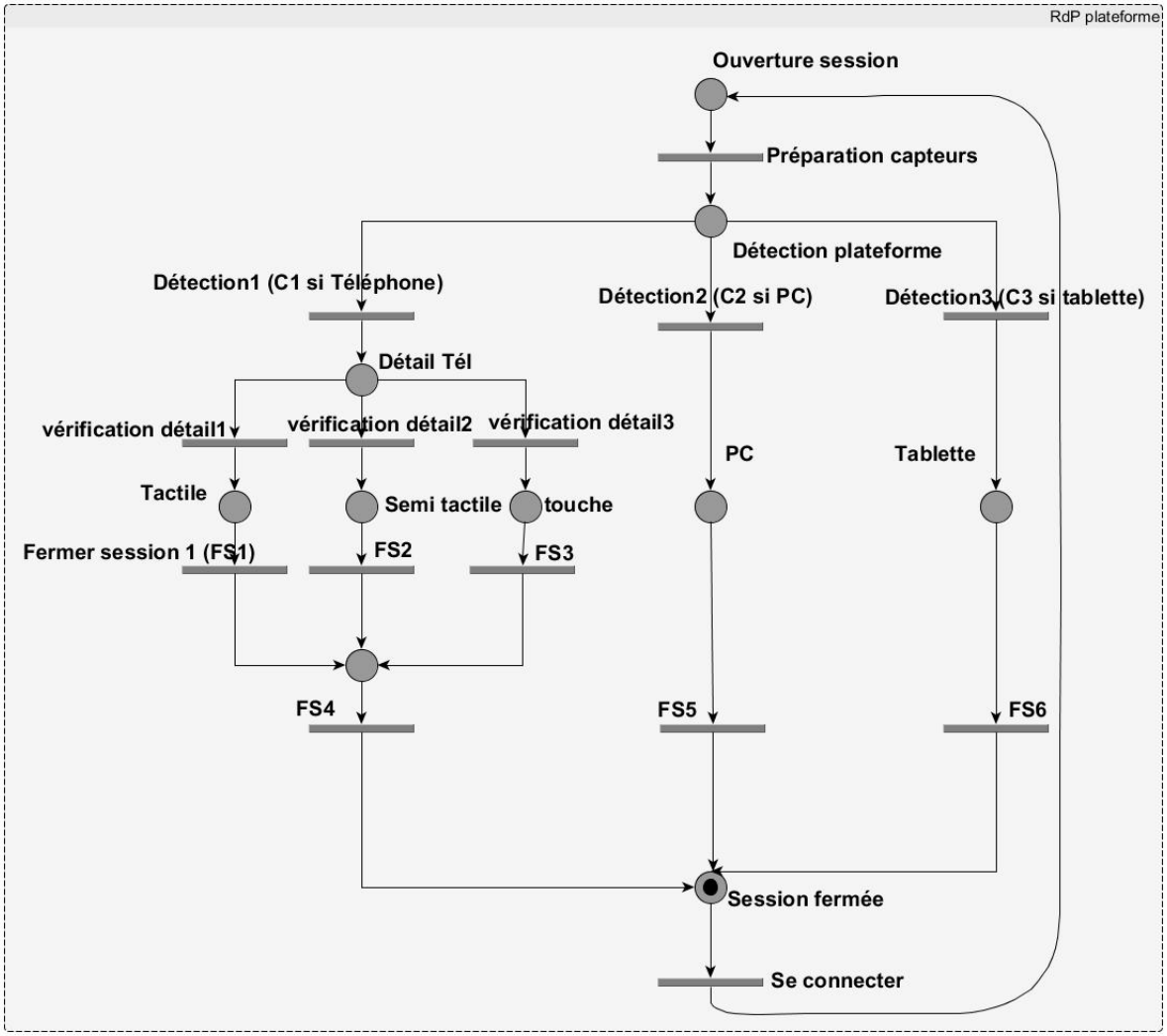


Figure 4.4 — RdP Plateforme.

Le marquage des trois réseaux déterminera l'état de notre SHM. Il faut à présent modéliser la tâche de l'utilisateur.

4.4.3.2 Modélisation de la tâche de l'utilisateur

Comme nous l'avons expliqué dans la section précédente, la tâche de l'utilisateur diffère selon l'état du patient. La figure 4.6 décrit les différentes interventions relatives à chaque valeur du taux de glucose (GL). Le modèle global de la tâche est constitué d'une composition alternative qui dépend de l'état du patient et de plusieurs compositions séquentielles qui décrivent l'enchainement des actions pour chaque intervention.

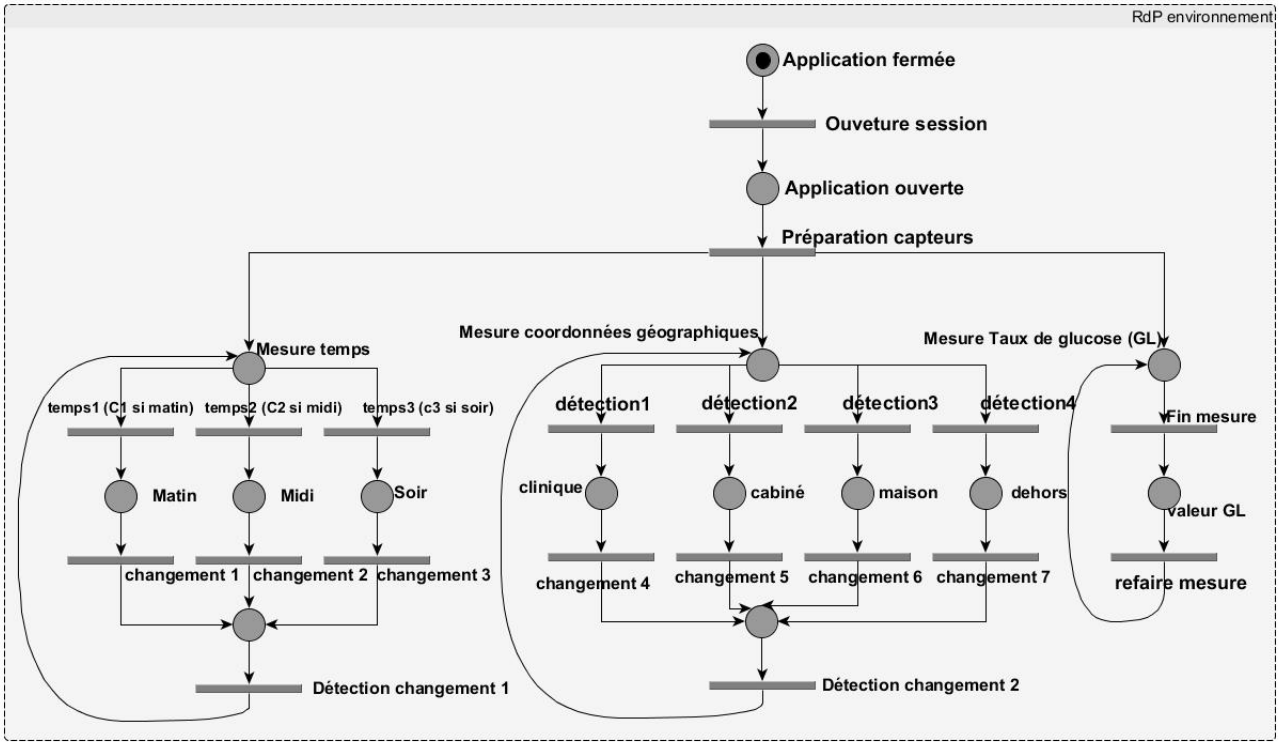


Figure 4.5 — RdP environnement.

4.4.4 Déduction de la situation courante (Ci,Ti)

Nos modèles étant prêts, l'étape qui suit représente la déduction des couples (Ci, Ti). Avant toute chose, il faut tout d'abord convertir nos modèles en PNML. L'objectif de cette étape est de traduire le modèle du SHM sensible au contexte en un langage compréhensible par la machine pour pouvoir l'exploiter par la suite. Pour ce faire, PetriNetEditor a été conçu et développé. PetriNetEditor représente un outil logiciel graphique pour *(i)* la génération automatique du code PNML pour un RdP donné, et *(ii)* la création et l'affichage d'un RdP à partir de son code PNML.

La création d'un RdP passe par trois étapes : la création de la place, de la transition et de l'arc qui lie la place à la transition (Figure 4.7).

Une fois le réseau créé, le fichier PNML est généré automatiquement. Prenons comme exemple un réseau basique qui contient deux places et une transition (Figure 4.8).

Le code PNML généré correspond à celui visible en Figure 4.9.

De ce fait, en utilisant cet outil, nous avons pu convertir tous nos modèles en du PNML. La

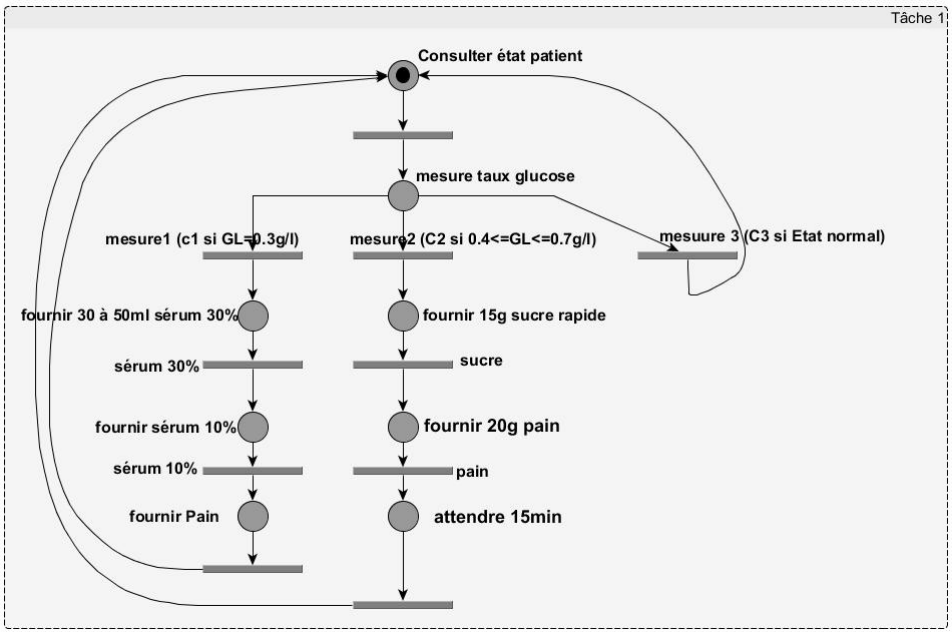


Figure 4.6 — Tâche de l'utilisateur.

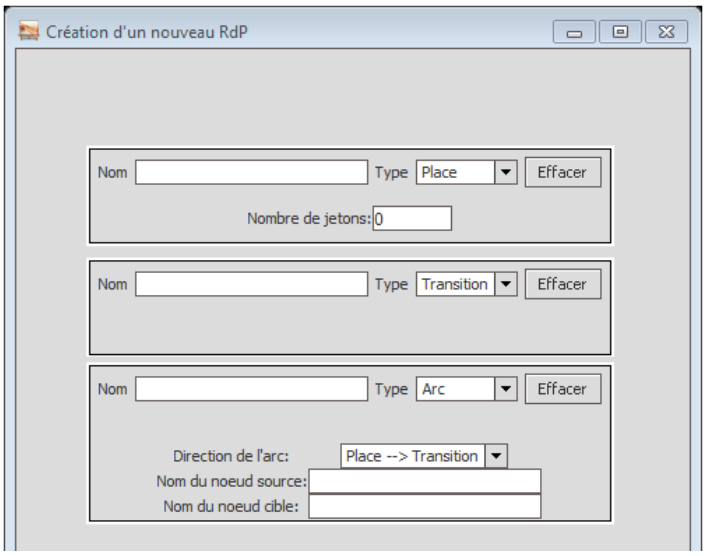


Figure 4.7 — Création d'un RdP.

figure 4.10 montre un exemple du code PNML relatif aux réseaux modélisant l'environnement et la plateforme.

Nos fichiers PNML étant prêt, nous avons utilisé des algorithmes de parcours de fichiers

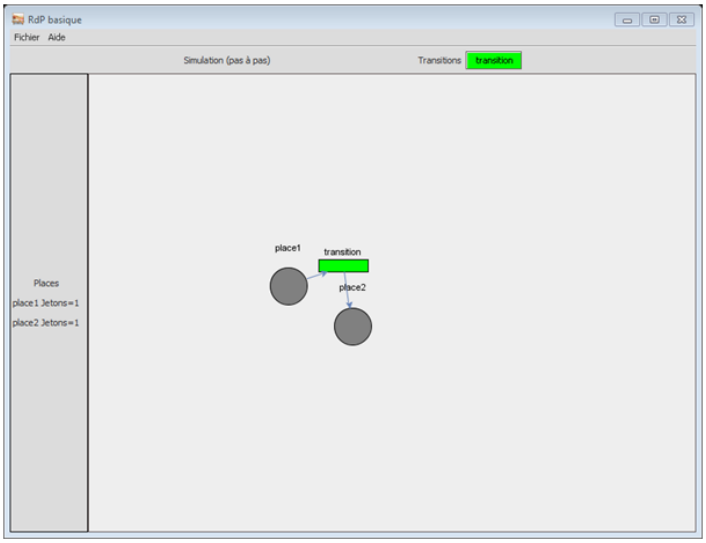


Figure 4.8 — RdP basique.

```
<pnml xmlns="http://www.informatik.hu-berlin.de/top/pnml/ptNetb">
<net>
<place><name><text>place1</text></name> <!-- Nom de la place -->
<token><text>0</text></token></place> <!-- Valeur du jeton -->
<place><name><text>place2</text></name>
<token><text>1</text></token></place>
<transition>
<name><text>transition1</text></name> <!-- Nom de la transition -->
</transition>
<arc id="a2" source="place1" target="transition1"/>
<arc id="a3" source="transition1" target="place2"/> <!-- Définition de l'
    arc -->
</net>
</pnml>
```

Figure 4.9 — PNML généré.

XML afin de déduire la valeur du triplet *<utilisateur, plateforme, environnement>*. Cette valeur se trouve dans chaque balise *<token>* de la balise *<place>* dans les fichiers PNML de l'utilisateur, de la plateforme et de l'environnement. Si cette valeur est égale à 1 alors le jeton appartient à cette place et par conséquent la valeur est déduite. Par exemple, prenons

```
<!-- Platform's PNML -->
<pnml xmlns="http://www.informatik.hu-berlin.de/top/pnml/ptNetb">
<net>
<!-- identification of the connected platform -->
<place><name><text>mobile_phone</text></name>
<token><text>1</text></token></place>
<!-- the connected platform is a screen touch phone -->
<place><name><text>screen_touch</text></name>
<token><text>1</text></token></place>
<place><name><text>tablet</text></name>
<token><text>0</text></token></place>
<transition><name><text>is_a_screen_touch</text></name></transition>...
<arc id="a2" source="mobile_phone" target="details_phone"/>
<arc id="a3" source="details_phone" target="screen_touch"/>...
</net>
</pnml>
<!-- Environment PNML -->
<pnml xmlns="http://www.informatik.hu-berlin.de/top/pnml/ptNetb">
<net>
<!-- identification of the environment's parameters -->
<!-- identification of the time -->
<place><name><text>morning</text></name>
<token><text>1</text></token></place>
<!-- identification of the user's location -->
<place><name><text>hospital</text></name>
<token><text>1</text></token></place>
<!-- identification of the patient's status -->
<place><name><text>hypoglycaemic</text></name>
<token><text>1</text></token></place>...
</net>
</pnml>
```

Figure 4.10 — PNML de l'environnement et de la plateforme.

le cas du PNML de la plateforme (Figure 4.10), la valeur de la balise *<token>* de la place *<screen-touch>* est égale 1, cela veut dire que l'utilisateur connecté utilise un Smartphone.

Afin de déduire la tâche adéquate, il faut parcourir à ce stade la BD « contexte, tâche ». En effet, cette base contient toutes les valeurs possibles et imaginables du contexte et à chaque

triplet nous associons une tâche bien particulière. Le tableau 5.1 montre un extrait de la base. Une fois la tâche identifiée, il est nécessaire de spécifier l'interface adéquate à cette situation.

Tableau 4.2 — Extrait de la base « Contexte, Tâche »

Utilisateur	**Plateforme**	**Temps**	**Lieu**	**Taux GL**	**Tâche**
Généraliste	Ecran tactile	Matin	Hôpital	0.6g/l	15g sucre rapide
Généraliste	Ecran tactile	Après-midi	Hôpital	0.6g/l	15g sucre rapide
Infirmière	Tablette	Matin	Hôpital	0.3g/l	50 ml sérum 30%
Infirmière	Tablette	Après-midi	Hôpital	0.3g/l	50ml sérum 30%
Infirmière	Tablette	Soir	Hôpital	0.3g/l	Perfusion sérum 10%

4.4.5 Spécification de l'interface utilisateur

Afin de spécifier les composantes de l'interface, il est primordial d'associer au niveau de chaque état du système les variables d'information nécessaires à son exécution.

Considérons la tâche de l'utilisateur représentée par la figure 1.6 ; nous allons nous intéresser aux états « consulter état patient », « fournir 15g sucre rapide », « fournir 20g pain », « fournir sérum 30% » et « fournir sérum 10% » qui représentent des états relatifs au changement du statut du patient. La liste des BIO associés à chacun de ces états est fournie ci-dessous.

Pour la place « consulter état patient », l'utilisateur a besoin de consulter le dossier médical du patient. Ceci est décrit par les variables : Out.P.Nom, Out.P.Prénom, Out.P.Sexe, Out.P.Dn, Out.P.Poids, Out.P.Prof, Out.P.Ts, Out.P.Mt, Out.P.LA, Out.P.Patho.

Pour les places « fournir 15g sucre rapide », « fournir 20g pain », « fournir sérum 30% » et « fournir sérum 10% » l'utilisateur a besoin de savoir le taux exact de GL, l'état du patient, le type d'allergie et le type d'utilisateur connecté. Ceci se traduit par les variables suivantes : Out.Gl, Out.P.Etat, Out.Type.U, Cs.TA. Si l'utilisateur est non expert alors il faut associer :

- Pour la place « fournir 15g sucre rapide », la quantité de sucre rapide à administrer Out.Qsr et une variable de commande afin d'appeler le médecin traitant Cmd.AMT. Si le patient est allergique au fructose alors Out.Qsr est remplacée par Out.Qsa ;
- Pour la place « fournir 20g pain », la quantité de sucre lent à administrer Out.Qsl et une variable de commande afin d'appeler le médecin traitant Cmd.AMT. Si le patient est allergique au gluten alors Out.Qsl est remplacée par Out.QPSG ;

- Pour la place « fournir sérum 30% », la quantité de sérum à perfuser Out.Qs.30 et une variable de commande afin d'appeler le médecin traitant Cmd.AMT ;
- Pour la place « fournir sérum 10% », la quantité de sérum à perfuser Out.Qs.10 et une variable de commande afin d'appeler le médecin traitant Cmd.AMT.

Si l'utilisateur est un diabétologue, alors toutes ces variables disparaissent (Figure 4.12). Il n'aura besoin que du dossier du patient.

Une fois les BIO déduits et associés au modèle de la tâche, il s'agit de spécifier l'interface. Cette étape est réalisable via l'association des BIO aux objets graphiques de l'interface conformément à des règles ergonomiques formalisées et stockées dans notre base comme le montre la Figure 4.11.

```
If <Outdoor_Use> Then Bright_Screen_Light
If <Hospital_Use> Then Switch_Ringtone_to_Vibrator
If <Parameter>=glucose level AND <Type_Parameter>="Informational"
AND <PlatformType>= "Tablet" Then
Use_Curve_Representation(parameter)
If <Parameter>=glucose level AND <Type_Parameter>="Informational"
AND <PlatformType>= "Phone" Then
Use_Digit_Representation(parameter)
If <Parameter>=Treating Doctor AND <Type_Parameter>="Control"
AND <PlatformType>= "Phone" Then
Use_Button_Representation(parameter)
```

Figure 4.11 — Exemples de règles ergonomiques.

A ce stade, nous pouvons déduire que DiaMon pourrait être composé de trois principales vues :

- Une vue de consultation si l'état du patient est normal ;
- Une vue d'intervention 1 si le patient présente une hypoglycémie modérée ;
- Une vue d'intervention 2 si le patient est en état critique (coma hypoglycémique).

Pour chacune de ces vues, on y trouvera des objets graphiques représentant les BIO associés aux états du modèle de la tâche. Par exemple, pour tout ce qui est variables d'informations et consignes, le système pourrait associer des zones de texte et/ou des labels. Pour les variables de commande, des boutons pourraient être attribués.

4.4.6 Génération de l'interface

Rappelons que cette étape ne concerne pas directement notre travail. En effet, suite aux spécifications obtenues, nous avons créé des maquettes d'interfaces afin de réaliser notre une étude et d'évaluer les interfaces contextualisées. Nous avons testé nos interfaces sur des médecins et nous avons collecté par la suite leur avis. Ces informations ont été divisées en des critères observables et des critères non observables. L'évaluation de nos interfaces a été régie par des critères ergonomiques bien précis à savoir l'utilité et l'utilisabilité.

Les différentes captures d'écran relatives à notre application seront exposées dans le prochain chapitre.

4.5 Conclusion

Nous avons démontré à travers ce chapitre, la faisabilité de notre approche formelle pour la spécification d'interfaces sensibles au contexte, sur un domaine d'application médicale : suivi de patients diabétiques. Nous avons proposé une nouvelle application nommée DiaMon destinée au corps médical afin de soutenir différents types d'acteurs concernés et de faciliter ainsi l'intervention médicale dans les cas les plus critiques.

La première partie de ce chapitre a été donc consacrée à la présentation de notre application. En effet, nous avons exposé le domaine d'application, pour enseuite définir les principes de DiaMon. Nous avons également présenté les signes et les traitements curatifs pour le diabète de type 2. Par la suite, les résultats de l'application des différentes étapes de la démarche, depuis l'étape de dédection des données contextuelles jusqu'à la génération de l'interface, ont été exposés.

Afin de valider nos interfaces utilisateur et notre application, nous avons mené une étude portant sur l'apport des interfaces contextualisées par rapport aux interfaces statiques. Nous avons mené cette étude auprès de médecins et nous avons recueilli leur avis afin de les analyser par la suite. Le chapitre 5 portera donc sur cette étude.

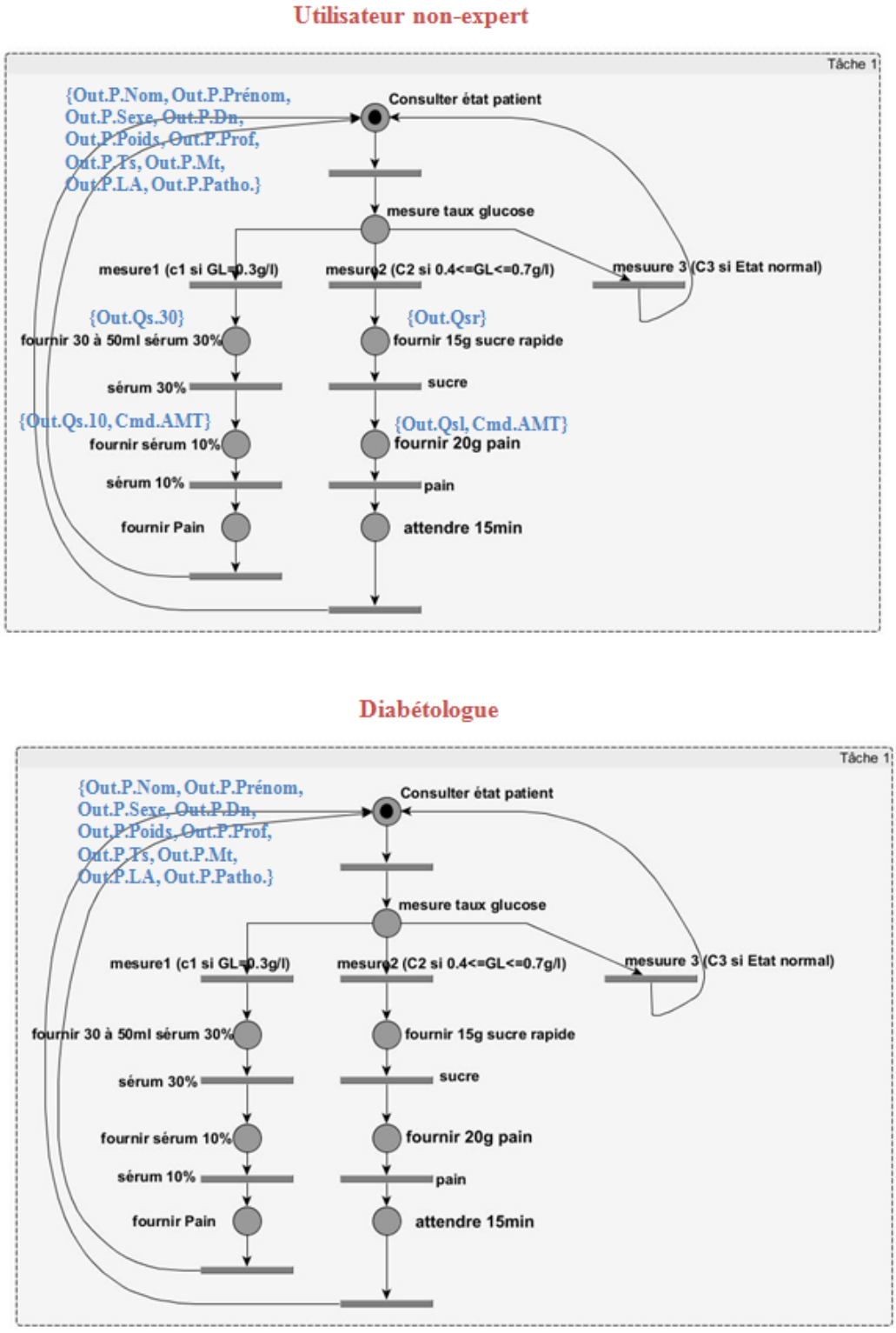


Figure 4.12 — Affectation des BIO aux places.

chapitre 5

Mise en place d'un protocole expérimental d'évaluation des interfaces sensibles au contexte

5.1 Introduction

Afin de valider notre approche et mesurer l'impact de l'application proposée sur des experts (médecins et pharmaciens), nous avons préparé une plateforme expérimentale et mené une étude sur terrain. Ce chapitre va donc exposer cette étude.

Nous allons en premier lieu, analyser notre SHM pour une interface classique et pour une interface contextualisée afin d'identifier les différences entre ces deux types d'interfaces. Nous allons présenter par la suite notre scénario d'exécution pour ensuite définir nos critères d'évaluation afin de comparer entre les interfaces statiques et les interfaces contextualisées. Les résultats de cette étude seront exposés en dernière partie de ce chapitre.

5.2 Analyse du Système Homme-Machine : analyse classique versus analyse contextualisée

Quelque soit le type d'analyse utilisé, la fonction principale de notre système reste inchangée et décomposée en deux sous-systèmes à savoir l'intervention en cas d'urgence et la mise à jour du dossier du patient. Les utilisateurs des interfaces créées, suite à une analyse classique ou contextualisée, demeurent les mêmes (secrétaire, infirmière, urgentiste, médecin

généraliste et spécialiste). Concernant les variables d'information, elles varient d'une analyse classique vers une analyse contextualisée. En effet, pour une analyse contextualisée, le système prend en considération les variables de l'environnement et s'adapte en fonction. C'est un système intelligent qui tient compte du profil de l'utilisateur et de la situation courante dans laquelle il évolue. De ce fait, le nombre de variables d'information est assez considérable. Pour une analyse classique, le système ne prend en considération que le métier de l'application et n'a aucune idée sur ce qui se passe autour de lui. Le tableau 5.1 expose les variables d'information relatives aux deux types d'analyses.

Tableau 5.1 — Les variables d'informations pour les deux types d'analyses

Analyse classique	**Analyse contextualisée**
Variables d'informations Entrées par l'utilisateur	
In.IdP, In.IdU, In.PU	
Variables d'informations de type sorties	
Out.P.Nom, Out.P.Prénom, Out.P.Sexe, Out.P.Dn, Out.P.Poids, Out.P.Prof, Out.P.Ts, Out.P.Mt, Out.P.LA, Out.P.Patho	Out.Gl, Out.P.Etat, Out.Type.U, Out.Qs.30, Out.Qs.10, Out.Qsr, Out.Qsl, Out.Qsa, Out.QPSG.
Variables d'informations de type consignes	
Cs.Pain, Cs.Resucre, Cs.perfusion	Cs.TA
Variables de commandes	
Cmd.FI	Cmd.AMT

La tâche de l'utilisateur dans un système classique ou contextualisé reste inchangée. Dans notre exemple, la tâche dépend de l'état du patient : état normal, hypoglycémie modérée ou coma hypoglycémique.

Lors de l'affectation des BIO au modèle de la tâche, le résultat obtenu diffère du type d'analyse utilisé. En effet, chaque analyse possède ses propres BIO. De ce fait, les interfaces obtenues vont être différentes. Ces interfaces ont fait l'objet d'un ensemble de test. En effet, après l'étape d'analyse, de modélisation et de spécification du SHM, nous avons obtenu deux types d'interface : une interface statique relative à l'analyse statique et une autre contextualisée. Nous avons donc voulu recueillir l'avis d'experts pour connaître leurs préférences en termes d'interfaces. Nous avons comparé les interfaces de DiaMon à des interfaces d'une application basique.

Pour ce faire, des objectifs de mesures ont été définis. En effet, nous avons voulu savoir si les interfaces obtenues étaient meilleures que les interfaces classiques et si nos interfaces

assuraient certains critères ergonomiques. Nous nous sommes concentrés sur l'évaluation de la qualité telle que la convivialité et la facilité d'utilisation ainsi que l'adaptabilité de nos interfaces. Afin d'atteindre nos objectifs, nous avons mis en place un scénario d'exécution.

5.3 Scénario d'exécution

Notre scénario se présente comme suit.

John est un jeune retraité âgé de 62 ans ex-responsable d'une agence immobilière. Ayant perdu sa femme depuis 2 ans et n'ayant pas eu d'enfant, il vit seul dans la région de Ariana en Tunisie. John est atteint de diabète de type 2 depuis 10 ans. Il s'injecte de l'insuline une fois par jour, généralement le matin avant de prendre son petit déjeuner.

Dû à sa situation familiale, John ne suit pas de régime alimentaire adapté. Il mange fréquemment en dehors de sa maison à des heures irrégulières. Etant de nature très active, John adore la marche à pied et sort tous les jours faire un tour dans son quartier.

John est suivi par le docteur Steven, professeur en diabétologie à la clinique « Ma clinique » à 15 km de chez lui. L'année dernière, John a été hospitalisé dans cette même clinique. Il a été victime d'une hypoglycémie sévère ayant abouti à un coma hypoglycémique. Depuis, John a été muni d'un capteur biologique « E-Health Sensor Shield » qui surveille en temps réel son taux de glycémie (GL) sanguin.

Par un beau matin ensoleillé, John décide d'aller faire le marché. Par peur de ne plus trouver de bon produit frais, il s'injecte de l'insuline et sort de chez lui rapidement en oubliant de prendre son petit déjeuner. Au bout d'une heure de courses, John commence à avoir quelques crampes gastriques. Il s'assoit par terre et là il commence à avoir des sueurs froides, il devient tout pâle et a terriblement envie de vomir. Au bout de quelques secondes, John se met à trembler et présente des troubles de la vue. Affolé, il prend son téléphone rapidement et appelle le service des urgences de la clinique « Ma clinique ».

Au bout de 5 minutes, l'ambulance arrive et l'urgentiste Peter le prend en charge. Muni d'une tablette « Samsung Galaxy Tab 3 », 10 pouces wifi/3G sous Android 4.4, Peter se connecte à l'application DiaMon via la technologie 3G et introduit le numéro de la carte d'identité de John. En même temps, il allonge John sur le brancard de l'ambulance et lui pose quelques questions.

La première interface (Figure 5.1) qui s'affiche informe Peter sur l'état de John. Elle affiche le taux exact de GL à savoir 0.5g/l. Via la géo-localisation, l'interface capte que le lieu d'intervention est l'ambulance donc s'adapte en fonction et augmente la luminosité de la ta-

blette. Puisque la plateforme utilisée est une tablette de 10 pouces, la zone d'affichage est assez restreinte. De ce fait, l'interface affiche seulement les informations nécessaires à l'utilisateur. Peter n'étant pas un diabétologue, l'interface adapte son contenu aux connaissances de l'utilisateur et le guide sur la manière d'accomplir sa tâche. En effet à cet instant, l'objectif de la tâche de l'utilisateur est le maintiens en vie de John.

John est allergique au gluten. Donc, au lieu d'afficher tout son dossier médical, l'interface n'affiche que le type d'allergie du patient pour que Peter prenne en considération cette information lors de l'accomplissement de sa tâche.

Lorsque Peter finit une tâche, il doit appuyer sur le bouton approprié pour que le système enregistre toutes les actions faites.

Figure 5.1 — Etat du patient.

John est toujours conscient et Peter essaye par tous les moyens de rétablir son taux de GL à la normale. Au bout de 10 min John recommence à trembler et le taux de GL continue à diminuer.

L'interface (Figue 5.2) s'adapte automatiquement à cette nouvelle situation et l'alerte, contenant le taux de GL, initialement en bas à gauche avec comme taille de police 18 devient au milieu de l'interface avec une taille de police égale à 36.

A 5 km de la clinique, le cas de John s'aggrave et il sombre dans un coma hypoglycémique. Le capteur biologique capte immédiatement la nouvelle valeur de GL et l'envoie au système. L'interface à son tour récupère cette valeur via la 3G et s'adapte à cette nouvelle situation. A cet instant, le type d'intervention doit changer. En effet, John inconscient ne peut plus

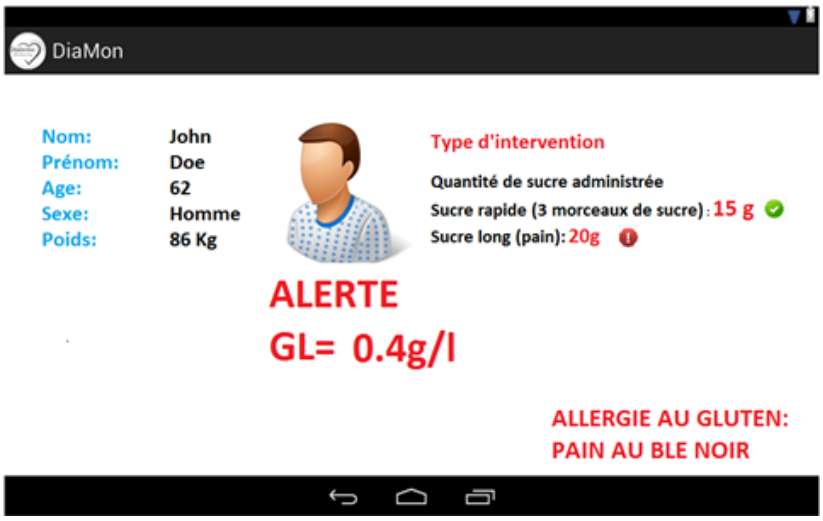


Figure 5.2 — Changement du taux de GL.

prendre par voie orale le sucre ou le pain. De ce fait, une perfusion de sérum glucosé doit être faite (Figure 5.3).

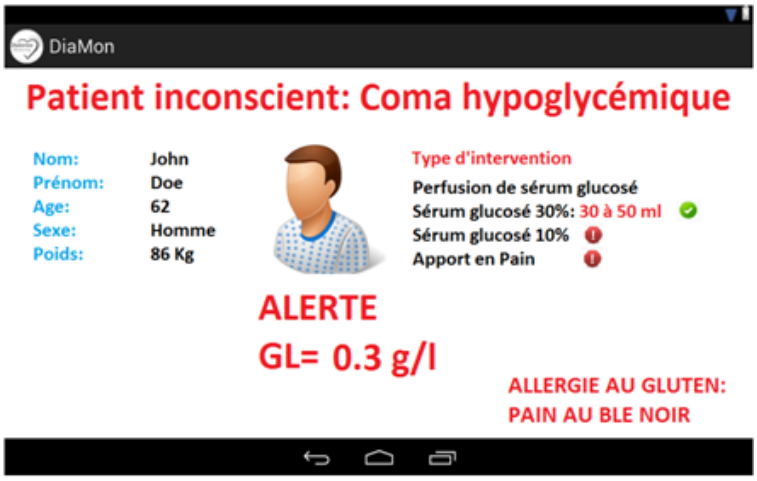


Figure 5.3 — Changement de l'état du patient.

Au moment où l'urgentiste lui fait une perfusion de 50 ml de sérum glucosé à 30 %, l'ambulance arrive à la clinique. Elle est à 100 m de la porte des urgences. Le réseau internet bascule de la 3G vers le wifi de la clinique. L'interface capte donc ce changement et s'adapte en fonction. En effet, à cet instant, l'objectif de la tâche de l'urgentiste est de trouver un diabétologue présent dans la clinique pouvant prendre en charge John. Du coup, quand le sys-

tème capte le changement de réseaux, l'interface s'adapte automatiquement à cette situation et la page-écran visible en figure 5.4 s'affiche. Toutes les informations concernant le dossier personnel du patient, le type d'intervention et le taux de GL disparaissent laissant place à la liste des médecins présents et disponibles dans la clinique avec le numéro du dossier du patient.

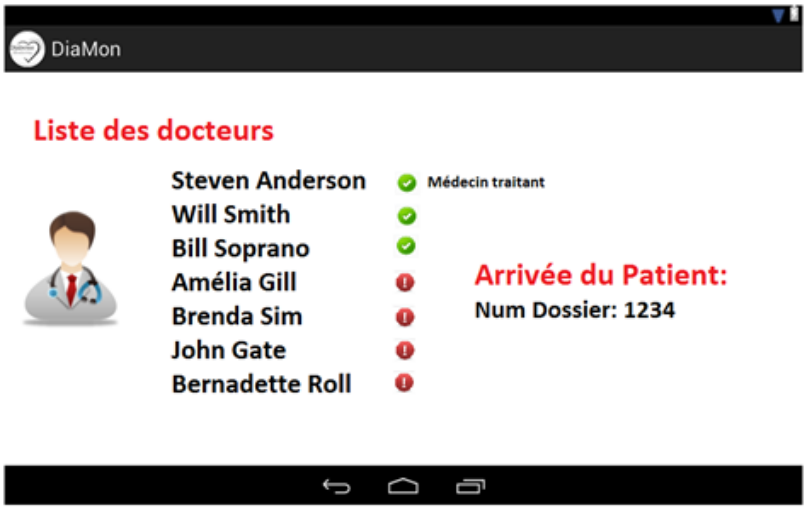


Figure 5.4 — Liste des docteurs disponibles.

Peter décide de choisir le docteur Steven puisqu'il est le médecin traitant de John et qu'il est disponible. En appuyant sur le nom du médecin, une notification (Figure 5.5) est envoyée afin de prévenir le Professeur Steven de l'arrivée de John.

Le Pr. Steven décide de prendre en charge John et appui sur le bouton valider. A cet instant Steven a besoin de savoir les tâches accomplies par l'urgentiste et dans quelle salle d'intervention John a été transporté. La page-écran visible en figure 5.6 s'affiche.

Dans la salle d'intervention, le Pr. Steven décide de se connecter sur l'ordinateur de bureau de la salle. Notre interface capte aussitôt ce changement de plateforme et s'adapte en conséquence. Toutes les informations relatives à John apparaissent avec son dossier médical, le traitement prescrit et la liste des analyses.

Afin d'évaluer nos objectifs de mesure, nous avons choisi de travailler sur une seule tâche de ce scénario. Nous nous sommes intéressés plus particulièrement à la tâche de l'urgentiste dans l'ambulance lors du transfert, qui tentait de maintenir en vie le patient. Les interfaces de cette tâche seront déployées et mises à la disposition des médecins pour le test et l'évaluation de nos interfaces.

Figure 5.5 — Notification pour le médecin disponible.

Figure 5.6 — Liste des tâches accomplies.

Donc, l'utilisateur (médecin ou pharmacien) va en premier lieu tester l'interface statique puis l'interface contextualisée en se mettant dans la peau du médecin urgentiste présent dans l'ambulance.

5.4 Implémentation

Après l'étape d'analyse, de modélisation et de spécification des interfaces, vient l'étape d'implémentation. La plateforme développée a été testée par la communauté d'experts.

5.4.1 Les interfaces statiques

Quand notre scénario commence, la première interface qui apparait (Figue 5.7) représente une boite de dialogue qui explique à l'utilisateur comment utiliser cette interface.

Figure 5.7 — Première interface.

Après avoir appuyé sur le bouton « commencer », l'utilisateur peut consulter le dossier du patient en appuyant sur les différents onglets. En cliquant sur l'onglet « info personnelles » la Figure 5.8 apparait.

En cliquant sur le bouton « info médicales » (Figure 5.9), les informations relatives au type de diabète, type sanguin du patient, ses éventuelles allergies, ses complications, les opérations effectuées et si le patient présente d'autre maladies seront affichées.

En cliquant sur l'onglet des analyses et des statistiques, les figures 5.10 et 5.11 apparaissent. Elles décrivent les analyses faites par le patient à savoir sa créatinine, microalbuminurie, triglycérides, etc.. Après avoir consulté le dossier du patient, l'utilisateur peut intervenir. Dans ce cas, une boite de dialogue sera affichée contenant toutes les interventions que peut accomplir l'utilisateur (Figure 5.12).

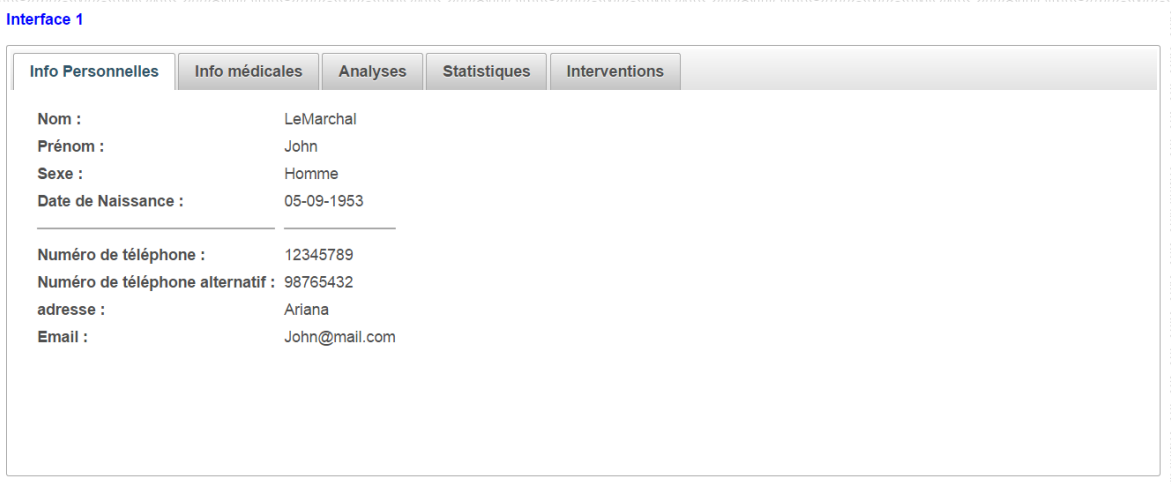


Figure 5.8 — Informations personnelles.

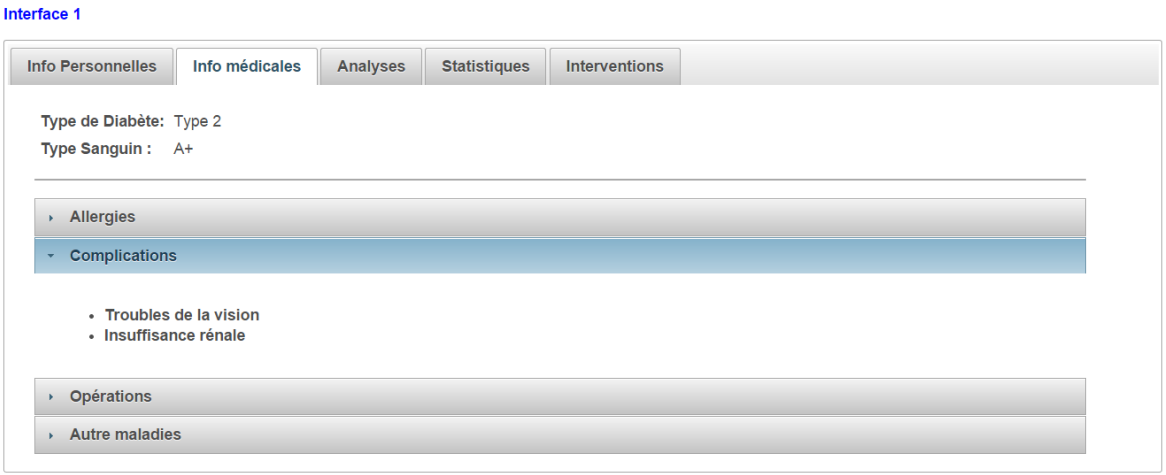


Figure 5.9 — Informations médicales.

En cliquant sur le bouton « finir l'intervention », l'utilisateur passe aux interfaces contextualisées.

5.4.2 Les interfaces contextualisées

Dans les interfaces contextualisées, tous les onglets disparaissent ne laissant place qu'aux informations nécessaires à l'utilisateur pour l'accomplissement de sa tâche. Le type d'intervention est identifié et affiché selon la variable GL. Un message apparait à l'utilisateur reflétant

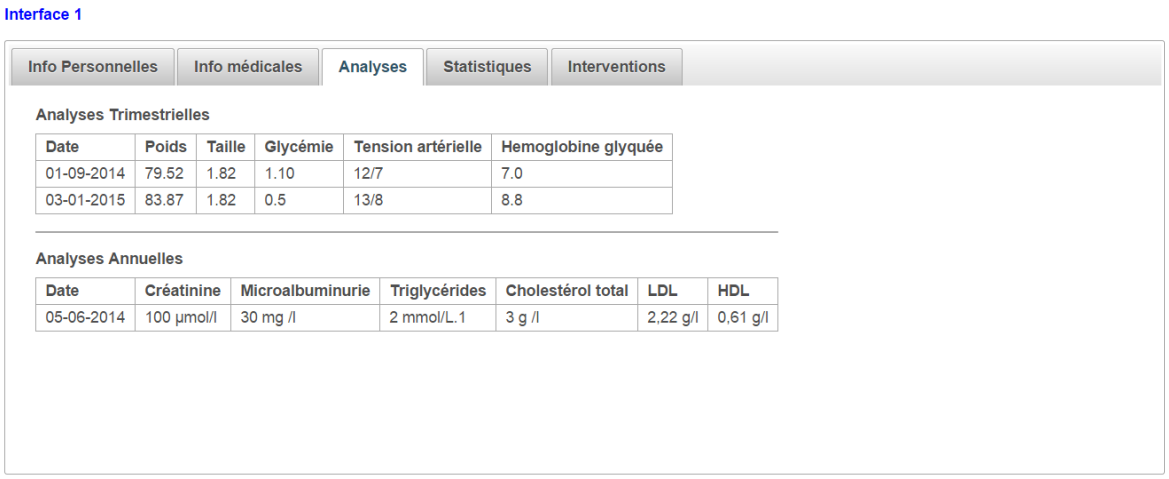


Figure 5.10 — Analyses trimestrielles et annuelles.

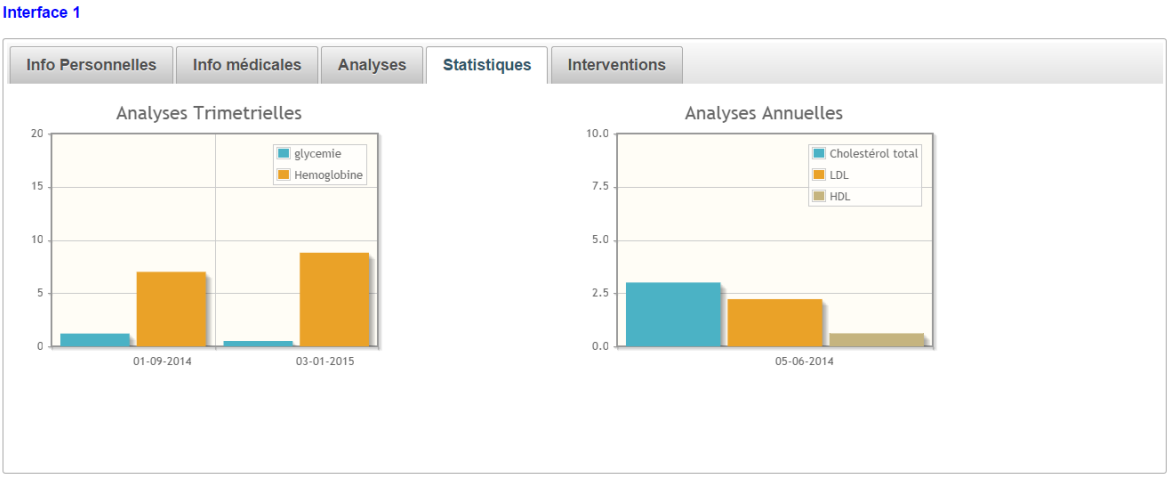


Figure 5.11 — Présentation graphique des analyses.

le taux de GL du patient et son type d'allergie. L'utilisateur appuie donc sur les boutons (représentés par des triangles) afin d'accomplir une tâche. L'interface qui suit reflète une hypoglycémie modérée du patient.

Quand le taux de GL continue à diminuer, le message d'alerte change de position et de taille afin de capter l'intention de l'intervenant. Un autre message de dialogue apparait afin d'ordonner à l'utilisateur d'intervenir (Figure 5.14).

Quand le patient sombre dans un coma hypoglycémique, un nouveau message apparait annonçant l'état grave du patient, le message du taux de GL change de valeur et le type

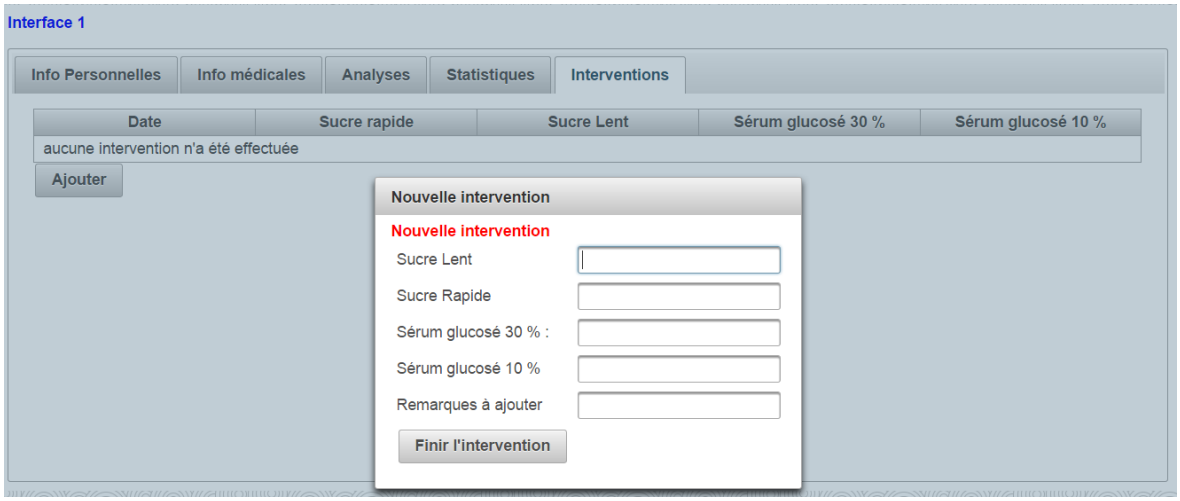


Figure 5.12 — Ajout d'une intervention.

Figure 5.13 — Patient en hypoglycémie.

d'intervention pour un patient conscient se transforme en l'intervention pour un patient inconscient (Figure 5.15).

Quand l'ambulance arrive à l'hôpital, l'application capte ce changement via le changement du réseau. La tâche de l'utilisateur est donc terminée. L'application propose à l'intervenant la liste des médecins disponibles afin de finir la simulation (Figure 5.16).

Figure 5.14 — Changement du taux de GL.

Figure 5.15 — Interface pour un coma hypoglycémique.

Afin de bien comparer les deux interfaces, nous avons tracé et sauvegardé dans un fichier toutes les actions accomplies par le sujet. Nous avons ensuite demandé à chaque personne de remplir un formulaire de satisfaction. Nous avons ensuite étudié les critères d'évaluation d'interfaces qui existent dans la littérature pour aboutir aux résultats.

Figure 5.16 — Interface pour l'arrivée du patient à l'hôpital.

5.5 Evaluation des interfaces utilisateur

Afin d'évaluer la qualité de nos interfaces, plusieurs critères de mesures ont été définis. Pour plus de clarté, nous avons décomposé nos critères d'évaluation en deux ensembles : les critères observables (reflètent les actions faites directement sur l'interface) et non observables (représentent les informations déduites suite à une analyse d'un questionnaire, interview ou autre technique de recueil d'information). Nous nous sommes basés sur les critères d'évaluation des IHM proposés par la communauté en ergonomie des logiciels.

5.5.1 Les paramètres observables

Ces paramètres représentent des variables qui vont être directement déduites à partir du fichier Log après chaque exécution du scénario par un individu. Le fichier log des interfaces statiques diffère de celui des interfaces contextualisées. Toutes les actions faites par les différents sujets sur nos interfaces vont être sauvegardées dans les fichiers Log.

Le tableau 5.2 présente les paramètres observés pour les interfaces statiques.

Concernant la simulation dynamique, les composantes de l'interface changent. De ce fait, les paramètres du fichier Log changent aussi. Le tableau 5.3 décrit ces paramètres.

Tableau 5.2 — Description du fichier Log de l'interface statique

Paramètres	Description
Début de la simulation	Il fournit l'heure du début de la simulation.
Les onglets choisis	Un ou plusieurs onglets peuvent être sélectionnés. Cela dépend de la navigation de l'utilisateur lors de la simulation. Le fichier Log enregistre le temps de clique sur l'onglet.
Les données saisies dans les champs de texte	Ces paramètres représentent : la quantité de sucre rapide, de sucre long, sérum 30%, sérum 10% et les remarques. L'utilisateur peut introduire une remarque relative au type d'allergie.
La Fin de la simulation	Il fournit la fin de la simulation

Tableau 5.3 — Description du fichier Log de l'interface contextualisée

Paramètres	Description
Début de la simulation	Il fournit l'heure du début de la simulation.
Boutons sélectionnés	Un ou plusieurs boutons peuvent être sélectionnés. Ils décrivent les actions faites par l'utilisateur.
La Fin de la simulation	Il fournit la fin de la simulation.

Ces paramètres seront sauvegardés dans un fichier Log. La structure de ces fichiers est représentée par le figure 5.17.

A la fin de chaque exécution de notre scénario, plusieurs questions vont être posées aux individus afin d'évaluer la qualité de nos interfaces et pouvoir élaborer une étude comparative entre les interfaces statiques et les interfaces contextualisées. La section qui suit présente ce questionnaire.

5.5.2 Les paramètres non observables

La rédaction du questionnaire a été guidée par des règles ergonomiques d'évaluation des interfaces utilisateurs, proposées dans le domaine de l'ergonomie des logiciels [Nielsen, 1993], [Coutaz, 1990] [Bastien et Scapin, 2001]. En effet, beaucoup de travaux d'évaluation des IHM sont proposés dans la littérature. Ces évaluations se basent sur différents critères de qualité de l'interface.

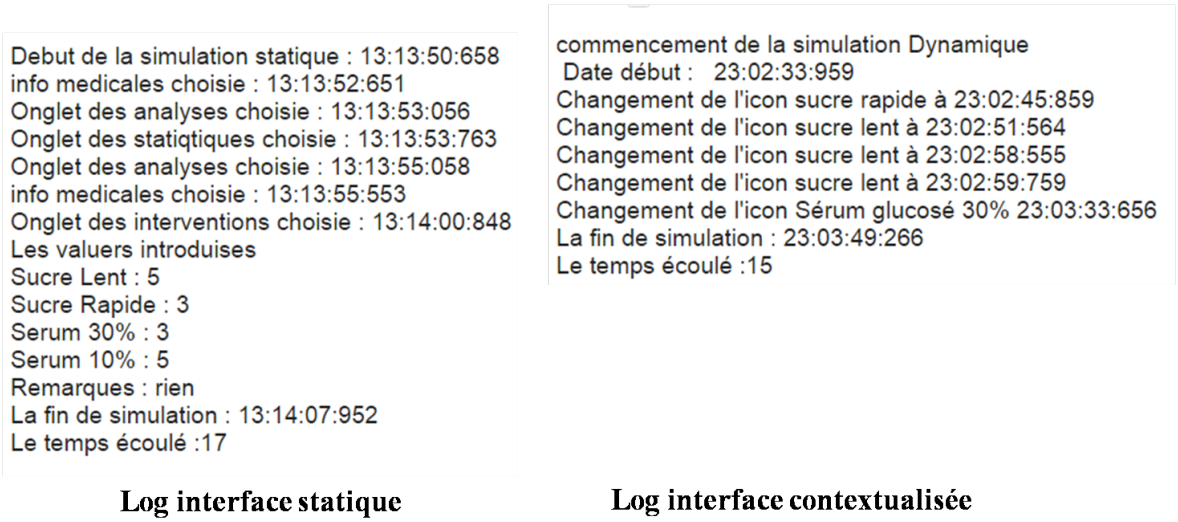


Figure 5.17 — Description des fichiers Log.

Pour notre étude expérimentale, nous avons convenu de faire l'évaluation des inerfaces proposées conformément aux critères d'utilité et d'utilisabilité. Pour l'utilisabilité, nous nous sommes intéressés à la qualité ergonomique de la présentation, l'adaptabilité au contexte et la facilité d'utilisation.

L'évaluation de l'utilité est vérifiée par les deux questions :

- L'IHM satisfait-elle les spécifications ?
- L'utilisateur peut-il accomplir sa tâche correctement ?

Concernant l'utilisabilité, elle est vérifiée par les questions :

- L'interface est-elle facile à utiliser ?
- L'interface est-elle adaptée à votre niveau de connaissances ?
- Les changements apparus dans l'interface sont-elles bénéfiques pour votre inervention ?

Ces paramètres peuvent être évalués via une évaluation expérimentale (observation, recueil de données, entretiens et questionnaires) ou une évaluation analytique (scénario d'utilisation et jugements d'experts). Dans nos travaux de recherche, l'évaluation expérimentale a été réalisée via le questionnaire et l'évaluation analytique a été faite via notre scénario et le questionnement d'expert du domaine (pharmaciens et médecins).

Ces critères d'évaluation concernent principalement les interfaces statiques. Nous avons remarqué durant ce travail la rareté des critères ergonomiques pour les interfaces contextualisées. Cependant, quelques critères proposés par J.M. Bastien et Scapin nous ont semblé intéressants pour l'évaluation d'interfaces dynamiques. Par exemple, nous pouvons citer le

critère d'adaptabilité qui concerne la capacité d'une interface à réagir selon le contexte et les besoins et préférences des utilisateurs [Bastien et Scapin, 2001].

Le tableau 5.4 décrit le questionnaire mis à disposition des différents sujets. Pour plus de clarté, nous allons expliquer à chaque question, le critère ergonomique employé pour sa rédaction.

Le sujet peut répondre à ces questions soit par « oui ou non » ou bien en cochant l'une des possibilités suivante : Médiocre, Moyen, Bon et excellent.

Après avoir défini nos critères d'évaluation, nous allons maintenant exposer les résultats de nos tests.

5.6 Résultats et analyses

5.6.1 Plan d'action des tests utilisateur

Pour pouvoir élaborer une étude basée sur les tests, nous devons suivre des étapes bien précises afin que les résultats fournissent de l'information fiable et utile. Tout d'abord, il faut commencer par identifier l'objectif de ces tests. Ces dernièrs ont été établies afin de mesurer les performances de notre approche et comparer entre les interfaces statiques et les interfaces contextualisées. Par la suite, il faut détecter la population cible.

DiaMon est essentiellement destinée au staff médical. Par faute de moyens, nous avons réussi à obtenir seulement douze personnes issues de différents domaines. Certains d'entre elles étaient des généralistes, d'autres étaient des diabétologues (privés et des internes aux hôpitaux tunisiens), un urgentiste et deux pharmaciens que nous avons sollicités pour nous donner leur avis.

De ce fait, nous avons choisi de faire une enquête-échantillon pour la collecte de nos données. En effet, nous allons recueillir nos informations d'une partie de la population qui représentera notre échantillon. A partir des données de l'échantillon, nous pouvons projeter les résultats pour la population dans son ensemble. Avant de définir la taille de notre échantillon, il faut fixer le taux du risque d'erreur. Plus la taille de l'échantillon est proche de la taille de la population, plus le biais est faible. C'est pour cette raison que nous avons fixé le taux d'erreur à 3%.

Après avoir définie le taux d'erreur, on peut désormais définir la taille de notre échantillon et faire par la suite notre échantillonnage à partir de la population cible. Il existe plusieurs

Tableau 5.4 — Questionnaire employé

Question	**Critère visé**
Est-ce que l'interface 1 est bien organisée par rapport à l'interface 2 ?	Ergonomie de l'IHM
Est-ce que l'interface 2 est bien organisée par rapport à l'interface 1 ?	
Est-ce que vous préférez l'interface 1 à l'interface 2 ?	Préférences de l'utilisateur
Est-ce que vous préférez l'interface 2 à l'interface 1 ?	
Est-ce que les informations affichées sur l'interface 1 sont suffisantes pour votre intervention ?	L'utilité
Est-ce que les informations affichées sur l'interface 2 sont suffisantes pour votre intervention ?	
Est-ce que vous avez bien accompli votre intervention en utilisant l'interface 1 ?	
Est-ce que vous avez bien accompli votre intervention en utilisant l'interface 2 ?	
Est-ce que vous êtes satisfait de l'interface 1 ?	Satisfaction de l'utilisateur
Est-ce que vous êtes satisfait de l'interface 2 ?	
Est-ce que l'interface 1 est facile à utiliser ?	Facilité d'utilisation
Est-ce que l'interface 2 est facile à utiliser ?	
Est-ce que la qualité des informations affichées dans l'interface 1 est meilleure que l'interface 2 ?	Qualité de l'IHM
Est-ce que la qualité des informations affichées dans l'interface 2 est meilleure que l'interface 1 ?	
Est-ce que l'interface 2 répond le mieux à vos attentes ?	L'utilisabilité
Est-ce que l'interface 2 est adaptée à votre niveau de connaissances ?	Adaptation au profil de l'utilisateur
Est-ce que les changements apparus dans l'interface 2 ont été bénéfiques pour votre intervention ?	Adaptabilité
Est-ce que le changement du label dans l'interface 2 « taux GL » vous a aidé pour votre intervention ?	Adaptation au contexte
Est-ce que l'apparition du label « patient inconscient » dans l'interface 2 a été bénéfique pour vous ?	
Est-ce que l'apparition de l'information « allergie au gluten » dans l'interface 2 vous a aidé à ne pas commettre d'erreur dans votre intervention ?	Adaptation aux connaissances de l'utilisateur

manières de faire l'échantillonnage à savoir l'échantillon aléatoire simple, systémique, avec une probabilité proportionnelle à la taille, stratifié etc. Pour notre étude, nous avons choisi de travailler avec l'échantillonnage aléatoire simple sans remise. Cette méthode donne à chaque individu de la population une chance égale d'appartenir à l'échantillon. Cette approche est parfaitement applicable dans notre cas puisque nous possédons la liste exhaustive de toute la population. Donc, nous allons en premier lieu, numéroter tous les individus de la liste avec des nombres comportant un même nombre de chiffres. En utilisant la fonction alea () du logiciel Excel, nous obtenons des nombres aléatoires comportant le nombre de chiffres désiré. En dernier lieu, nous sélectionnons les nombres qui coïncident avec la liste et nous nous arrêtons après avoir sélectionné n individus, avec n le nombre d'individus souhaités dans l'échantillon. Il existe sur le net plusieurs calculateurs de la taille d'un échantillon. Notre étude s'est basée sur le calculateur du groupe de recherche CUBE [cube recherche, 2016]. En fixant la taille de notre population à 12 personnes, le taux d'erreur à 3% et le niveau de confiance (reflète le degré de certitude de la marge d'erreur) à 95%, nous obtenons la taille de l'échantillon égale à la taille de la population.

Après avoir préparé le plan d'échantillonnage et identifié tous les éléments de notre étude, la prochaine étape consiste à analyser les résultats obtenus.

5.6.2 Résultat de l'étude

Comme nous l'avons mentionné auparavant, nous avons deux sortes de données à analyser : les informations des fichiers Log et les informations provenant du questionnaire.

5.6.2.1 Analyses des fichiers Log

A la lumière des données obtenues, nous nous sommes penchés sur le calcul du temps de la première action faite par le sujet, représenté par la Figure 5.18. Nous avons remarqué que le temps d'action pour l'interface contextualisée (égale à 2.1 secondes) est plus rapide par rapport à l'interface statique (égale à 5.3 secondes). En effet, ce résultat est compréhensible puisque pour l'interface statique, avant d'effectuer une action, l'utilisateur doit consulter le dossier du patient. En revanche, pour l'interface contextualisée, le système capte l'état du contexte et affiche les informations adéquates à cette situation. De ce fait, nous pouvons déduire que l'interface contextualisée est plus facile à comprendre et plus rapide en termes de temps d'exécution, propriété très importante dans les situations critiques.

Ensuite, nous nous sommes attardés sur les valeurs introduites durant l'intervention de

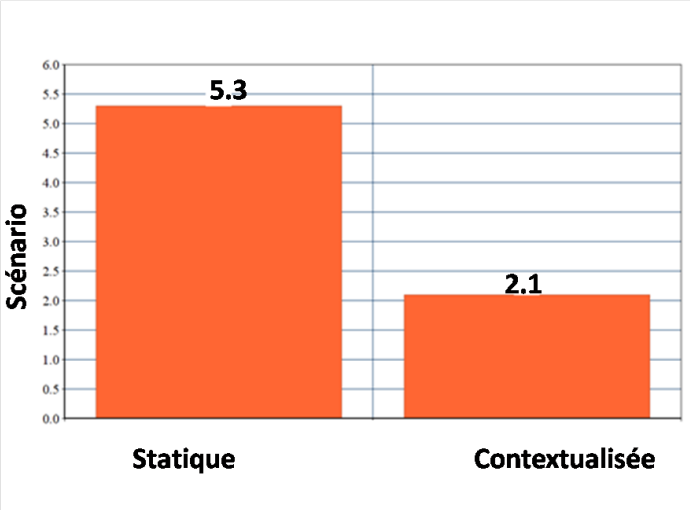


Figure 5.18 — Moyenne du temps de la première action de l'utilisateur.

l'échantillon dans l'interface statique et sur les contrôles effectués lors du traitement de l'interface contextualisée. Nous avons comparé ces deux types de données en calculant le nombre d'erreurs effectuées par les sujets. Cette comparaison a été faite sur la base des quantités de sucre rapide, de sucre long, de quantité de sérum 30% et 10% vues dans le chapitre précédent. Dans l'interface contextualisée, nous avons considéré comme erreur, les contrôles non effectués lors de l'intervention. Nous avons assigné pour chaque erreur 0.25 points pour les deux interfaces afin de faciliter le calcul de la moyenne des erreurs accomplies par notre échantillon. Les résultats sont représentés par la Figure 5.19.

Nous remarquons que les erreurs commises avec l'interface statique sont beaucoup plus importantes qu'avec l'interface contextualisée. Dans notre cas d'étude, l'utilisation d'interface contextualisée est conseillée. En effet, les erreurs au niveau de l'intervention dans l'interface statique peuvent amener à des résultats catastrophiques pour le patient pouvant conduire à son décès. Même si tout notre échantillon appartient au domaine médical et possède des connaissances assez importantes à ce sujet, intervenir dans un cas d'urgence demeure assez compliqué pour eux.

Nous allons passer à présent à l'analyse des données du questionnaire.

5.6.2.2 Analyses des données du questionnaire

Pour plus de clarté, nous avons décomposé les questions du questionnaire, énoncées dans la section précédente, en trois parties. En effet, les vingt questions posées comprennent :

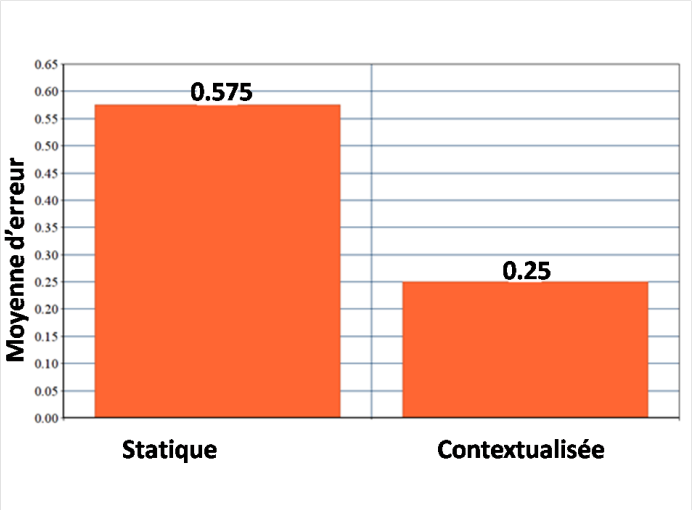


Figure 5.19 — Moyenne des erreurs faites lors de l'intervention de l'utilisateur.

- Quatre questions relatives à l'interface statique (son organisation, son utilité, etc.) ;
- Six questions relatives à la comparaison entre les deux interfaces ;
- Le reste des questions a été dédié aux interfaces contextualisées.

a- Analyse des questions relatives à l'interface statique et à l'interface contextualisée

La première question posée aux intervenants était la suivante : est-ce-que les informations affichées sur l'interface sont suffisantes pour accomplir votre l'intervention ?. Les réponses ont été comme suit :

- Quatre personnes sur douze ont affirmé que les informations affichées pour l'intervention dans l'interface statique sont médiocres. Cependant trois personnes les ont trouvées bonnes et une excellentes. En général, plus de la moitié des personnes interrogées affirment que les informations affichées au niveau de l'interface statique ne sont pas suffisantes pour accomplir leur intervention (Figure 5.20.a).
- Pour l'interface contextualisée, représentée par la Figure 5.20.b, plus de la moitié (9 personnes sur 12) ont affirmé que les informations affichées sont bonnes pour accomplir leurs tâches.

Concernant la question posée sur la satisfaction de l'utilisateur, les réponses ont été comme suit (Figures 5.21.a et 5.21.b) : Nous remarquons que cinq personnes sur douze ont trouvé l'interface statique bonne ou excellente : trois personnes ont affirmé qu'elle était médiocre et quatre ont trouvé qu'elle était moyenne (Figure 5.21.a). Pour l'interface contextualisée, six personnes sur douze ont répondu « bon » et deux « excellent ». L'interface contextualisée est

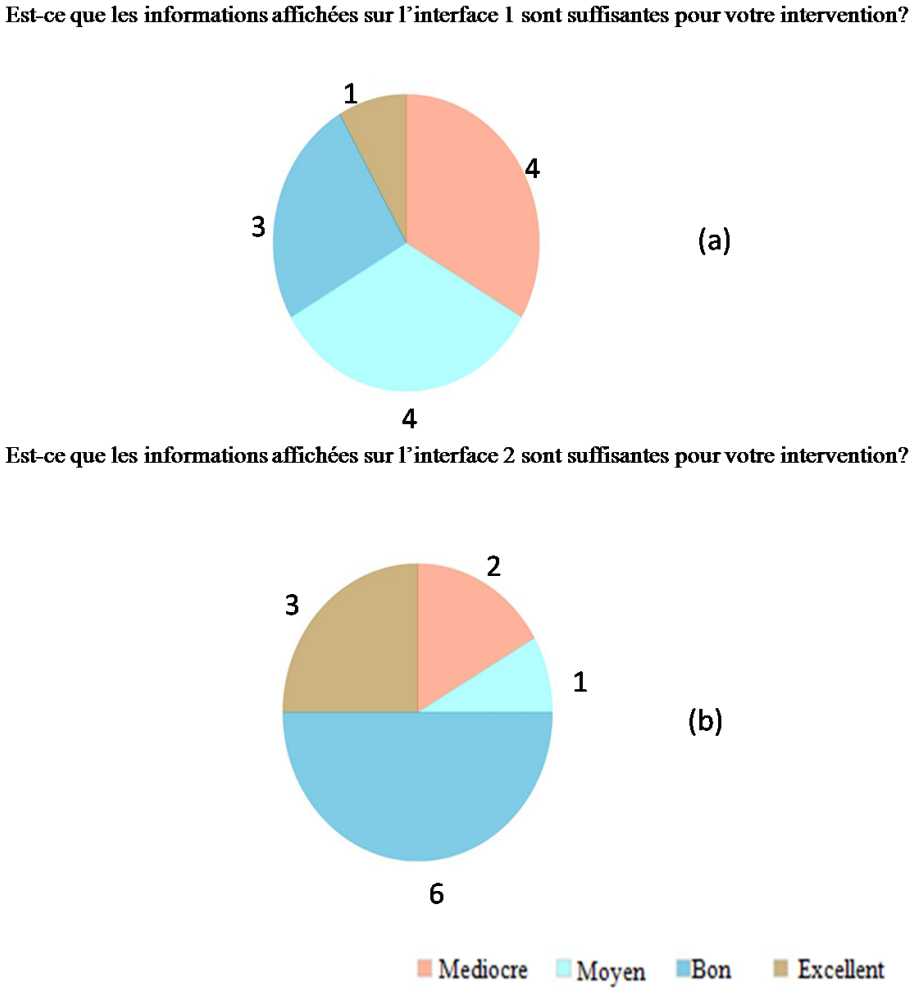


Figure 5.20 — Questions relatives aux interfaces statique et contextualisée.

considérée comme satisfaisante pour la majorité des participants, comme le montre la Figure 5.21.b.

Concernant la facilité d'utilisation les avis sont représentés par les figures 5.22.a et 5.22.b. Les réponses ont été partagées. En effet 50% des intervenants trouvent que l'interface statique est facile à utiliser et les autres 50% pensent le contraire (Figure 5.22.a). En résumé, les intervenants trouvent que l'interface est facile à utiliser mais la majorité d'entre eux ont refusé de remplir les champs relatifs à l'intervention par manque d'informations. Ils ont affirmé qu'ils ne pouvaient pas accomplir leurs tâches. Même constat pour l'interface contextualisée, les avis ont été partagés comme le montre la Figure 5.22.b.

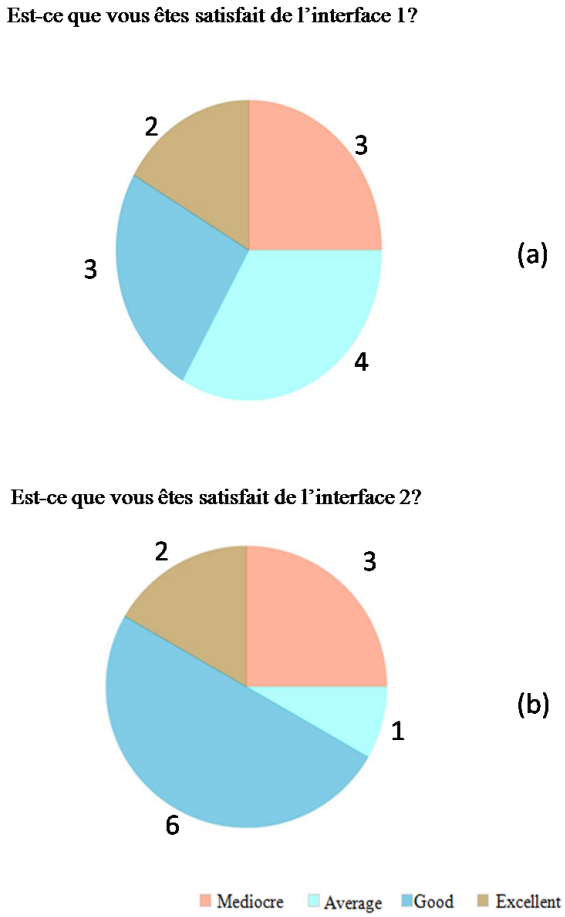


Figure 5.21 — Satisfaction des utilisateurs par rapport à l'interface statique et contextualisée.

Plusieurs questions ont été spécialement écrites pour l'évaluation des interfaces contextualisées. En effet, nous avons voulu savoir si les changements effectués au niveau des composantes graphiques de l'interface contextualisée lors de l'intervention, ont été bénéfiques.

Ici, nous pouvons voir dans la Figure 5.23, que huit personnes ont affirmé que l'apparition de ce « label » a été excellente pour eux, trois participants ont voté pour bonne et seulement une personne a affirmé qu'elle était moyenne.

Nous remarquons pour la Figure 5.24, que onze personnes sur douze ont affirmé l'utilité de l'apparition du label contenant le taux de glucose chez le patient durant l'intervention de l'utilisateur (sept ont répondu bon et quatre excellent).

Pareille pour la question portant sur les changements dans l'interface contextualisée (Figure 5.25), la majorité des participants (neuf sur douze) ont apprécié les changements des

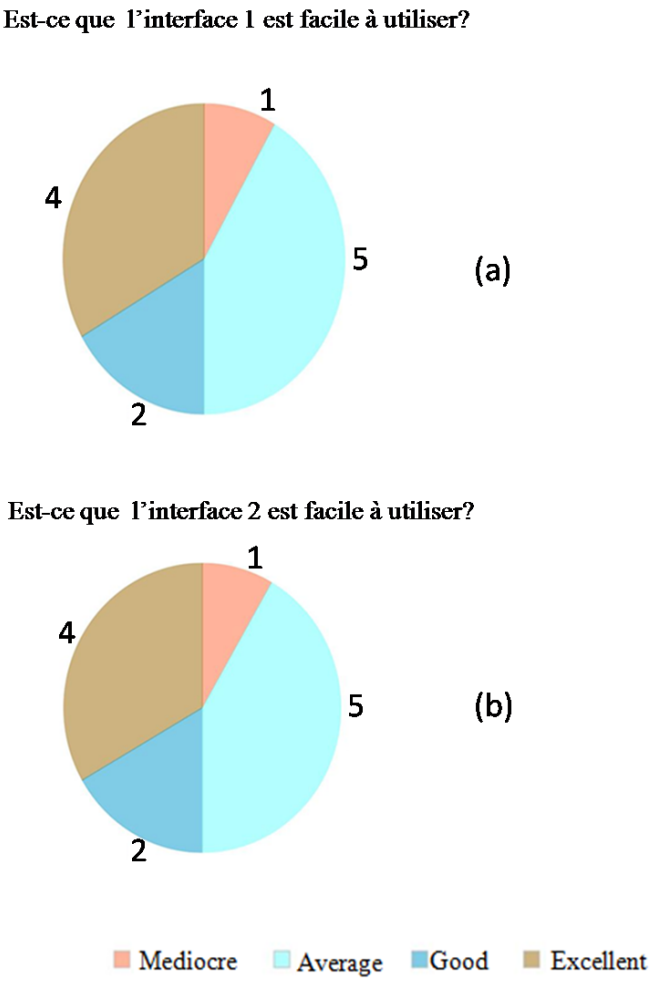


Figure 5.22 — Facilité d'utilisation de l'interface statique et contextualisée.

composantes graphiques survenus dans l'interface contextualisée. Ils ont affirmé l'avantage de l'apparition du message relatif au taux de glucose et au changement de l'intervention pour l'accomplissement de leurs tâches.

Concernant la question « est-ce que l'apparition de l'information « allergie au gluten » a été bénéfique ? », 100% des participants ont affirmé que cette information les a aidés à ne pas commettre d'erreur lors de leur intervention (quatre ont répondu excellent et huit bon comme le montre la figure 5.26).

b- Questions relatives à la comparaison entre les deux interfaces

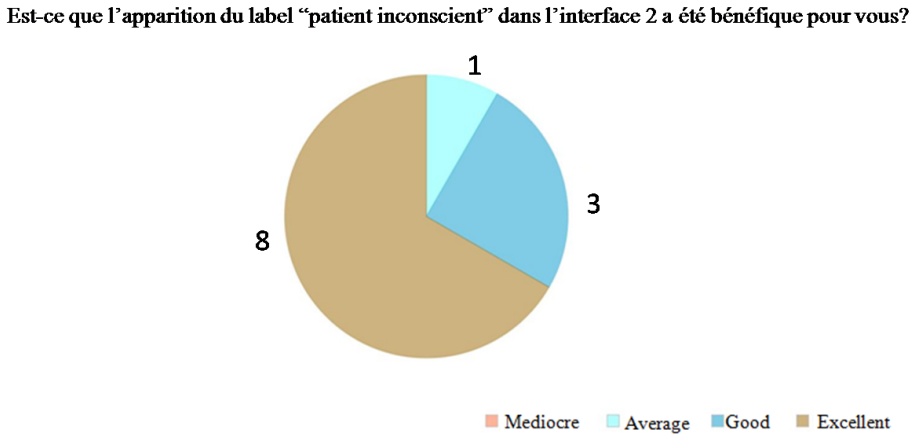


Figure 5.23 — Question relative à l'apparition du label « patient inconscient ».

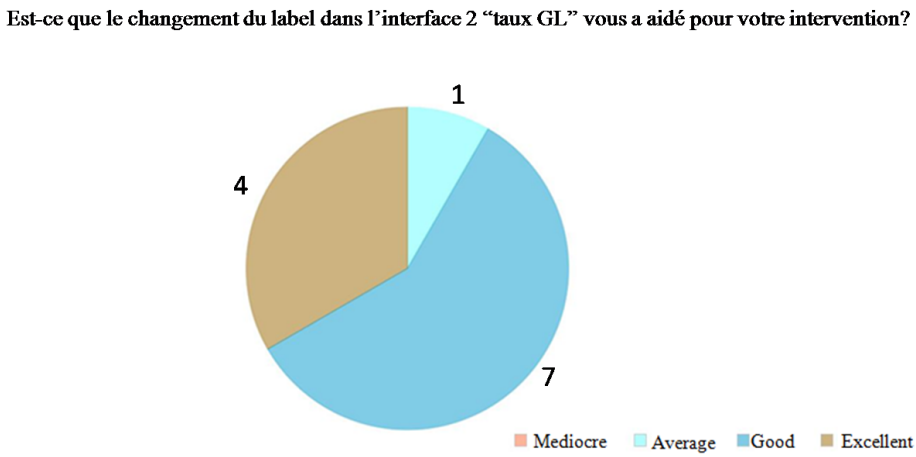


Figure 5.24 — Question relative à l'apparition du label « taux de GL ».

Nous avons posé des questions relatives à la préférence des utilisateurs : est-ce qu'ils préfèrent l'interface contextualisée à l'interface statique ? Leur réponse est représentée par la figure 5.27.

Dix participants sur douze préfèrent l'interface contextualisée à l'interface statique.

5.6.3 Discussion

Bien que la population visée ne soit pas très familière avec les technologies et les notions nouvelles (les capteurs biologiques, les tablettes et les applications sensibles au contexte) la

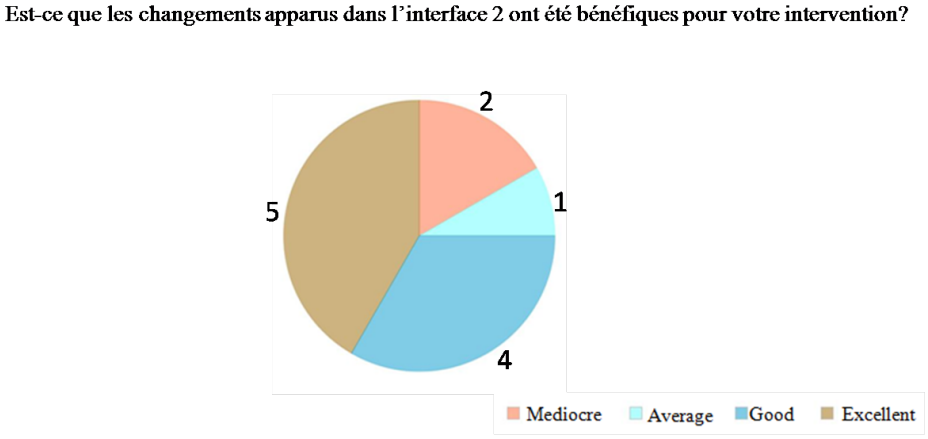


Figure 5.25 — Question relative aux changements dans l'interface contextualisées.

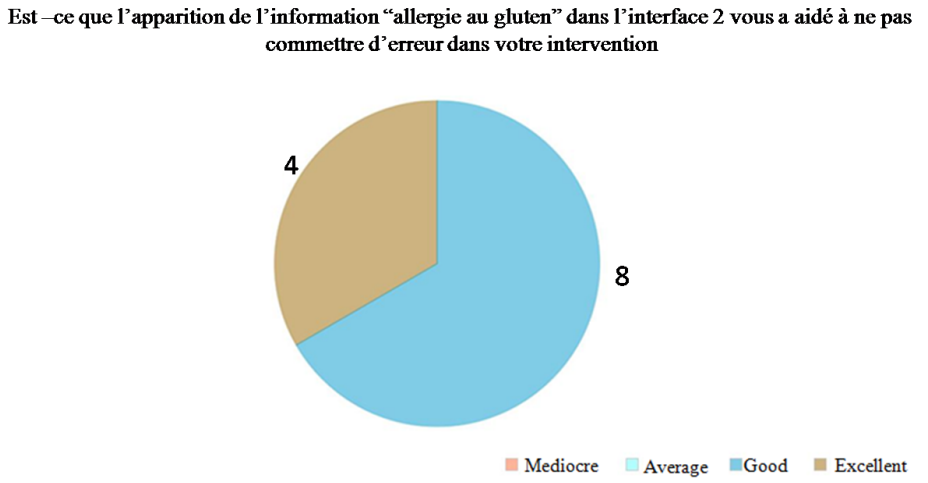


Figure 5.26 — Question relative à l'apparition du label « allergie au gluten ».

majorité des participants ont préféré l'interface contextualisée à l'interface statique. Ceci a été démontré à travers le questionnaire et surtout la dernière question (Figure 5.27). Ils ont considéré que les interfaces contextualisées sont beaucoup plus efficaces dans l'intervention en cas d'urgence. En tant que chercheurs, nous avons conclu que l'interface sensible au contexte remplissait les critères d'évaluation énoncés dans la section 5.5 tels que la facilité l'utilité et l'utilisabilité.

Les informations analysées des fichiers Log ont également démontré le même résultat. En effet, les interfaces sensibles au contexte sont moins sujettes à l'erreur et l'intervention de l'utilisateur est plus rapide.

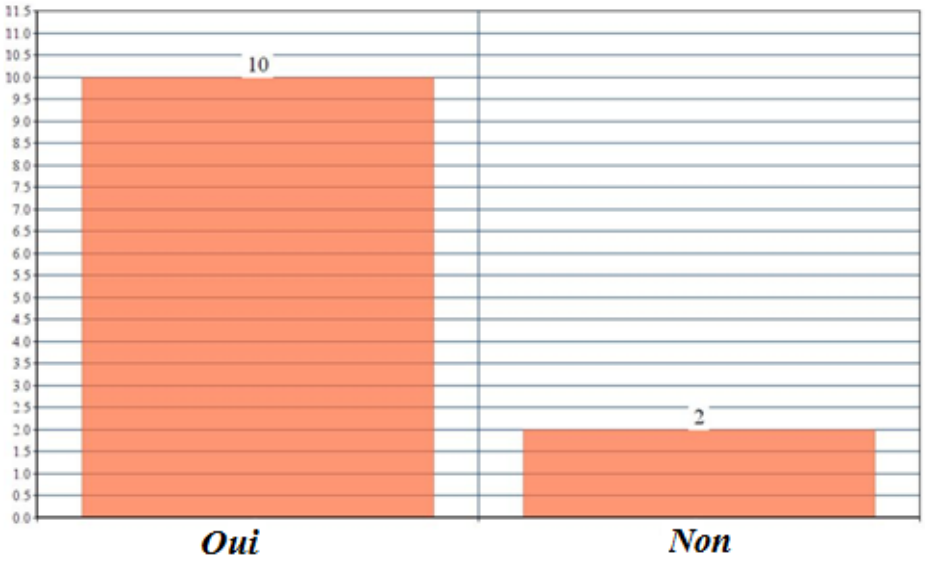


Figure 5.27 — Préférence des utilisateurs concernant l'interface conctextualisée ».

5.7 Conclusion

Nous avons présenté tout au long de ce chapitre notre plateforme expérimentale. Nous avons commencé par l'analyse des interfaces statiques et contextualisées afin de spécifier nos interfaces, puis nous avons exposé le scénario d'exécution suivi durant toute la phase de test. Les différentes interfaces obtenues (statiques et contextualisées) ont été également présentées.

Par la suite, nous avons défini nos critères d'évaluation basés sur les critères ergonomiques pour les interfaces statiques. Nous avons décomposé ces derniers en des paramètres observables et non-observables. Finalement, nous avons présenté et analysé les résultats de nos test.

Nous avons confirmé à travers l'analyse de nos résultats que les interfaces contextualisées étaient plus performantes dans les applications médicales critiques. Ces premiers résultats sont prometteurs et permettent d'envisager différentes perspectives de recherche, décrites dans la partie suivante.

Conclusion générale et perspectives

Nous avons proposé durant nos travaux de recherche, une approche formelle de spécification et de modélisation des interfaces sensibles au contexte dédiée au domaine médical. Nous avons essayé de proposer une démarche complète qui couvre progressivement toutes les étapes depuis la phase d'analyse du SHM sensible au contexte jusqu'à la phase de spécification et de génération des différentes vues graphiques.

Pour ce faire, nous avons établi une étude bibliographique portant sur ce domaine. En effet, dans le premier chapitre, nous avons introduit et défini les concepts de base à savoir, la notion du contexte, la sensibilité au contexte et l'adaptation. Nous avons ensuite exposé les différentes architectures des systèmes sensibles au contexte. Nous avons défini nos critères de comparaison pour pouvoir établir une étude comparative. L'un des critères les plus importants à nos yeux, était le modèle du contexte. En effet, la spécification d'interface contextualisée dans un domaine critique, comme le domaine médical, nécessite une modélisation formelle et rigoureuse pour garantir une interface valide et fiable.

Partant de ce postulat, nous avons présenté, dans le deuxième chapitre, les différentes approches de modélisation du contexte qui existent dans la littérature. Nous avons fixé par la suite des critères de comparaison, extraits à partir des exigences d'un système sensible au contexte dans un domaine critique, pour ensuite conclure avec l'outil choisi à savoir les réseaux de Petri.

Nous avons remarqué à travers notre étude bibliographique, que la majorité des systèmes proposés, ne prenaient pas en considération la tâche de l'utilisateur dès la phase d'analyse. La phase d'analyse du contexte était complètement séparée de la tâche de l'utilisateur bien qu'il existe une relation étroite entre ces deux concepts. En outre, la plupart des systèmes proposés dans le premier chapitre, utilisaient des approches de modélisation qui n'étaient pas assez formelles. Or cette étape représente la principale source des données du système.

A cet effet, nous avons essayé d'éviter toutes ses restrictions en proposant, dans le troisième chapitre, une approche formelle et complète de conception d'interfaces contextualisées. Notre démarche se démarque par les phases suivantes :

1. La détection des données contextuelles ;
2. L'analyse du SHM et de la tâche de l'utilisateur ;
3. La modélisation de la tâche et du contexte ;
4. La détection de la situation courante (Ci,Ti) ;
5. La spécification de l'interface utilisateur ;
6. La génération des interfaces.

La phase de génération des interfaces n'a pas été traitée dans ce travail. Nous avons effectué des maquettes sur les spécifications obtenues.

Dans le quatrième chapitre, nous avons présenté notre domaine d'application et nous avons appliqué notre démarche étape par étape. Nous avons défini DiaMon, qui représente une application de suivi des patients atteints de diabète type 2, destinée au corps médical (médecins et infirmiers).

Etant donné la rareté des travaux d'évaluation des interfaces contextualisées dans la littérature, nous avons établi, dans le cinquième chapitre, une étude portant sur les interfaces de DiaMon auprés du personnel médical. En se basant sur notre application et sur des critères ergonomiques d'évaluation des interfaces, nous avons établi une étude comparative entre les interfaces statiques et contextualisées.

Nos travaux de recherche laissent envisager à différentes perspectives de recherche. Premièrement, en raison de sa nature dynamique, plusieurs situations imprévues peuvent subvenir dans un environnement ubiquitaire. Que faire dans le cas ou une situation nouvelle qui n'est pas pris en charge par notre système, se présente ? Nous préconisons donc d'ajouter un autre module à notre architecture relative à la gestion des situations imprévues. Ce module sera basé sur un système expert qui décidera de la meilleure solution à fournir en cas d'imprévue.

La deuxième perspective concerne le traitement des données contextuelles captées. En effet, analyser et déduire l'information idoine à un moment précis n'est pas suffisant pour satisfaire les besoins de l'utilisateur en termes de données. Le nombre de capteurs dans un environnement ubiquitaire et l'état psychologique de l'utilisateur laissent une marge d'erreur assez élevée lors de l'analyse de l'information déduite et lors de l'étude des informations nécessaires à l'utilisateur. Nous envisageons d'ajouter un module entre la phase de dédection des données contextuelles et la phase d'analyse du SHM. Ce module aura pour objectif d'analyser la sémantique des données captées. Cette étape sera basé sur les ontologies.

La troisième perspective est relative à l'élaboration de la base des règles ergonomiques. En effet, de nos jours, rares sont les travaux qui ont proposé des règles ergonomiques pour la construction des interfaces contextualisées. De même pour l'évaluation des interfaces de ce type. Certes, nous avons proposé une évaluation de ces interfaces mais nous préconisons d'approfondir cette étude pour élaborer des règles et les exploiter dans différentes évaluations. Nous pouvons par exemple approfondir le critère d'adaptabilité, proposé par [Bastien et Scapin, 2001] et l'adapter pour les interfaces contextualisées.

La dernière perspective concerne notre application. Lors de notre étude, nous ne nous sommes pas intéressés aux cas de dysfonctionnements du système. En effet, comment peut-on capter une éventuelle intervention, si le dispositif mobile du médecin est en panne ? Ou encore, comment surmonter le cas où le capteur biologique est défaillant et les informations fournies sont incorrectes ? Nous préconisons donc d'utiliser les travaux établis par [Riahi, 2004] et [Moussa *et al.*, 2006] qui ont étudié les cas de dysfonctionnements dans les systèmes industriels, et de les adapter à nos travaux.

Bibliographie

[15909, 2007] 15909, I. S. I. (2007). *ISO/IEC, Software and Systems Engineering − High−level Petri Nets, Part 2 : Concepts, Definitions and Graphical Notation,*.

[AAP, 2013] AAP, A. A. O. P. (2013). Manuel de l'application aap essentials : Type 2 diabetes. https ://itunes.apple.com/us/app/aap-essentials-type-2-diabetes/id643786818 ?mt=8. Dernière consultation 11/02/2016.

[Abed et Ezzedine, 1998] Abed, M. et Ezzedine, H. (1998). Vers une démarche intégrée de conception−évaluation des systèmes homme−machine. *Journal of Decision Systems*, 7(1-4):147–175.

[Agostini *et al.*, 2006] Agostini, A., Bettini, C. et Riboni, D. (2006). Experience report : ontological reasoning for context-aware internet services. *In Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on*, pages 5–12.

[Agostini *et al.*, 2009] Agostini, A., Bettini, C. et Riboni, D. (2009). Hybrid reasoning in the care middleware for context awareness. *Int. J. Web Eng. Technol.*, 5(1):3–23.

[Akman et Surav, 1997] Akman, V. et Surav, M. (1997). The use of situation theory in context modeling. *Computational Intelligence*, 13:427–438.

[Atallah et Yang, 2009] Atallah, L. et Yang, G.-Z. (2009). The use of pervasive sensing for behaviour profiling a survey. *Pervasive and Mobile Computing*, 5(5):447–464.

[Augusto *et al.*, 2008] Augusto, J. C., Liu, J., McCullagh, P., Wang, H. et Yang, J.-B. (2008). Management of uncertainty and spatio-temporal aspects for monitoring and diagnosis in a smart home. *International Journal of Computational Intelligence Systems*, 1(4):361–378.

[Baas *et al.*, 2014a] Baas, M., Bernonville, S., Bricon-Souf, N., Hassler, S., Kolski, C. et Boy, G. A. (2014a). Towards a context model for human-centered design of contextual data entry systems in healthcare domain. *In Engineering Psychology and Cognitive Ergonomics*, pages 223–233. Springer.

[Baas *et al.*, 2014b] Baas, M., Bernonville, S. et Kolski, C. (2014b). Contextual data entry system design in the healthcare domain. *In Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, pages 1045–1050.

[Bardram, 2005] Bardram, J. (2005). The Java Context Awareness Framework (JCAF) - A Service Infrastructure and Programming Framework for Context−Aware Applications. *In* Gellersen, H.-W., Want, R. et Schmidt, A., éditeurs : *Pervasive Computing*, volume 3468 de *Lecture Notes in Computer Science*, pages 98–115. Springer Berlin Heidelberg.

[Barwise et Perry, 1980] Barwise, J. et Perry, J. (1980). *The Situation Underground*. Stanford University Press.

[Bastien et Scapin, 2001] Bastien, J. et Scapin, D. (2001). Évaluation des systèmes d'information et critères ergonomiques. *Environnements évolués et évaluation de l'IHM, Interaction homme-machine pour les SI*, 2:53–80, Hermes, Paris.

[Bauer *et al.*, 2003] Bauer, J., Kutsche, R.-D. et Ehrmanntraut, R. (2003). Identification and modeling of contexts for different information scenarios in air traffic. *Technische Universität Berlin, Diplomarbeit*.

[Ben Ismail, 2014] Ben Ismail, I. (2014). *Vers une approche d'adaptation dynamique et temps−réel du contenu informationnel d'une interface utilisateur dans un environnement ubiquitaire*. Thèse de doctorat, Ecole Nationale des Sciences de l'Informatique, Tunisie.

[Bettini *et al.*, 2010] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A. et Riboni, D. (2010). A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2):161–180.

[Billington *et al.*, 2003] Billington, J., Christensen, S., Hee, K., Kindler, E., Kummer, O., Petrucci, L., Post, R., Stehno, C. et Weber, M. (2003). The petri net markup language : Concepts, technology, and tools. *In* Aalst, W. et Best, E., éditeurs : *Applications and Theory of Petri Nets 2003*, volume 2679 de *Lecture Notes in Computer Science*, pages 483–505. Springer Berlin Heidelberg.

[Biotech, 2013] Biotech, U. (2013). Etude qualitative 2013 perspective d'utilisation des capteurs et biocapteurs dans la surveillance des maladies chroniques. Rapport technique, Optics Valley.

[Bouvin *et al.*, 2003] Bouvin, N. O., Christensen, B. G., Grønbæk, K. et Hansen, F. A. (2003). Hycon : A framework for context−aware mobile hypermedia. *Hypermedia*, 9(1):59–88.

[Bouzy et Cazenave, 1997] Bouzy, B. et Cazenave, T. (1997). Using the object oriented paradigm to model context in computer go. *In Proceedings of the first international and interdisciplinary conference on modeling and using context*, pages 279–289.

[Brown, 1996] Brown, P. J. (1996). The stick−e document : a framework for creating context−aware applications. *In Proceedings of EP'96, Palo Alto*, pages 259–272.

[Buttussi, 2008] Buttussi, F. (2008). A user−adaptive and context−aware architecture for mobile and desktop training applications. *In Proceedings of the 10th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '08, pages 543–543, New York, NY, USA. ACM.

[Calvary *et al.*, 2005] Calvary, G., Dâassi, O. et Coutaz, J. (2005). Des widgets aux comets pour la plasticité des systèmes interactifs. *Revue d'Interaction Homme Machine*, 6(1):33–53.

[Calvary *et al.*, 2004] Calvary, G., Demeure, A., Coutaz, J. et Dâassi, O. (2004). Adaptation des interfaces homme−machine à leur contexte d'usage plasticité des ihm. *Revue d'Intelligence Artificielle*, 18(4):577–606.

[Capra *et al.*, 2001] Capra, L., Emmerich, W. et Mascolo, C. (2001). Reflective middleware solutions for context−aware applications. *In Metalevel Architectures and Separation of Crosscutting Concerns*, pages 126–133. Springer.

[Card *et al.*, 1983] Card, S. K., Newell, A. et Moran, T. P. (1983). *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

[Chaari, 2007] Chaari, T. (2007). *Adaptation d'applications pervasives dans des environnements multi-contextes*. Thèse de doctorat, Institut national des sciences appliqués de Lyon, France.

[Chaari *et al.*, 2007] Chaari, T., Laforest, F. et Celentano, A. (2007). Adaptation in Context−Aware Pervasive Information Systems : The SECAS Project. *Int. Journal on Pervasive Computing and Communications(IJPCC)*, 3(4):400–425.

[Chaker et Mchirgui, 2013] Chaker, F. et Mchirgui, N. (2013). Le diabète sucré. Rapport technique 34, Faculté De Médecine Tunis.

[Chen, 2004] Chen, H. (2004). *An Intelligent Broker Architecture for Pervasive Context−Aware Systems*. Thèse de doctorat, Baltimore County : Department of CSEE, University of Maryland, United States.

[Chen *et al.*, 2009] Chen, L., Nugent, C., Mulvenna, M., Finlay, D. et Hong, X. (2009). Semantic smart homes : towards knowledge rich assisted living environments. *In Intelligent Patient Management*, pages 279–296. Springer.

[Cheng *et al.*, 2009] Cheng, B., de Lemos, R., Giese, H., Inverardi, P., Magee, J., Andersson, J., Becker, B., Bencomo, N., Brun, Y., Cukic, B., Di Marzo Serugendo, G., Dustdar, S., Finkelstein, A., Gacek, C., Geihs, K., Grassi, V., Karsai, G.,

Kienle, H., Kramer, J., Litoiu, M., Malek, S., Mirandola, R., Müller, H., Park, S., Shaw, M., Tichy, M., Tivoli, M., Weyns, D. et Whittle, J. (2009). Software engineering for self-adaptive systems : A research roadmap. *In* Cheng, B., de Lemos, R., Giese, H., Inverardi, P. et Magee, J., éditeurs : *Software Engineering for Self-Adaptive Systems*, volume 5525, pages 1–26. Springer Berlin Heidelberg.

[Chtcherbina et Franz, 2003] Chtcherbina, E. et Franz, M. (2003). Peer-to-peer coordination framework (p2pc) : Enabler of mobile ad-hoc networking for medicine, business, and entertainment. *In In Proceedings of International Conference on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet (SSGRR2003w)(L Aquila/Italy.*

[cooking hacks, 2013] cooking hacks (2013). e-health sensor platform v2.0 for arduino and raspberry pi [biometric / medical applications]. https ://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical. Dernière conultation 11/02/2016.

[Coutaz, 1990] Coutaz, J. (1990). *Interfaces homme-ordinateur : conception et réalisation.* Dunod. 455 pages.

[cube recherche, 2016] cube recherche (2016). Calculateurs en ligne. http ://www.cuberecherche.ca/frcalculateurs.php. Dernière consultation 11/02/2016.

[Da *et al.*, 2011] Da, K., Dalmau, M. et Roose, P. (2011). A Survey of adaptation systems. *International Journal on Internet and Distributed Computing Systems*, 2(1):1–18.

[Dalmau *et al.*, 2009] Dalmau, M., Roose, P. et Laplace, S. (2009). Context aware adaptable applications − a global approach. *CoRR*, abs/0909.2090.

[Danaei *et al.*, 2011] Danaei, G., Finucane, M. M., Lu, Y., Singh, G. M., Cowan, M. J., Paciorek, C. J., Lin, J. K., Farzadfar, F., Khang, Y.-H., Stevens, G. A. *et al.* (2011). National, regional, and global trends in fasting plasma glucose and diabetes prevalence since 1980 : systematic analysis of health examination surveys and epidemiological studies with 370 country-years and 2· 7 million participants. *The Lancet*, 378(9785):31–40.

[De Oliveira *et al.*, 2013] De Oliveira, K. M., Bacha, F., Mnasser, H. et Abed, M. (2013). Transportation ontology definition and application for the content personalization of user interfaces. *Expert Systems with Applications*, 40(8):3145–3159.

[Dey *et al.*, 2006] Dey, A., Sohn, T., Streng, S. et Kodama, J. (2006). icap : Interactive prototyping of context−aware applications. *In* Fishkin, K., Schiele, B., Nixon, P. et Quigley, A., éditeurs : *Pervasive Computing*, volume 3968 de *Lecture Notes in Computer Science*, pages 254–271. Springer Berlin Heidelberg.

[Dey *et al.*, 2001] Dey, A. K., Abowd, G. D. et Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context−aware applications. *Hum.−Comput. Interact.*, 16(2):97–166.

[Dey *et al.*, 1999] Dey, A. K., Salber, D., Futakawa, M. et Abowd, G. D. (1999). An architecture to support context−aware applications. Technical report, Georgia Institute of Technology.

[Diaper et Stanton, 2003] Diaper, D. et Stanton, N. (2003). *The handbook of task analysis for human-computer interaction*. CRC Press.

[Dobson *et al.*, 2006] Dobson, S., Denazis, S., Fernández, A., Gaïti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N. et Zambonelli, F. (2006). A survey of autonomic communications. *ACM Trans. Auton. Adapt. Syst.*, 1(2):223–259.

[Ferry, 2011] Ferry, N. (2011). *Adaptations dynamiques au context en informatique ambiante : proprieties logiques et temporelles*. Thèse de doctorat, Université de Nice, Sophia Antipolis, France.

[Fox et Clarke, 2009] Fox, J. et Clarke, S. (2009). Exploring approaches to dynamic adaptation. *In Proceedings of the 3rd International DiscCoTec Workshop on Middleware-Application Interaction*, pages 19–24. ACM.

[Geihs *et al.*, 2009] Geihs, K., Reichle, R., Wagner, M. et Khan, M. U. (2009). Modeling of context-aware self-adaptive applications in ubiquitous and service-oriented environments. *In Software engineering for self-adaptive systems*, pages 146–163. Springer.

[Genes, 2010] Genes, D. (2010). Diabetes diagnostics mody probability calculator. http ://www.diabetesgenes.org/content/mody-probability-calculator. Dernière conultation 11/02/2016.

[Gharsellaoui *et al.*, 2014] Gharsellaoui, A., Bellik, Y. et Jacquet, C. (2014). A system for user task monitoring and assistance in ambient intelligent settings. *In IHM'14, 26e conférence francophone sur l'Interaction Homme-Machine*, pages 130–138, Lille, France. ACM.

[Ghidini et Giunchiglia, 2001] Ghidini, C. et Giunchiglia, F. (2001). Local models semantics, or contextual reasoning= locality+ compatibility. *Artificial intelligence*, 127(2):221–259.

[Giulio *et al.*, 2002] Giulio, M., Fabio, P. et Carmen, S. (2002). Ctte : Support for developing and analyzing task models for interactive system design. *IEEE Trans. Softw. Eng.*, 28(8):797–813.

[Gottfried *et al.*, 2006] Gottfried, B., Guesgen, H. W. et Hübner, S. (2006). Spatiotemporal reasoning for smart homes. *In Designing Smart Homes*, pages 16–34. Springer.

[Gray et Salber, 2001] Gray, P. et Salber, D. (2001). Modelling and using sensed context information in the design of interactive applications. *In Engineering for Human-Computer Interaction*, pages 317–335. Springer.

[Gu *et al.*, 2005] Gu, T., Pung, H. K. et Zhang, D. Q. (2005). A service−oriented middleware for building context−aware services. *Journal of Network and Computer Applications*, 28(1):1–18.

[Haghighi *et al.*, 2008] Haghighi, P. D., Krishnaswamy, S., Zaslavsky, A. et Gaber, M. M. (2008). Reasoning about context in uncertain pervasive computing environments. *In Smart Sensing and Context*, pages 112–125. Springer.

[Hamid *et al.*, 2009] Hamid, R., Maddi, S., Johnson, A., Bobick, A., Essa, I. et Isbell, C. (2009). A novel sequence representation for unsupervised analysis of human activities. *Artificial Intelligence*, 173(14):1221–1244.

[Han *et al.*, 2008] Han, S., Song, S. K. et Youn, H. Y. (2008). Modeling and verification of context−awareness service for time critical applications using colored petri−net. *In Web Intelligence and Intelligent Agent Technology, 2008. WI−IAT'08. IEEE/WIC/ACM International Conference on*, volume 2, pages 71–74.

[Han et Youn, 2012] Han, S. et Youn, H. Y. (2012). Petri net-based context modeling for context-aware systems. *Artif. Intell. Rev.*, 37(1):43–67.

[Hanmansetty, 2004] Hanmansetty, R. (2004). *Model based approach for context aware and adaptive user interface generation.* Thèse de doctorat, Faculty of the Virginia Polytechnic Institute and State University.

[Held *et al.*, 2002] Held, A., Buchholz, S. et Schill, A. (2002). Modeling of context information for pervasive computing applications. *In Procceding of the World Multiconference on Systemics, Cybernetics and Informatics.*

[Henricksen et Indulska, 2004] Henricksen, K. et Indulska, J. (2004). Modelling and using imperfect context information. *In Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 33–37.

[Henricksen et Indulska, 2006] Henricksen, K. et Indulska, J. (2006). Developing context-aware pervasive computing applications : Models and approach. *Pervasive and Mobile Computing*, 2(1):37–64.

[Henricksen *et al.*, 2002] Henricksen, K., Indulska, J. et Rakotonirainy, A. (2002). Modeling context information in pervasive computing systems. *In Pervasive Computing*, pages 167–180. Springer.

[Henricksen *et al.*, 2003] Henricksen, K., Indulska, J. et Rakotonirainy, A. (2003). Generating context management infrastructure from high-level context models. *In 4th International Conference on Mobile Data Management (MDM)-Industrial Track*. Citeseer.

[Henricksen *et al.*, 2004] Henricksen, K., Livingstone, S. et Indulska, J. (2004). Towards a hybrid approach to context modelling, reasoning and interoperation. *In Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning And Management, in conjunction with UbiComp*, volume 2004.

[Hillah *et al.*, 2009] Hillah, L., Kindler, E., Kordon, F., Petrucci, L. et Tréves, N. (2009). A primer on the petri net markup language and iso/iec 15909−2. *Petri Net Newsletter*, (76):9–28.

[Hong *et al.*, 2009] Hong, X., Nugent, C., Mulvenna, M., McClean, S., Scotney, B. et Devlin, S. (2009). Evidential fusion of sensor data for activity recognition in smart homes. *Pervasive and Mobile Computing*, 5(3):236–252.

[Horrocks *et al.*, 2004a] Horrocks, I., Li, L., Turi, D. et Bechhofer, S. (2004a). The instance store : Dl reasoning with large numbers of individuals. *In 2004 International Workshop on Description Logics*, page 31.

[Horrocks *et al.*, 2004b] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M. *et al.* (2004b). Swrl : A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21:79.

[Horrocks *et al.*, 2003] Horrocks, I., Patel-Schneider, P. F. et Van Harmelen, F. (2003). From shiq and rdf to owl : The making of a web ontology language. *Web semantics : science, services and agents on the World Wide Web*, 1(1):7–26.

[HSPL, 2012] HSPL, H. S. P. L. (2012). Manuel de l'application diappbetes. https ://-play.google.com/store/apps/details ?id=com.xancu.diappbetes&hl=en. Dernière consultation 11/02/2016.

[Hussein *et al.*, 2011] Hussein, M., Han, J. et A., C. (2011). An architecture based approach to context−aware adaptive software systems. Rapport technique, Faculty of Information and Communication Technologies (FICT) Swinburne University of Technology, Australia.

[IGL, 1989] IGL, T. (1989). *SADT : Un langage pour communiquer*. Eyrolles.

[Indulska *et al.*, 2003] Indulska, J., Robinson, R., Rakotonirainy, A. et Henricksen, K. (2003). Experiences in using cc/pp in context-aware systems. *In Mobile Data Management*, pages 247–261.

[International, 2016] International, Diabetes, F. (2016). Idf clinical practice guidelines. http ://www.idf.org/guidelines. Dernière consultation 11/02/2016.

[Jacobson *et al.*, 2000] Jacobson, I., Booch, G. et Rumbaugh, J. (2000). *Le processus unifié de développement logiciel*. Eyrolles, Paris.

[Jameson, 2003] Jameson, A. (2003). The human-computer interaction handbook. chapitre Adaptive Interfaces and Agents, pages 305–330. L. Erlbaum Associates Inc.

[John et Kieras, 1996] John, B. E. et Kieras, D. E. (1996). The goms family of user interface analysis techniques : Comparison and contrast. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 3(4):320–351.

[Jüngel *et al.*, 2000] Jüngel, M., Kindler, E. et Weber, M. (2000). The petri net markup language. *Petri Net Newsletter*, 59(24-29):103–104.

[Kakousis *et al.*, 2010] Kakousis, K., Paspallis, N. et Papadopoulos, G. A. (2010). A survey of software adaptation in mobile and ubiquitous computing. *Enterprise Information Systems*, 4(4):355–389.

[Karantonis *et al.*, 2006] Karantonis, D., Narayanan, M., Mathie, M., Lovell, N. et Celler, B. (2006). Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring. *IEEE Transactions on Information Technology in Biomedicine*, 10(1):156–167.

[Keith-Hynes *et al.*, 2013] Keith-Hynes, P., Guerlain, S., Mize, B., Hughes-Karvetski, C., Khan, M., McElwee-Malloy, M. et Kovatchev, B. P. (2013). Dias user interface : a patient-centric interface for mobile artificial pancreas systems. *Journal of diabetes science and technology*, 7(6):1416–1426.

[Ketfi *et al.*, 2002] Ketfi, A., Belkhatir, N. et yves Cunin, P. (2002). Automatic adaptation of component-based software - issues and experiences. *In PDPTA*, pages 1365–1371. CSREA Press.

[Kieras, 1994] Kieras, D. (1994). Goms modeling of user interfaces using ngomsl. *In Conference Companion on Human Factors in Computing Systems*, CHI 94, pages 371–372, New York, NY, USA. ACM.

[Kindler, 2006] Kindler, E. (2006). The petri net markup language and iso/iec 15909-2 : Concepts, status, and future directions. *Entwurf komplexer Automatisierungssysteme*, 9:35–55.

[Klyne *et al.*, 2004] Klyne, G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M. et Tran, L. (2004). *Composite capability/preference profiles (CC/PP) : Structure and vocabularies 1.0*.

[Korpipaa *et al.*, 2003] Korpipaa, P., Mantyjarvi, J., Kela, J., Keranen, H. et Malm, E.-J. (2003). Managing context information in mobile devices. *Pervasive Computing, IEEE*, 2(3):42–51.

[Kubicki *et al.*, 2013] Kubicki, S., Lepreux, S. et Kolski, C. (2013). Distributed ui on interactive tabletops : Issues and context model. *In* Lozano, M. D., Gallud, J. A., Tesoriero, R. et Penichet, V. M., éditeurs : *Distributed User Interfaces : Usability and Collaboration*, Human−Computer Interaction Series, pages 27–38. Springer London.

[Kwon, 2004] Kwon, O. B. (2004). Modeling and generating context-aware agent-based applications with amended colored petri nets. *Expert Systems with Applications*, 27(4):609–621.

[Lenz et Oberweis, 2003] Lenz, K. et Oberweis, A. (2003). Inter−organizational business process management with xml nets. *In* Ehrig, H., Reisig, W., Rozenberg, G. et Weber, H., éditeurs : *Petri Net Technology for Communication−Based Systems*, volume 2472 de *Lecture Notes in Computer Science*, pages 243–263. Springer Berlin Heidelberg.

[Liebl *et al.*, 2013] Liebl, A., Henrichs, H. R., Heinemann, L., Freckmann, G., Biermann, E., Thomas, A., of the Working Group Diabetes Technology of the German Diabetes Association, C. G. M. W. G. *et al.* (2013). Continuous glucose monitoring : evidence and consensus statement for clinical use. *Journal of diabetes science and technology*, 7(2):500–519.

[Lowd et Domingos, 2005] Lowd, D. et Domingos, P. (2005). Naive bayes models for probability estimation. *In Proceedings of the 22nd international conference on Machine learning*, pages 529–536. ACM New York, NY, USA.

[McCarthy et Buvac, 1997] McCarthy, J. et Buvac, S. (1997). Formalizing context (expanded notes). Rapport technique, Stanford, CA, USA.

[Moalla, 1985] Moalla, M. (1985). Réseaux de petri interprétés et grafcet. *TSI-Technique et Science Informatique*, 14(1):17–30.

[Moussa, 2005] Moussa, F. (2005). *Vers une méthodologie globale de conception et de génération semi-automatique des IHM pour les systèmes industriels.* Habilitation universitaire en informatique, Faculté des Sciences de Tunis, Tunisie.

[Moussa *et al.*, 2015] Moussa, F., Ismail, I. et Jarraya, M. (2015). Towards a runtime evolutionary model of user-adapted interaction in a ubiquitous environment : the RADEM formal model. *Cognition, Technology & Work*, 17(3):391–415.

[Moussa *et al.*, 2006] Moussa, F., Kolski, C. et Riahi, M. (2006). Analyse des dysfonctionnements des systèmes complexes en amont de la conception des ihm : apports, difficultés, et étude de cas. *Revue d'Interaction Homme-Machine*, 7:79–111.

[Moussa *et al.*, 2002] Moussa, F., Riahi, M., Kolski, C. et Moalla, M. (2002). Interpreted petri nets used for human−machine dialogue specification. *Integrated Computer−Aided Engineering*, 9(1):87–98.

[Murata, 1989] Murata, T. (1989). Petri nets : Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580.

[Naïm *et al.*, 2011] Naïm, P., Wuillemin, P., Leray, P., Pourret, O. et Becker, A. (2011). *Réseaux bayésiens*. Algorithmes. Eyrolles.

[Nevatia *et al.*, 2004] Nevatia, R., Hobbs, J. et Bolles, B. (2004). An ontology for video event representation. *In Computer Vision and Pattern Recognition Workshop, 2004. CVPRW 04. Conference on*, pages 119–119. IEEE.

[Nielsen, 1993] Nielsen, J. (1993). *Usability engineering*. Academic Press.

[Palanque et Bastide, 1995] Palanque, P. et Bastide, R. (1995). Spécifications formelles pour l'ingénierie des interfaces homme-machine. *TSI. Technique et science informatiques*, 14(4):473–500.

[Park *et al.*, 2007] Park, A. H., Park, S. H. et Youn, H. Y. (2007). A flexible and scalable agent platform for multi−agent systems. *Transactions on engineering, computing and technology, Enformatika*, 19:1–6.

[Pascoe *et al.*, 2007] Pascoe, J., Thomson, K. et Rodrigues, H. (2007). Context−awareness in the wild : An investigation into the existing uses of context in everyday life. *In* Meersman, R., Tari, Z. et Herrero, P., éditeurs : *On the Move to Meaningful Internet Systems 2007 : OTM 2007 Workshops*, volume 4805 de *Lecture Notes in Computer Science*, pages 193–202. Springer Berlin Heidelberg.

[Paterno, 2000] Paterno, F. (2000). *Model-Based Design and Evaluation of Interactive Applications*. Springer-Verlag, London, UK, UK, 1st édition.

[Place *et al.*, 2013] Place, J., Robert, A., Brahim, N. B., Keith-Hynes, P., Farret, A., Pelletier, M.-J., Buckingham, B., Breton, M., Kovatchev, B. et Renard, E. (2013). Dias web monitoring : a real-time remote monitoring system designed for artificial pancreas outpatient trials. *Journal of diabetes science and technology*, 7(6):1427–1435.

[Rasmussen, 1986] Rasmussen, J. (1986). A framework for cognitive task analysis in systems design. *In* Hollnagel, E., Mancini, G. et Woods, D., éditeurs : *Intelligent Decision Support in Process Environments*, volume 21 de *NATO ASI Series*, pages 175–196. Springer Berlin Heidelberg.

[Reichart *et al.*, 2008] Reichart, D., Dittmar, A., Forbrig, P. et Wurdel, M. (2008). Tool support for representing task models, dialog models and user−interface specifications. *In* Graham, T. et Palanque, P., éditeurs : *Interactive Systems. Design, Specification, and Verification*, volume 5136 de *Lecture Notes in Computer Science*, pages 92–95. Springer Berlin Heidelberg.

[Reichel *et al.*, 2013] Reichel, A., Rietzsch, H., Ludwig, B., Röthig, K., Moritz, A. et Bornstein, S. R. (2013). Self-adjustment of insulin dose using graphically depicted self-monitoring of blood glucose measurements in patients with type 1 diabetes mellitus. *Journal of diabetes science and technology*, 7(1):156–162.

[Reignier *et al.*, 2007] Reignier, P., Brdiczka, O., Vaufreydaz, D., Crowley, J. L. et Maisonnasse, J. (2007). Context-aware environments : from specification to implementation. *Expert Systems*, 24(5):305–320.

[Rey, 2006] Rey, G. (2006). Méthode pour la modélisation du contexte d'interaction. *Ingénierie des Systèmes d'Information*, 11(5):141–166.

[Riahi et Moussa, 2014] Riahi, I. et Moussa, F. (2014). Formal modeling for pervasive design of human-computer interfaces. *In UBICOMM, The Eighth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, pages 35–43.

[Riahi et Moussa, 2015] Riahi, I. et Moussa, F. (2015). A formal approach for modeling context-aware human-computer system. *Computers & Electrical Engineering*, 44:241–261.

[Riahi *et al.*, 2013] Riahi, I., Moussa, F. et Riahi, M. (2013). Petri nets context modeling for the pervasive human−computer interfaces. *In* Brézillon, P., Blackburn, P. et Dapoigny, R., éditeurs : *CONTEXT*, volume 8175 de *Lecture Notes in Computer Science*, pages 316–329. Springer.

[Riahi *et al.*, 2011a] Riahi, I., Riahi, M. et Moussa, F. (2011a). Xml and petri nets in specification and generation of mobile hci. *In 15th IEEE International Conference on Intelligent Engineering Systems (INES)*, pages 239–244.

[Riahi *et al.*, 2011b] Riahi, I., Riahi, M. et Moussa, F. (2011b). Xml in formal specification, verification and generation of mobile hci. *In* Jacko, J., éditeur : *Human−Computer Interaction. Towards Mobile and Intelligent Interaction Environments*, volume 6763 de *Lecture Notes in Computer Science*, pages 92–100. Springer Berlin Heidelberg.

[Riahi, 2004] Riahi, M. (2004). *Contribution à l'élaboration d'une méthodologie de spécification, de vérification et de génération semi−automatique d'interfaces homme−machine : Application à l'outil Ergo−Conceptor +*. Thèse de doctorat, Université de Valenciennes et du Hainaut−Cambrésis.

[Ryan, 1999] Ryan, N. (1999). Contextml : Exchanging contextual information between a mobile client and the fieldnote server. *Computing Laboratory, University of Kent at Canterbury*.

[Ryan *et al.*, 1997] Ryan, N., Pascoe, J. et Morse, D. (1997). Enhanced reality fieldwork : the context aware archaeological assistant. *Bar International Series*, 750:269–274.

[Sabagh et Al-Yasiri, 2015] Sabagh, A. et Al-Yasiri, A. (2015). Gecaf : a framework for developing context-aware pervasive systems. *Computer Science - Research and Development*, 30(1):87–103.

[Samulowitz *et al.*, 2002] Samulowitz, M., Michahelles, F. et Linnhoff-Popien, C. (2002). Capeus : An architecture for context-aware selection and execution of services. *In New Developments in Distributed Applications and Interoperable Systems*, volume 70, pages 23–39.

[Schilit *et al.*, 1994] Schilit, B., Adams, N. et Want, R. (1994). Context-aware computing applications. *In Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, WMCSA '94, pages 85–90. IEEE Computer Society.

[Schmidt et Van Laerhoven, 2001] Schmidt, A. et Van Laerhoven, K. (2001). How to build smart appliances ? *Personal Communications, IEEE*, 8(4):66–71.

[Silva *et al.*, 2009] Silva, J. L., Campos, J. C. et Harrison, M. D. (2009). An infrastructure for experience centered agile prototyping of ambient intelligence. *In Proceedings of the 1st ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '09, pages 79–84. ACM.

[Simonin et Carbonell, 2007] Simonin, J. et Carbonell, N. (2007). Interfaces adaptatives Adaptation dynamique à l'utilisateur courant. *In Interfaces numériques*, page 18 pages. Hermès Science.

[Song *et al.*, 2007] Song, S. K., Han, S. et Youn, H. Y. (2007). A new agent platform architecture supporting the agent group paradigm for multi−agent systems. *In Intelligent Agent Technology, 2007. IAT '07. IEEE/WIC/ACM International Conference on*, pages 399–402.

[Strang et Linnhoff−Popien, 2004] Strang, T. et Linnhoff−Popien, C. (2004). A Context Modeling Survey. *In In : Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 − The Sixth International Conference on Ubiquitous Computing, Nottingham/England.*

[Sun *et al.*, 2010] Sun, J., Zhang, Y. et He, K. (2010). A petri-net based context representation in smart car environment. *In Proceedings of the 5th International Conference on Advances in Grid and Pervasive Computing*, GPC 10, pages 162–173, Berlin, Heidelberg. Springer-Verlag.

[Tabary et Abed, 2002] Tabary, D. et Abed, M. (2002). A software environment task object−oriented design (etood). *Journal of Systems and Software*, 60(2):129–140. Artificial and Computational Intelligence for Decisions, Control, and Automation in Engineering and Industrial Applications.

[Tardieu *et al.*, 1984] Tardieu, H., Rochfeld, A. et Colletti, R. (1984). La méthode merise-principes et outils.

[Thevenin et Coutaz, 1999] Thevenin, D. et Coutaz, J. (1999). Plasticity of User Interfaces : Framework and Research Agenda. *Proc Interact'99 Edinburgh A Sasse C Johnson Eds IFIP IOS Press Publ*, 99:110–117.

[Tigli *et al.*, 2009] Tigli, J.-Y., Lavirotte, S., Rey, G., Hourdin, V., Cheung-Foo-Wo, D., Callegari, E. et Riveill, M. (2009). Wcomp middleware for ubiquitous computing : Aspects and composite event-based web services. *Annals of telecommunications-annales des télécommunications*, 64:197–214.

[Timmerer *et al.*, 2010] Timmerer, C., Jabornig, J. et Hellwagner, H. (2010). A survey on delivery context description formats − a comparison and mapping model. *Journal of Digital Information Management*, 8(1):16–27.

[Turaga *et al.*, 2008] Turaga, P., Chellappa, R., Subrahmanian, V. S. et Udrea, O. (2008). Machine recognition of human activities : A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1473–1488.

[Van Biljon, 1988] Van Biljon, W. R. (1988). Extending petri nets for specifying man–machine dialogues. *International Journal of Man-machine studies*, 28(4):437–455.

[van der Veer et van Welie, 2000] van der Veer, G. et van Welie, M. (2000). Task based groupware design : putting theory into practice. *In Proceedings of the 3rd conference on Designing interactive systems : processes, practices, methods, and techniques*, pages 326–337. ACM.

[van Kasteren et Krose, 2007] van Kasteren, T. et Krose, B. (2007). Bayesian activity recognition in residence for elders. *In International Conference on Intelligent Environments, 2007. IE 07. 3rd IET*, pages 209–212.

[van Kasteren *et al.*, 2008] van Kasteren, T., Noulas, A., Englebienne, G. et Kröse, B. (2008). Accurate activity recognition in a home setting. *In Proceedings of the 10th International Conference on Ubiquitous Computing*, UbiComp 08, pages 1–9. ACM.

[Vanderdonckt, 2011] Vanderdonckt, J. (2011). User interface extensible markup language. http ://www.usixml.org. Dernière consultation 11/02/2016.

[Ward *et al.*, 1997] Ward, A., Jones, A. et Hopper, A. (1997). A new location technique for the active office. *Personal Communications, IEEE*, 4(5):42–47.

[Weiser, 1991] Weiser, M. (1991). The computer for the 21st century. *Scientific American*, 265(3):94–104.

[Wilson *et al.*, 1993] Wilson, S., Johnson, P., Kelly, C., Cunningham, J. et Markopoulos, P. (1993). Beyond hacking : a model based approach to user interface design. *People and computers*, pages 217–217, Cambridge University Press.

[World, 2009] World, H. O. (2009). *Global health risks : mortality and burden of disease attributable to selected major risks*. World, Health, Organization.

[Wurdel *et al.*, 2008] Wurdel, M., Propp, S. et Forbrig, P. (2008). Hci−task models and smart environments. *In* Forbrig, P., Paternò, F. et Pejtersen, A., éditeurs : *Human−Computer Interaction Symposium*, volume 272 de *IFIP International Federation for Information Processing*, pages 21–32. Springer US.

[Xue *et al.*, 2014] Xue, Q., Han, X., Li, M. et Liu, M. (2014). A conceptual architecture for adaptive human-computer interface of a pt operation platform based on context-awareness. *Discrete Dynamics in Nature and Society*, 2014.

[Yamada *et al.*, 2007] Yamada, N., Sakamoto, K., Kunito, G., Isoda, Y., Yamazaki, K. et Tanaka, S. (2007). Applying ontology and probabilistic model to human activity recognition from surrounding things. *Information and Media Technologies*, 2(4):1286–1297.

[Yang *et al.*, 2008] Yang, J.-Y., Wang, J.-S. et Chen, Y.-P. (2008). Using acceleration measurements for activity recognition : An effective learning algorithm for constructing neural classifiers. *Pattern recognition letters*, 29(16):2213–2220.

[Yang *et al.*, 2002] Yang, Z., Cheng, B. H. C., Stirewalt, R. E. K., Sowell, J., Sadjadi, S. M. et McKinley, P. K. (2002). An aspect-oriented approach to dynamic adaptation. *In Proceedings of the First Workshop on Self-healing Systems*, pages 85–92. ACM.

[Yau et Liu, 2006] Yau, S. S. et Liu, J. (2006). Hierarchical situation modeling and reasoning for pervasive computing. *In The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, 2006 and the 2006 Second International Workshop on Collaborative Computing, Integration, and Assurance. SEUS 2006/WCCIA 2006.*, pages 5–10. IEEE.

[Ye *et al.*, 2007a] Ye, J., Coyle, L., Dobson, S. et Nixon, P. (2007a). Ontology-based models in pervasive computing systems. *Knowl. Eng. Rev.*, 22(4):315–347.

[Ye *et al.*, 2007b] Ye, J., Coyle, L., Dobson, S. et Nixon, P. (2007b). Using situation lattices to model and reason about context. *Proceedings of MRC 2007 (coexist with CONTEXT 07)*, pages 1–12.

[Ye *et al.*, 2012] Ye, J., Dobson, S. et McKeever, S. (2012). Situation identification techniques in pervasive computing : A review. *Pervasive Mob. Comput.*, 8(1):36–66.

[Zhang *et al.*, 2005] Zhang, D., Gu, T. et Wang, X. (2005). Enabling context-aware smart home with semantic web technologies. *International Journal of Human-friendly Welfare Robotic Systems*, 6(4):12–20.