Ministère de l'enseignement supérieur et de la recherche scientifique
وزارة التعليم العالي والبحث العلمي

BADJI MOKHTAR-ANNABA UNIVERSITY
UNIVERSITE BADJI MOKHTAR-ANNABA

جامعة باجي مختار- عنابة

Faculté des sciences de l'ingéniorat
Département d'Informatique

Année : 2015/2016

THESE

présentée en vue de l'obtention du diplôme de

Doctorat en sciences

Intitulée :

Adaptation contextuelle et personnalisation pour les systèmes d'information ubiquitaires

Filière
Informatique

Présentée par
Mohamed-Salah BENSELIM

Devant le jury

Président	**Pr. MESLATI Djamel,**	**Université Badji Mokhtar, Annaba**
Rapporteur	**Pr. SERIDI Hassina,**	**Université Badji Mokhtar, Annaba**
Examinateur	**Pr. AMIRAT Abdelkrim,**	**Université Mohamed-Cherif Messaadia, Souk Ahras**
Examinateur	**Pr. MOKHATI Farid,**	**Université Larbi Ben M'Hidi, Oum El Bouaghi**
Examinateur	**Dr. LAFIFI Yacine,**	**Université 08 Mai 1945, Guelma**

بسم الله الرحمن الرحيم

« وَمَا بِكُمْ مِنْ نِعْمَةٍ فَمِنَ اللَّهِ »

REMERCIEMENTS

Je dois ma gratitude et ma reconnaissance à plusieurs personnes qui m'ont aidé et soutenu tout au long de la réalisation de cette thèse et auxquelles je dois, aussi, mes plus sincères remerciements.

Tout d'abord, je dois remercier la directrice de ma thèse, le Professeur Seridi-Bouchelaghem Hassina, pour ses précieux conseils et orientations durant toutes les années d'encadrement. Je la remercie, aussi, pour ses encouragements, son soutien et sa patience qui m'a beaucoup aidé pour mener à bien ce travail de thèse.

Je tiens à remercier également le Professeur Maslati Djamel d'avoir bien voulu présider le jury lors de ma soutenance de thèse.

Je remercie également le Professeur Amirat Abdelkrim, le Professeur Mokhati Farid et le Docteur Lafifi Yacine d'avoir accepté de lire et d'examiner cette thèse.

Je remercie beaucoup le Professeur Seridi Hamid pour l'intérêt et les conseils qu'il m'a accordé.

Je dois des remerciements très particuliers à ma femme et mes enfants pour leur amour, leur soutien et leur patience surtout pendant la période de finalisation de cette thèse. Sans eux, je ne serais jamais arrivé si loin.

Mes sincères remerciements s'étendent, aussi, à tous mes collègues, mes amis et ma famille, particulièrement mon père, pour leurs encouragements incessants.

Enfin, merci à toutes les personnes qui ont rendu possible cette thèse.

DÉDICACES

A la mémoire de ma mère,

A mon cher père pour son sacrifice et son aide,

*A mon épouse "**WIDAD**" pour sa patience et son appui,*

*A mes trésors, mes fils "**YAHIA**" et "**ANIS**",*

A mes frères et mes sœurs qui m'ont tant soutenu et encouragé,

Et à tous mes collègues et mes amis.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLES

ABRÉVIATIONS ET ACRONYMES

2TUP	2 Track Unified Process
3TUP	3 Track Unified Process
ACID	Atomicité, Cohérence, Isolation et Durabilité
CIM	Computation Independent Model
CC/PP	Composite Capabilities / Preference Profiles
CM	Contextual Model
CoOL	Context Ontology Language
CSCW	Computer Supported Cooperative Work
CSOA	Context-aware Service Oriented Architecture
CWM	Common Warehouse Metamodel
EDOC	Enterprise Distributed Object Computing
GPS	Global Positioning System
HTML	HyperText Markup Language
J2EE	Java Enterprise Edition
MDA	Model Driven Architecture
MDE	Model Driven Engineering
MM	Merged Model
MOF	Meta Object Facility
OCL	Object Constraint Language
OMG	Object Management Group
PDA	Personal Digital Assistant
PDM	Platform Description Model
PIM	Platform Independent Model
PSM	Platform Specific Model

PUMAS	Peer Ubiquitous Multi-Agent System
RDF	Ressource Description Framework
SECAS	Simple Environment for Context Aware Systems
UML	Unified Modeling Language
UP	Unified Process
WSDL	Web Service Description Language
XMI	XML Metadata Interchange
XML	eXtended Markup Language

RÉSUMÉ

Résumé

L'évolution gigantesque de l'informatique ubiquitaire nécessite énormément d'efforts quant à la conception et à la réalisation des systèmes d'information modernes. Aujourd'hui, le génie logiciel présente une grande tendance vers le développement d'applications destinées aux environnements ubiquitaires. Cette tendance sera contrainte par les modalités de prise en charge des paramètres spécifiques de l'ubiquité (mobilité, hétérogénéité, distribution, etc.). Les applications visées cherchent à satisfaire les préférences de l'utilisateur de plus en plus nomade et exigeant. La meilleure solution pour atteindre cette satisfaction est de concevoir des applications qui peuvent s'adapter aux changements continus subis par le contexte d'utilisation et qui peuvent fournir des informations plus pertinentes et personnalisées.

L'objectif de notre étude est de proposer des moyens et des outils capables d'assurer un certain degré d'adaptation et de personnalisation dans les systèmes d'information ubiquitaires. Par cette proposition, nous visons à enrichir le domaine de la sensibilité au contexte par des concepts qui peuvent contribuer au développement d'applications contextuellement adaptables. Notre contribution s'articule autour de trois volets : premièrement, la séparation des préoccupations contextuelles en introduisant le processus 3TUP (3 Track Unified Process) qui permet d'étudier les contraintes contextuelles indépendamment des autres types de contraintes (métiers et techniques). Deuxièmement, la personnalisation du langage UML (Unified Modeling Language) en proposant un profil UML destiné à la modélisation dans le domaine du context-awareness. Et troisièmement, l'adaptation contextuelle des applications en intégrant le modèle contextuel (représentation du contexte d'utilisation seulement) dans le cycle de vie de la MDA par une opération de fusion de modèles. A la fin de cette étude, nous construisons quatre diagrammes UML (classes, cas d'utilisation, séquences et activités) d'un exemple en utilisant les nouvelles notations du profil proposé.

Mots-clés : génie logiciel, système d'information, contexte d'utilisation, adaptation, personnalisation, UML, modélisation, séparation des préoccupations.

ABSTRACT

Abstract

The colossal emergence of new Web technologies and the evolution of ubiquitous computing require a lot of efforts as for design and development of information systems. Nowadays, software engineering moves toward the development of distributed and ubiquitous applications. This tendency is constrained by parameters such as mobility and heterogeneity that characterize the current situation of the user. Future applications try to satisfy all preferences of a user and to cover his special requirements. Designing adapted applications is the best way to reach this satisfaction.

The main goal of our study is to provide some means and tools that are able to guarantee a high degree of adaptation and personalization for ubiquitous information systems. In this study, we aim to enrich the domain of context-awareness by providing new concepts that can contribute to the development of context-aware applications. Our proposal is defined by three complementary sides. Firstly, we propose the separation of contextual concerns from other types of concerns (business and technical). In this first side, we introduce the notion of 3TUP (3 Track Unified Process). Secondly, we provide a new UML profile destined especially to the particular domain of context-awareness. The proposed profile comprises new UML notations that are obtained after extension of existing ones in order to be well adequate for context modeling. Thirdly, we propose a new vision of MDA approach that can take in charge the context of use in its development process. This can be possible by representing the context in a model and then integrating this contextual model in the MDA process by model fusion operation. At the end of this study, we construct four UML diagrams (class, use case, sequence and activity) of an example by using the new notations of the proposed profile.

Keywords: software engineering, information system, context of use, adaptation, personalization, UML, modelling, concerns separation.

ملـخـص

إن التطور الهائل للمعلوماتية المطلقة التواجد يستلزم بذل الكثير من الجهود من أجل تصميم و إنجاز أنظمة الإعلام المعاصرة. حاليا، تتوجه هندسة البرامج نحو تطوير تطبيقات موزعة و متوفرة مهما كانت وضعية المستعمل عبر الزمان و المكان. يتميز هذا التوجه بضرورة إدماج الخصوصيات الجديدة مثل تنقلات المستعمل، خيارات المستعمل و تنوع الأجهزة المستعملة. تهدف التطبيقات الجديدة إلى إرضاء المستعمل من خلال الأخذ بعين الاعتبار لكل خصائصه و خياراته. من بين الحلول العملية التي تستطيع تحقيق هذا الهدف هو تطوير تطبيقات و برامج قادرة على التأقلم و التكيف مع كل التغييرات التي تمس سياق الاستعمال و التي قد تحدث خلال عملية استغلال هذه البرامج.

الهدف من هذه الدراسة هو اقتراح وسائل جديدة قادرة على ضمان قدر معين من تكييف التطبيقات و تخصيص المعلومات الخاصة بأنظمة الإعلام المطلقة التواجد. من خلال هذا العمل، نبحث عن إثراء ميدان التكيف مع تحولات سياق الاستعمال و ذلك بتوفير مبادئ جديدة تستطيع أن تساعد على تطوير و إنجاز تطبيقات مكيفة. يتلخص اقتراحنا في ثلاث محاور رئيسية. أولا، نقوم بفصل العناصر المكونة لسياق الاستعمال عن كل العناصر الأخرى ، لأجل ذلك نقترح استعمال طريقة جديدة تعتمد على أساس « 3TUP : 3 Track Unified Process » و تمكن من دراسة خصائص الأنظمة من ثلاث جهات (الوظيفية و التكنولوجية و سياق الاستعمال) . ثانيا، تخصيص لغة النمذجة (UML) لكي تتلاشى مع خصوصية مكونات سياق الاستعمال. ثالثا، نقترح طريقة جديدة ترتكز على هندسة النماذج باستعمال أسس الـ MDA . تتمثل هذه الطريقة في إدخال نموذج خاص بسياق الاستعمال في دورة حياة الـMDA و ذلك باستعمال عملية دمج النماذج.

كلمات رئيسية: هندسة البرامج، أنظمة الإعلام، سياق الاستعمال، تكييف، تخصيص، UML ، نمذجة، فصل العناصر.

INTRODUCTION GÉNÉRALE

i. Contexte général du travail

Cette thèse s'inscrit dans le domaine de la sensibilité au contexte (Context-Awareness Domain) et plus particulièrement sur les notions d'adaptation et de personnalisation pour les systèmes d'information ubiquitaires.

Depuis sa naissance aux années 70, le concept de système d'information ne cesse de gagner du terrain dans les entreprises en devenant une clé magique pour assurer leur gestion et leur organisation. Ce concept tire sa force de sa capacité d'interagir avec les différents acteurs influents comme les décideurs, les exécutants, les facteurs de l'environnement extérieur dans le but d'optimiser la rentabilité des moyens humains, matériels, financiers et techniques disponibles. Un système d'information représente le principal responsable de la communication dans une organisation grâce au rôle de coordination informationnelle (collecte, stockage, traitement, diffusion, etc.) qu'il joue entre les services et les activités de cette organisation.

Mais, avec l'évolution gigantesque des différentes technologies (web, information, source d'information, matériel utilisé, etc.), cette catégorie de systèmes d'information telle qu'elle a été définie initialement a prouvé ses limites et est devenue, malheureusement, du type « classique » ou « traditionnel ».

Pour pouvoir suivre cette évolution, les systèmes d'information sensés être « modernes » doivent faire face aux multiples défis et facteurs tels que : la flexibilité, l'hétérogénéité, la distribution, la mobilité, la sécurité, etc. Et c'est à partir de ces défis que nait le terme d' « ubiquité » signifiant tout simplement la disponibilité de l'information n'importe où et n'importe quand. Dès lors, nous parlons de systèmes d'information ubiquitaires. Ces derniers se chargent de mettre à disposition de l'utilisateur toute l'information requise en temps et en espace. Ces systèmes peuvent interagir de manière efficace, harmonieuse et performante afin de pouvoir fournir les informations et les services attendus par l'utilisateur.

La question qui se pose, maintenant, est : est-ce que l'information retournée peut satisfaire les souhaits de l'utilisateur, ses préférences et ses exigences de plus en plus particulières ?

Pour répondre à cette question, il faut démontrer si les deux critères de l'ubiquité (temps et espace) sont suffisants ou non pour fournir l'information pertinente à l'utilisateur. En effet, la majorité des résultats suite à une requête donnée sont très nombreux, vagues et parfois confus. Pour restreindre et limiter le volume et la pertinence de ces résultats suivant les caractéristiques formulées par l'utilisateur, nous aurons besoin d'appliquer de nouveaux

critères susceptibles de satisfaire ces caractéristiques. Alors, la notion de « contexte d'utilisation » a été introduite pour regrouper toutes les préférences de l'utilisateur ainsi que toutes les contraintes spécifiques environnantes pouvant influencer la situation courante de cet utilisateur.
En plus des deux critères temps et espace, les systèmes d'information ubiquitaires doivent présenter un caractère sensible au contexte et nécessitent, par conséquent, la prise en considération des variations du contexte d'utilisation d'une situation à une autre. C'est-à-dire que ces systèmes doivent, selon le contexte de chaque situation d'exécution, fournir l'information correspondante. Pour l'utilisateur, cette information correspondante signifie qu'elle soit ajustée, adaptée et personnalisée suivant ses souhaits préférés comme le nombre, le contenu, la présentation, etc. D'où, l'utilité de faire appel aux processus d'adaptation et de personnalisation pour augmenter et améliorer l'application et la performance des systèmes d'information ubiquitaires. Par définition, l'adaptation consiste à apporter des modifications sur une application (ou à un système) dans le but d'assurer ses fonctions et d'améliorer ses performances ; tandis la personnalisation représente l'ajustement des informations fournies, des contenus retournés ou des services proposés au contexte dans lequel se trouve l'utilisateur. L'adaptation et la personnalisation constituent la solution appropriée pour filtrer l'information pertinente dissoute parmi tant d'autres généralement moins intéressantes ou même sans intérêt par rapport aux besoins de l'utilisateur.
C'est là que se classe notre étude présentée dans cette thèse. Ainsi, nous cherchons, à travers cette étude, de trouver des moyens et des outils concrets capables d'assurer l'adaptation contextuelle et la personnalisation pour les systèmes d'information ubiquitaires. Notre objectif s'étale, à la fois, sur les applications et sur les informations puisque la notion d'adaptation est toujours liée aux applications tandis la personnalisation est liée aux informations, et ce, en se basant sur le principe de la séparation des préoccupations contextuelles d'un système.

ii. Motivations

L'utilisation colossale du Web nécessite une nouvelle vision des systèmes d'information distants. L'évolution croissante des technologies hard (réseaux sans fil, dispositifs mobiles, PC portables, téléphones mobiles,…) destinées pour l'accès à une information distante a poussé la communauté du génie logiciel à suivre cette allure de croissance en garantissant des systèmes d'information plus adaptés et plus personnalisés. En effet, le Web nous permet de rechercher, d'accéder, d'exploiter et d'échanger les informations provenant de plusieurs sources d'information (hétérogènes et distantes) et en utilisant plusieurs dispositifs (fixes ou mobiles).
Le besoin d'assurer la disponibilité de l'information partout et à tout moment a conduit vers une forte émergence du concept de l'informatique ubiquitaire. Bénéficiant des avantages de ce concept, chaque utilisateur peut, de n'importe où et n'importe quand, interroger le système ou interagir avec une application pour avoir des réponses et des résultats bien définies. Ces réponses, même si elles sont justes, sont généralement très nombreuses et ne sont pas toutes aussi intéressantes et pertinentes; c'est-à-dire qu'elles ne répondent pas toutes aux souhaits de l'utilisateur. Celui-ci sera obligé de passer un temps considérable dans la recherche de l'information pertinente parmi les réponses proposées. L'objectif est de réduire la liste des réponses fournies par la prise en considération des souhaits de l'utilisateur et des conditions

d'exécution de l'application. Nous parlons, alors, d'adaptation et de personnalisation ; c'est-à-dire la prise en compte des préférences de l'utilisateur et du contexte d'utilisation par les applications et les systèmes d'information.
Les concepteurs de systèmes d'information ne doivent pas négliger ni ignorer les contraintes contextuelles qui entourent l'exécution d'une application et l'interaction avec l'utilisateur. Ceci permettra d'avoir des applications sensibles au contexte qui, d'une part, fournissent des informations personnalisées et pertinentes et, d'autre part, s'adaptent à la variation des conditions d'exécution issue de l'informatique ubiquitaire. La question est comment une application peut-elle prendre en charge les variations et les changements des conditions d'exécution (contexte d'utilisation et préférences de l'utilisateur), et à quel niveau (par rapport à l'application) faut-il introduire ou insérer les informations contextuelles capturées. Ceci nous amène à réfléchir sur la manière d'adapter les applications et de personnaliser les outils afin d'assurer un haut degré d'adaptation contextuelle et de personnalisation pour les systèmes d'information ubiquitaires caractérisés par les changements continus du contexte d'utilisation (changements dus à l'environnement, à l'application et à l'utilisateur).
Les nouvelles tendances s'orientent vers l'utilisation de la notion de « modèle » et l'ingénierie des modèles (MDE : Model Driven Engineering) qui présente de grandes facilités pour représenter, traiter, stocker et exploiter les informations. Dans le domaine de l'ingénierie des modèles, il existe deux produits de références et de grande réputation : l'approche MDA (Model Driven Architecture) et le langage UML (Unified Modeling Language). En effet, la MDA offre plusieurs opportunités dans le domaine du génie logiciel comme l'utilisation de modèles pour formaliser et gérer l'information, la séparation des différentes propriétés d'un système (fonctionnels et technologiques), la portabilité, l'interopérabilité et la réutilisation. De son côté, le langage UML présente plusieurs avantages (notations simplifiées et non ambiguës, formalisation précise des concepts, couverture importante des besoins) qui ont fait de lui un standard parmi les langages de modélisation disponibles.
Malgré ces avantages, ni l'approche MDA ni le langage UML ne prennent en charge, d'une façon claire et explicite, la notion de « contexte d'utilisation » permettant l'adaptation des applications développées. En effet, la MDA actuelle ne prévoit pas, distinctement, la prise en considération des informations contextuelles dans son processus de développement. Aussi, le langage UML ne propose pas de notations spécifiques pour le domaine particulier de la sensibilité au contexte « Context-Awareness Domain ». Cette notion, liée aux changements continuels de la situation courante de l'utilisateur, représente une partie cruciale pour l'adaptation des applications et pour la personnalisation des informations de l'utilisateur surtout lorsque nous évoluons dans un environnement nomade et ubiquitaire.
Tout cela peut justifier le réel besoin à proposer des moyens et des outils qui permettent de prendre en charge tous les éléments du contexte d'utilisation dans le but de développer des applications plus adaptées à ce contexte d'utilisation et de fournir, ainsi, des informations personnalisées suivant les besoins et les préférences des utilisateurs.

iii. Problématique

Dans un environnement ubiquitaire, l'utilisateur est toujours nomade et de plus en plus exigeant quant à la qualité des informations demandées (nature, contenu, affichage,…). L'utilisation de différents dispositifs mobiles conduit à un changement perpétuel (temps,

espace, environnement) de la situation courante de l'utilisateur. Ces paramètres affectent directement la nature et le mode d'exécution d'une tache. Dans ce cas, les développeurs auront en face le problème d'accès au même système en différentes situations et avec divers dispositifs. Aussi, ils seront contraints de prendre en charge et de résoudre le problème d'hétérogénéité qui concerne les informations, les sources d'information, les matériels et les logiciels.

Les applications qui opèrent dans les environnements ubiquitaires présentent de multiples défis comme la mobilité des utilisateurs, la flexibilité des architectures logicielles, la distribution massive des composants, la forte hétérogénéité des sources d'information, la forte hétérogénéité des ressources utilisées, l'énorme utilisation des données, la disponibilité des informations, la sécurité, etc. Toutes les informations manipulées par ces applications sont des informations liées au contexte d'utilisation qui doit être nécessairement pris en charge par de telles applications. C'est-à-dire que les résultats retournés par une de ces applications doivent être ajustés suivant les constituants du contexte d'utilisation. L'opération d'ajustement des résultats signifie l'adaptation des applications et/ou la personnalisation des informations. Ces deux notions ont été largement traitées lors des études menées dans le domaine de la sensibilité au contexte ; mais en proposant des solutions plus ou moins concrètes et convaincantes. En effet, la grande majorité des propositions faites dans le domaine de la contextualisation cherchent à adapter l'application pendant son exécution à l'aide d'outils ou de modules indépendants. Ceci exclut le caractère « automatique » du processus d'adaptation parce qu'il faut toujours effectuer des actions d'adaptation qui sont liées aux caractéristiques de l'architecture et de la composition de l'outil d'adaptation comme la nature des informations contextuelles, le type des capteurs, le mode de représentation et de traitement du contexte ; et parce qu'il sera très difficile, ou même impossible, de prendre en considération les nouveaux éléments du contexte qui n'auraient pas été prévus préalablement.

Aussi, parmi les travaux existants, nous distinguons ceux qui se sont penchées sur la modélisation du contexte d'utilisation avec modification du méta-modèle UML. Ceci posera trop de problèmes concernant la réutilisation et l'interopérabilité lors des futures utilisations du modèle contextuel dans d'autres systèmes se basant sur les notations du méta-modèle initial du langage UML.

D'autres approches traitent les informations contextuelles indifféremment des informations propres du système. Cette vision semble confuse parce que les informations contextuelles ne peuvent pas être des propriétés propres au système à cause du caractère de variation qui les fait distinguer de toutes les autres propriétés (fonctionnelles, organisationnelles ou techniques). Les informations du contexte d'utilisation sont complètement différentes de celles qui décrivent les fonctionnalités du système ou celles qui décrivent les spécifications techniques de la plateforme à utiliser. Même la modélisation du contexte sera une tache très difficile à accomplir. Par contre cela parait moins compliqué si nous traitions les informations contextuelles indépendamment des autres informations du système.

Dans ce sens, quelques travaux ont évoqué l'utilité de faire la distinction entre les préoccupations contextuelles et les préoccupations propres du système lors du développement d'applications sensibles au contexte et se sont limités à indiquer seulement l'importance du principe; mais le problème est que ces travaux ne proposent pas de méthodes, d'outils ou de réelles solutions sur la manière de réaliser cette séparation.

Parmi les langages de modélisation disponibles, il y a UML. Ce langage est devenu un standard de référence dans le domaine du génie logiciel grâce à ses multiples opportunités quant à la modélisation des systèmes complexes. Etant un langage plus général destiné à être utilisé dans tous les domaines, UML ne propose pas, de façon claire et explicite, des concepts ou des notations destinées spécialement pour le domaine de la sensibilité au contexte. Ce domaine très particulier, caractérisé par la variation et les changements de la situation courante de l'utilisateur, présente un réel besoin à avoir des outils de modélisation des informations spécifiques du contexte d'utilisation.

Par rapport à ce constat, il parait évident qu'il existe un besoin énorme, pour les systèmes d'information ubiquitaires, de proposer des solutions d'adaptation et de personnalisation capables de fournir l'information pertinente parmi plusieurs autres. L'adaptation contextuelle et la personnalisation sont deux processus qui se basent essentiellement sur les informations du contexte d'utilisation ; mais celles-ci sont toujours dissoutes dans des plages d'autres d'informations décrivant un système comme les informations fonctionnelles, organisationnelles et techniques. D'où, la nécessité de trouver une méthode permettant d'isoler les informations liées au contexte par rapport à l'ensemble des propriétés et contraintes du système. Cela permettra d'étudier, d'analyser et de concevoir ces informations spécifiques dans des branches (ou modules) indépendants conduisant, ainsi, à une représentation adéquate et convenable du contexte d'utilisation.

Ces solutions doivent prendre en considération, de façon claire et explicite, tous les besoins de l'utilisateur ainsi que les contraintes du contexte d'utilisation qui l'entourent.

Dans cette étude, nous cherchons à satisfaire les souhaits et les préférences des utilisateurs quelque soient les conditions environnantes de la situation courante de l'exécution d'une application. Pour cela, nous avons essayé de trouver un moyen pour adapter contextuellement les applications de façon qu'elles répondent aux exigences de l'utilisateur. D'autre part, notre problématique est d'apporter une valeur ajoutée aux notations UML pour qu'elles soient plus adéquates et appropriées dans la modélisation des contraintes contextuelles spécifiques.

iv. Objectifs

L'objectif de notre étude est d'assurer l'adaptation et la personnalisation des systèmes d'information ubiquitaires par rapport au contexte d'utilisation. Cette adaptation permettra de fournir des informations pertinentes aux utilisateurs selon leurs préférences souhaitées. Notre but est double : adapter contextuellement les applications et personnaliser le langage UML. Le premier objectif consiste à rendre possible le développement d'applications sensibles au contexte qui soient capables de prendre en charge les variations continues du contexte d'utilisation et qui soient plus adaptées aux différentes situations d'exécution. Dans ce cas, nous insistons sur l'utilité de faire la séparation entre les informations contextuelles et les informations propres du système. Pour cela, nous visons à proposer une solution concrète pour réaliser cette séparation avant toute modélisation du contexte. Ensuite, nous devons intégrer le modèle contextuel obtenu dans une approche, par exemple la MDA, destinée au développement d'applications. Notre deuxième objectif est d'augmenter la sémantique de quelques concepts UML de façon qu'ils soient capables de prendre en charge les contraintes spécifiques du contexte d'utilisation et, par conséquent, de modéliser ces contraintes avec des

notations et des formes plus appropriées. Cette spécialisation du langage UML permettra d'atteindre l'objectif de personnalisation des informations retournées à l'utilisateur.

v. Contributions

Par ce modeste travail, nous espérons contribuer à la satisfaction des nouvelles exigences des systèmes d'informations ubiquitaires. Notre contribution majeure comporte trois parties :

a. La séparation des préoccupations contextuelles : dans le premier volet de notre contribution, nous proposons un nouveau type de processus appelé : 3 Track Unified Process (3TUP), et ce, par analogie avec le processus existant : 2 Track Unified Process (2TUP). Cette proposition consiste à séparer les contraintes spécifiques liées au contexte qui sont dissoutes dans l'ensemble des caractéristiques d'un système et de les étudier de façon indépendante. Le principe de séparation se résume par la définition de trois types de contraintes qui peuvent influencer un système : métiers, contextuelles et techniques. Les contraintes métiers représentent les fonctionnalités et les informations propres du système. Les contraintes contextuelles sont celles qui sont liées au contexte d'utilisation incluant l'utilisateur et l'environnement de la situation courante. Les contraintes techniques décrivent les spécifications de la plateforme à utiliser. L'étude séparée des contraintes contextuelles conduit, nécessairement, à la modélisation adéquate du contexte d'utilisation et offre, ainsi, de grandes opportunités pour les processus d'adaptation contextuelle et de personnalisation des systèmes d'information ubiquitaires.

b. La personnalisation du langage UML : le deuxième volet de notre contribution comporte la proposition d'un profil UML spécifique pour le domaine de la sensibilité au contexte (Context-Awareness Domain). Ce profil UML est composé d'un ensemble de nouvelles notations UML (sous formes d'extensions) définies pour être utilisées dans la modélisation du contexte d'utilisation. Les notations proposées, décrites par des stéréotypes, des contraintes et des étiquettes, sont obtenues par extension des notations UML existantes. Un stéréotype permet d'augmenter la sémantique d'un élément existant du méta-modèle UML afin qu'il soit plus adéquat dans la modélisation de certaines contraintes particulières. Les étiquettes définissent et spécifient les attributs d'un stéréotype crée. Les contraintes fixent les conditions et les limites d'utilisation des nouveaux stéréotypes. Le profil UML, ainsi construit, regroupe des notations étendues relatives à quatre diagrammes UML (classes, cas d'utilisation, séquences et activités) pouvant décrire et couvrir les principales vues du système à étudier (vue fonctionnelle, vue statique et vue dynamique). L'implémentation et l'expérimentation de ce profil UML ont été réalisées sous la plateforme StarUML.

c. L'adaptation contextuelle des applications : le troisième volet de notre contribution propose une nouvelle vision de l'approche MDA qui permet de développer des applications adaptées au contexte d'utilisation. Cette proposition se base essentiellement sur le principe de la séparation des préoccupations contextuelles introduit au début de notre contribution. En utilisant ce principe, toutes les propriétés d'un système sont classées en trois types de contraintes : métiers, contextuelles et techniques. Dans ce cas, l'approche MDA se voit confrontée à faire une étude à trois

branches parallèles (au lieu de deux comme définie initialement). Chacune de ces trois branches se charge d'étudier un seul type de contraintes en construisant un modèle précis. Ainsi, la branche des contraintes métiers donne naissance au modèle PIM (Platform Independent Model), la branche des contraintes techniques permet de construire le modèle PDM (Platform Description Model) et la branche des contraintes contextuelles nous donne un nouveau modèle appelé CM (Contextual Model). A ce stade de la MDA classique, nous procèdons à une transformation du modèle métier PIM vers un modèle spécifique PSM (Platform Specific Model) suivant les spécifications techniques de la plateforme contenues dans le modèle PDM. Mais pour notre version proposée, nous devons d'abord fusionner les deux modèles (PIM et CM) pour garantir la prise en compte du contexte d'utilisation ; et ensuite, nous transformons le modèle fusionné MM (Merged Model) vers un modèle PSM suivant les spécifications de la plateforme choisie (PDM). L'importance de l'approche proposée réside, d'une part, dans la meilleure prise en charge du contexte d'utilisation lors des étapes de développement d'applications sensibles au contexte, et ce, grâce à la représentation claire des éléments de ce contexte dans un modèle indépendant, et d'autre part, dans la préservation de tous les avantages qui ont fait de la MDA une référence dans le domaine du génie logiciel comme l'utilisation de modèles pour formaliser et gérer l'information, la séparation des différentes propriétés d'un système, la portabilité, l'interopérabilité et la réutilisation.

vi. Organisation du document

Ce manuscrit comporte sept chapitres répartis en trois parties. La première partie, composée de trois chapitres, introduit les principaux concepts utilisés dans ce manuscrit et expose un état de l'art du domaine de recherche avec une étude comparative sur les travaux voisins menés antérieurement. La seconde partie présente les détails des contributions apportées à travers cette thèse. Tandis que la troisième partie expose l'expérimentation d'un exemple illustratif du profil UML proposé sous forme d'un cas d'étude.

Première partie

Dans la première partie, nous exposons un état de l'art comportant des explications théoriques nécessaires à la compréhension de la suite du travail ainsi qu'une description généralisée des notions relatives à notre étude. Le chapitre 1 présente une vue globale sur les systèmes d'information ubiquitaires en rappelant les définitions et les caractéristiques de ces systèmes. Le chapitre 2 expose les données principales nécessaires à la compréhension de la notion du contexte d'utilisation et met l'accent sur l'utilité et l'importance de cette notion pour l'adaptation des applications et pour la personnalisation des informations.

Nous clôturons cette partie par le chapitre 3 dans lequel nous présentons une étude comparative qui synthétise et résume les principaux travaux antérieurs proches de notre domaine d'étude. Ce chapitre montre la réalisation d'une double comparaison faite sur les travaux voisins, d'un côté, pour les évaluer les uns par rapport aux autres et, d'autre côté, pour les comparer avec notre étude afin de mieux se situer et de mieux tracer les objectifs à atteindre.

Deuxième partie

La deuxième partie, quant à elle, comporte trois chapitres décrivant nos contributions via cette thèse. Le chapitre 4 aborde la première contribution (séparation des préoccupations contextuelles) en introduisant le nouveau processus 3TUP. Celui-ci met en évidence l'importance d'isoler les contraintes contextuelles par rapport aux autres types de contraintes (métiers et technologiques) lors du processus de développement d'une application. Cette séparation jouera un rôle très important pour l'augmentation des degrés d'interopérabilité et de réutilisation entre les différents systèmes. La seconde contribution (personnalisation du langage UML) est présentée dans le chapitre 5. Cette contribution permet de spécialiser les notations du langage UML pour être utilisé adéquatement dans le domaine de la sensibilité au contexte. Toutes les notations UML spécialisées seront regroupées dans un profil UML qui représentera un outil facile à utiliser pour le développement d'applications sensibles au contexte. La dernière contribution (adaptation contextuelle des applications) de notre étude propose une nouvelle vision de l'approche MDA capable de prendre en charge le contexte d'utilisation sous forme d'un modèle contextuel inséré (par opération de fusion de modèles) dans son processus. Les détails de cette dernière proposition sont exposés dans le chapitre 6.

Troisième partie

La troisième partie, comportant le chapitre 7, expose une expérimentation d'un exemple illustratif pour démontrer la faisabilité du profil UML proposé. Ainsi, toutes les étapes d'implémentation et de mise en œuvre de ce profil sont réalisées sous la plateforme « StarUML ».

Conclusion générale et perspectives

Enfin, ce document s'achève par un bilan concluant le travail réalisé et une présentation de quelques perspectives pour les futures recherches.

PARTIE I

ETAT DE L'ART

CHAPITRE 1

SYSTÈMES D'INFORMATION UBIQUITAIRES

Introduction

Le système d'information représente le noyau de toute organisation qui peut lui garantir plusieurs qualités comme la rapidité, la fiabilité et la pertinence. Il se rapporte à toutes les données et à tous les modes de traitement des informations utilisées dans cette organisation et il ne se limite pas à ce qui est informatisé. En effet, c'est un élément primordial du pilotage d'une telle organisation et de ses activités pour aider à la prise de décision. Pour faire face à la concurrence grandissante, les entreprises s'appuient sur la qualité et l'efficacité de leurs systèmes d'information pour évoluer et pour satisfaire les besoins de ses clients de plus en plus exigeants.

Aujourd'hui, le grand défi des systèmes d'information est de prendre en charge la mobilité des utilisateurs et l'hétérogénéité des sources d'information ainsi que la grande diversité des dispositifs utilisés.

Le développement rapide de l'informatique mobile (téléphone mobile, assistants personnels, ordinateur portables…) permet d'envisager la mise en œuvre de systèmes d'information souples et dynamiques appelés : systèmes d'information ubiquitaires (ou pervasifs). A travers ces systèmes, l'utilisateur dispose à tout moment et à tout endroit d'un accès à l'information dont il a besoin.

L'ubiquité des systèmes d'information requiert la personnalisation et l'adaptation de ces systèmes avec les préférences des utilisateurs et le contexte d'utilisation de chaque service fourni. Ainsi, ces systèmes satisferont les caractéristiques, les besoins et les objectifs des utilisateurs.

Les systèmes d'information ubiquitaires représentent une version améliorée des systèmes d'information classiques qui doivent vérifier et satisfaire les contraintes d'ubiquité comme la distribution, la mobilité, l'hétérogénéité, l'adaptation et la personnalisation.

1.1. Rappels sur les systèmes d'information

1.1.1. Définitions

Un système d'information est un ensemble d'acteurs, de données, d'opérations, de flux, de localisations et de fonctions applicables aux données concourant à l'activité d'une entité (entreprise, administration, etc.) [Salvan, 2013]

Selon le dictionnaire wikipedia, un système d'information est un ensemble organisé de ressources (technologique, personnel, données et procédures) qui permet de collecter, stocker, traiter et diffuser de l'information dans un environnement donné.

Une autre définition très répandue dans la littérature : Un système d'information est l'ensemble des moyens et procédures utilisés en vue de restituer aux utilisateurs une information directement utilisable au bon moment.

Pour récapituler les autres définitions existantes, nous retenons qu'en bref, un système d'information est composé de moyens humains et techniques nécessaires au stockage et au traitement de l'information dans une organisation.

1.1.2. Caractéristiques et composants

Un système d'information de l'entreprise peut présenter plusieurs caractéristiques comme [Laassiri, 2013] :

- Aide à la décision : l'administration de toute entreprise nécessite la prise des bonnes décisions aux bons moments dans le but d'agir rapidement et efficacement et assurer, ainsi, son bon fonctionnement.
- Outil de communication : un bon système d'information doit être capable d'assurer le rôle d'intermédiaire en termes d'échanges et d'acheminement des informations entre les différents acteurs d'une entreprise.
- Fiabilité : pour être fiable, toute information doit être correcte, précise et continuellement mise à jours.
- Disponibilité : la disponibilité de l'information consiste surtout à minimiser le temps d'accès, à préciser l'endroit de mémorisation et à bien choisir le mode de mémorisation.

Concernant la composition d'un système d'information, il existe généralement trois grandes familles de composants [http://www.numeraladvance.com/Systeme_d_Information] :

- La matière informationnelle
- La structure qui organise cette matière
- Les traitements qui transforment la matière

1) La matière informationnelle comprend :
   - La donnée : c'est l'élément de base de la matière informationnelle. Elle se présente souvent comme une donnée chiffrée ou un ensemble de données non traitées.

- L'information : l'information peut être une donnée synthétique ou un résultat obtenu après la réalisation d'un traitement par un système informatique.
- La connaissance : La connaissance est une formulation particulière de l'information. Nous parlons de connaissance lorsque l'information correspondante implique une expérience prédéterminée.

2) La structure permet de gérer les grandes masses d'informations tout en offrant la possibilité d'accéder facilement à ces informations et de les exploiter. La structure comprend les principales méthodes ou techniques d'organisation comme :
    - L'arborescence
    - Le classement
    - Le tableau
    - La matrice
    - Le graphe
    - Le texte

3) Les traitements consistent à réaliser les transformations des données en information ou la transformation d'une matière informationnelle en une autre. Comme exemple de traitements, il y a :
    - Le tri
    - La recherche
    - La sélection (alphabétique, alphanumérique, numérique...)
    - La transformation géométrique
    - La transformation algébrique
    - La projection
    - La régression
    - La recherche d'un minimum

1.1.3. Fonctions et objectifs

Un système d'information a pour mission de récolter, stocker, traiter, mettre en forme et diffuser l'ensemble des informations qui sont nécessaires pour atteindre les objectifs opérationnels et stratégiques d'une organisation.
D'où, les principales fonctions d'un système d'information sont :

- Collecte de l'information
- Saisie de l'information
- Stockage et mémorisation de l'information
- Traitement de l'information
- Restitution de l'information
- Transmission et diffusion de l'information

Toute organisation peut être vue comme étant une entité globale composée de trois systèmes qui s'interagissent entre eux : système de décision, systèmes d'information et système opérant. Le système d'information assure le lien et le couplage entre le système opérant et le

système de pilotage en prenant en compte les contraintes de l'environnement extérieur. Le système d'information fournit au système opérationnel les informations détaillées nécessaires à l'accomplissement de ses missions. Les flux entrants et sortants sont constitués exclusivement d'informations.

Le rôle principal d'un système d'information est d'atteindre les objectifs suivants:

- Assurer l'échange d'information entre :
  - Le système de décision et le système opérant
  - L'organisation et l'environnement extérieur
- Fournir au système de décision des informations concernant :
  - l'état de fonctionnement du système opérant afin de prendre les décisions nécessaires
  - l'environnement extérieur pour prévoir les décisions d'adaptation
  - L'état global du fonctionnement de l'organisation
- Fournir au système opérant les informations nécessaires à son fonctionnement à partir des ordres et des décisions reçues du système de décision.

1.1.4. Défis des systèmes d'information modernes

La conception des systèmes d'information modernes s'articule autour de quelques points sensibles :

- La flexibilité des architectures logicielles
- La distribution massive des composants
- La forte hétérogénéité des sources d'information
- La forte hétérogénéité des ressources utilisées
- L'utilisation massive des données
- Mobilité, disponibilité et sécurité.

1.2. Notions sur l'informatique ubiquitaire

1.2.1. Définitions

Par définition, tirée du dictionnaire Larousse, le mot « Ubiquité » désigne le fait d'être présent partout à la fois ou en plusieurs lieux en même temps.

Selon le dictionnaire wikipédia, le terme « ubiquitaire » désigne un environnement dans lequel les ordinateurs et réseaux sont enfouis, intégrés et omniprésents dans le monde réel. L'utilisateur a accès à un ensemble de services au travers d'interfaces distribuées se voulant intelligentes, dont il est entouré.

La notion d'informatique ubiquitaire (Ang.: ubiquitous computing ou ubicomp), est introduite pour la première fois par Mark Weiser en 1991 [Weiser, 1991]. C'est un paradigme assez récent qui désigne l'informatique omniprésente et qui permet à un utilisateur d'accéder aux données où qu'il soit et quand il le désire ; c'est-à-dire que le déplacement dans l'espace et la variation du temps ne soient pas un obstacle envers l'accès aux informations même si celles-ci présentent des caractéristiques hétérogènes et proviennent, parfois, de sources d'information

distantes et distribuées. L'ubiquité fait appel aux réseaux ambiants (ubiquitous networks) qui se caractérisent par des entités mobiles (terminaux, routeurs, PDA, téléphones cellulaires,…etc.) communicantes et parfois de très petite taille. L'utilisation de ces réseaux sera contrainte par les problèmes de mobilité, de sécurité et de sûreté (confidentialité des données, fiabilité des applications), de continuité de services (tolérance aux pannes) et de la qualité de ces service.

L'informatique ubiquitaire offre plusieurs opportunités quant à l'accès aux informations comme l'utilisation aléatoire, en temps et en espace, des différents dispositifs mobiles par plusieurs types d'utilisateurs incluant les personnes et les systèmes distants. A cause de la mobilité des utilisateurs, non seulement le temps et le lieu sont en constante variation mais aussi tous les objets (y compris les personnes) entourant l'utilisateur ainsi que les facteurs environnementaux. La modification ou le changement de l'état d'un élément du contexte d'utilisation entraîne une transition de ce contexte vers un nouvel état de la situation courante de l'utilisateur. Donc, l'utilisateur doit faire face à chaque nouvelle situation avec laquelle il doit s'adapter.

Un environnement ubiquitaire englobe tous les éléments (naturels ou artificiels) qui entourent un objet et qui peuvent interagir, communiquer et coopérer avec cet objet. Ces éléments facilitent l'accès aux services offerts partout et à tout moment tout en mettant à disposition les informations requises.

1.2.2. Tendances technologiques de l'informatique ubiquitaire

Parmi tous les domaines émergents de l'informatique, l'environnement ubiquitaire se distingue par de nombreux tendances technologiques et développements [Kouadri, 2008] : les réseaux, les capteurs, les matériels d'accès, et les intergiciels (middleware).

1.2.2.1. Les Réseaux

Le succès de l'informatique ubiquitaire est lié étroitement à la disponibilité des différentes technologies de réseaux. En effet, la prise en compte de la grande mobilité des utilisateurs nécessite l'existence de canaux de communication efficaces pour permettre l'acheminement instantané des informations distantes et variables en temps et en espace. A chaque niveau de mobilité doit correspondre un type de réseaux précis. Exemples de réseaux sans fil : WPAN, WMAN.

1.2.2.2. Les Capteurs

Dans un environnement ubiquitaire, toutes les informations requises par l'utilisateur doivent être collectées, extraites et mises à jours d'une façon continue. Cette opération est appelée : capture d'information, et elle est assurée par des capteurs matériels ou logiciels implantés sur les systèmes. Le rôle de ces capteurs est, d'une part, de transformer les données réelles ou naturelles en informations exploitables par l'application, et d'autre part, de choisir les actions correspondantes aux informations capturées.

1.2.2.3. Les matériels d'accès

Un matériel d'accès représente le lien d'interaction entre l'utilisateur et le système. Pour les systèmes ubiquitaires, les matériels d'accès doivent respecter les caractéristiques de l'ubiquité telles que la mobilité des utilisateurs, l'hétérogénéité des informations et la distribution des sources d'information. Pour ce fait, il existe plusieurs types de matériels d'accès qui vérifient ces conditions comme les téléphones mobiles, les PDA, les tablettes, PC, etc.

1.2.2.4. Les intergiciels (middleware)

Les intergiciels permettent la gestion et l'harmonie des trois technologies précédentes : technologie des réseaux, technologie des capteurs et technologie des matériels d'accès. Ceci est assuré par l'élaboration de plusieurs procédures et traitements qui définissent le mode de fonctionnement des différentes parties du système.

1.2.3. Particularités de l'informatique ubiquitaire

Le domaine de l'informatique ubiquitaire est très spécifique du fait qu'il se présente comme un ensemble de plusieurs propriétés particulières et sensibles comme la mobilité, la miniaturisation et l'interdépendance.

1.2.3.1. La mobilité

La mobilité représente tout ce qui peut changer de place ou de position. Cette particularité de l'informatique ubiquitaire désigne les changements subis par l'environnement lors de l'exécution d'une situation. L'informatique ubiquitaire doit présenter la capacité et le pouvoir de l'utilisateur (ou tout autre objet) à communiquer avec les autres objets qui l'entourent tout en prenant en compte les changements de l'environnement. Ceci nécessite la présence de moyens d'adaptation qui peuvent assurer l'accès facile aux différents services, données et traitements dont a besoin l'utilisateur mobile.

1.2.3.2. La miniaturisation

Afin d'assurer une communication efficace et en temps réelle entre les utilisateurs mobiles et les objets environnants, l'informatique ubiquitaire doit offrir du matériel plus petit et plus légers capable d'embarquer les données et les traitements requis par l'utilisateur de plus en plus mobile.

1.2.3.3. L'interdépendance

L'informatique ubiquitaire est caractérisée, aussi, par l'existence d'une grande dépendance, d'une part, entre l'utilisateur et l'environnement et d'autre part, entre les objets qui constituent cet environnement. Cette dépendance doit être garantie par une actualisation instantanée des données et par un choix adéquat du service requis.

1.2.4. Éléments de l'informatique ubiquitaire

Lorsqu'il s'agit de développer une application dans un environnement ubiquitaire, il faut s'attendre à être confronté face à différentes situations d'exécution. Pour s'adapter à ces situations, l'interactivité des différentes machines nécessite la prise en considération de trois éléments fondamentaux : le matériel, l'application et l'environnement [Banavar, 2002].

1.2.4.1. Le matériel

En plus qu'il soit un lieu de stockage et de traitement des données, le matériel représente un portail pour s'introduire dans une application ou dans un espace de données.

1.2.4.2. L'application

Une application n'est pas seulement un logiciel destiné à exploiter les capacités d'un matériel, mais, c'est aussi un moyen efficace qui permet à l'utilisateur d'améliorer les performances d'une tache.

1.2.4.3. L'environnement

Un environnement d'exécution ne se réduit pas à un espace virtuel qui existe pour stocker et exécuter un logiciel, mais englobe, aussi, toutes les informations relatives aux objets environnants d'un utilisateur.

Mais ces éléments semblent insuffisants pour garantir un bon fonctionnement (exécution efficace) de l'application dans toutes les situations possibles et assurer son adaptation par rapport aux éventuels changements qui peuvent survenir. D'où l'utilité d'ajouter un élément « l'utilisateur » que nous jugeons indispensable et crucial pour atteindre un haut degré d'adaptation surtout lorsqu'on opère dans des environnements ubiquitaires. L'utilisateur est la source principale de la majorité des changements subis par une situation d'exécution. Ainsi, c'est lui qui décide des déplacements à faire, du matériel à utiliser, des ressources à mettre à disposition, etc. D'où, l'utilisateur est l'élément central autour duquel s'articulent tous les autres éléments à prendre en considération lors du développement d'une application évoluant dans un environnement ubiquitaire.

1.2.5. Cycle de vie de l'informatique ubiquitaire

Banavar propose un cycle de vie formé de trois étapes [Banavar, 2000]:

- Temps de conception (design-time)
  C'est le temps nécessaire à un développeur pour créer, maintenir ou optimiser une application.
- Temps de chargement (load-time)
  C'est un temps durant lequel le système compose, adapte et charge les composants d'une application.
- Temps d'exécution (run-time)
  C'est le temps pendant lequel l'utilisateur exploite l'application et utilise ses fonctionnalités.

1.2.6. Niveaux de l'informatique ubiquitaire

L'informatique ubiquitaire peut être considérée comme étant un système à trois niveaux où chaque niveau présente ses propres rôles et fonctions [Tiako, 2009].

Niveau 1 : gestion des taches

- Contrôler et planifier les taches de l'utilisateur

- Contrôler le contexte
- Contrôler les préférences
- Gérer et décomposer les taches complexes

Niveau 2 : gestion de l'environnement

- Contrôler les ressources et les capacités de l'environnement
- Planifier les services requis

Niveau 3 : l'environnement

- Choisir et contrôler la ressource appropriée
- Gérer l'efficacité des ressources

1.2.7. Clefs de recherche dans le domaine de l'informatique ubiquitaire

Pour mener de bonnes études dans le domaine de l'informatique ubiquitaire, il faut s'appuyer sur trois éléments de base qui représentent des clefs de recherche pour ce domaine [Banavar, 2000].

- Espace intelligent (smart spaces)

C'est une zone bien définie, ouverte ou enclose, qui renferme plusieurs systèmes incorporés (ordinateurs, capteurs, interfaces utilisateur et les infrastructures de services).

- Transparence (invisibility)

Les utilisateurs ne sont pas, nécessairement, préoccupés par leur interaction avec les technologies de l'informatique ubiquitaire.

- Ascension située (localized scalability)

S'occupe de la gestion des échanges d'information entre les utilisateurs et leur entourage (objets environnants).

1.3. Systèmes d'information ubiquitaires

Aujourd'hui, l'accès à distance aux informations et aux services est devenu très important. Ceci exige que les procédés et les outils qui gèrent l'information soient de nature ubiquitaire. Les systèmes d'information ubiquitaires se manifestent considérablement dans les domaines de développement d'applications, de recherche d'informations, d'organisation, de gestion des entreprises, etc. Ces systèmes doivent être fortement disponibles pour tous les utilisateurs, n'importe où, n'importe quand et sous n'importe quelle circonstance. Par conséquent, les concepteurs des systèmes d'information ubiquitaires doivent prendre en compte toutes les contraintes relatives à l'ubiquité comme l'hétérogénéité, la mobilité, la sécurité, etc.

L'ubiquité des systèmes d'information peut être atteinte par la personnalisation et l'adaptation de ces systèmes avec les préférences des utilisateurs et le contexte d'utilisation de chaque service. Ainsi, ces systèmes satisferont tous les besoins et les souhaits des utilisateurs toujours exigeants.

1.3.1. Principes des systèmes d'information ubiquitaires

L'informatique ubiquitaire envisage quatre paradigmes fondamentaux : la décentralisation, la diversification, la connectivité et la simplicité d'emploi [Hansmann, 2003].

1.3.1.1. Décentralisation

Avec l'émergence colossale des petits appareils de communication, l'informatique ubiquitaire s'oriente vers la distribution des traitements et des calculs en les faisant embarquer sur ces appareils. Cette décentralisation doit prendre en compte des paramètres spécifiques comme l'architecture des systèmes distribués, la synchronisation de l'information et la gestion des applications

1.3.1.2. Diversification

Ce paradigme est étroitement lié aux fonctionnalités et aux performances des systèmes de l'ordinateur. Ces fonctionnalités sont proposées par une grande variété d'appareils caractérisés par un usage précis et un public précis. L'informatique ubiquitaire est contrainte par l'incidence de la diversité des plateformes matérielles et logicielles sur les applications et sur les échanges d'information ainsi que par une grande gamme de machines ayant des caractéristiques diverses comme les capacités de stockage, les types de processeurs, les systèmes d'exploitation, etc. Pour remédier à cette diversité et résoudre les problèmes d'interopérabilité y afférents, il faut respecter les standards industriels disponibles.

1.3.1.3. Connectivité

Cette connectivité offre la possibilité d'utiliser n'importe quelle application sur n'importe quel appareil en utilisant n'importe quel réseau. Les appareils de l'informatique ubiquitaire sont fortement connectés grâce à l'utilisation des nouvelles technologies de connexion comme le Wifi, L'infrarouge, le Bluetooth, etc.

1.3.1.4. Simplicité d'emploi

Les appareils de l'informatique ubiquitaire sont des outils très spécialisés (pour certain usage et pour certain public) et ne sont pas conçus pour une utilisation générale (ou universelle). Pour garantir la facilité d'utilisation, il faut réduire le temps d'accès aux informations, optimiser la gestion des informations, rendre les informations disponibles et commodes.

1.3.2. Caractéristiques des systèmes d'information ubiquitaires

Les systèmes d'information ubiquitaires représentent une extension des systèmes d'information classiques et qui doivent vérifier les contraintes d'ubiquité comme la distribution, la mobilité, l'hétérogénéité, l'adaptation et la personnalisation.

1.3.2.1. Distribution

Lors de l'utilisation très vaste du Web et des codes mobiles, une application a besoin d'exécuter des codes téléchargés à travers les clients et les serveurs. Le téléchargement se fait à partir de sources diverses et probablement distantes. L'utilisateur de l'application devient nomade, c'est-à-dire qu'il aura la possibilité d'accéder aux données de cette application à n'importe quel moment et de n'importe quel endroit. Ceci peut être assuré par l'intermédiaire de différents moyens d'accès présentés sous formes de terminaux nomades (un téléphone GSM associé à un PDA, une télévision interactive ou WebTV). La conception d'application doit, donc, prendre en considération de nouveaux paramètres (sécurité, disponibilité et mobilité) qui peuvent influencer la distribution des systèmes d'information.

La sécurité : Les applications utilisées par les systèmes d'information font appel à de nombreux composants fournis et administrés par de multiples organisations dont le degré de confiance n'est pas toujours à la hauteur. Il convient donc d'établir un espace de confiance afin d'avertir l'application [Donsez, 1999]. Cet espace de confiance est créé pendant l'étape d'initialisation de l'application et est maintenu jusqu'à l'étape de terminaison. La grande sûreté et l'assurance absolue de l'espace de confiance conduisent logiquement à une meilleure sécurité des systèmes d'informations distribués, et ce, indépendamment des conditions de fonctionnement et des contraintes d'utilisation. Les informations peuvent être sécurisées par la mise en place de mécanismes de contrôle d'accès qui ont la possibilité de restreindre l'utilisation des systèmes d'information seulement aux utilisateurs autorisés (vérification de l'authenticité). Aussi, il faut toujours garder la trace des informations échangées et de leurs sources, ceci permettra un meilleur contrôle (en termes d'intégrité) des flux d'informations.

La disponibilité : L'exécution d'une application nécessite la disponibilité de quelques composants seulement et non pas de tous ceux possibles. Le concepteur peut définir quel est le type de fonctionnement d'une application quand la configuration ne comporte que quelques composants disponibles, et quelles sont alors les conditions d'utilisation [Donsez, 1999].

La mobilité : La notion de mobilité désigne surtout la capacité de se déplacer continuellement dans l'espace. Dans les systèmes d'information ubiquitaires, chaque utilisateur est considéré comme "nomade", donc, il peut changer sa localisation (endroit) selon ses besoins et ses désirs (préférences). La prise en charge de la mobilité consiste à assurer, à tout moment, la capacité d'identification de la nouvelle localisation d'un utilisateur après ses mouvements ou ses déplacements, et offrir, ainsi, les chemins d'accès appropriés de cette nouvelle localisation. La variation de la localisation entraîne un changement du contexte d'utilisation, par conséquent, les systèmes d'information doivent être capables de rétablir les paramètres du contexte d'utilisation (liés à l'ancienne localisation) et de les faire adapter avec la nouvelle localisation (dès que l'utilisateur reprenne la main sur l'application).

1.3.2.2. Mobilité

La gestion de la mobilité nécessite l'identification et la définition de quelques besoins en termes de bases de données de localisation (localisation de téléphones cellulaires, gestion de flottes de véhicules, etc.). Pour définir ces besoins, il faut identifier les dimensions relatives au problème de la mobilité dans les accès aux données [Canals, 2002] :

a. Gestion de données spatio-temporelles : l'objectif des bases de localisation est de déterminer clairement l'endroit et l'emplacement d'un utilisateur ou du moins l'approximer avec une faible marge d'erreur. Cette approximation permet une meilleure modélisation des données, une expression claire des requêtes et une évaluation efficace de ces mêmes requêtes.
b. Modèles d'accès à l'information : la mobilité nécessite d'envisager de nouveaux modèles d'accès aux données plus souples et plus efficaces que les modèles existants (par exemple, le modèle client-serveur). En effet, un modèle de souscription permet à un utilisateur d'être notifié automatiquement lorsqu'un événement satisfaisant son

abonnement est publié. Un modèle de diffusion permet à un serveur de diffuser de façon répétitive une information qui peut être captée par les utilisateurs (clients) intéressés. Autres modèles basés sur une diffusion épidémique de l'information peuvent être envisagés pour permettre le passage à l'échelle [Canals, 2002].

c. Synchronisation et cohérence transactionnelle : la prise en charge de la mobilité nécessite souvent un travail en mode déconnecté. Celui-ci provoque généralement des problèmes d'hétérogénéité et de dissemblance entre la version originale (souvent hébergée sur le réseau fixe) et ses différentes copies (hébergées sur des dispositifs mobiles). Lors des reconnexions, des protocoles complexes de synchronisation/réconciliation doivent être appliqués pour rétablir la convergence des différentes copies tout en respectant les règles de causalité et préservation de l'intention de l'utilisateur. La mobilité nécessite aussi la mise en œuvre des propriétés transactionnelles ACID (Atomicité, Cohérence, Isolation et Durabilité) parce que certaines applications ont des besoins plus forts de cohérence qui nécessitent le respect de ces propriétés.
d. Gestion des données embarquées : une bonne gestion de la mobilité pousse les concepteurs à concevoir et à produire des composants bases de données destinés à être embarqués sur des dispositifs mobiles ultra-légers. L'architecture de chaque dispositifs se caractérise par des propriétés bien définies (portabilité, autonomie électrique, coût, sécurité, etc.) et doit tenir compte des contraintes matérielles imposées (taille, débit, capacité mémoire, etc.).
e. Confidentialité des données : l'objectif majeur des utilisateurs mobiles est d'accéder plus facilement aux données. C'est pourquoi celles-ci sont stockées sur un serveur fixe et gérées, par exemple, par un hébergeur de données sur l'Internet (Database Service Provider). C'est alors que surgit le problème de confidentialité et de sécurité de ces données. Des modèles de sécurité de bout en bout restent à définir afin de résister à toutes formes d'attaques et de menaces provenant d'un intrus ou de l'hébergeur lui-même.
f. Adaptabilité : l'objectif de l'adaptabilité est d'offrir à l'utilisateur, indépendamment de sa localisation et son type de terminal, un service aussi proche que possible de celui fourni dans un contexte fixe. A cet effet, toute information téléchargée sur un dispositif mobile doit dépendre, par exemple, des capacités d'affichage, de stockage et du temps de connexion nécessaire au téléchargement.

1.3.2.3. Hétérogénéité

L'hétérogénéité est la principale propriété qui caractérise les systèmes d'information modernes. En effet, cette caractéristique marque la majorité des éléments intervenant dans un système d'information. Nous distinguons :

- L'hétérogénéité des données,
- L'hétérogénéité des sources d'information,
- L'hétérogénéité des ressources utilisées,
- L'hétérogénéité des interfaces,
- L'hétérogénéité des logiciels.

1.3.2.4. Adaptation et personnalisation

L'évolution des systèmes d'information s'oriente actuellement vers plus d'adaptation et de personnalisation dans un environnement informatique marqué par l'hétérogénéité des sources d'information et par la mobilité incessante des utilisateurs.

Avec l'apparition de l'informatique ubiquitaire, le véritable challenge est d'accéder à une information pertinente dans un environnement caractérisé par un contexte mobile et fortement dynamique. La personnalisation permet d'améliorer la pertinence de l'information retournée en prenant en considération tous les facteurs susceptibles d'influencer l'exécution d'une situation particulière.

Conclusion

Les systèmes d'information ubiquitaires ont fait leur apparition suite à l'émergence colossale des nouvelles technologies utilisées pour accéder et exploiter les informations indépendamment des contraintes qui caractérisent la situation de l'utilisateur. Dans ce premier chapitre, nous avons exposé un état de l'art qui synthétise la description des principales notions relatives aux systèmes d'information. Cette description comporte les définitions, caractéristiques, éléments de composition, principes de fonctionnement ainsi que les objectifs et avantages des notions étudiées. Ces notions étudiées sont telles que : les systèmes d'information, l'informatique ubiquitaire et les systèmes d'information ubiquitaires.

CHAPITRE 2

ADAPTATION CONTEXTUELLE ET PERSONNALISATION

Introduction

Aujourd'hui, l'accès aux informations distantes se fait à travers différents dispositifs tels que les ordinateurs de bureaux, les ordinateurs portables ou même des dispositifs mobiles comme les téléphones portables ou les PDA (Personal Digital Assistant).

Les informations requises par l'utilisateur doivent être adaptées, au plus possible, à son profil d'utilisation et à ses préférences. Ces dernières sont considérées comme des paramètres variables à cause de la diversité des comportements des utilisateurs, de la variation des éléments de l'environnement ainsi que l'évolution des ressources disponibles. Par conséquent, le résultat attendu va être certainement influencé par le changement de ces paramètres ; c'est-à-dire qu'il doit prendre en compte le contexte d'utilisation dans lequel se trouve l'utilisateur.

L'étude du contexte d'utilisation permet de déceler la nature des interactions entre l'utilisateur, les applications et l'environnement. Ces interactions comprendront non seulement les états physiques et logiques, mais aussi, les états émotionnels et les comportements de tous les objets environnants.

Les approches actuelles destinées au développement des applications se trouvent obligées de s'adapter avec les nouvelles contraintes relatives à la mobilité des utilisateurs et à l'hétérogénéité des dispositifs utilisés. Ces paramètres affectent directement l'état de la situation courante de l'utilisateur. En effet, les changements imprévus et continuels subis par une situation peuvent entraver la bonne exécution des taches. Donc, à un instant donné, chaque situation est caractérisée par des éléments bien définis qui peuvent influencer l'exécution des taches de cette situation. Ces éléments constituent le contexte d'utilisation de la situation courante d'un utilisateur.

Le contexte d'utilisation inclut tous les objets, physiques ou morales, qui peuvent agir ou interagir avec l'utilisateur, l'application ou les deux (y compris l'utilisateur et l'application

eux même). Généralement, la notion de contexte d'utilisation comprend des éléments tels que la localisation, l'utilisateur (préférences, comportement,…), le temps, les ressources disponibles, les personnes avoisinantes, etc.
Les applications qui utilisent le contexte d'utilisation s'appellent : applications sensibles au contexte. Ces applications cherchent à s'adapter suivant les changements du contexte d'utilisation et suivant les préférences imposées par l'utilisateur.

2.1. Notions sur le contexte

2.1.1. Définitions

Plusieurs définitions du terme « contexte » ont été introduites dans les différents dictionnaires de la langue française. Parmi ces définitions, nous citons :

- « Ensemble des circonstances dans lesquelles se produit un événement ou se situe une action » [Dictionnaire Larousse].
- « Ensemble des circonstances dans lesquelles s'insère un fait » [Dictionnaire Le petit robert]
- « Ensemble des éléments qui entourent un fait et permettent de le comprendre » [Dictionnaire Hachette]

Dans le domaine de l'informatique ubiquitaire, le mot contexte est synonyme du contexte d'utilisation qui regroupe tous les éléments et les facteurs environnant une situation quelconque.
Ci-après, nous citons quelques définitions qui ont marqué le début de l'ère de l'informatique ubiquitaire et qui, jusqu'à présent, demeurent des références dans la définition du contexte d'utilisation.
Schilit et Theimer [Schilit, 1994b] ont défini le contexte comme étant l'endroit, l'identité des personnes voisins et objets et les changements de ces objets.
Pas loin de la définition précédente, Brown et Chen [Brown, 1997] montrent que le contexte représente l'endroit, l'identité des gens qui entourent l'utilisateur, le temps, la saison, la température,…etc.
Ryan [Ryan, 1997] définit le contexte comme l'endroit de l'utilisateur, l'environnement, l'identité et le temps.
Pascoe [Pascoe, 1998] définit le contexte comme un sous-ensemble d'états physiques et conceptuels relatifs à une entité particulière.
Dey [Dey, 1998], pour sa part, fournit une définition plus large : le contexte représente l'état émotionnel de l'utilisateur, la concentration, l'endroit, la date, le temps, les objets et les personnes présentes dans l'environnement de l'utilisateur.
Dans [Dey, 1999a] Dey donne une autre définition : le contexte symbolise l'état de l'utilisateur et cet état peut être du type physique, social, émotionnel ou informationnel.

Pour résumer ces définitions, Dey, Abowd et Salber [Dey, 2001] donnent une définition plus significative du contexte :
"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves".

Ce qui peut être traduit par :
« Le contexte est toute information utilisable pour caractériser la situation d'une entité. Une entité peut être une personne, un endroit, ou tout objet considéré pertinent pour l'interaction entre l'utilisateur et l'application incluant l'utilisateur et l'application eux-mêmes ».

2.1.2. Composants du contexte

Les principaux éléments qui peuvent constituer le contexte sont :

- L'endroit : caractérise la position et la situation d'un utilisateur ou d'une entité particulière par rapport aux éléments avoisinants et par rapport à l'espace
- L'environnement (ou l'activité) : est l'ensemble de tous les éléments et tous les événements qui peuvent intervenir lors de l'exécution d'une situation
- L'identité : représente la nature (définition) exacte d'une entité ainsi que sa description comportementale.
- Le temps : décrit le moment (instant) du déroulement d'une tache

Ces éléments du contexte seront considérés comme des types primaires (premier niveau), et à partir desquels, nous pouvons dériver d'autres types de second niveau qui seront utiles pour mieux décortiquer le contexte d'une entité particulière. Les entités envisagées peuvent être réparties en trois classes :

- Les places : sont des régions de l'espace géographique
- Les personnes : peuvent être sous forme d'individus ou groupes, unies ou réparties
- Les objets : peuvent être des objets physiques ou des composants logiciels.

Les parties constituantes de l'environnement sont [Schilit, 1994a]:

- Environnement informatique : machines disponibles, processeurs, réseaux, etc.
- Environnement de l'utilisateur : endroit, personnes voisins, situation sociale, etc.
- Environnement physique : niveau de lumière, niveau de son, etc.

2.1.3. Caractéristiques du contexte

Une information contextuelle doit présenter des propriétés de qualité très particulière pour pouvoir être prise en compte d'une manière efficace [Ranganathan, 2005] :

- Confiance : c'est la probabilité pour que le contexte capturé soit correct.
- Précision (exactitude) : représente le pourcentage d'erreur des contextes capturés ou déduits.
- Fraîcheur : c'est l'intervalle de temps qui sépare les différentes interprétations d'un certain type de contexte.
- Résolution (détermination) : c'est un intervalle qui délimite l'information de localisation.

D'autres caractéristiques peuvent être liées au contexte comme celles citées dans [Lombardi, 2014] :

- Le contexte doit être abstrait pour avoir une signification claire

- Les capteurs d'acquisition du contexte peuvent être distribués et hétérogènes
- Le contexte peut avoir plusieurs représentations alternatives
- Le contexte a un caractère dynamique (dans la majorité des cas)
- L'information contextuelle est imparfaite et incertaine

Plusieurs bénéfices sont attendus de la prise en compte du contexte [Privat, 2007] :

- Désambiguïser l'information issue des inputs primaires,
- Fiabiliser un système,
- Augmenter les performances adaptatives d'un système.
- Amélioration du confort d'utilisation
- Rendre l'application plus proche d'un agent humain permettant de lui conférer du « bon sens »
- Prise en compte de la performance des utilisateurs par rapport au système
- Prendre en compte des besoins latents, même implicites et non formulés par l'utilisateur
- Adaptation intelligente à l'utilisateur par apprentissage incrémental, apprentissage de préférences des utilisateurs
- Eviter de demander des entrées d'information inutiles ou redondantes
- Eviter de présenter des informations en sortie inutiles ou redondantes
- Fournir des informations en sortie plus pertinentes, adaptées et personnalisées
- Adapter la présentation des informations aux conditions de consultation (terminaux disponibles)
- S'adapter à l'environnement de l'utilisateur et à l'utilisateur lui-même, à son activité, à sa situation, à son état, etc.
- Resituer le contexte distant et enrichir la conscience de l'utilisateur en situation de communication distante (adaptation non automatique).

Le contexte peut être utilisé selon plusieurs façons possibles [Privat, 2007] :

- Déclenchement ou arrêt automatique du système
- Adaptation automatique instantanée du système
- Apprentissage incrémental incorporé dans le système
- Information annexe fournie en sortie : état du système ou contexte distant
- Contrôle de l'activité externe du système

2.1.4. Cycle de vie du contexte

Le cycle de vie du contexte est composé de cinq étapes fondamentales (Figure 1):

- Capture du contexte : c'est l'acquisition des informations contextuelles nécessaires à l'exécution d'une action. Cette opération est faite à l'aide de capteurs externes (physiques) ou internes (logiciels).
- Stockage du contexte : c'est la mémorisation de toutes les informations brutes capturées pour une éventuelle exploitation immédiate ou future.

- Modélisation du contexte : cette opération permet de représenter les informations brutes sous des formes abstraites (modèles) pour faciliter leur traitement et étude.
- Traitement du contexte : consiste à extraire les informations contextuelles en manipulant les modèles construits à partir des informations brutes.
- Répartition du contexte : permet de dispatcher et de distribuer les informations contextuelles nécessaires à l'exécution de certaines actions.

Figure 1. Cycle de vie du contexte.

2.1.5. Traitement du contexte

Dans la thèse [Boystov, 2011], l'auteur propose une architecture générale du processus de traitement du contexte. Ce processus montre le passage, via des étapes, de l'information brutes jusqu'à la prise en compte de l'information contextuelle. Ces étapes de traitement du contexte sont :

- Capturer les informations (ou préférences de l'utilisateur)
- Fusionner et valider les données hétérogènes
- Comprendre et maitriser le contexte (vue générale)
- Comprendre et maitriser la situation (vue particulière)
- Adapter les commandes pour chaque situation
- Exécuter les commandes (actions)

2.1.6. Domaines du contexte

Les domaines de contexte pouvant être prises en compte s'articulent autour de plusieurs dimensions qui influencent l'exécution d'une situation [Ranganathan, 2005] [Privat, 2007]:

- Contexte physique

- Spatial (localisation, position, orientation, attitude, etc.)
- Temporel (heure, jour, date, semaine, saison, …)
- Environnemental (éclairage, niveau de bruit, température, pression, et plus généralement toutes les variables physiques pertinentes pouvant être acquises à partir de l'environnement)

- Contexte utilisateur

- Profil (rôle, préférences, permissions)
- Capacités (connaissances du domaine, niveau d'expertise)
- Activité (marche, travail, voiture), tache au sens procédure (niveau en dessous d'une activité)
- Présence disponibilité, …

- Contexte du groupe

- Activité du groupe

- Relations et liens sociaux,
- Autres personnes

- Contexte social

- Activité sociale en cours
- Contexte organisationnel, hiérarchique, politique, etc.
- Normes sociales implicitement acceptées et partagées

- Contexte de l'application

- Emails reçus
- Sites Web visités
- Fichiers précédemment ouverts

- Contexte « système »

- Logiciel (middleware et applications), bases de connaissances
- Existence de stratégies et procédures d'action
- Nature et état du réseau utilisé
- Etat matériel (capacités permanentes et actuelles, performances, charge, disponibilité, niveaux de batterie, etc.)

2.1.7. Catégorisation du contexte

La diversité des types (ou catégories) du contexte découle de la grande variation du sens attribué par les chercheurs à la notion de « contexte ».

D'après les définitions précédentes, nous pouvons ressortir quelques types du contexte. Ainsi pour Schilit [Schilit, 1994a], les catégories du contexte tournent autour des idées (Où êtes-vous ?, Avec qui vous êtes ? et Quelles sont les ressources disponibles dans le voisinage ?), mais les auteurs excluent ou ne prennent pas en compte d'autres informations plus importantes comme l'activité ou le temps. En définissant le contexte, Ryan [Ryan, 1997] propose quatre types de contexte qui sont : l'endroit, l'environnement, l'identité et le temps. De leur part, Dey et Abowd [Dey, 1999b], ont repris ces mêmes types de contexte sauf qu'ils ont remplacé l'environnement par un nouveau type qui est l' « activité ». Cette étude de [Dey, 1999b] montre qu'il existe des types primaires et des types secondaires parce qu'en ayant un type de contexte (primaire) qui sera considéré comme index, nous pouvons déduire des indices sur d'autres informations du contexte (secondaire).

Par exemple, en sachant l'identité d'une personne (type primaire), nous sommes capables de connaître facilement d'autres informations (de type secondaire) comme les numéros de téléphones, les adresses, la liste des amis, les autres personnes de l'environnement, etc. Aussi, en ayant l'endroit d'une entité, par exemple une voiture, nous pouvons ressortir d'autres informations pouvant être pertinentes car le contexte de l'entité « voiture » change et diffère lorsque cette voiture est stationnée dans un garage ou bien si elle circule sur autoroute.

Pour résumer l'étude de la catégorisation du contexte, Perera [Perera 2013] a synthétisé les résultats de plusieurs travaux antérieurs et il a collecté un ensemble de facteurs qui influencent le contexte et qui constituent ces principaux types. La liste de ces types comporte :

l'utilisateur, le système, l'environnement, l'historique, l'état social, les réseaux, les objets, les capteurs, l'identité (Qui), l'endroit (Ou), le temps (Quand), l'activité (Quoi), les causes (Pourquoi), capturé, statique, dérivé, résultat d'un profile, opérationnel, conceptuel, objectif, cognitif, externe (physique), interne (logique), observable (bas-niveau), non observable (haut-niveau).

L'étude du contexte nous aide à comprendre l'exécution et le comportement d'un système par rapport aux changements subis par les différents éléments de l'environnement. Parmi les points positifs tirés de cette étude, nous citons [Sonawane, 2012] :

- Le contexte influence et favorise une meilleure perception de l'information
- Le contexte peut gérer un grand volume d'information de notre environnement
- Le contexte propose des solutions qui nous guident à travers les informations de notre environnement
- Le contexte permet de faire la différence entre les informations importantes et pertinentes et celles qui ne le sont pas
- Le contexte nous aide à s'adapter avec le milieu environnant.

2.2. La sensibilité au contexte (context-aware)

2.2.1. Définitions

Après avoir répondu aux questions (qui, quoi, quand et où), et une fois nous avons identifié les types de contexte, nous sommes aptes, maintenant, à aborder la question « comment ? », et ce, pour parvenir à mieux comprendre le mode d'utilisation du contexte dans le monde de l'informatique nomade et plus particulièrement dans la conception des applications de l'informatique ubiquitaire.

Le terme « context-aware » est utilisé pour mettre en évidence l'utilisation du contexte, la sensibilité à ce contexte ainsi que sa prise en compte par les applications dans les différents domaines. Ce terme englobe toutes les connaissances et les informations relatives à l'utilisation du contexte par les développeurs d'applications.

La notion de l'informatique sensible au contexte « context-aware computing », introduite par Schilit et Theimer en 1994 [Schilit, 1994b], désigne tout logiciel qui peut s'adapter avec les changements des utilisateurs et des objets dans un endroit d'utilisation précis et pour un temps donné. Cette définition exige, donc, que les applications soient auto-adaptables avec le contexte.

Toutes les définitions de l'informatique sensible au contexte convergent vers deux catégories : l'utilisation du contexte et l'adaptation au contexte [Dey, 1999b].
Dans le cadre de l'utilisation du contexte, Hull [Hull, 1997] et Pascoe [Pascoe, 1998a] [Pascoe, 1998b] résument le sens de l'informatique sensible au contexte en l'aptitude d'assumer les quatre taches caractéristiques: la détection, la perception, l'interprétation et la réponse vis-à-vis de l'environnement de l'utilisateur. Par contre, Dey [Dey, 1998] réduit cette définition à la relation de l'application avec l'interface homme-machine. Salber [Salber, 1998]

décrit la sensibilité au contexte comme étant la capacité de fournir un maximum de flexibilité d'un service basé sur la perception et la capture du contexte en temps réel. Pour la catégorie de l'adaptation au contexte, nous parlerons plutôt d'applications sensibles au contexte.

2.2.2. Les applications sensibles au contexte

De point de vue adaptation au contexte, les travaux [Schilit, 1994a], [Ward, 1997], [Korteum, 1998], [Brown, 1997], [Davies, 1998], [dey, 1997], [Abowd, 1997] admettent que les applications sensibles au contexte sont des applications qui changent dynamiquement ou bien qui adaptent leur comportement suivant le contexte de l'application et de l'utilisateur.
Aussi, Brown [Brown, 1998] définit les applications utilisant le contexte comme étant des applications qui fournissent automatiquement l'information et/ou décident de l'action à prendre selon le contexte détecté.

En s'inspirant des différentes définitions précédentes, Dey et Abowd [Dey, 1999b] ont fourni une définition très significative et plus récapitulative du context-aware, et que nous allons adopter dans notre étude :
"A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task".

Ce qui peut être traduit par :
« Un système est sensible au contexte s'il utilise ce contexte pour fournir une information pertinente et/ou des services à l'utilisateur, sachant que cette pertinence dépend des taches de l'utilisateur ».

Cette définition est plus générale que celles citées précédemment. En effet, si nous considérons une application où le comportement est invariable mais qui utilise les éléments du contexte seulement en affichage ou en consultation, et nous voulons connaître si cette application est context-aware ou bien non, alors dans ce cas le résultat ne sera pas le même en appliquant les différentes définitions. C'est-à-dire qu'une application peut ne pas être context-aware pour les définitions prétendues réduites (ou étroites) et qu'elle peut l'être pour d'autres définitions qui soient plus générales et plus vastes (par exemple la définition de Dey et Abowd [Dey, 1999b]).

2.2.3. Caractéristiques des applications sensibles au contexte

Chaque application sensible au contexte doit vérifier les caractéristiques suivantes [Perera, 2013]:

- Présentation : toute information ou service est présentée à l'utilisateur sous des formes précises ; et c'est le contexte qui est responsable de choisir et de décider sur le type approprié de cette présentation.
- Exécution : plusieurs services incluent l'exécution automatique de certaines actions. La prise en charge automatique de ces actions doit être basée sur le contexte.
- Marquage (annotation) : puisque le contexte est acquis à partir de différents capteurs, alors il faut créer un lien entre chaque type de contexte et le capteur correspondant en

les marquant par une annotation. Ceci permettra de mieux comprendre la nature du contexte et de garder la trace des sources d'origine de chaque contexte.

Pour Sonawane [Sonawane, 2012], la sensibilité au contexte possède les propriétés suivantes :

- Adaptation des interfaces (capture et acquisition du contexte)
- Adaptation des applications sensibles au contexte
- Augmentation du degré de précision lors de l'extraction d'information
- Détermination de l'ensemble des données les plus pertinentes pour l'application
- Traitement du contexte (modélisation, représentation et stockage)
- Facilité du mode d'interaction de l'utilisateur
- Intégration de la sensibilité au contexte dans les différentes architectures logicielles
- Sécurité et confidentialité des données contextuelles

2.2.4. Catégorisation des applications sensibles au contexte

En se basant sur la définition du context-aware, nous déduisons toutes les informations pertinentes relatives au contexte. Maintenant, la question est de savoir comment utiliser ces informations dans le développement des applications. Pour cela, les chercheurs ont proposé la catégorisation des applications utilisant le contexte pour une meilleure prise en compte de cette notion.

La catégorisation des applications sensibles au contexte offre deux bénéfices majeurs [Sonawane, 2012] :

- Spécification facile des types d'application que les chercheurs envisagent développer
- Description précise des types de propriétés requises pour la construction des futures applications sensibles au contexte

Pour commencer, Schilit [Schilit, 1994c] s'appuie sur l'idée que la prise en compte du contexte doit vérifier les aspects les plus importants : Où êtes-vous ? , Avec qui vous êtes ? Et Quelles sont les ressources (dans le voisinage) ? Et il démontre que le contexte n'est pas fonction seulement de l'endroit (l'espace) mais aussi de tous les objets mobiles susceptibles d'être changeants et qui se manifestent prés de l'utilisateur, et il inclut même des paramètres comme la lumière, le son, le réseau, la communication, la situation sociale…etc. D'où, il propose une catégorisation du contexte basée sur la nature de la tache, c'est-à-dire si cette tache perçoit une information ou exécute une commande, ou bien qu'elle s'effectue manuellement ou automatiquement. Ce qui revient à dire qu'on peut avoir quatre catégories d'applications utilisant le contexte obtenues par le croisement de deux axes orthogonaux ; le premier est lié à la nature de la tache (information ou commande), tandis que le deuxième désigne le mode d'exécution (manuel ou automatique). Ces catégories sont résumées dans la figure 2 [Schilit, 1994c] :

- Sélection de proximité : C'est une technique d'interaction utilisateur qui consiste à proposer tous les objets proches se trouvant dans le voisinage de l'utilisateur.
- Reconfiguration contextuelle automatique : C'est un processus qui permet d'ajouter de nouveaux composants, de supprimer des composants déjà existants ou de modifier les liens entre les composants suivant les changements du contexte.

- Commandes contextuelles : Selon le contexte d'exécution, une même commande peut produire des résultats différents.
- Déclencheurs contextuels : Ce sont des simples règles SI-ALORS utilisées pour spécifier comment les systèmes utilisant le contexte seront adaptés.

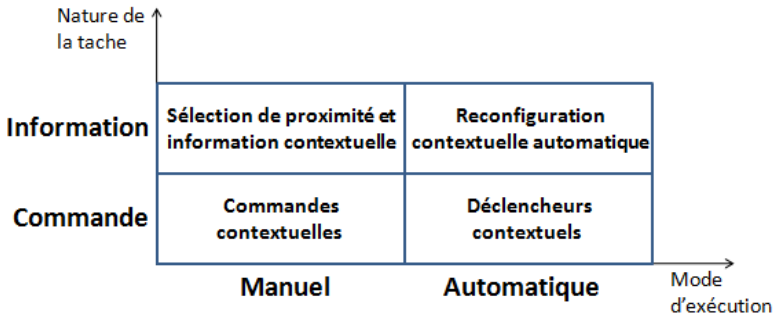


Figure 2. Les dimensions des applications sensibles au contexte [Schilit, 1994c].

Une autre forme de catégorisation du contexte a été proposée par Chen et Kotz [Chen, 2000]. En effet, ils introduisent la notion d'activité et de passivité pour la prise en compte du contexte ; par conséquent, ils définissent deux catégories d'applications sensibles au contexte :

- Utilisation du contexte actif : adapter automatiquement une application à un nouveau contexte en changeant le comportement de cette application.
- Utilisation du contexte passif : une application propose un contexte (nouveau ou modifié) à l'utilisateur et/ou sauvegarde ce contexte pour une éventuelle utilisation ultérieure.

De sa part, Pascoe [Pascoe, 1998a] a proposé une autre catégorisation des applications sensibles au contexte ; cette catégorisation s'appuie sur l'identification des principales caractéristiques et des facteurs qui influent la prise en compte du contexte dans le développement des applications. Ces caractéristiques sont :

- Perception (ou Capture) contextuelle : c'est la possibilité de détecter les informations contextuelles et de les présenter à l'utilisateur.
- Adaptation contextuelle : c'est la possibilité d'exécuter ou de modifier automatiquement un service en s'appuyant sur le contexte courant.
- Découverte des ressources contextuelles : les applications utilisant le contexte devront être capables de localiser et d'exploiter les ressources et les services supposés pertinents au contexte de l'utilisateur.
- Augmentation contextuelle : c'est la possibilité d'associer des données numérisées avec les éléments du contexte de l'utilisateur.

2.2.5. Perception et capture du contexte

La manipulation de n'importe quel outil exige l'existence et la disponibilité de cet outil. De même, pour que les applications soient en mesure de bien prendre en compte les changements du contexte, il faut leur fournir tous les éléments susceptibles d'influencer ce contexte. C'est-à-dire qu'il faut d'abord capturer et acquérir ces éléments pour ensuite permettre aux

applications de les utiliser. Pour cela, plusieurs mécanismes ont été élaborés pour capturer et percevoir le contexte courant et le présenter sous forme d'information.

Parmi ces mécanismes, nous présentons ceux établis par Chen et Kotz dans [Chen, 2000] :

2.2.5.1. Perception de l'endroit

Le lieu de l'utilisateur est un élément très important du contexte parce que le déplacement de cet utilisateur d'un emplacement vers un autre implique automatiquement un changement du lieu et par conséquent la modification du contexte. Pour capturer l'endroit, nous utilisons utiliser un système qui va garder la trace des mouvements de l'utilisateur dans l'espace. Par exemple, nous obligeons l'utilisateur à insérer son badge (ou carte d'identité magnétique) dans le lecteur approprié ou bien appuyer à l'aide d'un doigt sur une zone de reconnaissance automatique des empreintes. Une autre solution pour déterminer la localisation d'un utilisateur consiste à utiliser le GPS (Global Positioning System) qui peut nous fournir les coordonnées spatiales de n'importe quel endroit. L'inconvénient du système GPS est qu'il présente des insuffisances et des défaillances lorsqu'il s'agit de localiser des utilisateurs se trouvant à l'intérieur des constructions parce que le signal est si faible qu'il ne peut s'infiltrer ou traverser les obstacles solides tels que les murs des constructions.

2.2.5.2. Perception des contextes de bas niveau

On appelle contexte de bas niveau toute information contextuelle qui soit dans un état brut et qui peut être perçue ou mesurée directement ; par exemple, nous avons : le temps, la lumière, le son, la température, etc.

- Le temps : cette information contextuelle est généralement obtenue à partir de l'heure système de la machine. Néanmoins, il existe d'autres formes de temps à percevoir telles que les jours de la semaine, les jours du mois, les mois de l'année, les quatre saisons, les zones du faisceau horaire, … etc.
- Les objets voisins : tous les objets et les personnes proches peuvent influencer l'environnement de l'utilisateur et par conséquent modifient le contexte d'utilisation; d'où, le besoin d'enregistrer, périodiquement, la variation de l'état de ces objets (y compris les personnes).
- L'orientation : la direction est aussi une information contextuelle très importante qu'il faut mesurer pour, ensuite, la fournir aux applications.

Il existe d'autres éléments du contexte qu'il faut capturer à l'aide de techniques et d'outils appropriés. A cet effet, nous utilisons la photodiode pour détecter le niveau de la lumière, nous faisons appel à deux accéléromètres pour nous fournir les mesures de l'inclinaison et de la vibration, nous pouvons placer des senseurs infrarouges pour détecter la présence humaine, nous mettons un microphone pour capter les sons, ainsi que des senseurs pour mesurer la température, la pression, volume de gaz dégagé, etc.

2.2.5.3. Perception des contextes de haut niveau

Les contextes de haut niveau sont étroitement liés à l'activité courante de l'utilisateur et à son comportement social. Dans ce cas, nous pouvons capturer ces modes de contextes en utilisant

un système de surveillance (par vision) se basant sur la technologie de la vidéo (camera) associée aux méthodes de traitement d'images. Nous pouvons, aussi, consulter le calendrier de taches d'un utilisateur pour extraire ses activités prévues à un temps donné. Une autre solution pour la perception des contextes de haut niveau consiste à appliquer les techniques de l'intelligence artificielle sur les informations contextuelles déjà capturées. C'est-à-dire que nous pouvons reconnaître un contexte complexe à partir de la combinaison des autres senseurs de contextes de bas niveau [Schmidt, 1998].

2.2.5.4. Perception des changements du contexte

La liste des informations contextuelles proposées aux applications ne doit pas être figée mais, par contre, elle doit être actualisée après chaque modification du contexte. Il ne suffit pas, donc, de capturer les changements du contexte, mais d'essayer, aussi, de les prévoir en utilisant les techniques de sondage ou de simulation. Le sondage permet de prévoir le délai et la périodicité pour lesquels la liste des contextes sera mise à jours.

2.2.6. Modélisation du contexte

La modélisation des informations du contexte est une formalisation de données pour exprimer ces informations sous des formes bien adaptées (des modèles) afin d'être utilisées ou étudiées. La diversité des propriétés des éléments du contexte entraîne une grande variété de modèles, la majorité d'eux se penche vers la modélisation des informations basée sur l'endroit, d'autres font appel aux structures de données pour modéliser ces informations.

2.2.6.1. Modèles liés à l'endroit

Ce sont les modèles qui permettent de manier facilement la mobilité des objets dans l'espace et d'avoir la localisation plus ou moins exacte de ces objets.

- Modèle symbolique : l'endroit est représenté par des symboles abstraits ou des noms ; Exemple : un système basé sur l'accès par carte magnétique (Active Badge).
- Modèle géométrique : l'endroit est représenté par des coordonnées géométriques ; Exemple: Global Positioning System (GPS).
- Modèle combiné : l'endroit peut être représenté, à la fois, par des noms symboliques et des coordonnées géométriques ; les deux représentations peuvent se convertir l'une vers l'autre par application de prédicats prédéfinis.

2.2.6.2. Modèles liés aux structures de données [Strang, 2004], [Chaari, 2005]

- Paires (clé, valeur) : consiste à associer une valeur à chaque clé (ou un attribut) pour former une paire qui servira à la modélisation des informations contextuelles. Par exemple, le « context toolkit » de Dey [Dey, 2001] utilise ce type de modélisation.

- Production d'étiquette : ce type de modèles est basé sur un système de balisage. Il consiste à utiliser des étiquettes et des champs, ces mêmes champs peuvent contenir d'autres étiquettes et champs ; ce qui ramène le système vers un déroulement récursif. L'exemple de ConteXtML [Ryan, 2007] propose un protocole basé sur XML pour échanger les informations

contextuelles (sous forme de messages regroupés dans des balises) entre un client mobile et un serveur.

- Les RDF (Ressource Description Framework) : permettent de créer de nouveaux profils qui peuvent décrire les capacités des terminaux et les préférences des utilisateurs. La représentation la plus connue dans cette approche est une extension du profil W3C CC/PP [Indulska, 2003].

- Les modèles graphiques : le contexte est modélisé à l'aide de représentations graphiques. Le moyen le plus connu dans la modélisation à base d'outils graphiques est le langage UML. Celui-ci propose plusieurs diagrammes qui peuvent décrire les différentes vues d'un système. Par exemple les diagrammes UML utilisés dans [Bauer, 2003].

- Modèles orienté objet : un objet est caractérisé par ses états et ses méthodes. Les informations contextuelles seront exprimées par les états de l'objet tandis que les méthodes permettront d'accéder, manipuler et modifier ces états. Cette modélisation tire sa force des avantages connus de l'orienté objet (encapsulation, héritage, réutilisation, etc.). Ces modèles sont, par exemple, utilisés dans le projet TEA [TEA, 1998]

- Modèles basé sur la logique : les données du contexte sont exprimées formellement sous formes de faits (ou facteurs) dans un système à base de règles. L'avantage est qu'il est possible d'ajouter de nouvelles règles à la base de données à chaque besoin. Un exemple de ces modèles est proposé dans [McCarty, 1993] pour modéliser le contexte.

- Les modèles à base d'ontologies : ces modèles permettent d'utiliser les ontologies pour décrire et représenter les connaissances d'une façon très appropriée. Par exemple, le modèle CoBrA se base sur l'idée d'un courtier de contexte visant à la création d'ontologies pour construire des systèmes sensibles au contexte.

2.3. Notions sur l'adaptation

2.3.1. Définitions

Dans le sens général de l'adaptation, le dictionnaire français [Dictionnaire Larousse] donne une définition du terme « adapter » comme étant une action qui permet de « modifier, appliquer ou ajuster quelque chose pour l'approprier à une autre ».
Dans le domaine de l'informatique, nous citons la définition du dictionnaire [Granddictionnaire] qui décrit l'adaptation comme suit : « opération qui consiste à apporter des modifications à un logiciel ou à un système informatique dans le but d'assurer ses fonctions et d'améliorer ses performances ».

2.3.2. Motivations de l'adaptation

Pour assurer son évolution et sa pérennité, toute application a besoin d'être adaptée à tout type de changement. L'adaptation est nécessaire pour mettre à niveau le fonctionnement de l'application avec les nouveaux besoins qui peuvent surgir avant ou pendant son exécution.

L'étude menée dans [Ketfi, 2002] montre pourquoi nous avons besoin d'adapter les applications et classe les motivations et les raisons de l'adaptation en quatre catégories :

- Adaptation correctionnelle : en cas de disfonctionnement d'une application en cours d'exécution, nous procédons à un remplacement des parties défaillantes dans le but de corriger les défauts tout en conservant les mêmes fonctionnalités des éléments remplacés.
- Adaptation adaptative : dans certains cas, même qu'elle s'exécute correctement, l'application peut être adaptée pour satisfaire les nouveaux changements affectant son environnement d'exécution.
- Adaptation évolutive : toute application peut être adaptée par extension des éléments existants. Cette extension permet l'ajout de nouvelles fonctionnalités si celles-ci n'ont pas été prises en compte au moment du développement de l'application.
- Adaptation perfective : nous faisons recours à ce type d'adaptation lorsque nous avons besoin d'améliorer ou d'optimiser les performances d'une application.

2.3.3. Classification de l'adaptation

L'adaptation peut être appliquée de différentes manières, et selon cette application il existe de nombreuses classifications qui ont marqué les travaux antérieurs d'adaptation. L'étude menée par Chassot [Chassot, 2006] récapitule ces classifications comme suit :

- Statique ou dynamique : l'adaptation est dite statique lorsque la ressource (document, image, video, etc.), préalablement définie, est adaptée manuellement avant de l'utiliser dans son nouveau contexte d'utilisation. Et nous disons que l'adaptation est dynamique lorsque les transformations sur la ressource sont effectuées automatiquement au cours de son utilisation.
- Centralisée ou distribuée : ce type de classification est du à la nature de la ressource utilisée puisque celle-ci peut être locale à une seule machine ou distribuée sur plusieurs machines. Dans le cas d'une ressource locale, l'adaptation est tout simplement centralisée. Mais si la ressource est distribuée, le processus d'adaptation peut être centralisé sur une seule machine ou distribué sur les différentes machines où les parties de cette ressource sont stockées.
- Comportementale ou architecturale : l'adaptation est dite comportementale lorsqu'elle peut modifier le comportement d'une ressource sans affecter sa structure interne. Mais, si la structure interne de la ressource peut être modifiée suivant les besoins de l'utilisateur, nous disons que l'adaptation est architecturale.

2.3.4. Performances de l'adaptation

Un processus d'adaptation nécessite la vérification de plusieurs conditions et paramètres qui permettent d'évaluer ses performances [Ketfi, 2002] :

- La cohérence : l'opération d'adaptation doit préserver la cohérence de l'application à adapter. Elle doit choisir avec précaution le moment de son lancement pour ne pas déstabiliser l'exécution normal de l'application. La cohérence d'une application comporte la sécurité, la complétude, les bons moments de lancer l'adaptation et la possibilité d'annuler une adaptation lancée.

- Degré d'automatisation : c'est la capacité d'une application à réaliser sa propre adaptation. En effet, chaque application possède toutes les informations nécessaires pour lancer une adaptation sans l'intervention de l'utilisateur ou de l'administrateur.

2.3.5. Adaptation des applications au contexte

L'adaptation d'une application au contexte implique l'adaptation des éléments qui la composent ; c'est-à-dire chercher à adapter le comportement, le contenu et la présentation des données suivant le contexte d'utilisation (Figure 3).

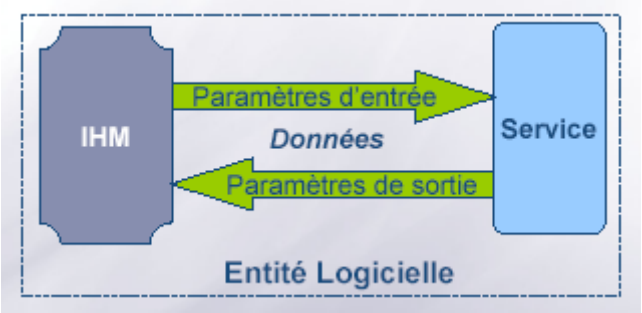


Figure 3. Vue « utilisateur » d'une application [Chaari, 2005].

2.3.5.1. Adaptation de comportement (services)

Le comportement peut être représenté par un graphe de dépendance où les nœuds symbolisent les services et les arcs désignent les relations entre les services. L'adaptation de comportement consiste à transformer un graphe de dépendance de services en un autre graphe de dépendance de services adaptés. Cette transformation porte sur les nœuds ainsi que sur les arcs. Une solution, citée dans [Chaari, 2005], est d'intégrer une couche d'adaptation sur les services de l'application en utilisant plusieurs opérateurs comme les opérateurs de filtrage (projection, sélection, augmentation, union), les opérateurs sur les instances de services (SélectionVersion, AjoutVersion) ou les opérateurs d'adaptation du graphe (VerrouillerService, AjouterService, InsérerService).

2.3.5.2. Adaptation de contenu (données)

C'est la modification de quelques propriétés des informations (les données) proposées à l'utilisateur de façon à satisfaire ses préférences. Par exemple, nous avons :

a. L'adaptation de format : modifier complètement ou partiellement le format d'une donnée (ex. réduire le nombre de couleurs d'une image)
b. La traduction : convertir un texte suivant les préférences de l'utilisateur
c. La compression de données : peut être une compression brute (ex. zip), une compression multimédia (ex. jpeg) ou une compression sémantique (ex. résumé d'un texte).
d. La décomposition de données : sélectionner seulement une partie d'un objet
e. L'agrégation de données : c'est le rassemblement de plusieurs données différentes. Elle peut être de deux types : agrégation spatiale (ex. un objet image plus un objet texte) ou agrégation temporelle (ex. montage de séquences vidéos).

2.3.5.3. Adaptation de présentation (IHM)

Consiste à assouplir l'interface utilisateur de façon à ce qu'elle soit maniable plus facilement et selon les préférences de cet utilisateur. Par exemple, si l'utilisateur souhaite des formes d'affichages spécifiques (ex. les courbes graphiques au lieu des données tabulaires). L'adaptation de présentation consiste aussi à fournir des facilités de navigation entre les entités logicielles (services+interfaces+données) pour pouvoir bénéficier de tous les services de l'application [Chaari, 2005].

2.4. Notions sur la personnalisation

2.4.1. Définitions

D'après le dictionnaire Larousse, le mot « personnalisation » est définit comme suit :
« Adaptation d'un produit ou d'un service à la personnalité de celui à qu'il est destiné » et « Action d'identifier le pouvoir à la personne qui l'exerce »

Dans le domaine de l'informatique, nous citons la définition du dictionnaire « granddictionnaire » qui décrit la personnalisation comme suit : « mécanisme qui permet de personnaliser le comportement et l'apparence d'un composant logiciel, et ce, de façon conviviale à l'intérieur d'un outil de développement ».

La personnalisation permet de faciliter l'expression du besoin de l'utilisateur et de lui permettre d'obtenir des informations pertinentes lors de ses accès à un système d'information [Bouzeghoub, 2005].

En d'autre terme, la personnalisation consiste à [Abbas, 2008]:

- Filtrer un flux d'information entrant pour éliminer le bruit
- Guider la navigation dans un vaste espace d'information
- Recommander un ensemble d'information à l'utilisateur
- Ajuster le résultat d'une requête suivant une interface
- Adapter l'interaction à la situation contextuelle de l'utilisateur

2.4.2. Avantages de la personnalisation

Etroitement liée aux besoins des utilisateurs, la personnalisation présente plusieurs avantages pouvant satisfaire ces besoins et préférences. Parmi ces avantages, nous citons [http://www.abc-netmarketing.com] :

- Meilleure adaptation aux besoins des utilisateurs
- Fidélisation
- Différentiation
- Augmentation de la marge
- Réduction des coûts

2.4.3. Domaines de la personnalisation

La personnalisation est un principe qui vise à prendre en charge le maximum des préférences des utilisateurs en leur offrant des résultats plus pertinentes et des services très satisfaisants.

L'importance de ce principe se mesure par son omniprésence dans divers domaines de recherche. La classification des domaines d'activité de la personnalisation varie suivant le type d'applications qui utilisent ce principe ou bien suivant les technologies de base qui permettent de développer ces applications [Bouzeghoub, 2005]. Cette classification met en évidence plusieurs domaines de la personnalisation comme :

- Commerce électronique : ce domaine concerne les opérations électroniques de l'achat et la vente des produits tout en proposant des choix de produits prédéfinis basés sur l'historique des opérations précédentes de l'utilisateur et de son comportement antérieur.
- Dissémination sélective d'information : c'est la diffusion des informations quotidiennes comme la recherche d'emploi, les forums, la veille technologique, etc. Ce domaine consiste à filtrer le flux d'information en se basant sur un profil constitué des principaux centres d'intérêt de l'utilisateur, sa langues, sa position, etc.
- Apprentissage assisté par ordinateur : ce domaine est lié aux environnements informatiques pour l'apprentissage humain. Ici, la personnalisation cherche à adapter le système d'apprentissage en proposant une formation sur mesure qui tient en compte le style d'apprentissage préféré, les connaissances de l'apprenant, son comportement, sa réaction, etc.
- Accès aux bibliothèques électroniques : dans ce domaine, la personnalisation propose l'accès aux différents types de documents (classiques ou multimédias) suivant plusieurs façons liées aux préférences de l'utilisateur : utiliser une liste prédéfinie, répondre à une nouvelle requête, ou bien recommander des choix en fonction de l'historique de l'utilisateur.
- Systèmes d'information mobiles : dans ce domaine, la personnalisation cherche à réduire l'effort et le temps nécessaire à la recherche de l'information pertinente propre à chaque situation et à chaque instant.
- Configuration de logiciels : les différentes configurations de logiciels (systèmes d'exploitation, protocole réseaux, services web) en fonction des besoins et des préférences des utilisateurs sont considérées, pour ce domaine, comme des types de personnalisation.
- Systèmes de bases de données : les principales contributions de personnalisation dans le domaine des bases de données sont liées à la nature et au type du langage de manipulation de données choisi. Ces contributions comportent la définition des vues, les critères de filtrage, ordre de préférence des critères, ordre de préférence des résultats, etc.
- Moteurs de recherche d'information : ce sont les moteurs de recherche web qui introduisent les notions de profil et d'adaptation en manipulant les données personnelles, les données de sécurité, le filtrage des résultats, la customisation de l'interface, etc.
- Interface homme-machine : la personnalisation des interfaces homme-machine inclut à la fois le choix de l'interface graphique et le choix du mode de communication (langue du dialogue, niveau du dialogue, type d'interaction).

2.4.4. Dimensions de la personnalisation

D'après [Bouzeghoub, 2005b], il est important d'étudier et d'explorer le processus de personnalisation d'une façon verticale. C'est-à-dire connaitre les différentes dimensions à partir desquelles il faut définir les profils des utilisateurs. Pour cela, un ensemble de huit dimensions a été identifié.

- Données personnelles : représentent la partie statique de l'utilisateur,
- Centre d'intérêt : exprime le domaine d'expertise de l'utilisateur,
- Ontologie du domaine : explique la sémantique des termes et opérateurs employés par l'utilisateur,
- Qualité attendues des résultats délivrés : permet d'exprimer les préférences externes,
- Customisation : exprime les modularités de présentation des résultats,
- Sécurité (confidentialité) : concerne les données, les informations et les requêtes,
- Retour de préférences : regroupe les informations collectées sur le comportement de l'utilisateur,
- Informations diverses : regroupe les attributs non classables dans les autres dimensions.

2.4.5. Mécanismes clefs de la personnalisation

La personnalisation est un processus de contextualisation parce que son objectif principal est d'adapter les informations, les contenus et les services par rapport au contexte dans lequel se trouve l'utilisateur. Cet objectif ne peut être atteint que lorsque certains mécanismes clefs de la personnalisation soient vérifiés [Mignot, 2014]:

- Spécialisation : les applications sont bien définies et dédiées à des profils utilisateurs typiques et importants
- Adaptation : tous les éléments de l'application changent selon les actions de l'utilisateur
- Prédiction : les actions et les propositions proactives doivent être capables d'anticiper les besoins

2.4.6. Services de la personnalisation

Chaque système doit fournir des services biens définis à partir desquels il peut ajuster les données en fonction des besoins et des préférences de l'utilisateur.

Ces services de personnalisation comportent : la recommandation, le filtrage des résultats et l'enrichissement des requêtes [Abbas, 2008].

- La recommandation : ce service est un processus de personnalisation qui permet de proposer à un utilisateur un ensemble de conseils déduits à partir de l'expérience des autres utilisateurs. Ces conseils peuvent augmenter le nombre et la qualité des choix de décisions à prendre quant à la recherche d'information. Parmi les approches utilisant le service de recommandation, nous citons l'exemple de l'approche basée sur la notion de profil utilisateur et l'approche basée sur la technique de fouilles des données.

- Le filtrage des résultats : ce service est basé sur un filtrage des informations demandées par l'utilisateur (sous forme de requêtes) afin d'ignorer et supprimer les données non pertinentes. Le principe de base consiste à créer une image du contenu de chaque document, ensuite cette image est comparée aux profiles. Comme exemples de filtrage de résultats, nous pouvons citer : le filtrage cognitif (basé sur le contenu) et le filtrage collaboratif (basé sur la pertinence des autres résultats de recherche).
- L'enrichissement des requêtes : consiste à ajouter les éléments (sous forme de prédicats) du profil utilisateur dans sa requête. Chaque requête est reformulée suite à son enrichissement à partir du profil utilisateur. La méthode type utilisant ce service est celle de Koutrika [Koutrika, 2004].

Conclusion

L'informatique ubiquitaire nécessite la disponibilité de l'information n'importe où et n'importe quand. Les applications opérant dans des environnements ubiquitaires doivent être en mesure de s'exécuter normalement même si le contexte d'utilisation change. Au cours de ce deuxième chapitre, nous avons présenté une synthèse des meilleures définitions décrivant les concepts relatifs à la notion de « contexte ». En effet, nous avons rappelé quelques définitions élémentaires du contexte d'utilisation ainsi que ses composants, ses caractéristiques, son cycle de vie, sa catégorisation, etc. Nous avons présenté, aussi, quelques détails sur le domaine de la sensibilité au contexte et les applications développées dans ce domaine. A la fin de ce chapitre, nous avons exposé un aperçu sur deux notions étroitement liées au contexte d'utilisation, à savoir : l'adaptation et la personnalisation.

CHAPITRE 3

ETUDE COMPARATIVE

« CLASSIFICATION DES TRAVAUX VOISINS ET POSITIONNEMENT DU PROBLÈME »

Introduction

Les futures applications doivent fournir des résultats exacts et des informations personnalisées, et elles doivent être adéquates avec tous les changements du contexte d'utilisation subis par la situation courante de l'utilisateur. Les systèmes d'information ubiquitaires se distinguent par un ensemble de caractéristiques spécifiques comme la mobilité des utilisateurs, l'hétérogénéité des sources d'information et la distribution des systèmes. Ces contraintes, regroupées, participent à la définition de la notion « contexte d'utilisation ». Cette notion représente l'élément central autour duquel s'articulent toutes les recherches menées dans le domaine de la sensibilité au contexte comme la modélisation contextuelle, la gestion des profils, l'adaptation des applications, la personnalisation des informations, etc.

Dans ce chapitre, nous allons exposer une synthèse des travaux antérieurs ayant un rapport avec notre domaine de recherche. La synthèse portera sur un échantillon de 28 travaux du domaine de la sensibilité au contexte choisis aléatoirement dans la limite de leur disponibilité sur le web. Cette synthèse sera présentée sous forme d'une étude comparative dans laquelle nous effectuerons une comparaison à deux niveaux. Dans un premier temps, nous allons comparer les travaux étudiés entre eux en les classant en groupes homogènes (suivant les thèmes étudiés et les notions proposées) et ensuite, dans un second temps, nous allons comparer ces travaux avec les nôtres qui représentent les principaux axes de cette thèse. La double comparaison sera accompagnée d'une évaluation qui doit mettre en évidence les apports et les avantages des travaux étudiés, et qui doit, aussi, soulever leurs insuffisances et leurs limites. Suivant les résultats des opérations de comparaison et d'évaluation, nous allons essayer de positionner notre problématique par rapport à ces travaux déjà réalisés dans le domaine de la contextualisation, et nous allons essayer de se situer clairement en mettant l'accent sur l'utilité, l'importance et la qualité de notre travail relativement aux autres. A la

fin de cette partie, nous présenterons une section « motivations et objectifs » dans laquelle nous allons mettre en évidence, motiver et argumenter les contributions proposées dans cette thèse ainsi que les valeurs ajoutées espérées.

3.1. Classification des travaux voisins étudiés

3.1.1. Présentation des travaux voisins

Plusieurs études ont été menées dans le domaine de la sensibilité au contexte et surtout concernant la prise en compte des changements contextuels qui environnent l'exécution d'une application ainsi que les préférences souhaitées d'un utilisateur. La majorité de ces études ont essayé de représenter le contexte d'utilisation à l'aide de modèles afin de concevoir des applications adaptées capables de fournir des informations personnalisées. Dans ce qui suit, nous allons exposer un bilan qui résume les travaux antérieurs ayant un rapport avec notre domaine de recherche.

Dans sa thèse [Abbas, 2008], Abbas présente un système d'accès personnalisé à l'information qui est basé sur le principe de la gestion des profils et qui est fondé globalement sur trois principes : les profils, le contexte et les services. Les profils contiennent essentiellement des informations et des connaissances sur l'utilisateur. Le contexte regroupe un ensemble de paramètres décrivant l'environnement d'utilisation du système. Les services sont des programmes autonomes construits pour exécuter les différentes tâches de la personnalisation. Ce système de personnalisation s'articule autour de trois éléments : l'utilisateur qui est responsable d'accéder à des informations à l'aide de requêtes, les ressources extraites à partir des sources de données variées et le logiciel qui permet de gérer la personnalisation de l'accès à l'information.

Dans [Ayed, 2007], les auteurs proposent une approche de développement basée sur les modèles qui permet de concevoir des applications sensibles au contexte indépendamment de la plateforme. Cette approche comporte plusieurs phases : identification des informations contextuelles requises, définition de la variabilité (structure, comportement et paramètres) de l'application, identification des mécanismes de collecte du contexte, identification des mécanismes d'adaptation, définition des transformations et la génération du code. Les transformations utilisées sont basées sur la séparation entre préoccupations fonctionnelles et préoccupations non fonctionnelles de l'application. Pour la concrétisation de cette approche et de ses phases, un profil UML a été proposé afin d'assurer la modélisation des applications sensibles au contexte. Ces applications pourront être introduites facilement, via le profil proposé, dans tout logiciel basé sur UML.

Les auteurs de [Bouzeghoub, 2014] proposent un modèle de profil générique qui permet de classifier les informations contenues dans les profils. Ce type de profil est principalement défini par six dimensions : les données personnelles, le centre d'intérêt, la qualité attendue, les préférences de livraison, la sécurité et l'historique des interactions de l'utilisateur. Aussi, ils proposent un ensemble d'opérateurs de manipulation de profils et une plateforme de gestion de profils consistant à : créer un profil, évoluer un profil, importer un profil, intégrer plusieurs profils, valider un profil, apparier des profils, etc.

Le projet SECAS (Simple Environment for Context Aware Systems) s'intéresse à l'adaptation contextuelle en proposant une plateforme de conception et déploiement d'applications sensibles au contexte d'utilisation [Chaari, 2005]. Pour assurer une telle adaptation, ce projet fait appel aux services Web. L'architecture générale de SECAS est définie par cinq étapes : capture du contexte, interprétation du contexte, stockage du contexte, dissémination du contexte et adaptation au contexte. L'adaptation de l'application au contexte comporte l'adaptation de comportement, l'adaptation de contenu et l'adaptation de présentation. Il faut noter, ici, que les auteurs du projet SECAS parlent de l'importance de la séparation entre les données contextuelles et les données de l'application pour la modélisation du contexte mais sans donner des détails précisant la manière de réalisation de cette séparation.

Chaari a poursuivi ses recherches sur l'adaptation contextuelle lors de sa thèse [Chaari, 2007] en proposant une stratégie plus complète, générique et évolutive d'adaptation d'applications au contexte d'utilisation sur trois volets : adaptation fonctionnelle des services offerts à l'utilisateur, adaptation des données renvoyées par ces services et adaptation de la présentation des données à l'utilisateur. Pour valider la conception et la réalisation de son approche d'adaptation, l'auteur a présenté un prototype de la plateforme SECAS incluant un nouveau modèle contextuel. Cette conception comporte trois types de modélisation : modélisation fonctionnelle, modélisation dynamique et modélisation statique.

Une étude, faite aussi dans le domaine de la sensibilité au contexte, est présentée par Chen [Chen, 2000]. D'une part, cette étude explore les systèmes et les applications sensibles au contexte suivant plusieurs points comme : les types de contextes utilisés, les modèles des informations contextuelles, les systèmes qui supportent la capture et la diffusion du contexte, les applications adaptables aux changements du contexte ; et d'autre part, elle synthétise les différentes notions relatives au domaine de la sensibilité au contexte comme : la définition des termes « contexte » et « sensibilité au contexte », les applications sensibles au contexte déjà développées, les différentes approches de capture et de modélisation du contexte, les différentes infrastructures supportées ainsi que les éléments de sécurité.

A.K. Dey présente dans [Dey, 2001] une boite à outils « context toolkit » qui peut servir comme support principal pour l'adaptation des applications au contexte en passant par trois étapes : capturer le contexte, l'interpréter et le fournir à l'application. Cette boite à outils est constituée principalement des éléments suivants : les composants graphiques (widgets), les interpréteurs (interpreters), les collecteurs (aggregators), les enregistreurs (discoverers) et les services (services). Le fonctionnement du toolkit débute lorsqu'un élément de contexte surgit. Cette nouvelle opportunité (modification de l'environnement) est sauvegardée par un enregistreur. Celui-ci prévient les collecteurs pour aller chercher les widgets et les interpréteurs appropriés, et invite les applications à choisir les meilleurs agrégateurs, widgets et interpréteurs. Un senseur fournit les données au widget qui sauvegarde le contexte, le widget fait appel à un interpréteur pour convertir ces données et en obtenir des informations de haut niveau d'abstraction, ce qui rendra l'information contextuelle plus disponible et mieux exploitable par les applications ou les autres composants. L'agrégateur recueille les parties du

contexte à partir des différents widgets et prépare les réponses appropriées aux requêtes formulées par les applications [Dey, 2001].

Dans [Fuentes, 2008], les auteurs présentent un processus pour la conception et l'implémentation des applications sensibles au contexte. Ce processus est basé sur l'utilisation de la notion d'aspects qui contribue à l'encapsulation des préoccupations contextuelles dans des modules bien définis. Dans ce processus, les applications pervasives sont modélisées à l'aide d'un profil UML basé sur les aspects.

Le travail cité dans [Gensel 2008] expose des travaux d'adaptation contextuelle menés sur la définition, la représentation et l'exploitation de modèles de contexte dans le cadre des systèmes d'information web à usage collaboratif. A cet effet, un modèle à objets du contexte a été proposé intégrant des éléments du contexte physique de l'utilisateur (localisation, dispositif utilisé …) et ceux du contexte collaboratif dans lequel il évolue (notions de groupe, de rôle, d'activité ...).

Hsu a développé un framework à plusieurs niveaux de contexte pour supporter les environnements ubiquitaires et adapter les applications au contexte [Hsu, 2011]. Ces niveaux de contexte sont au nombre de six : niveau des capteurs, niveau de l'information, niveau des services, niveau de présentation, niveau du matériel et niveau d'application. Le framework proposé introduit les technologies web 2.0 qui représentent une partie élémentaire des systèmes sensibles au contexte pour faciliter l'échange et le partage des ressources contextuelles (y compris les informations et les services).

Dans une autre étude [Hsu, 2012], l'auteur Hsu a présenté un profil UML pour la modélisation des applications sensibles au contexte incluant les technologies web 2.0. Ce profil permettra de faciliter l'interopérabilité entre les systèmes hétérogènes évoluant dans les environnements sensibles au contexte. A travers ce travail, l'auteur cherche à mettre en place un outil visuel basé sur les technologies web 2.0 pour la modélisation des applications sensibles au contexte et construire, ainsi, un modèle conceptuel propre au contexte d'utilisation.

Dans [Hsu, 2013], l'auteur propose une approche basée sur les principes de la MDA (Model Driven Architecture) sous forme d'un profil UML nommé "Web2.0MUML" et destiné à la modélisation des 'mashups' qui caractérisent la technologie spécifique du Web 2.0. Ce profil consiste à étendre les notations du langage UML en utilisant les mécanismes d'extensibilité dans le but de modéliser les 'mashups' pour représenter les propriétés structurelles les plus pertinentes du web 2.0 au niveau conceptuel.

Dans l'étude [Hussein, 2010], les auteurs présentent une nouvelle approche pour la modélisation et la réalisation d'un système logiciel adaptable au contexte d'utilisation, et ce, en proposant une représentation spécifique des relations contextuelles sous forme d'un modèle contextuel adaptable permettant l'adaptation et l'évolution de ce système au cours du processus d'exécution (runtime). La proposition comporte un nouveau modèle contextuel CAMP (Context-aware Adaptive systeMs comPonent) et un méta-modèle qui peuvent générer

l'implémentation du système et qui permettent de réduire les erreurs de développement en prenant en considération toutes les relations contextuelles pendant l'étape de modélisation.

L'étude de [Jrad, 2008] présente quelques travaux sur la modélisation contextuelle des utilisateurs destinés spécialement pour la personnalisation. Le modèle contextuel, basé sur le contexte de navigation des utilisateurs, contient plusieurs catégories comme l'historique de navigation, les préférences et la tache courante. Ce travail montre aussi la façon dont le modèle contextuel est alimenté par extraction de données en utilisant les techniques d'analyse du comportement de l'utilisateur à travers sa navigation (Web Usage Mining).

Dans [Kirsch-Pinheiro, 2006], l'auteur propose une méthode d'adaptation contextuelle et de personnalisation de l'information en focalisant sur la modélisation du contexte, la manipulation du contexte et le filtrage guidé par le contexte. Ce travail étudie la modélisation du contexte en proposant un model contextuel évolutif qui cherche à séparer les informations contextuelles de celles propres au domaine d'application en utilisant un formalisme par objets du type classe/association. Pour la personnalisation des informations, l'auteur propose un processus « filtrage guidé par le contexte » qui est capable de sélectionner un sous-ensemble d'informations suivant lequel il peut adapter les résultats à l'individu et à ses préférences en lui retournant des informations pertinentes et personnalisées.

Dans [Kostadinov, 2014], l'auteur présente un état de l'art des approches de personnalisation existantes comme les services de personnalisation, le contenu des profils, les techniques de construction des profils, l'utilisation des données du profil et la sécurité des informations. Il propose, aussi, un modèle de profil générique qui prend en charge les aspects de la personnalisation et qui met en évidence les relations existantes entre les composants.

Hamid Mcheick décrit un ensemble d'éléments (les liens lors de l'exécution, la flexibilité et la performance) et de techniques (liens statiques et liens dynamiques) nécessaires pour l'adaptation d'une application au contexte [Mcheick, 2014]. Ce travail présente un model contextuel caractérisé par sa capacité d'interagir et d'adapter intelligemment et dynamiquement par rapport aux changements de l'environnement et aux besoins des utilisateurs. Ceci doit permettre aux systèmes sensibles au contexte de s'auto-adapter et de s'auto-évoluer au cours de l'exécution. L'avantage de cette proposition est qu'elle réalise un haut niveau d'abstraction et qu'elle supporte la construction de systèmes sensibles au contexte en se basant sur la modularité et la séparation des préoccupations.

Hervé Mignot présente, dans [Mignot, 2014], un exposé dans lequel il définit la notion de personnalisation comme étant l'adaptation des informations fournies par rapport au contexte dans lequel se trouve l'utilisateur ; et il conclut, ensuite, que la personnalisation signifie tout simplement la contextualisation. Enfin, il fait une comparaison entre la personnalisation avec d'autres notions comme la customisation et l'adaptation et propose les éléments d'une bonne stratégie de personnalisation (caractéristiques, composants, risque, etc.).

Dans l'étude citée dans [Ou, 2006], les auteurs ont conçu un modèle contextuel utilisant les ontologies dans le but de modéliser les informations contextuelles, ensuite ils proposent une nouvelle architecture MDIA (Model Driven Integration Architecture). Cette architecture permet d'intégrer les spécifications des modèles complexes et de générer l'implémentation des applications sensibles au contexte.

L'étude présentée dans [Seridi, 2012] utilise les opportunités de la notion de 'modèle' et de l'approche MDA (Model Driven Architecture) pour développer des services web sensibles au contexte. Cette étude propose un méta-modèle qui permet de modéliser toutes les informations contextuelles en utilisant la séparation des contraintes contextuelles pour permettre la prise en charge du contexte pendant les premières étapes du processus de développement. Globalement, le développement des services sensibles au contexte est défini par trois parties : séparation des préoccupations contextuelles, adaptation des services au contexte et les étapes de développement (modélisation et transformation).

Sheng et Benatallah ont proposé un langage de modélisation sous forme d'un méta-modèle "Context UML" destiné au développement des services web sensibles au contexte [Sheng, 2005]. Leur proposition, basée sur les modèles, consiste à modifier quelques méta-classes UML et à construire une extension du méta-modèle UML (syntaxe et notation) qui sera utilisé pour la spécification des informations nécessaires à la conception des services web sensibles au contexte.

Le profil de modélisation du contexte CMP (Context Modeling Profile) a été proposé dans le travail de [Simons, 2007] dans le but de construire des modèles contextuels pour les systèmes distribués et mobiles. Ce profil se présente comme un langage visuel qui fournit un ensemble de règles spécifiques aux modèles de contexte nécessaires au développement des applications mobiles du domaine de la sensibilité au contexte. Le profil proposé est construit à partir de plusieurs extensions du langage UML sans modification du méta-modèle UML. Ceci lui permettra d'être intégré facilement dans tous les outils de modélisation UML.

Dans [Sindico, 2009], les auteurs présentent une méthode de développement basée sur les modèles qui est définie par une extension du langage UML et qui est destinée à la modélisation d'applications évoluant dans le domaine de la sensibilité au contexte. Cette extension utilise un langage spécifique appelé CAMEL (Context Awareness ModEling Language) qui permet d'enrichir les modèles UML avec des éléments et des comportements liés directement au contexte. Les modèles UML ainsi construits en utilisant le langage CAML vont subir des opérations de transformation de modèles pour générer un code exécutable relatif à la plateforme choisie.

Dans [Skillen, 2012], les auteurs présentent un profil utilisateur qui est basé sur les ontologies et qui peut fournir un modèle extensible de profil utilisateur reposant sur la modélisation des aspects statiques et dynamiques de l'utilisateur et facilitant, ainsi, son utilisation dans les applications sensibles au contexte. Ce profil utilisateur est basé sur l'identification et la caractérisation des propriétés de l'utilisateur. L'ontologie utilisée est définie sous l'éditeur

d'ontologie 'Protégé' et elle est exprimée en langage OWL (Web Ontology Language). L'objectif principal, ici, est de permettre la personnalisation des informations d'applications lorsque l'utilisateur se déplace entre les environnements mobiles.

Une autre étude sur l'accès personnalisé à l'information a été présentée dans [Tamine, 2005]. Cette étude démontre, par des approches et des techniques, que la modélisation de l'utilisateur représente la clé de la personnalisation. Les auteurs de cette étude présentent, aussi, quelques systèmes d'accès personnalisé à l'information en mettant l'accent sur leur architecture qui comprend la présentation des principales fonctions liées à la gestion des profils utilisateurs et l'exploitation de ces profils dans le processus d'accès à l'information.

Vale et Hammoudi [Vale, 2008] utilisent l'approche MDA dans la modélisation du contexte et dans le développement des applications sensibles au contexte, et pour cela, ils ont conçu le méta-modèle contextuel CSOA (Context-aware Service Oriented Architecture) utilisant les principes de l'approche MDA et se basant sur les vues EDOC-ECA (Enterprise Distributed Object Computing- Enterprise Computing Architecture).

Le travail de [Vale, 2009] présente une architecture interopérable pour le développement de services sensibles au contexte en utilisant les principes de l'ingénierie des modèles et les ontologies. Ce travail se base sur la séparation des préoccupations pour garantir l'interopérabilité et la réutilisation pour tout développement d'applications sensibles au contexte. A cet effet, un méta-modèle de contexte est défini suivant le méta-modèle ODM (Ontology Definition Metamodel) et conformément aux principes de l'approche MDA. Par ailleurs, ce même travail étudie l'adaptation des applications suivant les détails techniques des plateformes utilisées où les auteurs proposent une vue d'adaptation, sous forme d'un modèle spécifique de plateforme (PSM), composée de plusieurs composants : composants métiers, composants de contexte et composants de communication (connectors).

Le travail cité dans [Van Den Bergh, 2005] présente une analyse détaillée qui décrit comment le langage UML et ses mécanismes d'extension (profil UML) peuvent être utilisés dans la modélisation des aspects structurels et dynamiques des interfaces utilisateurs liées au contexte. La conception de ces interfaces utilisateurs est basée principalement sur trois modèles : un modèle de taches qui décrit les taches d'un utilisateur interagissant avec un système pour un but précis, un modèle de dialogue qui décrit la manière selon laquelle les taches de l'utilisateur sont accomplies et un modèle de présentation qui spécifie la structure de l'interface utilisateur.

Hormis ces travaux apparemment liés à notre champ d'étude, nous avons aussi étudié un ensemble de travaux qui se sont focalisés sur la construction de profils UML ou qui ont proposé des extensions de notations UML mais pour des domaines autres que celui de la sensibilité au contexte. L'étude de ce type de travaux nous a permis de mieux comprendre les principes de construction d'un profil UML, la mise en place et le fonctionnement des mécanismes d'extensibilité, le choix des informations à personnaliser, etc. Dans la table 1,

nous listons quelques uns de ces travaux (hors du domaine de la sensibilité au contexte) avec leurs domaines respectifs.

Travaux et/ou Etudes	Domaines
UML profile for aspect-oriented software development [Aldawud, 2003]	Développement de logiciels orienté aspects
Towards an UML profile for the description of software architecture [Amirat, 2009]	Description des architectures logicielles
Extending UML to Support Ontology Engineering for the Semantic Web [Baclawski, 2001]	Les ontologies
Extending UML for modeling web applications [Baresi, 2001]	Modélisation hypermédia
A Metamodel and UML Profile for Rule-Extended OWL DL Ontologies [Brockmans, 2006]	Les ontologies OWL
Extending UML to Model Multi-Agent Systems [Da Silva, 2004]	Les systèmes multi-agents
Representing WSDL with Extended UML [De Castro, 2004]	Langage WSDL (Web Service Description Language)
A UML Profile for OWL Ontologies [Djuric, 2004]	Les ontologies OWL
UML Profiles for Real-Time Systems and their Applications [Gherbi, 2006]	Les systèmes temps réel
A UML Profile to Model Mobile Systems [Grassi, 2004]	Les systèmes mobiles
Towards a UML profile for the description of dynamic software architectures [Hadj-Kacem, 2005]	Description des architectures logicielles dynamiques
Towards a UML Profile for Service-Oriented Architectures [Heckel, 2003]	Les architectures orientées service SOA (Service-Oriented Architectures)
Rational UML Profile for Business Modeling [Johnston, 2004]	modélisation des processus métiers
Towards a UML profile for software architecture descriptions [Kandé, 2000]	Description des architectures logicielles
Extending the UML 2 Activity Diagram with Business Process Goals and Performance Measures and the Mapping to BPEL [Korherr, 2006]	Processus métiers et langage BPEL (Business Process Execution Language)
A UML 2 profile for business process modeling [List, 2005]	Modélisation des processus métiers
UML Profile for the Platform Independent Modelling of Service-Oriented Architectures [Lopez-Sanz, 2007]	Les architectures orientées service SOA (Service-Oriented Architectures)
A UML profile for multidimensional modelling in data warehouses [Lujan-Mora, 2006]	Modélisation multidimensionnelle
Extending UML for Agents [Odell, 2000]	Les agents

New Approach for Conception and Implementation of Object Oriented Expert System Using UML [Touzi, 2009]	Les systèmes experts orienté objet
A UML Profile for Agent-Oriented Modelling [Wagner, 2002]	La modélisation orientée agent
A UML Profile for Communicating Systems [Werner, 2006]	Les systèmes communicants
Towards a UML profile for software product lines [Ziadi, 2004]	Lignes de produits logiciels

Table 1. Liste des travaux présentant des extensions UML pour des domaines hors de la sensibilité au contexte

3.1.2. Analyse « thématique » des travaux voisins

Les travaux concernés par cette étude comparative sont ceux qui touchent le domaine de la contextualisation à travers un (ou plusieurs) thème(s) parmi ceux qui ont été choisis relativement au contexte d'utilisation et à la sensibilité au contexte. Ces thèmes sont les suivants (figure 4):

- Adaptation des applications par rapport au contexte d'utilisation (A.A)
- Gestion des profils se basant sur le contexte d'utilisation (G.P)
- Modélisation du contexte d'utilisation (MOD)
- Personnalisation des informations suivant le contexte d'utilisation (P.I)
- Séparation des préoccupations contextuelles (S.P)
- Profil UML destiné à la modélisation du contexte d'utilisation (UML)

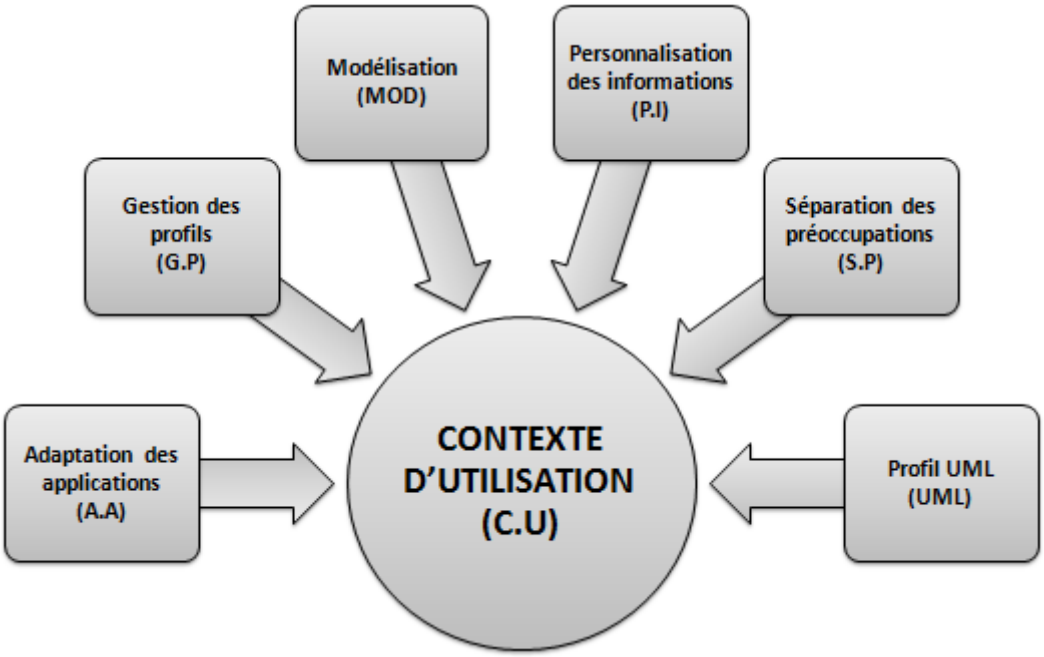


Figure 4. Thèmes de recherches relatives au contexte d'utilisation

La table 2 expose les résultats d'une recherche bibliographique faisant ressortir les principaux thèmes de recherches étudiés dans chaque travail voisin à notre étude. Il faut noter, ici, qu'un même travail peut inclure un seul thème de recherche comme il peut inclure plusieurs thèmes à la fois.

N.B : la colonne intitulée par le symbole de sommation Σ représente le nombre cumulé de thèmes étudiés par un seul travail voisin.

Travaux et/ou Etudes	Thèmes de recherches se basant sur le « Contexte d'utilisation»						Σ
	A.A	G.P	MOD	P.I	S.P	UML	
Système d'Accès Personnalisé à l'Information : Application au domaine médical [Abbas, 2008]		✓					**1**
MDD approach for the development of context-aware applications [Ayed, 2007]			✓		✓	✓	**3**
Personnalisation de l'information : aperçu de l'état de l'art et définition d'un modèle flexible de profils utilisateurs [Bouzeghoub, 2014]		✓		✓			**2**
L'adaptation dans les systèmes d'information sensibles au contexte d'utilisation: approche et modèles. Le projet SECAS : Simple Environment for Context Aware Systems [Chaari, 2005]	✓				✓		**2**
Adaptation d'applications pervasives dans des environnements multi-contextes [Chaari, 2007]	✓						**1**
A survey of context-aware mobile computing research [Chen, 2000]			✓				**1**
A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications [Dey, 2001]	✓						**1**
Aspect-Oriented Executable UML Models for Context-Aware Pervasive Applications [Fuentes, 2008]	✓					✓	**2**
Modèles de contexte pour l'adaptation à l'utilisateur dans des Systèmes d'Information Web Collaboratifs [Gensel 2008]			✓				**1**
An Architecture of Mobile Web 2.0 Context-aware Applications in Ubiquitous Web [Hsu, 2011]	✓						**1**
Extending UML to model Web 2.0-based context-aware applications [Hsu, 2012]			✓			✓	**2**
Visual modelling for Web 2.0 applications using model driven architecture approach [Hsu, 2013]						✓	**1**
Integrated Modelling of Context-Aware Adaptive Software Systems [Hussein, 2010]	✓		✓				**2**
Modèle contextuel pour la personnalisation			✓	✓			**2**

[Jrad, 2008]							
Adaptation Contextuelle et Personnalisée de l'Information de Conscience de Groupe au sein des Systèmes d'Information Coopératifs [Kirsch-Pinheiro, 2006]	✓		✓	✓	✓		**4**
Personnalisation de l'information et gestion des profils utilisateurs [Kostadinov, 2014]		✓		✓			**2**
Modeling Context Aware Features for Pervasive Computing [Mcheick, 2014]	✓		✓		✓		**3**
Personnalisation : enjeux, stratégies et outils [Mignot, 2014]				✓			**1**
A Model Driven Integration Architecture for Ontology-Based Context Modelling and CAA Development [Ou, 2006]			✓				**1**
Development of context-aware web services using the MDA approach [Seridi, 2012]			✓		✓		**2**
ContextUML: A UMLBased Modeling Language for Model-Driven Development of Context-Aware Web Services [Sheng, 2005]			✓				**1**
CMP: A UML Context Modeling Profile for Mobile Distributed Systems [Simons, 2007]			✓			✓	**2**
Model driven development of context aware software systems [Sindico, 2009]			✓			✓	**2**
Ontological User Profile Modeling for Context-Aware Application Personalization [Skillen, 2012]		✓		✓			**2**
Accès personnalisé à l'information : Approches et techniques [Tamine, 2005]		✓		✓			**2**
Towards Context Independence in Distributed context-aware applications by the Model Driven Approach [Vale, 2008]	✓		✓				**2**
An Architecture for the Development of Context-aware Services based on MDA and Ontologies [Vale, 2009]	✓		✓		✓		**3**
Using UML 2.0 and Profiles for Modelling Context Sensitive User Interfaces [Van Den Bergh, 2005]		✓	✓			✓	**3**

Table 2. Liste des travaux voisins avec les thèmes de recherches correspondants

Le nombre cumulé des thèmes, désigné par $\sum$, indique le degré de « voisinage » des travaux étudiés par rapport à notre domaine d'étude. C'est à partir de ce nombre qu'on peut juger si le travail voisin étudié est, par exemple, très proche ou moins proche de nos propres travaux. Ceci nous aidera à décider sur la pertinence des travaux étudiés. La table 3 montre une correspondance entre les valeurs possibles du nombre cumulé $\sum$ et leurs définitions respectives relativement au degré de « voisinage » des travaux étudiés. Il faut noter qu'il est évident lorsque $\sum$ est égal à 0, le travail correspondant est jugé complètement différent de

notre domaine d'étude. Par contre, si le nombre $\sum$ est égal à 6, alors le travail correspondant sera considéré comme fortement similaire ou identique.

$\sum$	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Degré de « voisinage »	**Différent**	**Voisin**	**Plus Voisin**		**Très Voisin**		**Similaire**

Table 3. Définition des degrés de « voisinage »

Dans la table 4, nous exposons quelques résultats statistiques concernant la répartition des thèmes de contextualisation par rapport aux travaux étudiés.

Code requête	Désignation des requêtes	Nombre	Taux %
R1	Nombre total des travaux	**28**	**100**
R2	Nombre des travaux incluant le thème « A.A »	**10**	**36**
R3	Nombre des travaux incluant le thème « G.P »	**06**	**21**
R4	Nombre des travaux incluant le thème « MOD »	**16**	**57**
R5	Nombre des travaux incluant le thème « P.I »	**07**	**25**
R6	Nombre des travaux incluant le thème « S.P »	**06**	**21**
R7	Nombre des travaux incluant le thème « UML »	**07**	**25**
R8	Nombre de travaux ayant cumulé un seul thème	**10**	**36**
R9	Nombre de travaux ayant cumulé deux thèmes	**13**	**46**
R10	Nombre de travaux ayant cumulé trois thèmes	**04**	**14**
R11	Nombre de travaux ayant cumulé quatre thèmes	**01**	**04**
R12	Nombre de travaux ayant cumulé cinq thèmes	**00**	**00**
R13	Nombre de travaux ayant cumulé six thèmes	**00**	**00**
R14	R2 OU R4	**21**	**75**
R15	Moyenne de thèmes par travail	(52/28) = **1.86**	

Table 4. Etat statistique des travaux voisins

3.1.3. Interprétation et Evaluation

En lisant les résultats présentés dans l'état statistique des travaux voisins étudiés (table 4), nous pouvons dégager quelques remarques sous forme d'interprétation faisant éclaircir des points de comparaison entre ces travaux. Ces remarques sont résumées comme suit :

1. Notons d'abords que la moyenne de thèmes par travail est égale à 1.86 (Requête R15). Ce chiffre montre que les travaux étudiés sont bien voisins à notre étude puisque chacun d'eux traite presque 2 thèmes à la fois parmi les principaux thèmes du domaine de la sensibilité au contexte.
2. Le thème le plus traité dans les travaux voisins étudiés est celui de la modélisation du contexte avec un nombre de 16 parmi 28 et un taux égal à 57 % (R4). Ceci étant logiquement évident parce que la majorité des travaux liés à la sensibilité au contexte démarrent par la représentation des éléments du contexte sous forme d'un modèle.
3. Les thèmes les moins traités dans les travaux voisins étudiés sont ceux relatifs à la gestion des profils et à la séparation des préoccupations avec le même nombre de travaux 6 parmi 28 et un taux égal à 21 % (respectivement R3 et R6).
4. Les travaux ayant traité un seul thème sont au nombre de 10 parmi 28 soit un taux de 36 % (R8). Ces travaux seront jugés juste « voisins ».
5. Les travaux jugés « plus voisins » sont ceux qui ont cumulé un total d'au moins 2 ou 3 thèmes étudiés. Le nombre total de ces travaux est égal à 17 avec un taux de 61 % (R9 et R10).
6. Parmi tous les travaux voisins étudiés, il existe un seul qui a totalisé un cumul de 4 thèmes (R11). Ce travail est considéré comme celui le « très voisin » à notre domaine d'étude puisqu'il n'y a pas de travaux qui ont cumulé 5 ou 6 thèmes (respectivement R12 et R13).
7. D'après le résultat de la requête R14 (21 parmi 28, soit 75 %), la majorité absolue des travaux menés dans le domaine de la contextualisation ont traité l'un ou l'autre des deux thèmes : adaptation des applications (A.A) ou modélisation du contexte d'utilisation (MOD).

Ci-après, nous allons exposer un classement des travaux voisins étudiés par groupes suivant les similitudes et les ressemblances des thèmes étudiés ou des notions proposées. Les groupes seront cités dans un ordre décroissant du nombre total des travaux liés à chaque thème. Au cours de ce classement, nous allons faire une évaluation de ces travaux en interprétant les résultats obtenus et en faisant ressortir d'éventuelles insuffisances et/ou limites.

1er rang (Groupe du thème « MOD ») : les travaux de ce groupe occupent le premier rang avec un total de 16 parmi 28. L'objectif de ces travaux est la modélisation du contexte d'utilisation ; c'est-à-dire la représentation du contexte à l'aide de notations claires et précises pour faciliter sa prise en charge par les applications particulières (ou systèmes). Nous remarquons facilement que plus de la moitié (57 %) des travaux étudiés portent sur l'abstraction du contexte pour construire un modèle contextuel facile à exploiter lors du développement d'applications sensibles au contexte ou lors des processus d'adaptation et de personnalisation. Ce qu'il faut reprocher à quelques uns de ces travaux c'est qu'ils proposent des nouveaux modèles de contexte avec modification du méta-modèle UML ; ceci posera

beaucoup de problèmes de compatibilité des notations UML lors de futures utilisations de ce modèle contextuel avec d'autres outils basés sur le méta-modèle initial de UML.

2ème rang (Groupe du thème « A.A) : ce groupe compte 10 travaux et un taux de 36 % (deuxième rang). Ces travaux cherchent à assurer l'adaptation des applications par rapport au contexte d'utilisation en proposant des méthodes et des outils pouvant assurer une certaine adaptation des applications. Mais la majeure partie de ces travaux considère l'application comme un produit fini sur lequel il faut appliquer des techniques (ou outils) pour l'adapter aux conditions spécifiques du contexte d'utilisation. Dans ce cas, il est indispensable de connaitre, toujours et d'avance, plusieurs paramètres essentiels comme le type d'information contextuelle à capturer, la source de cette information, le type de capteur à utiliser, etc. Ceci implique la nécessité d'intervention humaine et exclut, par conséquent, le caractère automatique du processus d'adaptation.

3ème rang (Groupe du thème « P.I ») : au 3ème rang (avec 7 travaux parmi 28, soit 25 %) se placent les travaux qui visent la personnalisation des informations suivant le contexte d'utilisation des situations particulières ou variables. Pour ces travaux, la personnalisation des informations se base sur la prise en compte des éléments suivants : les données personnelles, les préférences, l'historique de navigation, les caractéristiques de la tache courante, les filtres, etc. Ces éléments semblent importants pour le processus de personnalisation mais portent tous sur l'utilisateur et l'application. Nous souhaiterions de voir d'autres éléments relatifs à l'information elle-même qui peuvent la rendre plus spécialisée (en contenu et en présentation) et plus pertinente (en nombre et en qualité).

4ème rang (Groupe du thème « UML ») : avec le même résultat que le groupe précédent (7 travaux parmi 28 et un taux égal à 25 %), le groupe du thème « UML » se place au 4ème rang et comprend les travaux qui ont proposé des notations étendues (extensions) du langage UML, sous forme de profil, destinées à la prise en charge des propriétés contextuelles d'un système. La majorité de ce type de travaux se sont limités à fournir quelques extensions du langage UML sans pour autant se foncer dans la construction d'un profil UML complet. La preuve est que les auteurs de ces travaux ne parlent pas, dans leur majorité, de profil UML destiné au domaine de la contextualisation mais, plutôt, ils introduisent des notations étendues précises dont ils ont besoin pour l'accomplissement de leurs objectifs d'adaptation ou de personnalisation.

5ème rang (Groupe du thème « G.P ») : ce groupe renferme les travaux qui utilisent le contexte d'utilisation comme élément de base servant à la gestion des profils (principalement le profil utilisateur). Occupant le 6ème rang avec un total de 6 travaux parmi 28 et un taux de 21 %, ces travaux présentent les principales techniques et fonctions liées à la gestion des profils utilisateurs et l'exploitation de ces profils dans le processus d'accès à l'information. Ces techniques se rapportent sur la construction d'un profil, l'évolution d'un profil, l'utilisation des données d'un profil, la sécurité des informations, etc. Les profils utilisateurs proposés sont définis par des modèles obtenus après modélisation des différentes propriétés de l'utilisateur. Nous remarquons, ici, que tous les travaux incluant la gestion des profils convergent vers la

manipulation du profil « utilisateur », mais nous savons que l'utilisateur n'est pas le seul acteur principal dans la composition du contexte d'utilisation qui comprend aussi l'application et l'environnement.

6ème rang (Groupe du thème « S.P ») : ce groupe se place au 6ème rang avec un total de 6 travaux parmi 28 et un taux égal à 21 %. Les travaux concernés sont ceux qui ont utilisé la notion de séparation des préoccupations dans le domaine de la sensibilité au contexte. La séparation est appliquée, d'une part, entre les préoccupations fonctionnelles et les préoccupations non fonctionnelles d'un système et, d'autre part, entre les préoccupations contextuelles et celles propres au domaine d'application. L'objectif visé est de garantir un haut degré d'interopérabilité et de réutilisation lors de tout développement d'applications sensibles au contexte. L'importance de séparer les préoccupations contextuelles est de pouvoir étudier et modéliser toutes les contraintes du contexte d'utilisation dans une branche (ou module) indépendante des autres propriétés (métiers et techniques) de l'application. La majorité de ces travaux évoquent et insistent sur cette importance mais sans donner des solutions pratiques et fiables pour réaliser le processus de séparation.

3.2. Positionnement du problème

Après avoir fait un bilan bibliographique qui met en évidence les principaux travaux menés dans le domaine de la sensibilité au contexte, et après avoir eu une idée globale sur les thèmes étudiés ainsi que sur les notions proposées, nous voilà, maintenant, obligé de se situer et de se positionner par rapport à ces travaux voisins.

L'objectif de notre étude est de compléter les travaux existants (des autres auteurs) par des propositions qui peuvent contribuer à assurer l'adaptation contextuelle et la personnalisation pour les systèmes d'information ubiquitaires. Notre problématique s'articule sur l'étude des insuffisances et des points faibles des autres approches pour venir combler ces manques par des idées et des propositions concrètes et réalisables.

Il faut rappeler qu'évidemment notre travail de thèse soit une consolidation d'un ensemble de recherches qui ont été menés antérieurement.

3.2.1. Présentation de nos recherches antérieures

Avant de commencer la comparaison de nos travaux avec les autres, nous allons présenter une description résumée de ces travaux que nous avons menés depuis le début de nos recherches sur le domaine de la sensibilité au contexte. En effet, dans nos travaux [Benselim, 2009a], [Benselim, 2009b] et [Benselim, 2009c], nous avons présenté une nouvelle approche de développement qui peut prendre en compte tous les changements du contexte d'utilisation pendant le processus de développement d'une application et nous avons proposé une nouvelle vision de l'approche MDA (Model Driven Architecture). Ensuite, nous avons démontré qu'il est possible de séparer (ou isoler) les propriétés contextuelles d'un système et qu'il est possible de traiter et développer les contraintes contextuelles indépendamment des contraintes fonctionnelles propres (ou métiers) d'un système et des contraintes techniques de la plateforme choisie [Benselim, 2011a]. Cette séparation nous permet d'étudier toutes les informations contextuelles dans une branche à part et nous évite de refaire tout le processus de développement à cause des changements du contexte. Comme résultat, nous avons proposé un

modèle contextuel conforme au méta-modèle proposé et en utilisant le langage UML standard. Mais, malgré ce résultat, nous n'étions pas complètement satisfaits parce que les concepts de UML standard ne supportent pas tous les spécificités du contexte d'utilisation dans une forme appropriée et adéquate. Pour cela, nous avons décidé de chercher une meilleure solution pour représenter le contexte d'une façon explicite et en utilisant une notation spécifique. UML est un langage de modélisation orienté objet fourni par l'OMG (Object Management Group) et qui est utilisé pour la modélisation des différents aspects d'un système logiciel. UML possède plusieurs caractéristiques parmi lesquelles nous nous sommes intéressés aux mécanismes d'extensibilité (stéréotypes, contraintes et valeurs marquées) qui sont nécessaires pour l'introduction de nouveaux éléments UML. Ceux-ci sont utilisés pour la modélisation d'un domaine particulier comme le domaine de la sensibilité au contexte. Donc, UML peut être étendu afin de pouvoir modéliser les propriétés contextuelles qui influencent la situation courante d'un utilisateur. Cette extension est mise en œuvre par l'ajout de nouveaux éléments au langage unifié standard qui contribuent à la construction d'un profil UML complet pour le développement d'applications sensibles au contexte. C'est dans ce sens que nous avions décidé de se lancer dans la construction d'un profil UML destiné au domaine de la sensibilité au contexte, et dont nous espérions qu'il soit un outil de conception pratique qui facilitera la modélisation des applications sensibles au contexte. Dans nos premiers travaux sur le profil UML [Benselim, 2011b], [Benselim, 2012a] et [Benselim, 2012b], nous avons introduit la notion d'extension des notations et des concepts du langage standard UML pour faciliter la modélisation adéquate des contraintes spécifiques relatives au contexte d'utilisation, et nous avons proposé, à travers ces travaux, des principes d'extensibilité avec des syntaxes claires capables de spécialiser l'utilisation du langage UML pour un domaine particulier. L'article cité dans [Benselim, 2013] s'enfonce dans les détails du processus d'extension du langage UML en proposant des nouvelles notations UML étendues à partir de celles existantes dans la version UML standard. Il faut noter, ici, que les toutes les notations proposées concernent uniquement le diagramme de classes UML et portent sur ses concepts de base comme la classe, l'association, l'attribut, etc. Ces nouvelles notations sont créées suivant les principes des mécanismes d'extensibilité et se présentent sous formes de stéréotypes, contraintes et étiquettes. Ensuite, dans le travail [Benselim, 2017], nous avons développé encore notre profil UML pour engendrer plus de notations et plus de diagrammes UML. En effet, nous avons proposé des nouvelles notations UML relatives à quatre diagrammes UML différents (classes, cas d'utilisation, séquences et activités). Le choix de ces diagrammes n'était pas arbitraire mais parce que ce sont des diagrammes qui peuvent refléter les principales vues d'un système (fonctionnalités, vue statique, vue dynamique).

3.2.2. Comparaison avec les travaux voisins

Maintenant, nous allons procéder à une comparaison de nos travaux avec les autres. La fiabilité des résultats de toute comparaison exige l'application des mêmes critères pour les deux parties à comparer. De ce fait, nous avons dressé le même format de la table 2 incluant les mêmes thèmes de recherches, et nous avons obtenu les résultats mentionnés dans la table 5.

Nos travaux et/ou Etudes	Thèmes de recherches se basant sur le « Contexte d'utilisation»						Σ
	A.A	G.P	MOD	P.I	S.P	UML	
Contextual adaptation of ubiquitous information systems [Benselim, 2009a]	✓						**1**
Une approche pour le développement d'applications sensibles au contexte [Benselim, 2009b]	✓				✓		**2**
Développement d'applications sensibles au contexte en utilisant l'approche Model Driven Architecture [Benselim, 2009c]	✓		✓				**2**
Development of context-aware applications in ubiquitous information systems [Benselim, 2011a]	✓		✓		✓		**3**
Modelling context with extended UML [Benselim, 2011b]			✓	✓		✓	**3**
Extended UML for the Development of Context-Aware Applications [Benselim, 2012a]	✓			✓		✓	**3**
Modelling context with extended UML [Benselim, 2012b]			✓	✓		✓	**3**
Development of context-aware web services using the MDA approach [Seridi, 2012]			✓		✓		**2**
Extending UML Class Diagram Notation for the Development of Context-aware Applications [Benselim, 2013]	✓			✓		✓	**3**
Towards A UML Profile for Context-Awareness Domain [Benselim, 2017]			✓	✓		✓	**3**

Table 5. Liste de nos travaux antérieurs avec les thèmes de recherches correspondants

Dans la table 6, nous exposons un état comparatif comprenant des résultats statistiques obtenus après application des mêmes requêtes sur les deux parties à comparer (travaux voisins / nos travaux).

Code requête	Désignation des requêtes	Travaux voisins		Nos travaux	
		Nombre	Taux %	Nombre	Taux %
R1	Nombre total des travaux	**28**	**100**	**10**	**100**
R2	Nombre des travaux incluant le thème « A.A »	**10**	**36**	**06**	**60**
R3	Nombre des travaux incluant le thème « G.P »	**06**	**21**	**00**	**00**
R4	Nombre des travaux incluant le thème « MOD »	**16**	**57**	**06**	**60**
R5	Nombre des travaux incluant le thème « P.I »	**07**	**25**	**05**	**50**

R6	Nombre des travaux incluant le thème « S.P »	**06**	**21**	**03**	**30**
R7	Nombre des travaux incluant le thème « UML »	**07**	**25**	**05**	**50**
R8	Nombre de travaux ayant cumulé un seul thème	**10**	**36**	**01**	**10**
R9	Nombre de travaux ayant cumulé deux thèmes	**13**	**46**	**03**	**30**
R10	Nombre de travaux ayant cumulé trois thèmes	**04**	**14**	**06**	**60**
R11	Nombre de travaux ayant cumulé quatre thèmes	**01**	**04**	**00**	**00**
R12	Nombre de travaux ayant cumulé cinq thèmes	**00**	**00**	**00**	**00**
R13	Nombre de travaux ayant cumulé six thèmes	**00**	**00**	**00**	**00**
R14	R2 OU R4	**21**	**75**	**10**	**100**
R15	Moyenne de thèmes par travail	(52/28) = **1.86**		(25/10) = **2.5**	

Table 6. Etat statistique comparatif (travaux voisins / nos travaux)

Après lecture des résultats statistiques présentés dans la table 6, nous pouvons déduire quelques conclusions permettant de mieux connaitre la position de nos travaux par rapport aux autres travaux voisins. En effet, ces résultats montrent une comparaison claire concernant les thèmes étudiés, la pertinence des travaux, etc. Parmi ces conclusions, nous citons :

1. Pratiquement, tous nos travaux antérieurs sont bien liés au domaine de la sensibilité au contexte avec une moyenne de 2.5 thèmes par travail (R15). Cette moyenne est largement supérieure à celle des travaux voisins (1.86).
2. Les thèmes les plus traités dans nos travaux sont ceux relatifs à l'adaptation et à la modélisation avec un taux de 60 % chacun.
3. Le thème relatif à la gestion des profils utilisateurs n'a pas été étudié dans nos travaux précédents (R3).
4. La grande majorité de nos travaux antérieurs ont traité trois thèmes (R10) avec un taux de 60 % (6 travaux parmi 10).
5. Les résultats de la majorité des requêtes présentent une différence considérable. En effet, les taux des requêtes (R2, R4, R5, R6, R7, R10, R14 et R15) présentent des taux supérieurs pour nos travaux par rapport à ceux des autres travaux. Cela peut signifier, d'une part, que nos études cernent bien le domaine de la sensibilité au contexte (du côté quantitatif des thèmes) et d'autre part, que nous avons pu apporter qualitativement, à travers ces études, des idées complémentaires et des nouvelles propositions qui peuvent enrichir les recherches de ce domaine. Ces apports qui représentent les atouts et les valeurs ajoutées de cette thèse seront détaillés dans la partie « contributions ».

3.3. Critiques et suggestions

3.3.1. Critiques

Après avoir fait cette double comparaison qui nous a permis, d'une part, d'évaluer les travaux voisins les uns par rapport aux autres et, d'autre part, de pouvoir se positionner et se situer

relativement à ces travaux, nous avons jugé utile de mentionner les critiques et les insuffisances soulevées lors de cette étude d'analyse et de comparaison. C'est à partir de ces critiques que nous avons pu définir clairement les principales tendances de notre problématique ainsi que les objectifs à atteindre. Parmi ces critiques, nous citons :

- Le peu de travaux qui ont cité la notion de séparation des préoccupations contextuelles n'ont pas donné une définition claire de cette notion et n'ont pas proposé une solution précise (technique, outil, …) pour la réaliser concrètement.
- La majorité des travaux menés sur l'adaptation contextuelle des applications procèdent à une adaptation pendant l'exécution « run-time » ; c'est-à-dire qu'ils supposent l'application comme un produit fini qui nécessite un outil externe qu'il faut appliquer au cours de l'exécution pour la rendre adaptée à un certain contexte d'utilisation.
- Plusieurs travaux menés sur la modélisation du contexte portent des changements radicaux dans la structure du méta-modèle UML ; c'est-à-dire que ces travaux construisent des modèles de contexte avec modification du méta-modèle UML. Les modèles contextuels ainsi obtenus auront beaucoup de problèmes liés à la compatibilité des notations et à l'interopérabilité avec d'autres systèmes ou plateformes utilisant UML.
- Dans la majorité des travaux qui semblent proposer la construction de profils UML pour la sensibilité au contexte, cette construction se limite à la création de quelques extensions UML nécessaires aux processus d'adaptation ou de personnalisation.

3.3.2. Suggestions

Suivant les critiques décelées lors de l'étude des travaux voisins et qui ont été mentionnées ci-dessus, nous essayons dans ce qui suit de donner quelques suggestions susceptibles de combler les insuffisances remarquées et de résoudre les problèmes y afférents.

- Il faut définir un processus bien précis qui réalise clairement la séparation entre les préoccupations contextuelles et les autres préoccupations du système (métiers et techniques). Nous proposons, ici, d'utiliser les principes du processus unifié à deux chemins 2TUP (2 Tracks Unified Process) et les projeter sur un processus à trois chemins (contextuel, métiers, technique).
- Pour une bonne adaptation des applications au contexte, il faut essayer d'introduire les contraintes contextuelles au cours de la phase de conception « design-time » au lieu de le faire pendant l'exécution « run-time ». Nous proposons, ici, d'utiliser toutes les opportunités de l'approche MDA (Model Driven Architecture) pour pouvoir prendre en charge les contraintes contextuelles (sous forme d'un modèle) pendant la phase de conception d'une application.
- Pour garder une certaine harmonie des différents systèmes et plateformes, et pour garantir un haut degré de réutilisabilité, nous suggérons de construire les nouveaux modèles contextuels conformément au méta-modèle standard UML et sans apporter de modifications sur ce méta-modèle. Ceci permettra de maintenir la compatibilité des notations de ces nouveaux modèles avec celles des autres systèmes utilisant le méta-modèle UML non modifié.

- Les profils UML sont définis comme étant une spécialisation des notations standards UML pour un domaine particulier. D'autre part, la personnalisation des informations et le développement d'applications sensibles au contexte s'appuient fortement sur ces profils. D'où, la nécessité de construire un profil UML complet offrant tous les concepts nécessaires à la construction de modèles contextuels adéquats.

3.4. Motivations et objectifs

Pour mieux motiver et argumenter nos choix relatifs aux objectifs visés et aux contributions proposées, nous nous sommes posé quelques questions de types Quoi, Pourquoi et Comment dont nous devrions expliquer dans ce qui suit :

Question 1 **: Quelle est la relation entre 'Adaptation' et 'Personnalisation' ?**

Pour monter la relation qui existe entre l'adaptation et la personnalisation, nous citons quelques points de comparaison comme suit :

- La personnalisation représente une partie du processus d'adaptation.
- L'adaptation est liée aux systèmes et aux applications ; tandis que la personnalisation est liée aux utilisateurs et à l'information
- Une personnalisation implique toujours une adaptation, mais une adaptation n'implique pas toujours une personnalisation (Figure 5).

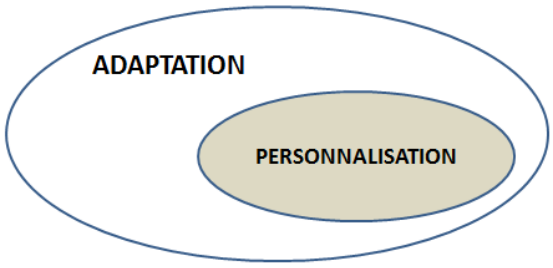


Figure 5. Relation entre adaptation et personnalisation.

En général, toute adaptation est liée à un changement ; c'est-à-dire que tout changement nécessite une adaptation. Ces changements concernent :

- L'utilisateur
- L'application
- L'environnement

L'adaptation cherche toujours à faire évoluer le système. Un système d'information doit évoluer pour s'adapter aux différents changements (nouvelle activité, nouveaux acteurs, nouvelles données, etc.)

L'adaptation des systèmes d'information ubiquitaires comprend :

- Adaptation suivant la mobilité des utilisateurs,
- Adaptation suivant l'hétérogénéité des matériels utilisés et des informations,
- Adaptation suivant la distribution des systèmes et des sources de données,
- Adaptation suivant l'environnement (maison, bureau, extérieur, intérieur, etc.)

La personnalisation dans les systèmes d'information ubiquitaires comprend :

- Personnalisation du profil utilisateur
- Personnalisation de l'information

Question 2 **: Quand fait-on appel à l'adaptation et à la personnalisation ?**

L'adaptation des applications est la clé de satisfaction des utilisateurs. En effet, elle permet de tailler l'exécution de ces applications afin de les rendre « sur mesure » par rapport aux exigences des utilisateurs. Le besoin à l'adaptation et à la personnalisation a énormément augmenté surtout avec l'émergence de l'informatique ubiquitaire. Ce type de science a offert de larges opportunités aux utilisateurs pour accéder et exploiter les applications informatiques distantes et hétérogènes. L'évolution technologique propose une large gamme d'outils mobiles et intelligents pour accéder aux informations n'importe où et n'importe quand. Cette diversité des applications et des outils utilisés rend délicat l'organisation et la pertinence des résultats retournés. D'où, la nécessité de modifier ou de donner une certaine forme aux applications pour qu'elles soient capables de réagir convenablement avec la variation des caractéristiques entrant dans la composition du contexte général d'une situation d'exécution.
Et c'est là l'utilité de faire appel aux notions d'adaptation et de personnalisation pour que les applications puissent retourner des résultats pertinents et conformes avec les requêtes des utilisateurs.

Question 3 **: Est-ce que les systèmes d'information ubiquitaires ont vraiment besoin de l'adaptation contextuelle et la personnalisation ?**

L'importance de l'adaptation contextuelle et de la personnalisation pour les systèmes d'information ubiquitaires réside dans la couverture et la prise en charge de tous les changements contextuels qui peuvent survenir lors de l'exécution d'une application. Vu les contraintes très spécifiques qui caractérisent ce type de systèmes d'information, l'adaptation et la personnalisation semblent indispensables pour la survie et l'efficacité de ces systèmes. En effet, l'adaptation anticipe l'apparition d'une contrainte contextuelle spécifique et la prend en charge lors du processus de modélisation pour pouvoir ajuster (en nombre, en forme et en qualité) les résultats retournés lors de l'exécution.

Question 4 **: Comment justifier le choix d'un profil UML pour assurer cette adaptation ?**

Actuellement, le langage UML est le standard des langages de modélisation parce qu'il présente plusieurs avantages comme la clarté des notations, la facilité d'utilisation, etc. Ce langage propose, aussi, la possibilité de s'auto-étendre pour devenir adaptable aux spécificités des domaines particuliers. Les notations standards du langage UML sont générales mais extensibles. Le but de cette extension est d'avoir de nouveaux concepts capables de modéliser les contraintes contextuelles spécifiques sous des formes plus adéquates et appropriées. L'ensemble des nouveaux éléments de modélisation sont regroupés dans un profil UML personnalisé pour la modélisation des applications sensibles au contexte. Etant une partie de la notion d'adaptation, la personnalisation conduit toujours à contribuer à cette notion. Par conséquent, la personnalisation du langage UML peut participer à l'adaptation des applications œuvrant dans les environnements ubiquitaires.

Question 5 **: Comment atteindre les objectifs de cette thèse ?**

A partir du titre de la thèse « Adaptation contextuelle et personnalisation pour les systèmes d'information ubiquitaires », nous pouvons ressortir les objectifs à atteindre. En effet, deux objectifs majeurs sont apparents : adaptation contextuelle et personnalisation. Le premier objectif consiste à trouver un moyen ou un outil qui peut assurer un certain degré d'adaptation (suivant le contexte d'utilisation) des applications disponibles dans les systèmes d'information ubiquitaires. Pour cela, nous avons pensé à proposer une démarche (sous forme d'une version améliorée de l'approche MDA) qui peut prendre en charge toutes les contraintes contextuelles dans son processus de développement d'applications. Ainsi, les applications obtenues auront la capacité de fournir des informations plus pertinentes qui varient et s'ajustent selon les variations subies par les différentes situations possibles d'un utilisateur. Le deuxième objectif visé par cette thèse est la personnalisation. Cette dernière est principalement liée aux utilisateurs et aux informations qu'ils manipulent. La personnalisation consiste, aussi, à spécialiser un phénomène ou un concept pour le rendre adéquat avec les cas particuliers d'un système. Dans ce sens, nous avons réfléchi à personnaliser les notations du langage UML pour que nous puissions l'utiliser dans la modélisation du domaine de la sensibilité au contexte (Context-Aware Domain).

D'autre part, et afin de bien réaliser les idées proposées (adaptation et personnalisation), nous avons jugé très utile d'étudier toutes les contraintes contextuelles indépendamment des autres contraintes du système. Ceci nous a amené à isoler les éléments contextuels d'un système dans une branche d'étude séparée. Cela peut assurer une étude profonde et efficace du caractère contextuel de ce système.

La table 7 résume les contributions de cette thèse par rapport au titre prédéfini.

<table>
<tr><th>Contributions de la thèse</th><th>Lien</th><th>Titre de la thèse (objectif)</th></tr>
<tr><td rowspan="2">- Séparation des préoccupations contextuelles
- Personnalisation du langage UML
- Adaptation contextuelle des applications</td><td>(assure)➔</td><td rowspan="2">Adaptation contextuelle et personnalisation pour les systèmes d'information ubiquitaires</td></tr>
<tr><td>←(requiert)</td></tr>
</table>

Table 7. Correspondance entre contributions et titre de la thèse.

Pour atteindre les objectifs visés, nous avons essayé d'approfondir nos recherches autour d'une idée de base qui résume un processus d'adaptation composé de trois volets (Figure 6).

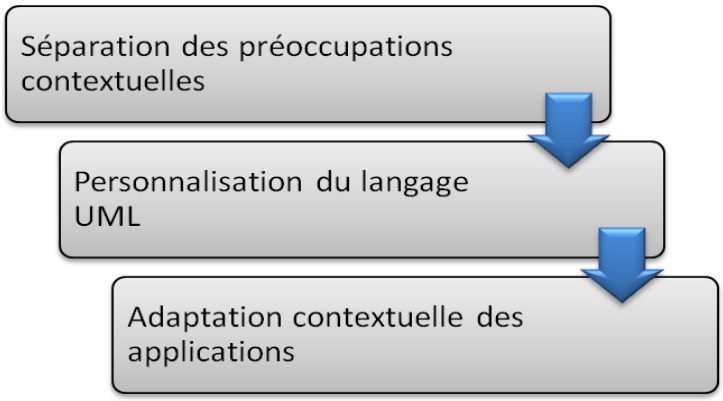


Figure 6. Idée générale du processus d'adaptation proposé.

Conclusion

Dans ce chapitre, nous avons exposé une étude comparative dans laquelle nous avons réalisé un bilan de quelques travaux voisins en les comparants entre eux suivant les thèmes traités et les notions proposées. Cette première comparaison nous a permis d'évaluer la pertinence de ces travaux par rapport au domaine de la sensibilité au contexte et, aussi, de faire ressortir quelques insuffisances et limites pour lesquelles nous avons essayé de suggérer des solutions. Une deuxième comparaison a été faite entre ces travaux et ceux que nous avons menés antérieurement. Ceci étant nécessaire pour mieux positionner notre problématique par rapport à ces travaux et pour mettre en évidence les valeurs ajoutées apportées à travers cette thèse.

PARTIE II

CONTRIBUTIONS

CHAPITRE 4

SÉPARATION DES PRÉOCCUPATIONS CONTEXTUELLES

Introduction

Le principe de la séparation des préoccupations offre une grande souplesse dans le domaine de développement des logiciels. En effet, cette séparation permet, comme dans le cas de la MDA, de définir un modèle métier indépendant de toute plateforme technique et de générer automatiquement du code vers la plateforme choisie. Ainsi, le processus de développement ne sera pas mis en cause à cause de l'évolution possible des supports technologiques.

L'apport de l'informatique ubiquitaire demande une étude spécifique et minutieuse des changements subis par le contexte d'utilisation d'une situation. Les éléments de ce contexte doivent être traités indépendamment des autres contraintes parce qu'ils présentent des caractéristiques spécifiques et très variables (mobilité, hétérogénéité, …).

Notre étude propose de créer un nouveau type de préoccupations à séparer des autres (métiers et techniques). Ce nouveau type sera consacré pour l'étude des contraintes contextuelles. Ainsi, nous aurons un processus de développement qui s'articule autour de trois branches indépendantes en entrée (branche fonctionnelle, branche technologique et branche contextuelle). A ce propos, nous avons introduit la notion du processus 3TUP (3 Track Unified Process) qui peut compléter, surtout dans le domaine du contexte-aware, au processus de développement en « Y » de la MDA.

4.1. Rappels sur le processus de développement en « Y »

Une meilleure description de l'architecture du processus de développement en «Y» a été détaillée par Roques et Valée dans le livre « UML en action » [Roques, 2003] et dont nous présentons quelques notions élémentaires.

4.1.1. Processus de développement logiciel

Un processus de développement représente une séquence d'étapes ordonnées qui coopèrent entre elles dans le but de créer un nouveau système logiciel ou bien de mettre à jour un

système existant. L'objectif d'un tel processus est la production de meilleurs logiciels qui satisferont pleinement les besoins des utilisateurs.

4.1.2. Processus Unifié (UP: Unified Process)

Un processus unifié est un type de processus de développement logiciel qui est construit sur UML. Il est itératif et incrémental, centré sur l'architecture, conduit par les cas d'utilisation et piloté par les risques.

La gestion d'un processus UP est organisée suivant les quatre (4) phases suivantes:

- Préétude
- Elaboration
- Construction
- Transition.

Un processus UP doit répondre aux caractéristiques suivantes :

- Il est itératif et incrémental
- Il est piloté par les risques
- Il est construit autour de la création et de la maintenance d'un modèle
- Il est orienté composant
- Il est orienté utilisateur.

Les activités de développement d'un processus UP sont définies par cinq (5) disciplines fondamentales qui décrivent :

- La capture des besoins
- L'analyse
- La conception
- L'implémentation
- Le test et le déploiement.

4.1.3. Le processus 2TUP (2 Track Unified Process)

Un processus 2TUP est un processus unifié qui doit satisfaire toutes les contraintes de changements subies par les systèmes d'information de l'entreprise tout en contrôlant les capacités d'évolution et de correction de ces systèmes. D'après l'architecture en «Y», tous les changements imposés à un système informatique peuvent être classés en deux types de contraintes: les contraintes fonctionnelles et les contraintes techniques (Figure 7). C'est-à-dire que le processus peut suivre deux chemins de développement indépendants. D'où l'appellation de « 2 Track » qui signifie littéralement « 2 chemins ».

Figure 7. Le système d'information soumis à deux types de contraintes.

Le principe du processus 2TUP consiste à traiter toute évolution imposée au système d'information sur deux axes distincts et parallèles qui sont l'axe fonctionnel (préoccupations métiers) et l'axe technique (préoccupations technologiques). Ces deux axes qui forment deux branches d'entrée seront fusionnés dans une branche commune qui représente la phase de réalisation du processus global. Cette fusion conduit à l'obtention d'un processus de développement en forme de Y (Figure 8) [Roques, 2003].

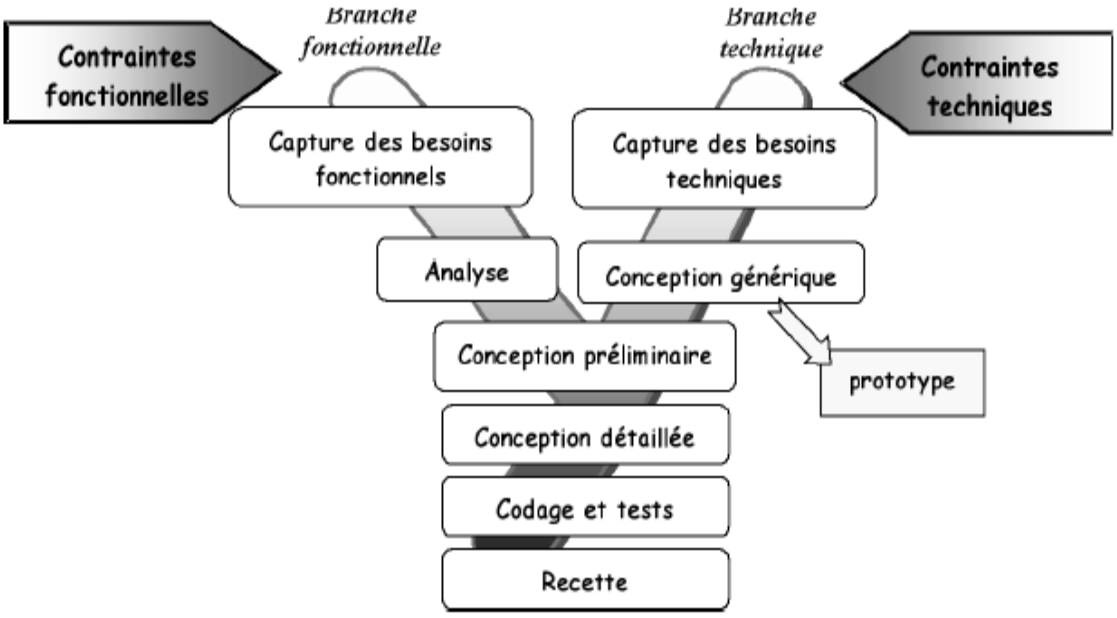


Figure 8. Le processus de développement en « Y » [Roques, 2003].

4.2. Processus de développement proposé

4.2.1. Le processus 3TUP (3 Track Unified Process)

Le processus 3TUP peut être défini comme suit [Benselim, 2011a]:

Un processus 3TUP est un processus UP qui prend en charge toutes les contraintes de changement subies par les systèmes d'information ubiquitaires tout en assurant le contrôle des capacités d'évolution et de correction de ces systèmes. La particularité majeure des contraintes imposées à un système d'information ubiquitaire réside dans le changement continuel du contexte d'utilisation de la situation courante d'un utilisateur.

Les éléments du contexte d'utilisation (préférences de l'utilisateur, localisation, temps, ressources et environnement) ne peuvent appartenir ni aux spécifications fonctionnelles ni aux spécifications technologiques.

4.2.2. Architecture du processus de développement 3TUP

Comme illustré dans la figure 9, le processus de développement proposé s'articule autour de quatre (4) phases:

- Une branche technique
- Une branche fonctionnelle
- Une branche contextuelle

- Une phase de réalisation

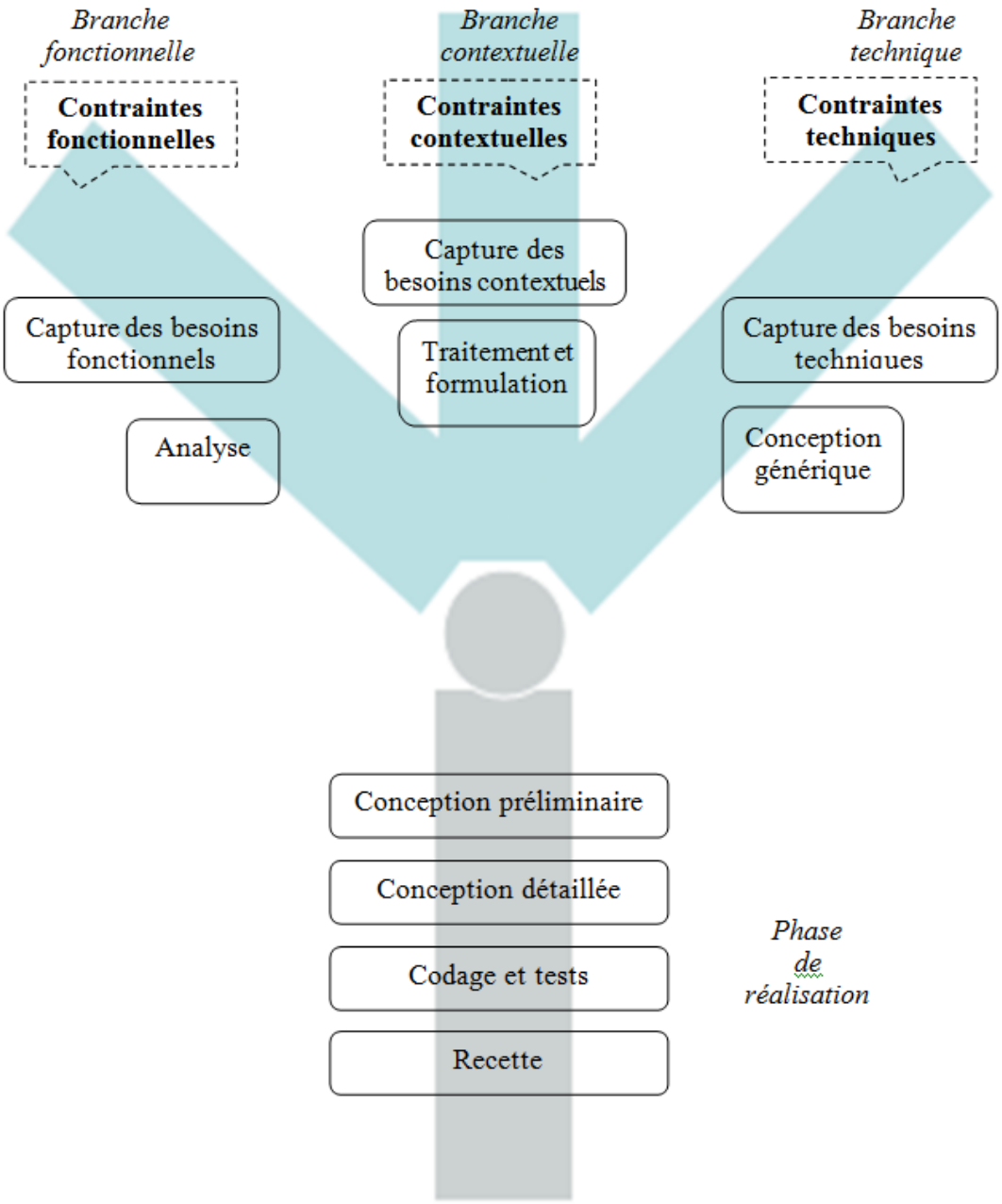


Figure 9. Architecture du nouveau processus de développement.

4.2.3. Pourquoi séparer les préoccupations contextuelles ?

En effet, pour répondre à cette question, il faut démontrer que chacun des éléments contextuels ne peut être traité ni par la branche fonctionnelle ni par la branche technique. Par exemple, dans l'informatique ubiquitaire, les notions de temps et d'espace sont très importantes et elles couvrent l'ubiquité (n'importe où et n'importe quand) des systèmes d'information et des utilisateurs de plus en plus nomades. Aussi, le changement des objets environnants peut affecter directement l'état de la situation courante d'un utilisateur. D'autre

part, chaque utilisateur peut présenter des préférences particulières quant au contenu, à la présentation ou à l'affichage des informations demandées.

La figure 10 présente une architecture générale décrivant le principe de base de la séparation des préoccupations. Tout type de préoccupations (métiers, techniques ou contextuelles) sera exprimé à l'aide d'un modèle bien précis (respectivement : PIM, PSM ou CM). La construction de chacun de ces modèles est réalisée conformément au méta-modèle correspondant. Comme résultat, nous obtenons des modèles séparés et indépendants qui représente, chacun, un type précis de préoccupations.

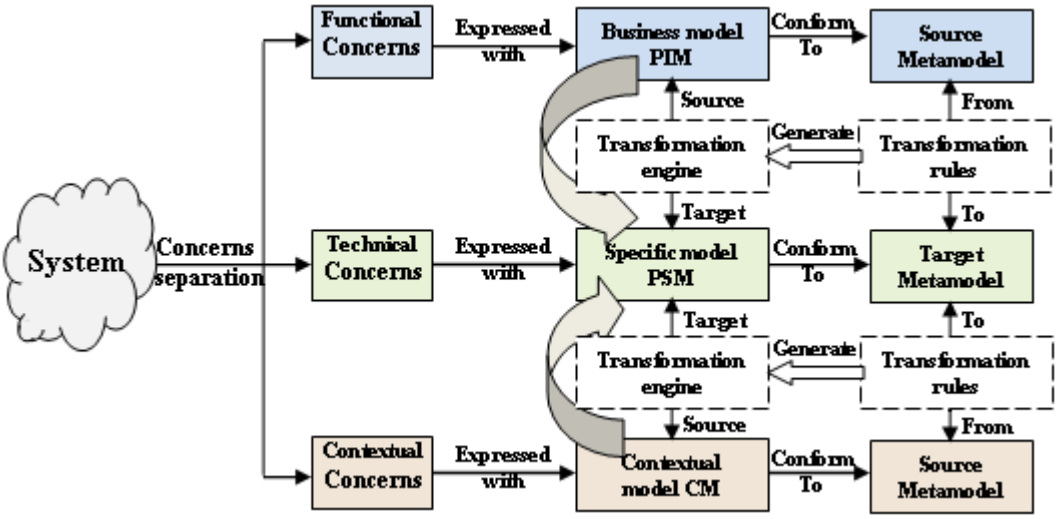


Figure 10. Séparation des préoccupations [Benselim, 2011a].

A partir des définitions précédentes de la notion de « contexte » au chapitre 2, nous pouvons extraire quelques éléments qui constituent les composants principaux du contexte d'utilisation. Ces éléments sont : l'utilisateur, la localisation, le temps et l'environnement. Dans le domaine de l'informatique ubiquitaire, ces éléments présentent une caractéristique commune qui est le changement et la variation, et c'est sur ce point que se base notre démonstration.

4.2.3.1. L'utilisateur

L'entité « utilisateur » inclut l'identité, les activités et le profil (préférences). Ces éléments changent continuellement et sont spécifiques à chaque utilisateur. Celui-ci a besoin de personnaliser ses activités ou bien les informations demandées en présentant quelques préférences ou des souhaits particuliers. Donc, l'entité « utilisateur » ne peut pas être une contrainte fonctionnelle du système ni une contrainte technique.

4.2.3.2. La localisation

La localisation désigne l'endroit et l'emplacement d'une situation particulière. L'endroit est changeable suivant l'espace d'exécution de cette situation et suivant la mobilité des utilisateurs et des ressources. A partir de la localisation, le système peut fournir des valeurs

par défaut de quelques paramètres comme la langue, la monnaie, unités de mesure,…et qui ne sont pas spécifiés par l'utilisateur. La variation incessante de l'endroit fait de lui un élément qui ne pas appartenir aux fonctionnalités du système ni, bien sûr, aux spécifications technologiques de la plateforme à utiliser.

4.2.3.3. Le temps

L'élément contextuel « Temps » peut inclure le temps, la date, les saisons…etc. Cet élément peut fournir plusieurs indications sur le climat, la température, la pression…et qui représentent des informations pertinentes pour les utilisateurs nomades. Le temps est une variable primordiale dans les systèmes d'information ubiquitaires, donc sa prise en charge ne doit pas être effectuée ni parmi les contraintes fonctionnelles ni parmi les contraintes techniques.

4.2.3.4. L'environnement

Cet élément possède une large signification en renfermant les matériels, les ressources, les réseaux, les personnes avoisinantes ainsi que tout objet capable d'agir ou d'interagir avec l'utilisateur, l'application ou bien les deux à la fois. Chacun de ces éléments de l'environnement peut affecter et modifier l'exécution d'une situation suivant ses propres changements.

L'informatique ubiquitaire offre plusieurs possibilités d'accès à l'information comme l'utilisation aléatoire, en temps et en espace, des dispositifs mobiles par des utilisateurs de plus en plus exigeants et nomades. A cause de la mobilité des utilisateurs, non seulement le temps et l'endroit qui changent constamment mais aussi les objets environnants (y compris les personnes avoisinantes). Donc l'utilisateur se trouve confronté à une situation différente, c'est-à-dire dans un nouveau contexte d'utilisation et dont il doit s'y adapter.

La modification ou le changement de l'un des éléments du contexte d'utilisation implique la transformation de ce contexte vers un nouvel état de la situation courante de l'utilisateur. La prise en charge de ces changements ne doit pas remettre en cause tout le processus de développement, mais seulement la partie qui traite ce type de contraintes.
D'où l'utilité de regrouper toutes les spécifications contextuelles qui représente le contexte d'utilisation dans une nouvelle branche indépendante appelée : **branche contextuelle**.
Cette séparation de la branche contextuelle permettra de développer les préoccupations contextuelles d'un système d'une façon indépendante, d'une part, des fonctionnalités propres de ce système et d'autre part, des contraintes technologiques de la plateforme choisie.

Le processus 3TUP propose un cycle de développement à trois branches. Ce cycle met en évidence la séparation entre les préoccupations techniques, les préoccupations fonctionnelles et les préoccupations contextuelles.

Ceci nous amène à répartir les contraintes subies par un système d'information ubiquitaire en trois types distincts (Figure 11):

- Contraintes fonctionnelles

- Contraintes techniques
- Contraintes contextuelles

C'est-à-dire que le processus peut suivre trois chemins de développement indépendants. D'où l'appellation de « **3 Track** » qui signifie littéralement « 3 chemins ».

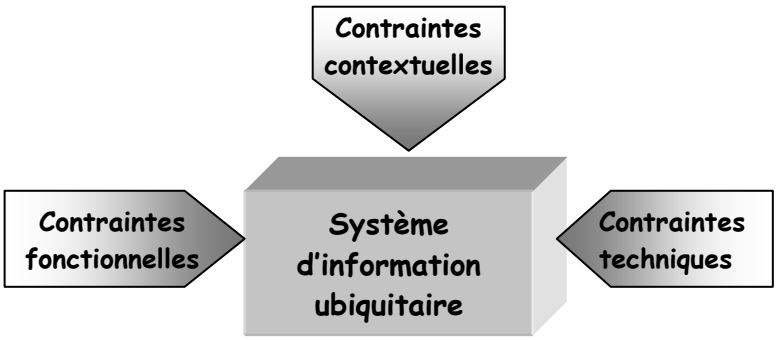


Figure 11. Le système d'information soumis à trois types de contraintes [Benselim, 2011a].

4.2.4. Description de la branche contextuelle

La branche contextuelle regroupe l'ensemble des contraintes relatives au contexte d'utilisation d'un système ou d'une application. Cette branche est très cruciale puisque l'informatique ubiquitaire offre de grandes opportunités pour l'utilisation nomade des dispositifs et des services. Ceci aura un impact direct sur le contexte d'utilisation d'une situation quelconque.

Le développement de la branche contextuelle passe par deux étapes majeures : capturer les besoins contextuels et ensuite les interpréter.

a. Capture des besoins contextuels : cette étape consiste à capturer et collecter tous les éléments susceptibles d'influencer la situation courante d'un utilisateur. Ces éléments représentent les composants du contexte d'utilisation incluant l'utilisateur, l'application et l'environnement.

b. Traitement et formulation des données : les éléments capturés sont traités et transformés en données conceptuelles. Cette étape sert à préparer les données contextuelles pour être exploitables lors de l'étape de la conception préliminaire (1ère étape de la phase de réalisation).

4.2.5. Avantages de la séparation des préoccupations contextuelles

- Eliminer la dépendance qui peut exister entre les fonctionnalités propres du système et les contraintes contextuelles.
- Prise en compte du contexte d'utilisation indépendamment de l'évolution technologique des plateformes choisies.
- La possibilité de développement en parallèle de plusieurs axes à la fois.
- L'évolution de la technologie ne peut pas affecter tout le processus de développement.
- Protéger les fonctionnalités du système de l'évolution et des changements subis par le contexte d'utilisation.

- Augmenter le degré de maintenabilité.
- Minimiser les risques de remise en cause de tout le processus de développement.
- Intégration facile des nouvelles contraintes contextuelles qui peuvent apparaître.

Conclusion

Un système d'information est caractérisé par un ensemble de propriétés et contraintes bien définies. Ces propriétés définissent la structure de ce système et son mode fonctionnement. L'étude de ces propriétés comme étant un tout ou un ensemble risque de soulever des taches complexes et fastidieuses à cause des ambiguïtés qui peuvent avoir lieu telles que la redondance d'information, l'interférence des missions, le type de contraintes, etc. Afin de remédier à ces insuffisances, les concepteurs ont proposé le principe de la séparation des préoccupations pour pouvoir étudier distinctement chaque type de contraintes auxquelles sont soumis les systèmes d'informations. Selon le processus de développement en « Y », il existe deux types de préoccupations à séparer, à savoir : les préoccupations métiers et les préoccupations techniques. Dans ce chapitre, nous avons proposé la séparation d'un nouveau type de préoccupations qui comporte seulement les contraintes contextuelles. L'importance de la séparation des préoccupations contextuelles est issue de l'environnement ubiquitaire (mobilité, hétérogénéité distribution) dans lequel les utilisateurs utilisent leurs applications et dispositifs. Dans cette proposition, nous avons introduit un nouveau processus de développement appelé : processus 3TUP (3 Track Unified Process) qui se charge de séparer trois types de préoccupations : métiers, techniques et contextuelles.

CHAPITRE 5

PERSONNALISATION DU LANGAGE UML

Introduction

Dans le chapitre précédent, nous avons démontré qu'il est possible d'isoler les contraintes contextuelles d'un système sous formes de préoccupations séparées. La modélisation de ces préoccupations spécifiques au domaine du context-aware requiert des concepts et des notations bien particulières capables d'assurer une représentation adéquate et appropriée d'un système. Pour cela, nous avons réfléchi à personnaliser le langage UML en étendant ses notations standards pour qu'elles deviennent en mesure de modéliser les préoccupations spécifiques du contexte d'utilisation.

5.1. Pourquoi personnaliser UML ?

UML est devenu un langage standard dans le domaine de la modélisation orientée objet. Ce langage a été proposé par l'OMG pour modéliser les différents aspects des systèmes logiciels. UML se distingue par plusieurs caractéristiques et propriétés qui font de lui, actuellement, l'un des meilleurs langages de modélisation. Parmi ces caractéristiques, nous nous sommes intéressés aux mécanismes d'extensibilité (stéréotypes, contraintes et étiquettes) qui fournissent la possibilité d'introduire de nouveaux éléments de modélisation. Ces nouveaux éléments sont créés, par extension, à partir de concepts UML existants et seront destinés à modéliser les systèmes spéciaux ayant des propriétés spécifiques. Ici, nous admettant que le langage UML a été spécialisé et personnalisé pour être utilisé par un domaine particulier.

L'extension du langage UML permettra de mettre à disposition un ensemble de nouvelles notations étendues UML qui soient plus ou moins adaptées à garantir une modélisation claire et appropriée des aspects spécifiques que présentent quelques domaines. Parmi ces domaines, nous mentionnons celui qui étudie la sensibilité au contexte « context-awareness » dont le noyau est le contexte d'utilisation. La particularité de ce domaine découle de la spécificité des contraintes qu'il présente surtout qu'il opère dans des environnements ubiquitaires caractérisés par la mobilité des utilisateurs, l'hétérogénéité des informations, la distribution des systèmes et des sources d'informations, etc. Ces contraintes particulières ne sont pas toutes modélisables, de façon précise et adéquate, par les concepts existants dans la version

standard du langage UML. Par exemple, les développeurs de logiciels trouvent du mal à modéliser les changements fréquents et continus que subit un utilisateur lors de l'exécution d'une application parce que ces changements influencent directement la situation courante de cet utilisateur et la rendent très variable d'un instant à un autre, d'un endroit à un autre, d'un matériel à un autre, etc.

Etant donné ces limites de modélisation de la version standard de UML, et vu que le langage UML propose des moyens d'extensibilité permettant de couvrir ces insuffisances, nous avons décidé d'étendre les notations de UML pour être capable de modéliser convenablement les contraintes spécifiques du domaine de la sensibilité au contexte. Toutes ces notations étendues seront regroupées, sous forme d'un paquetage, dans un profil UML destiné spécialement au développement d'applications sensibles au contexte. Ci-après, nous allons exposer les détails du profil UML proposé, et ce, en expliquant sa structure et ses composants. Ces détails portent essentiellement sur la description du paquetage de ce profil, la création des stéréotypes, la création des contraintes et la création des étiquettes (tagged values).

5.2. Architecture du profil UML proposé

Le contexte d'utilisation peut être représenté à l'aide de notations spécifiques UML. D'où l'utilité d'étendre ce langage de modélisation en ajoutant de nouveaux éléments qui peuvent représenter les contraintes contextuelles de façon appropriée. Chacune de ces contraintes doit être représentée par une notation adéquate du langage UML. Pour cela, nous proposons une extension de la version standard du langage UML pour construire un profil UML qui est composé de stéréotypes, contraintes et étiquettes et qui soit capable d'adapter les concepts de UML au domaine de la sensibilité au contexte. Un stéréotype permet de définir et d'ajouter une nouvelle sémantique à un élément existant du méta-modèle UML. Les étiquettes sont toujours attachées au stéréotype correspondant et leur rôle est de spécifier les attributs du stéréotype crée. Les contraintes permettent de définir des conditions appliquées sur les nouveaux éléments pour fixer les champs et les limites de leur utilisation. Notre profil UML proposé est un diagramme de structure qui décrit les mécanismes d'extensibilité du langage UML en spécifiant les nouveaux stéréotypes, étiquettes et contraintes. Toutes les nouvelles notations des diagrammes UML sont obtenues après extension des éléments existants dans le méta-modèle UML [Benselim, 2017].

Les diagrammes UML concernés par l'extension de notations sont quatre :

- Les diagrammes de classes
- Les diagrammes de cas d'utilisation
- Les diagrammes de séquences
- Les diagrammes d'activités

Ces quatre diagrammes n'ont pas été choisis arbitrairement mais ce choix est justifié par le fait que ces diagrammes soient représentatifs des différentes vues d'un système. Ainsi, les fonctionnalités d'un système peuvent être représentées par un diagramme de cas d'utilisation, la vue statique est représentée par un diagramme de classes et la vue dynamique peut être

illustrée par un diagramme de séquences ou par un diagramme d'activités. Aussi, les quatre diagrammes choisis par notre étude offrent les principaux outils qui peuvent couvrir les besoins de développement d'une application lors des différentes étapes d'analyse, de conception et d'implémentation.

Le profil UML proposé « UML Context-Aware » peut être représenté par un paquetage englobant d'autres sous profils (correspondants aux différents diagrammes UML) qui étendent le méta-modèle UML.

De même, le paquetage global du profil UML proposé est composé de quatre principaux paquetages représentant chacun un profil UML précis : ClassUML, UsecaseUML, SequenceUML ou ActivityUML (Figure 12).

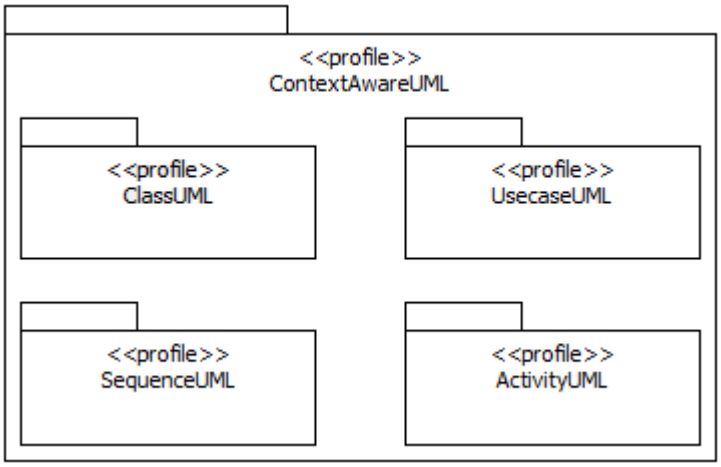


Figure 12. Paquetage global du profil proposé « UML Context-Aware » [Benselim, 2017].

Chacun de ces paquetages hérite sa structure et ses propriétés à partir de la méta-classe UML '*Package*' dans lequel seront regroupées toutes les notations étendues du diagramme UML correspondant (Table 8).

Nom du profil	**Méta-modèle de base**	**Elément du méta-modèle (méta-classe)**	**Description**
« ClassUML profile »	Méta-modèle UML	'Package'	Étend et personnalise les notations du diagramme de classes UML afin de supporter la modélisation sensible au contexte

« UsecaseUML profile »	Méta-modèle UML	'Package'	Étend et personnalise les notations du diagramme de cas d'utilisation UML afin de supporter la modélisation sensible au contexte
« SequenceUML profile »	Méta-modèle UML	'Package'	Étend et personnalise les notations du diagramme de séquences UML afin de supporter la modélisation sensible au contexte
« ActivityUML profile »	Méta-modèle UML	'Package'	Étend et personnalise les notations du diagramme d'activités UML afin de supporter la modélisation sensible au contexte

Table 8. Description des sous profils inclus dans le profil UML proposé.

Dans la table 9, nous présentons les principaux éléments du méta-modèle UML qui vont être étendus lors de la création du profil UML proposé « UML Context-Aware ».

Nom du diagramme UML	**Stéréotype proposé**	**Elément du méta-modèle UML (méta-classe)**
Diagramme de classes	ContextClass	'Class'
	ContextAssociation	'Association'
	ContextAttribute	'Attribute'
	ContextConstraint	'Constraint'
Diagramme de cas d'utilisation	ContextActor	'Actor'
	ContextUseCase	'Use case'
	ContextDependency	'Relationships' (use, extend, include)
Diagramme de séquences	ContextObject	'Object'
	ContextLifeLine	'LifeLine'
	ContextMessage	'Message'
Diagramme d'activités	ContextActivity	'Activity'
	ContextTransition	'Transition'

Table 9. Liste des éléments du méta-modèle UML à étendre.

Contrairement aux notations du langage UML standard, les extensions proposées permettent de modéliser adéquatement les concepts de tout système context-aware. Les entités de ce type de systèmes qui possèdent des propriétés et des contraintes ne pouvant pas être représentées par les notations standard doivent être modélisées à l'aide des stéréotypes proposées. Les caractéristiques spécifiques comme les conditions, les limites ou la sémantique seront prises en charge par les contraintes et les étiquettes attachées aux nouveaux stéréotypes.

Généralement, tous les concepts UML peuvent être stéréotypés (étendus et personnalisés).
Il faut noter que pour les approches dirigées par les modèles (par exemple la MDA), la création d'un profil UML suppose que le langage standard UML soit étendu au niveau méta-modèle (niveau M2). Par conséquent, toutes les extensions proposées sont réalisées au niveau méta-modèle (M2) ; ensuite, les notations des diagrammes (au niveau modèle : M1) sont construits conformément aux concepts du méta-modèle correspondant.

Ci-après, nous allons décrire les quatre profils UML proposés et nous exposerons les principaux concepts qui personnalisent les notations du langage UML pour les diagrammes de classes, de cas d'utilisation, de séquences et d'activités. Pour la clarté de ce travail, nous allons détailler tous les mécanismes d'extensibilité proposés pour chacun des quatre diagrammes, et ce, suivant le plan suivant :

- Description du paquetage du profil
- Description des stéréotypes
- Description des contraintes
- Description des étiquettes

Tout d'abords, il faut rappeler une brève définition des mécanismes d'extensibilité utilisés lors de la description du profil UML proposé.

- Les stéréotypes permettent de personnaliser les notations du langage UML dans le but de créer de nouveaux éléments de modélisation spécifiques à un domaine particulier. Ils définissent comment les méta-classes UML existantes peuvent être étendues (par augmentation de la sémantique) dans un profil.

- Les contraintes représentent les conditions et les restrictions qui règlementent l'utilisation des nouveaux concepts (stéréotypes) et elles sont toujours attachées à ces stéréotypes. Ces contraintes mettent en évidence les principales différences entre les stéréotypes proposés et les concepts existants du langage standard UML.
  Généralement, une contrainte peut être exprimée de plusieurs façons telles que :
    - Langage naturel (anglais, français, etc.)
    - Langage de programmation (Java, C++, etc.)
    - Notations mathématiques et logiques (AND, OR, +, =, etc.)
    - Diagrammes graphiques (UML, etc.)
    - Langage OCL (Object Constraint Language)

- Les étiquettes sont toujours attachées aux stéréotypes et elles sont utilisées pour spécifier les attributs de ces stéréotypes. Elles sont considérées comme méta-attributs des méta-classes. Chaque étiquette peut être définie par une paire d'information : un nom d'attribut et sa valeur. D'où la syntaxe générale d'une étiquette :

<Nom_Etiquette> : <Nom_Attribut> = <Valeur_Attribut>

5.3. Description du profil « ClassUML » : Diagrammes de classes

5.3.1. Paquetage du profil « ClassUML »

La figure 13 présente une vue générale de l'extension proposée et montre comment les méta-classes UML 'Class' et 'Association' sont étendues pour donner naissance à de nouveaux éléments UML pouvant jouer un rôle spécifique dans la modélisation sensible au contexte.

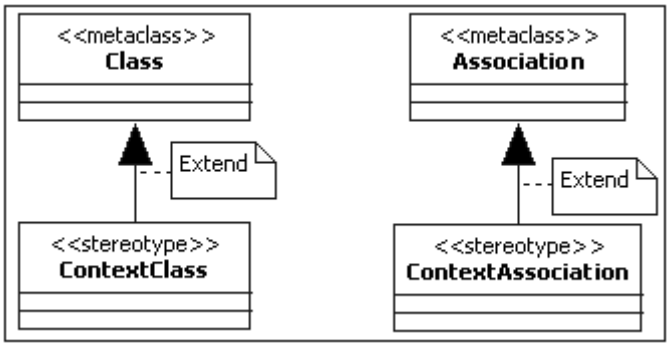


Figure 13. Paquetage du profil « ClassUML » [Benselim, 2013].

Le stéréotype "ContextClass" est une nouvelle méta-classe qui a été obtenue après extension de l'élément existant "Class" appartenant au méta-modèle UML. Ce stéréotype peut être associé à d'autres méta-classes et son rôle se résume à représenter le contexte d'utilisation d'une situation quelconque. "ContextClass" est défini par un ensemble d'éléments "ContextElement" qui représentent toutes les contraintes contextuelles. Ces éléments contextuels "ContextElement" peuvent avoir des propriétés très variables à cause de la mobilité des utilisateurs et à cause de la distribution des applications. Pour assurer une bonne représentation du contexte lors du développement d'applications distantes, nous proposons la création de nouveaux stéréotypes pouvant prendre en charge tous les composants de ce contexte et pouvant aider à construire un modèle contextuel approprié.

5.3.2. Stéréotypes du profil « ClassUML »

Dans la figure 14, nous montrons quelques stéréotypes qui sont associés au stéréotype "ContextElement" et qui définissent les contraintes spécifiques liées au contexte d'une situation. "ContextRelevancy" indique le degré de pertinence d'un élément, c'est-à-dire le niveau d'importance que présente un élément contextuel par rapport au contexte général. Le type d'un élément contextuel "ContextType" est déterminé par l'une des entités suivantes : l'utilisateur, l'application, l'environnement ou le comportement [Benselim, 2013].

Par conséquent, chaque élément contextuel sera modélisé par l'un des stéréotypes suivants issus de la spécialisation du stéréotype "ContextType" :

- "UserContext" : lorsque l'information est directement liée à l'utilisateur,
- "ApplicationContext" : si l'information provient de l'application,
- "EnvironmentContext" : si l'information est fournie par les objets environnants (autre que l'utilisateur et l'application),

- "BehaviorContext": représente l'aspect comportemental de l'information lorsqu'elle est issue de l'interaction entre les trois entités: l'utilisateur, l'application et l'environnement.

"ContextAttribute" est un stéréotype qui permet de représenter les propriétés de chaque classe englobant un élément contextuel et de définir toutes les valeurs qui peuvent être attachées aux instances de cette classe.

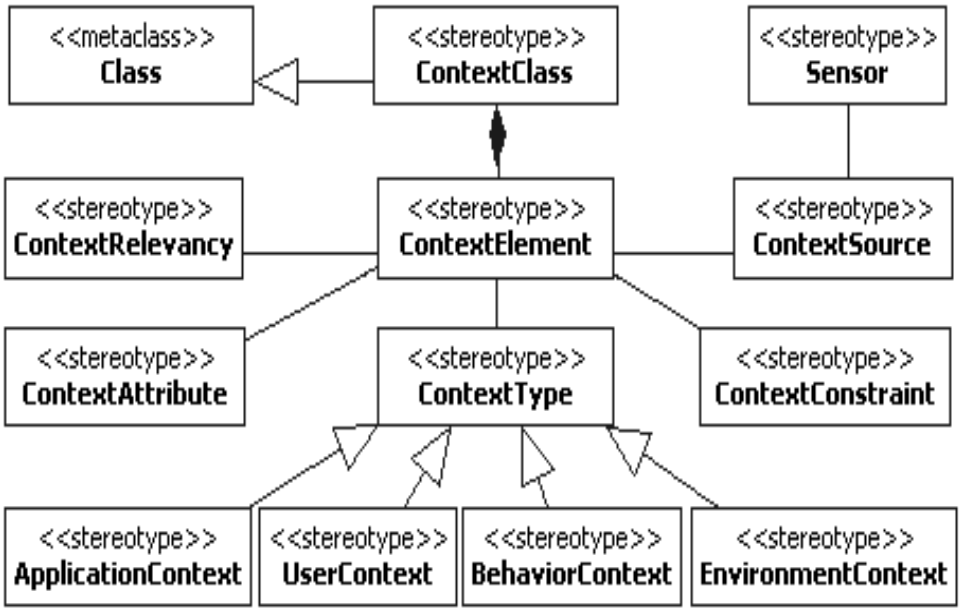


Figure 14. Les stéréotypes proposés de la méta-classe "Class".

Les stéréotypes "ContextConstraint" définissent les conditions et les restrictions appliquées sur une classe et sont utilisés pour modéliser les limites d'utilisation ou les champs d'application de cette classe. La source d'information d'un élément contextuel est représentée à l'aide du stéréotype "ContextSource" qui permet de modéliser le matériel ou le logiciel spécifique utilisé pour la capture de l'information contextuelle requise. "ContextSource" assure le traitement et l'acheminement de toute information acquise via le capteur "Sensor".

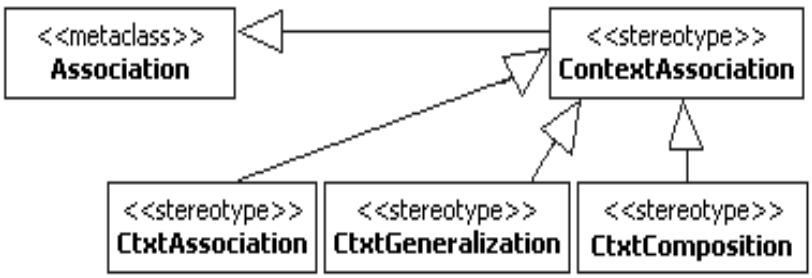


Figure 15. Les stéréotypes proposés de la méta-classe "Association".

De son côté aussi, la méta-classe UML "Association" peut être stéréotypée et étendue en "ContextAssociation" afin de définir les nouvelles relations qui associent les stéréotypes "Class" nouvellement créés (Figure 15).

Relation	Classe de base	Classes associées
IsSpecializedTo	CtxtGeneralization	Class, ContextClass
IsComposedBy	CtxtComposition	ContextClass, ContextElement
HasRelevancy	CtxtAssociation	ContextElement, ContextRelevancy
HasAttribute	CtxtAssociation	ContextElement, ContextAttribute
HasType	CtxtAssociation	ContextElement, ContextType
HasConstraint	CtxtAssociation	ContextElement, ContextConstraint
HasSource	CtxtAssociation	ContextElement, ContextSource
IsProvidedFrom	CtxtAssociation	ContextElement, ContextSource
CapturedBy	CtxtAssociation	ContextSource, Sensor
IsSpecializedTo	CtxtGeneralization	ContextType, ApplicationContext
IsSpecializedTo	CtxtGeneralization	ContextType, UserContext
IsSpecializedTo	CtxtGeneralization	ContextType, BehaviorContext
IsSpecializedTo	CtxtGeneralization	ContextType, EnvironmentContext

Table 10. Liste des relations pouvant associer les stéréotypes de "Class".

Chaque stéréotype "ContextAssociation" doit être spécialisé pour pouvoir fournir des relations spécifiques telles que : "CtxtAssociation", "CtxtGeneralization" ou "CtxtComposition". Ces stéréotypes d'association, représentant respectivement une simple association, une généralisation (ou spécialisation) ou une composition, assurent le lien entre deux stéréotypes "Class".
Dans la table 10, nous présentons une brève description des associations proposées qui peuvent relier les stéréotypes de la méta-classe "Class".

5.3.3. Contraintes du profil « ClassUML »

Les contraintes attachées aux stéréotypes définis permettent d'étendre la sémantique du langage UML. Elles participent à la spécification des restrictions qui seront considérées comme des conditions d'utilisation lors du processus de modélisation.

La table 11 décrit, en langage naturel, quelques contraintes attachées aux stéréotypes proposés.

Stéréotype	Description de la contrainte attachée
ContextClass	Doit être étendu à partir de la méta-classe UML "Class"
ContextElement	Doit être associé au stéréotype "ContextClass" par relation de composition
ContextElement	Doit être associé aux stéréotypes "ContextType", "ContextRelevancy", "ContextAttribute", "ContextConstraint" et "ContextSource" par simple association
ContextRelevancy	Doit être associé au stéréotype "ContextElement" par simple association
Sensor	Doit être associé au stéréotype "ContextSource" par simple association
ApplicationContext	Doit être associé au stéréotype "ContextType" par relation de généralisation
UserContext	Doit être associé au stéréotype "ContextType" par relation de généralisation
BehaviorContext	Doit être associé au stéréotype "ContextType" par relation de généralisation
EnvironmentContext	Doit être associé au stéréotype "ContextType" par relation de généralisation

Table 11. Les contraintes attachées aux stéréotypes proposés « ClassUML ».

Comme indiqué précédemment, les contraintes peuvent être exprimées en utilisant plusieurs outils : langage naturel, langage UML, langage OCL, etc. Ici, nous donnons un exemple de contrainte exprimée en langage OCL. Supposons que nous voulons limiter l'intervalle de valeurs du champ de la pertinence (relevancy) d'un élément contextuel, et que les valeurs minimales et maximales (MinValue et MaxValue) de cet intervalle sont prédéfinies.

Dans ce cas, la contrainte correspondante sera attachée au stéréotype "ContextElement" et aura le code OCL suivant :

```
Context ContextClass inv:
   Self.ContextElement->forAll(e : ContextElement |
         e.relevancy <= self.MaxValue
      and e.relevancy >= self.MinValue)
```

5.3.4. Etiquettes du profil « ClassUML »

La table 12 présente quelques exemples d'étiquettes attachées à différents stéréotypes. En effet, pour chaque stéréotype nous devons déterminer les étiquettes correspondantes en

spécifiant le nom d'un attribut et les valeurs possibles de cet attribut. Par exemple, un utilisateur peut présenter une certaine préférence pour un mode d'affichage précis des données retournées. Afin de prendre en charge ce besoin d'utilisateur, le stéréotype <<UserContext>> a été doté d'un attribut 'Favorite_Data_Display' dont les valeurs possibles sont : "Text", "Sound", "Picture", "Video", "Graphics". Cette liste de valeurs peut satisfaire les préférences particulières formulées par l'utilisateur lorsqu'il s'agit, pour cet exemple, de choisir ou de modifier le mode d'affichage des résultats.

Nom du diagramme UML	**Stéréotype d'attachement**	**Nom de l'attribut**	**Valeurs de l'attribut**
Diagramme de classes	UserContext	Favorite_Data_Display	"Text", "Sound", "Picture", "Video", "Graphics"
	Sensor	Type_Of_Sensor	"Hardware", "Software"
		Source_Of_Information	"Local", "Distant"
	ContextElement	Type_Of_Element	"Basic", "Optional"

Table 12. Les étiquettes attachées aux stéréotypes proposés « ClassUML ».

5.4. Description du profil « UseCaseUML » : Diagrammes de cas d'utilisation

5.4.1. Paquetage du profil « UseCaseUML »

Pour les diagrammes de cas d'utilisation, l'extension des notations UML existantes est utile pour créer de nouveaux concepts qui soient plus adaptés à modéliser les fonctionnalités spécifiques et les scénarios possibles d'un système opérant dans des environnements sensibles au contexte. Le paquetage du profil « cas d'utilisation » comprend toutes les extensions des classes de base liées aux diagrammes de cas d'utilisation (Figure 16).

Le stéréotype <<ContextActor>> est obtenu par extension de la méta-classe UML "Actor" en personnalisant sa syntaxe et sa sémantique. Cette personnalisation permet de spécialiser le rôle du concept "Actor" pour le rendre capable à modéliser convenablement les types particuliers d'utilisateurs et entités comme les utilisateurs nomades, les voyageurs, les systèmes distribués, les appareils mobiles, etc. Les principales méta-classes "Actor", "UseCase" et "Dependency" sont respectivement étendues à l'aide des stéréotypes <<ContextActor>>, <<ContextUseCase>> et <<ContextDependency>>.

La différence entre les concepts UML standard et nouveaux réside dans la possibilité de pouvoir attacher des contraintes et des étiquettes aux concepts créés (stéréotypes). Les contraintes sont utilisées pour fixer les limites et les restrictions suivant les spécifications particulières du contexte d'utilisation. Les étiquettes montrent les nouvelles propriétés (attributs) de chaque stéréotype.

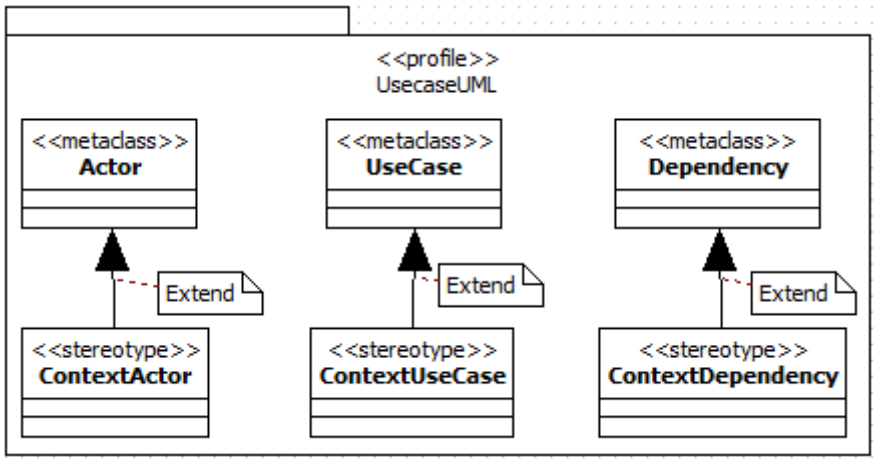


Figure 16. Paquetage du profil « UseCaseUML » [Benselim, 2017].

5.4.2. Stéréotypes du profil « UseCaseUML »

Dans les diagrammes de cas d'utilisation, le stéréotype <<ContextActor>> est utilisé pour modéliser les entités qui présentent des contraintes contextuelles et ayant des propriétés variables. Ces entités peuvent être des utilisateurs, des systèmes, des matériels, des logiciels, etc. <<ContextUseCase>> est un stéréotype qui modélise les fonctionnalités particulières d'un système caractérisées par des actions contextuelles et des taches variables dans le temps et dans l'espace. Le stéréotype <<ContextDependency>> modélise les relations spécifiques qui relient les stéréotypes <<ContextUseCase>> (Table 13).

Nom diagramme UML	**Stéréotype**	**Méta-classe UML**	**Description**
Cas d'utilisation	ContextActor	Actor	Représente les rôles joués par les utilisateurs nomades ou autres systèmes et entités mobiles
	ContextUseCase	UseCase	Spécifie un ensemble d'actions contextuelles ou de taches exécutées par un système mobile
	ContextDependency	Dependency	C'est une relation qui indique qu'un élément de modélisation a besoin (ou utilise) un autre élément tel que *ContextActor ou ContextUseCase*

Table 13. Les stéréotypes proposés du profil « UseCaseUML ».

Afin de modéliser un utilisateur nomade de façon appropriée, nous avons besoin de quelques notations spécifiques qui peuvent représenter explicitement les propriétés particulières de cet utilisateur. Ces propriétés particulières influencent l'exécution d'une application en modifiant continuellement le contexte d'utilisation des situations. Elles portent, par exemple, sur l'état de l'utilisateur (position assise, position debout, en marchant, en dormant, en voyageant), la localisation actuelle (domicile, bureau, hôtel, aéroport), le matériel utilisé (PC de bureau, PC

portable, téléphone mobile, PDA), les personnes avoisinants, les ressources disponibles, les réseaux existants, etc. La prise en considération de ces propriétés se fait en les insérant sous formes d'étiquettes (attributs) attachées au stéréotype <<ContextActor>>.

5.4.3. Contraintes du profil « UseCaseUML »

Les contraintes du profil « UseCaseUML » sont définies par des conditions claires qui montrent comment les stéréotypes proposés (<<ContextActor>>, <<ContextUseCase>>, <<ContextDependency>>) sont utilisés lors du processus de modélisation (Table 14).

Nom diagramme UML	**Stéréotype**	**Description de la contrainte attachée**
Cas d'utilisation	ContextActor	- Doit être étendu à partir de la méta-classe UML "Actor" - Doit être associé au stéréotype *ContextUseCase* en utilisant le stéréotype *ContextDependency*
	ContextUseCase	- Doit être étendu à partir de la méta-classe UML "UseCase" - Doit être associé à un autre stéréotype *ContextUseCase* en utilisant *ContextDependency*
	ContextDependency	- Doit être étendu à partir de la méta-classe UML "Dependency" - Relier deux stéréotypes *ContextUseCase* - Montrer le type de dépendance (use, extend, include) qui peut exister entre les stéréotypes *ContextUseCase*

Table 14. Les contraintes attachées aux stéréotypes proposés « UseCaseUML ».

La figure 17 présente graphiquement les contraintes proposées du profil « cas d'utilisation » et le type de relation entre les stéréotypes ainsi que les multiplicités correspondantes.

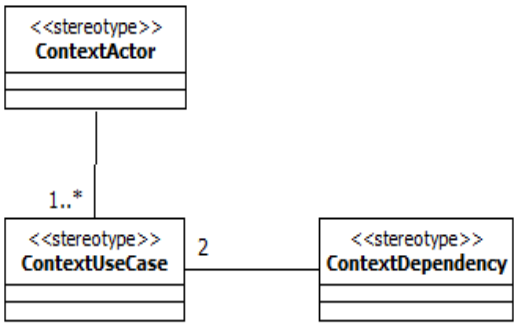


Figure 17. Représentation graphique des contraintes du profil « UseCaseUML ».

En appliquant ces contraintes, nous fixons à '2' le nombre d'instances <<ContextUseCase>> pouvant être associées par une relation <<ContextDependency>> et nous autorisons que la même instance <<ContextActor>> peut déclencher une ou plusieurs instances de cas d'utilisation <<ContextUseCase>>.

5.4.4. Etiquettes du profil « UseCaseUML »

La mobilité des utilisateurs et la grande diversité des systèmes (distribution et hétérogénéité) rendent fastidieux et très difficile le processus de modélisation des entités concernées. Le but d'un diagramme de cas d'utilisation est d'assurer la représentation adéquate de toutes les entités et les fonctionnalités d'un système sous formes d'Acteurs et de Cas d'utilisation; mais ce but ne sera pas atteint convenablement lorsque le système opère dans des environnements ubiquitaires où le contexte d'utilisation est constamment variable d'une situation à une autre.
Pour cela, le profil « UseCaseUML » propose des étiquettes (attributs des stéréotypes) qui peuvent couvrir et prendre en charge les propriétés particulières des systèmes sensibles au contexte. Par exemple, le changement de la position actuelle (de son domicile à son bureau de travail ou bien de son bureau vers son hôtel, etc.) d'un utilisateur va affecter directement le contexte de la situation courante parce que chaque lieu possède ses propres spécificités et paramètres comme la disponibilité de réseaux, le type de matériel utilisé, etc.

Nom du diagramme UML	**Stéréotype d'attachement**	**Nom de l'attribut**	**Valeurs de l'attribut**
Cas d'utilisation	ContextActor	Actor_Type	"Human", "Software", "Hardware"
		State_Of_User	"Walking", "Sitting", "Standing", "Traveling"
		Actual_Location	"Home", "Office", "Hotel"
		Spoken_Language	"Arabic", "French", "English
		Used_Money	"Dollar", "Euro", "Dinar"
		Used_Device	"PC", "Laptop", "Mobile", "PDA"
	ContextUseCase	Usecase_Type	"Local", "Distant"
		Usecase_Sharing	"Individual", "Common"

Table 15. Les étiquettes attachées aux stéréotypes proposés « UseCaseUML ».

Pour ce cas, les changements de situations contextuelles sont prises en considération par la présence de l'attribut 'Actual_Location' attaché au stéréotype <<ContextActor>> pour indiquer, à tout moment, l'endroit actuel de l'utilisateur. Ceci permettra de mettre à jour les paramètres de l'application pour être mieux adaptée aux changements continus du contexte d'utilisation. Dans la table 15, nous identifions une liste d'étiquettes du profil « UseCaseUML » sous formes d'attributs et leurs valeurs correspondantes.

La figure 18 illustre l'utilisation des concepts proposés dans la modélisation des utilisateurs nomades.

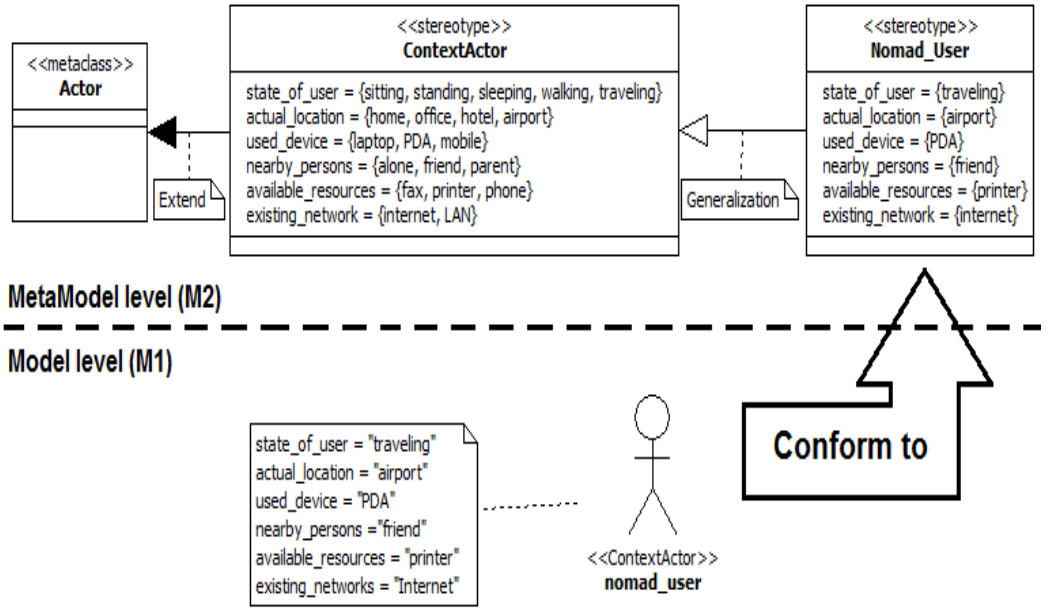


Figure 18. Exemple de modélisation graphique d'un utilisateur nomade [Benselim, 2017].

5.5. Description du profil « SequenceUML » : Diagrammes de séquences

5.5.1. Paquetage du profil « SequenceUML »

Les diagrammes de séquences permettent de représenter la succession d'interaction d'un cas d'utilisation (ou d'un scénario) dans un ordre chronologique. Ces diagrammes illustrent l'interaction des objets en mettant l'accent sur les messages émis et reçus par ces objets. Puisque les objets sont des instances de classes et puisque nous avons proposé l'utilisation de nouvelles classes personnalisées au domaine sensible au contexte (les stéréotypes <<ContextClass>>), alors nous sommes obligés, ici, de travailler avec le concept <<ContextObject>> comme étant une instance du stéréotype <<ContextClass>>.

La figure 19 montre comment les méta-classes "Object", "LifeLine" et "Message" sont respectivement étendues par les stéréotypes <<ContextObject>>, <<ContextLifeLine>> et <<ContextMessage>>. Ces stéréotypes créés ajoutent un nouveau sens et une juste sémantique aux méta-classes UML existantes en spécifiant l'objectif exact pour lequel elles vont être utilisées [Benselim, 2017].

Les nouveaux stéréotypes sont capables de modéliser n'importe quel objet qui varie suivant le contexte d'utilisation d'une situation. Les caractéristiques variables d'un objet sont prises en charge par les contraintes et les étiquettes qui seront attachées au stéréotype <<ContextObject>> correspondant. Ceci permettra une meilleure personnalisation des notations des diagrammes de séquences UML pour le domaine sensible au contexte et permettra, aussi, une manière de modélisation explicite et précise des facteurs contextuels continuellement variables.

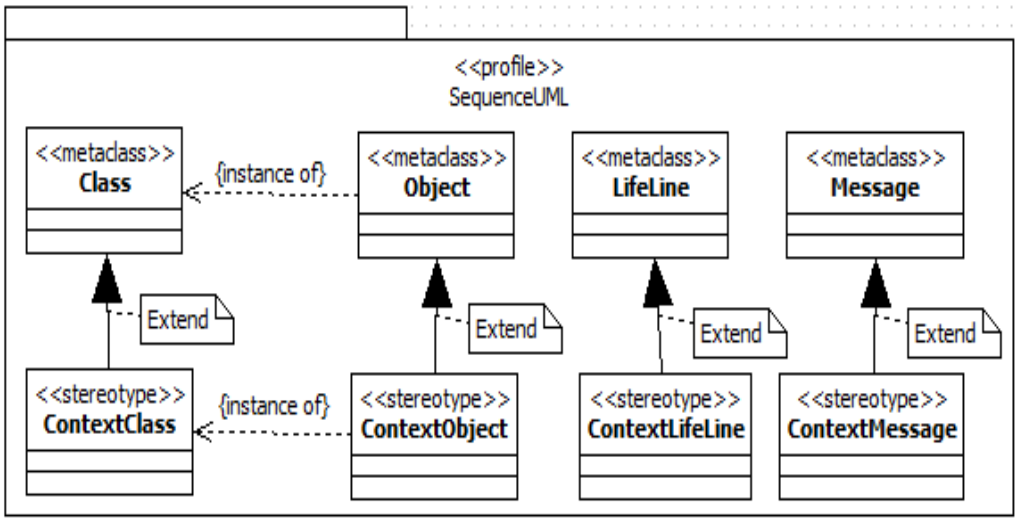


Figure 19. Paquetage du profil « SequenceUML ».

5.5.2. Stéréotypes du profil « SequenceUML »

Dans le profil « SequenceUML », nous proposons les stéréotypes <<ContextObject>>, <<ContextLifeLine>> et <<ContextMessage>> issus respectivement par extension des méta-classes UML "Object", "LifeLine" et "Message" (Table 16).

La communication ou l'échange de message entre deux stéréotypes <<ContextObject>> est caractérisée par l'émergence de contraintes spécifiques provenant de ces stéréotypes. Ceci peut affecter le déroulement de l'interaction à cause des éventuels changements qui peuvent survenir pendant l'exécution de l'application. Ici, nous introduisons la notion d' « interaction contextuelle ». Une interaction contextuelle est une communication entre deux objets <<ContextObject>> échangeant des messages <<ContextMessage>> et susceptible d'être influencée par le changement du contexte d'utilisation.

Nom diagramme UML	**Stéréotype**	**Méta-classe UML**	**Description**
Diagramme de séquences	ContextObject	Object	Représente un participant unique (ContextObject ou son instance) dans une interaction contextuelle
	ContextLifeLine	LifeLine	- Doit être associé au stéréotype

			ContextObject - Représente la période de vie au cours de laquelle un ContextObject peut participer à une ou plusieurs interactions contextuelles
	ContextMessage	Message	Définit un type de communication particulière (échange d'information) entre deux ContextLifeLine correspondant à deux ContextObject.

Table 16. Les stéréotypes proposés du profil « SequenceUML ».

5.5.3. Contraintes du profil « SequenceUML »

Les contraintes du profil « SequenceUML » sont définies par des restrictions qui organisent et réglementent l'utilisation des stéréotypes proposés (<<ContextObject>>, <<ContextLifeLine>>, <<ContextMessage>>) pendant la modélisation (Table 17).

Nom diagramme UML	**Stéréotype**	**Description de la contrainte attachée**
Diagramme de sequences	ContextObject	- Doit être étendu à partir de la méta-classe UML "Object" - Peut interagir avec un autre stéréotype *ContextObject* à l'aide d'un stéréotype *ContextMessage*
	ContextLifeLine	- Doit être étendu à partir de la méta-classe UML "LifeLine" - Met en évidence chronologiquement les points d'envoi et de réception du stéréotype *ContextMessage*
	ContextMessage	- Doit être étendu à partir de la méta-classe UML"Message" - Son point de départ doit être le stereotype *ContextLifeLine* (correspondant au stéréotype source *ContextObject*) - Son point d'arrivée doit être un autre *ContextLifeLine* (correspondant au stéréotype cible *ContextObject*)

Table 17. Les contraintes attachées aux stéréotypes proposés « SequenceUML ».

Aussi, les contraintes liées au profil « SequenceUML » peuvent être représentées graphiquement comme illustré dans la figure 20. Tous les stéréotypes proposés sont dotés de contraintes, sous forme de conditions, qui limitent leur utilisation dans le domaine de la sensibilité au contexte. Ces contraintes montrent comment les stéréotypes proposés peuvent être associés et fixent le nombre d'instances correspondantes pouvant participer à ces associations au niveau modèle (M1).

En effet, chaque instance du stéréotype <<ContextObject>> doit avoir une et une seule instance du stéréotype <<ContextLifeLine>>, et chaque instance <<ContextMessage>> doit être reliée à exactement deux instances <<ContextLifeLine>> parce qu'un message représente

toujours un échange d'information (lors d'une interaction) entre deux objets en se pointant sur les deux lignes de vie correspondantes.

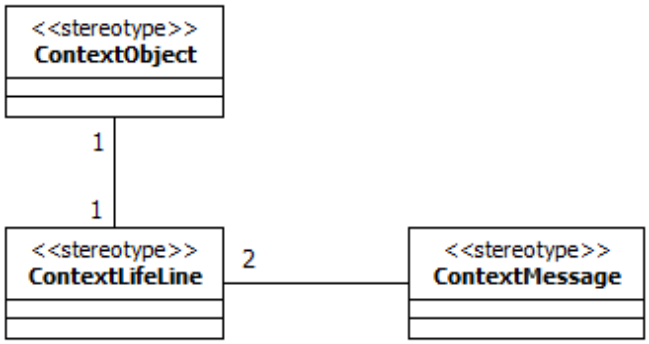


Figure 20. Représentation graphique des contraintes « SequenceUML ».

5.5.4. Etiquettes du profil « SequenceUML »

Les stéréotypes proposés du profil « SequenceUML » sont marqués par des étiquettes qui font apparaitre les caractéristiques spécifiques liées à la variation du contexte d'utilisation. En effet, chaque caractéristique est prise en charge sous forme d'un attribut attaché au stéréotype correspondant et dont nous devons définir toutes les valeurs possibles qu'il peut avoir. La table 18 présente quelques étiquettes proposées qui sont attachées aux différents stéréotypes et qui sont définies, chacune, par un nom d'attribut et ses valeurs possibles.

Nom du diagramme UML	**Stéréotype d'attachement**	**Nom de l'attribut**	**Valeurs de l'attribut**
Diagramme de séquences	ContextObject	Object_Nature	"Internal", "External"
		Object_Type	"Permanent", "Temporary"
		Object_Role	"Creator", Destructor"
	ContextMessage	Message_Quality	"High", "Medium", "Low"
		Is_Recursive	"True", "False"
		Is_Synchrone	"True", "False"
		Is_Asynchrone	"True", "False"

Table 18. Les étiquettes attachées aux stéréotypes proposés « SequenceUML ».

Par exemple, pour le stéréotype <<ContextObject>> nous avons proposé trois attributs 'Object_Nature', 'Object_Type', 'Object_Role' qui mettent en évidence le côté contextuel (variable) d'un objet et qui spécifient respectivement la nature de cet objet (interne ou externe), son type (permanent ou temporaire) et son rôle (créateur ou destructeur).

5.6. Description du profil « ActivityUML » : Diagrammes d'activités

5.6.1. Paquetage du profil « ActivityUML »

Les deux principales méta-classes UML ("Activity" et "Transition") d'un diagramme d'activités sont étendues respectivement par deux stéréotypes (<<ContextActivity>> et <<ContextTransition>>) (Figure 21).

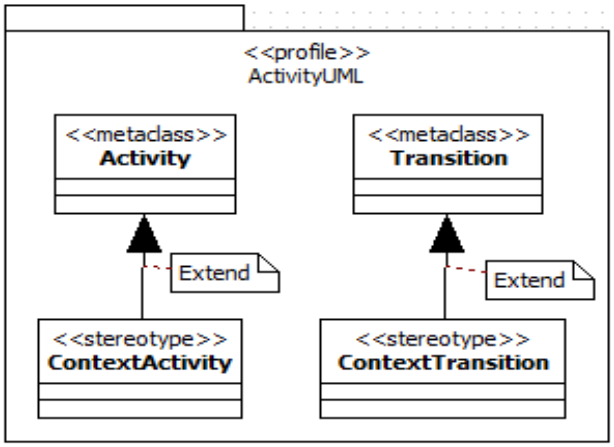


Figure 21. Paquetage du profil « ActivityUML ».

Pour les systèmes mobiles, les taches (ou activités) exécutent des actions contextuelles qui sont influencées par plusieurs facteurs variables et instables à cause de la grande mobilité des utilisateurs. De telles activités ne peuvent pas être représentées, de façon appropriée, par les notations standard du langage UML parce que chacune d'elles peut prendre plusieurs formes différentes selon que le contexte d'utilisation change d'une situation à une autre. D'où, la nécessité d'utiliser les stéréotypes proposés dans le profil « ActivityUML » pour pouvoir modéliser convenablement les activités présentant un caractère contextuel.

5.6.2. Stéréotypes du profil « ActivityUML »

Pour le profil « ActivityUML », deux stéréotypes (<<ContextActivity>> et <<ContextTransition>> ont été créés par extension des deux méta-classes UML respectives ("Activity" et "Transition") (Table 19).

Nom diagramme UML	**Stéréotype**	**Méta-classe UML**	**Description**
Diagramme d'activités	ContextActivity	Activity	C'est un ensemble d'actions (opérations ou méthodes) contextuelles qui peuvent être appliquées sur un stéréotype *ContextClass*
	ContextTransition	Transition	Indique une transition reliant la fin d'un stéréotype *ContextActivity* vers le début d'un autre *ContextActivity*

Table 19. Les stéréotypes proposés du profil « ActivityUML ».

Le stéréotype <<ContextActivity>> est capable de modéliser toutes les actions qui peuvent être modifiées suite aux modifications subies par le contexte d'utilisation d'une situation. Le stéréotype <<ContextTransition>> modélise le passage d'une activité <<ContextActivity>> vers une autre activité <<ContextActivity>> avec, éventuellement, la précision d'une garde conditionnelle.

5.6.3. Contraintes du profil « ActivityUML »

Les contraintes attachées aux stéréotypes du profil « ActivityUML » sont définies pour préciser et fixer le mode d'utilisation de ces stéréotypes lors du processus de modélisation. Les contraintes proposées, ici, permettent de spécifier les règles de création et d'utilisation des stéréotypes correspondantes (Table 20). Par exemple, pour le stéréotype <<ContextTransition>>, nous avons proposé une contrainte permettant d'ajouter une garde qui exige la réalisation et la satisfaction d'une certaine condition pour pouvoir transiter d'une activité <<ContextActivity>> vers une autre activité <<ContextActivity>>.

Nom diagramme UML	**Stéréotype**	**Description de la contrainte attachée**
Diagramme d'activités	ContextActivity	- Doit être étendu à partir de la méta-classe UML "Activity" - Doit être associé à un autre stéréotype *ContextActivity* à l'aide d'un stéréotype *ContextTransition*
	ContextTransition	- C'est une relation qui associe deux stéréotypes *ContextActivity* - Peut avoir une garde qui conditionne le passage entre deux *ContextActivity*

Table 20. Les contraintes attachées aux stéréotypes proposés « ActivityUML ».

Dans la figure 22, nous présentons graphiquement une contrainte d'un diagramme d'activités montrant le type d'association qui doit exister entre les stéréotypes <<ContextActivity>> et <<ContextTransition>> ainsi que les multiplicités correspondantes. En effet, chaque instance du stéréotype <<ContextTransition>> peut relier exactement deux instances du stéréotype <<ContextActivity>>

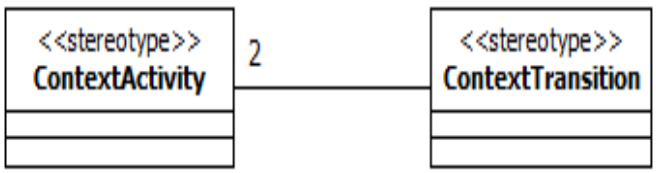


Figure 22. Représentation graphique des contraintes « ActivityUML ».

5.6.4. Etiquettes du profil « ActivityUML »

Par définition, les étiquettes représentent les attributs d'un stéréotype. Dans le profil « ActivityUML », nous avons proposé quelques étiquettes qui sont attachées aux deux stéréotypes créés <<ContextActivity>> et <<ContextTransition>> (Table 21). Ainsi, l'attribut 'ActivityVisibility' prendra en charge les éventuels changements du type de visibilité d'une activité <<ContextActivity>> et propose, pour cela, une liste de valeurs indiquant le type public, privé ou protégé.

Nom du diagramme UML	**Stéréotype d'attachement**	**Nom de l'attribut**	**Valeurs de l'attribut**
Diagramme d'activités	ContextActivity	Activity_Visibility	"Public", "Private", "Protected"
		Type_Of_Activity	"Internal", "External"
	ContextTransition	Context_Of_Guard	"Day", "Night", "Indoor", "Outdoor"
		Is_Periodic	"True", "False"

Table 21. Les étiquettes attachées aux stéréotypes proposés « ActivityUML ».

Conclusion

UML est un langage de modélisation qui propose des concepts et des notations permettant de représenter graphiquement les différentes vues d'un système. Une limite d'utilisation du langage UML est qu'il est conçu pour modéliser les systèmes de tous les domaines d'ordre général. Par contre, ce langage présente des possibilités d'extension capables de le rendre utilisable pour les domaines particuliers présentant des caractéristiques spécifiques. Notre étude porte sur le domaine particulier de la sensibilité au contexte « context-awareness » et lors de la modélisation du contexte d'utilisation, nous avons rencontré beaucoup de difficultés à trouver des notations UML plus adéquates pour modéliser les éléments contextuelles qui sont très variables d'un moment à un autre, d'un endroit à un autre, d'un dispositif à un autre, etc. Pour cela, nous avons décidé, dans ce chapitre 4, de personnaliser le langage UML en étendant et en spécialisant ses notations pour être utilisé adéquatement dans le développement d'applications sensibles au contexte. Les nouvelles notations proposées sont regroupées dans un paquetage sous forme de profil UML destiné au domaine du context-aware.

CHAPITRE 6

ADAPTATION CONTEXTUELLE DES APPLICATIONS

Introduction

L'adaptation des systèmes d'information (ou des applications) aux préférences de l'utilisateur et au contexte d'utilisation se base essentiellement sur la manière dont les éléments de ce contexte sont pris en charge. En effet, cette prise en charge des contraintes contextuelles varie selon d'abords le mode de représentation de ces contraintes (règles, modèles, fichiers XML, etc.) et, aussi, selon le moment de leur insertion par rapport au cycle de vie de l'application (développement ou exécution).

Généralement, la prise en compte du contexte d'utilisation et des préférences utilisateurs par les systèmes d'information ubiquitaires doit passer par plusieurs étapes :

- **a.** Capture de l'information contextuelle (collecte d'informations via les capteurs et les senseurs).
- **b.** Traitement de l'information capturée (interprétation, stockage, …).
- **c.** Modélisation des données et formulation des règles contextuelles.
- **d.** Adaptation des systèmes d'information en utilisant les nouveaux modèles de données (on fera appel à une approche de conception des systèmes d'information sur laquelle seront insérés les nouveaux modèles de données).

Dans ce qui suit, nous allons présenter notre proposition visant à améliorer l'approche MDA pour qu'elle soit capable de prendre en charge les éléments du contexte d'utilisation lors de son processus de développement.

6.1. Présentation des solutions envisagées

Nous avons opté pour une approche basée sur les modèles (MDA : Model Driven Architecture) [OMG, 2003] comme approche de conception. La MDA consiste à appliquer des transformations successives sur les modèles (CIM – PIM – PSM) pour aboutir à la

génération du code exécutable d'une application [OMG, 2003]. La notion de "contexte d'utilisation" n'est pas clairement prise en charge par le cycle de vie de la MDA même si dans la définition du modèle CIM [OMG, 2003] nous faisons inclure les informations portant sur l'environnement, mais ceci demeure insuffisant, d'une part, pour faire face aux gigantesques variations subies par cet environnement à cause des grandes opportunités offertes par l'informatique nomade, et d'autre part, pour bien satisfaire les préférences des utilisateurs de plus en plus exigeants.

Pour cela, nous envisageons d'introduire ou bien d'insérer les informations contextuelles dans le cycle de vie de la MDA, cette insertion peut se faire à plusieurs niveaux [Benselim, 2009a]:

1er cas : Lors de l'élaboration du modèle CIM

On peut augmenter la définition du modèle CIM [OMG, 2003] pour englober l'aspect contextuel de la situation courante (application et utilisateur). Cette solution est réalisable par la substitution du terme "environnement" dans la définition du CIM par le terme "contexte d'utilisation" qui réunira, en plus de l'environnement, l'application et l'utilisateur. Cette substitution permet de définir clairement chacun des paramètres du contexte d'utilisation et d'inclure l'aspect comportemental et les préférences d'un utilisateur. Par exemple, les concepteurs des systèmes d'information peuvent introduire, dès la préparation du modèle CIM, des variables (éléments du modèle CIM) réservées pour les paramètres régionaux (langue, date, heure, nombres, unités de mesure,…) détectables automatiquement selon la localisation de l'utilisateur (élément de l'environnement). L'environnement, dans ce cas, ne garantit pas la personnalisation des résultats car si cet utilisateur est un touriste étranger qui utilise son PDA à partir de la localisation détectée et qui présente un profil différent (langue, mode d'affichage,..), il ne sera pas satisfait.

2ème cas : Lors de la transformation PIM-PSM

Pour insérer les informations contextuelles, nous pouvons utiliser les différentes approches (ou techniques) de transformations des modèles MDA proposées par l'OMG dans [OMG, 2003]. Parmi ces techniques nous pouvons choisir deux qui conviennent le plus à nos fins (simples, claires, faciles à mettre en œuvre,…); et qui sont :

- Technique de marquage de modèles [OMG, 2003]: c'est une technique qui consiste à transformer un modèle source (PIM) vers un modèle résultat (PSM) en utilisant les annotations. Celles-ci sont extraites à partir du "Mapping" (ensemble de règles de transformation relatives aux contraintes technologiques d'une plateforme), et sont, ensuite, appliquées sur les éléments du PIM. Nous obtenons ainsi un nouveau modèle appelé "PIM marqué" qui renferme les éléments marqués (transformables) et non marqués du PIM initial. Enfin, chacune des règles du "Mapping" est appliquée sur l'élément marqué correspondant (appartenant au PIM marqué) pour avoir un élément du PSM (Figure 23).

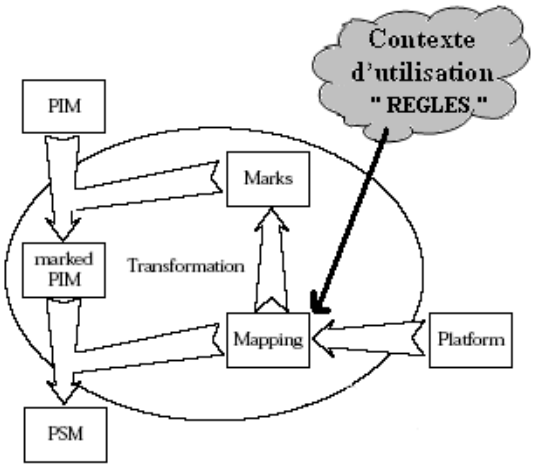


Figure 23. Architecture proposée d'une transformation par marquage [Benselim, 2009a].

- Technique de fusion de modèles [OMG, 2003]: cette technique consiste à fusionner le modèle à transformer (PIM) avec un autre modèle (à définir) pour obtenir un modèle résultat (PSM) (Figure 24).

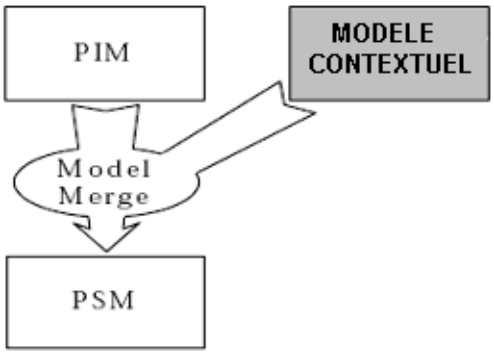


Figure 24. Architecture proposée d'une transformation par fusion [Benselim, 2009a].

Dans la figure 26, nous présentons l'architecture des solutions envisagées et nous montrons les étapes des deux variantes possibles (**a**) et (**b**) pour la prise en compte du contexte par la MDA.

Variante (a) : En utilisant la technique de marquage de modèles (Figure 23), nous proposons que l'ensemble des règles de transformation "Mapping" puisse être enrichi par les informations contextuelles (sous forme de règles contextuelles), c'est-à-dire que ce "mapping contextuel" (Figure 25) s'alimentera de deux sources différentes à savoir la plateforme et le contexte d'utilisation. A partir du mapping contextuel, nous déduisons les annotations (ou marques) qui seront utilisées pour marquer les éléments du PIM à transformer. Nous obtenons alors un "PIM marqué" sur lequel nous appliquerons les règles du mapping contextuel (plateforme et contexte d'utilisation) du même système pour avoir enfin un modèle PSM ayant

les spécifications techniques de la plateforme et aussi les spécifications contextuelles et comportementales du système.

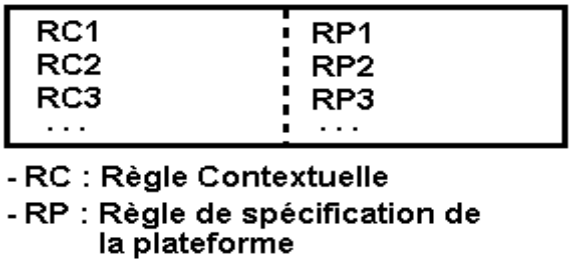


Figure 25. Contenu du "mapping contextuel".

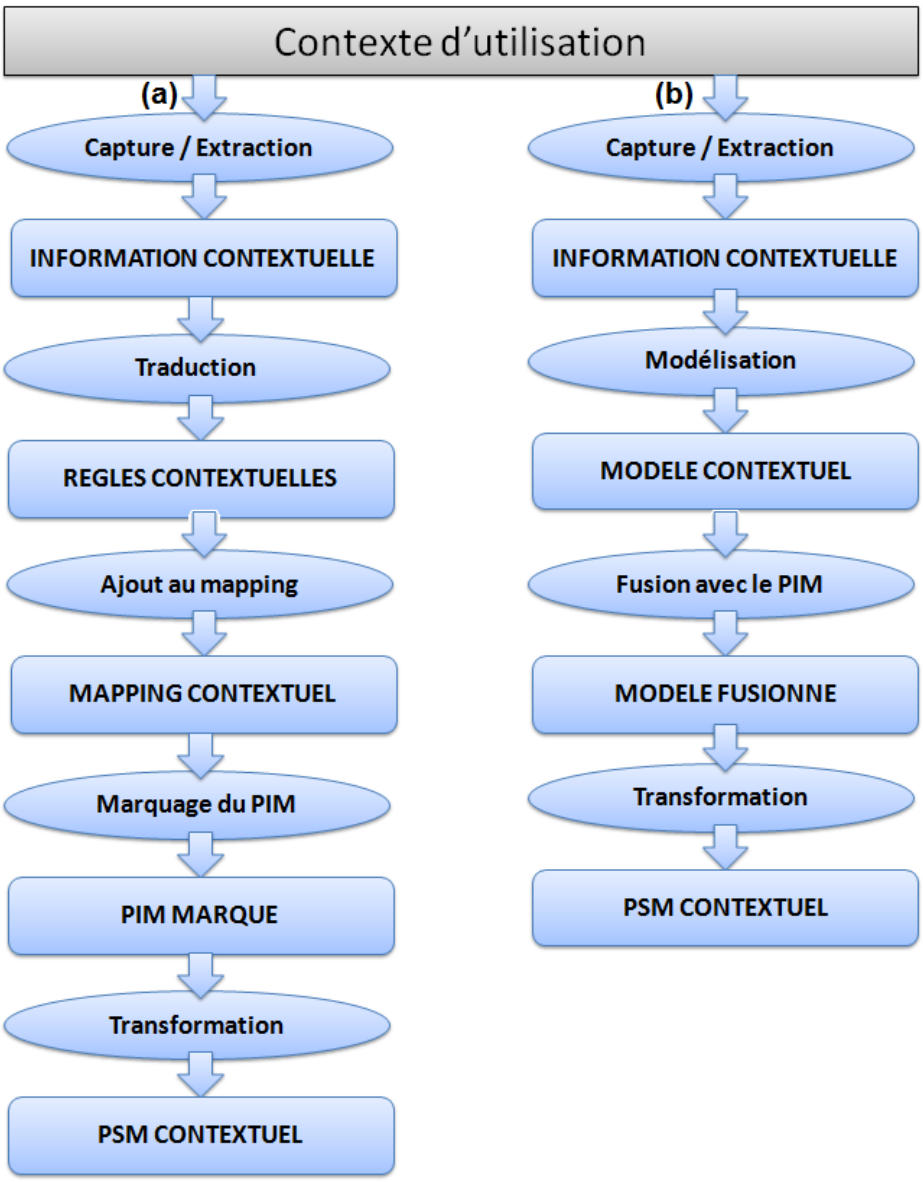


Figure 26. Architecture des solutions envisagées [Benselim, 2009a].

Variante (b) : En utilisant l'autre technique de transformation qui consiste à fusionner le modèle PIM avec un autre modèle (Figure 24), nous proposons que cet autre modèle soit un nouveau modèle contextuel qui contiendra toutes les informations contextuelles du système. Nous envisageons, donc, d'ajouter une étape préliminaire qui sera consacrée à la modélisation du contexte et qui aura comme résultat le modèle contextuel.
Dans ce nouveau modèle contextuel, nous allons formaliser toutes les contraintes contextuelles d'utilisation et les préférences des utilisateurs en données contextuelles. L'étape suivante consiste à faire la combinaison (fusion) de ce modèle contextuel avec le modèle PIM pour assurer la transformation vers le modèle PSM. Le principe de la fusion des deux modèles est décomposé en quatre phases: [Kolovos, 2006]
- comparaison: identification des correspondances (accords) entre les éléments des modèles à fusionner
- test: vérification de la conformité des éléments correspondants identifiés dans le but d'éliminer les conflits entre eux
- fusion: application de l'outil de fusion (algorithme)
- restructuration: nettoyage du modèle obtenu de toutes les inconsistances et les anomalies.

6.2. La modélisation du contexte

Actuellement, l'utilisation de la notion de « modèle » marque son grand retour avec l'apport de l'ingénierie des modèles (MDE : Model Driven Engineering) qui présente de grandes facilités pour représenter, traiter, stocker et exploiter les informations. Les informations contextuelles, déjà identifiées par séparation des préoccupations, doivent être modélisées à l'aide d'un langage de modélisation. Ceci permettra d'avoir un modèle contextuel qui formalise toutes les contraintes du contexte d'utilisation.

Parmi les approches existantes destinées pour la modélisation du contexte, nous avons choisi la modélisation graphique utilisant le langage UML (Unified Modeling Language). Ce choix est argumenté par les multiples avantages que présente UML comme la disponibilité de notations simplifiées et non ambiguës, la formalisation précise des concepts et la couverture importante des besoins en modélisation.

6.2.1. La représentation du contexte

Plusieurs auteurs [Bauer, 2003] [Chaari, 2005] estiment qu'actuellement il n'existe pas une méthode standard qui montre comment modéliser un contexte ni comment ce contexte peut-il influencer la conception d'un système d'information.

L'utilisation d'une information contextuelle nécessite sa description et sa représentation. A cet effet, Bauer propose plusieurs possibilités [Bauer, 2003]:

- Le langage humain
- Les diagrammes UML
- La composition de modèles

Pour décrire une information contextuelle, chaque élément contextuel doit faire l'objet d'une description textuelle contenant les informations qui ne peuvent pas être exprimées par des diagrammes, et qui contient aussi :

- Une description générale de la situation
- Les détails de l'information supportée
- Le processus de la distribution de l'information
- Les processus décrivant l'utilisation de l'information

Les procédures qui interviennent et qui sont liées à l'environnement doivent être exprimées par des diagrammes. Le langage UML (Unified Modelling Language) représente le meilleur outil standard pour la modélisation de l'information. UML propose plusieurs diagrammes qui peuvent décrire les différents aspects ou vues d'un système.

La représentation d'un contexte dépend étroitement de son utilisation [Bauer, 2003]. Par exemple, si le contexte est utilisé pour décrire l'état d'une entité alors ce contexte doit être exprimé par un diagramme de classes (représentation par diagrammes statiques). Par contre, si le contexte est utilisé pour spécifier une information contextuelle dépendante de l'interaction de l'utilisateur alors le contexte sera exprimé à l'aide de diagramme approprié (représentation par diagrammes dynamiques).
Donc les diagrammes UML peuvent être utilisés pour représenter le contexte. Par conséquent, toutes les informations relatives au contexte (composants, procédure, taches, …) peuvent être exprimées et modélisées par l'un de ces diagrammes.

6.2.2. Méthode d'utilisation du contexte

Généralement, l'utilisation du contexte passe par trois étapes [Bauer, 2003]:

a. Identification et analyse du contexte

Cette étape consiste à décrire clairement les informations contextuelles et de les étudier pour faire ressortir tous les éléments constituants le contexte ainsi que leurs propriétés.

b. Abstraction des informations contextuelles

Les éléments du contexte précédemment identifiés et analysés sont soumis à un processus d'abstraction destiné à mettre en évidence les constituants les plus importants du contexte. Ceci doit tenir compte de quelques paramètres :

- L'information requise par l'utilisateur
  - Structure du modèle : par exemple, le diagramme de classes
  - Type du support d'information : exprimé par un diagramme de séquences
- La présentation de l'information
- La source d'information

c. Modélisation et conception

La modélisation des systèmes d'information sensibles au contexte se base essentiellement sur la notion de contexte. Ces systèmes agissent comme étant un assistant qui est capable de

fournir l'information demandée malgré les changements continuels subis par la situation courante de l'utilisateur.

6.2.3. Modélisation graphique du contexte

Principe

Pour modéliser graphiquement le contexte d'utilisation, nous proposons un processus à six phases :

1ère phase : Identification : définir tous les objets et les contraintes susceptibles d'influencer la situation courante d'un utilisateur. L'identification des facteurs et des contraintes qui influencent la situation courante d'un utilisateur peut être déduite lors de l'opération de séparation des préoccupations vue précédemment (Chapitre 4). Le résultat de cette séparation permet d'identifier clairement tous les objets du système qui affectent le contexte d'utilisation. Ces objets peuvent être liés soit à l'utilisateur lui-même (identité, profil, préférences, etc.), soit à l'application (logiciels, matériels, réseaux, etc.) ou bien à l'environnement (localisation, temps, température, personnes avoisinantes, ressources disponibles, etc.).

2ème phase : Nomenclature : Attribuer un nom à chaque objet. Les objets identifiés sont définis, chacun, par un nom unique qui représentera le nom d'une classe ou d'un stéréotype de classe dans le futur modèle. La liste des objets collectés comprend par exemple: {Utilisateur, Personne, Localisation, Temps, Matériel, ressource, PC, Laptop, PDA, Téléphone mobile «Mobile», Session, Profil, Préférence, Contenu, Présentation, Comportement, Rue, Bâtiment, Aéroport, Gare de train «Gare», Bureau, Maison, Cyber}.

3ème phase : Hiérarchisation : Organiser les objets selon leurs relations dans le système. C'est-à-dire qu'il faut représenter graphiquement les liens et les dépendances hiérarchiques entre les différents éléments identifiés lors des phases précédentes et qui vont composer le contexte d'utilisation.

4ème phase : Choix du méta-modèle : La représentation d'un diagramme doit être conforme aux règles spécifiées dans le méta-modèle correspondant. Le méta-modèle permet fixer les règles de construction d'un modèle. D'où, il est indispensable de préparer un méta-modèle précis avant d'entamer la construction du modèle désiré.

5ème phase : Sélection du diagramme : Choisir un diagramme adéquat pour représenter les objets qui contiennent des informations contextuelles. Pour représenter le modèle contextuel, nous avons opté pour le diagramme de classes proposé par le langage UML. Ce type de diagramme illustre la structure statique des classes d'un système ainsi que les relations qui existent entre ces classes. Après avoir fixé l'ensemble des éléments contextuels du système, le diagramme de classe représente un excellent formalisme de base pour modéliser ces éléments. La notation est assez simple et met en œuvre des rectangles contenant les classes et des flèches

montrant soit une association soit un lien d'héritage. Il faut noter, ici, qu'il est possible d'utiliser les notations étendues du profil UML proposé dans le chapitre 5.

6ème phase : Construction du modèle : Schématiser tous les constituants du modèle à partir des objets identifiés et en utilisant les concepts du diagramme choisi. Lors de cette dernière phase, nous essayons de consolider tous les éléments nécessaires à la construction du modèle contextuel. Ainsi, toutes les classes doivent être décrites en précisant leurs attributs et leurs opérations (méthodes). Ensuite, il faut définir les associations utilisées dans notre modèle ainsi que leurs types et leurs cardinalités.

6.3. Intégration du modèle contextuel dans l'approche MDA

L'ingénierie de modèles (MDE : Model Driven Engenering) se base sur la gestion et la manipulation des modèles pour atteindre à mieux les objectifs de conception et de développement.
La MDA est une approche qui utilise la notion de "modèles" pour concevoir une application. Pour cette approche, toute chose (objet ou entité) peut être considéré comme un modèle. Un modèle est une description formelle d'un objet. Les modèles, qui représentent les éléments de base dans l'approche MDA, peuvent subir plusieurs types d'opérations comme la création, la suppression, la modification, la fusion, la transformation,etc. Ainsi, la fusion de modèles représente l'une de ces opérations les plus délicates vu la spécificité des contraintes que doivent vérifier tous les éléments nécessaires à l'exécution de cette opération. Par exemple, des contraintes sur le mode de représentation des modèles en entrée, sur la composante de l'ensemble des règles de mapping (qui inclut les règles de fusion et les règles de transformation), sur les détails de déroulement du processus de fusion (union et combinaison) et à quel niveau d'abstraction se fera cette fusion (modèle, méta-modèle ou méta-méta-modèle). Le développement des applications sensibles au contexte rencontre un grand défi quant à l'intégration du contexte d'utilisation.

Notre proposition d'adaptation contextuelle se distingue de la majorité des autres travaux par la prise en compte des contraintes contextuelles pendant les étapes du processus de développement d'une application et non pas après. Le modèle contextuel, obtenu après séparation des préoccupations contextuelles et modélisation, doit faire l'objet d'une prise en charge par une approche de développement.
Cette proposition envisage une amélioration (dans le sens de complémentarité) de l'approche MDA dans laquelle sera intégré le modèle contextuel. Pour accomplir cette intégration, nous avons utilisé une opération de fusion de modèles qui va combiner les éléments du modèle métier (PIM) avec ceux du modèle contextuel (CM : Contextual Model). Le processus de fusion passe par quatre étapes : comparaison des paires d'éléments, vérification de conformité des éléments comparés, fusion et combinaison des éléments vérifiés et, enfin, la restructuration des éléments combinés. Le résultat de l'opération de fusion est un modèle fusionné (MM : Merged Model) qui sera transformé vers un modèle spécifique (PSM) suivant les spécifications de la plateforme choisie.

6.3.1. Définition de l'opération de fusion

La fusion de deux ou plusieurs modèles est la combinaison de tous leurs composants (entités, attributs, associations) suivant des contraintes et des règles de passage bien définies dans le but de construire un nouveau modèle cible.

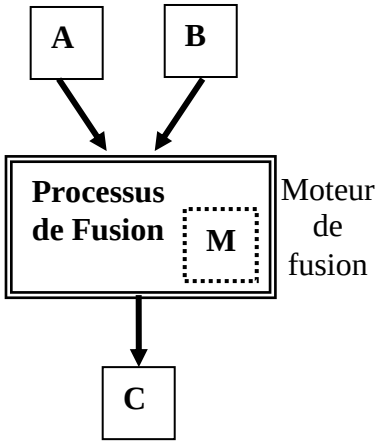


Figure 27. Schéma général d'une opération de fusion.

La figure 27 représente une opération de fusion de deux modèles en entrée A et B pour construire un modèle résultat C suivant le mapping M. P**o**ur notre cas, nous avons:

- Le modèle en entrée **A** : représente le modèle métier du système.
- Le modèle en entrée **B** : désigne le modèle contextuel du système.
- Le modèle résultat **C** : c'est le modèle obtenu après combinaison et transformation des modèles en entrée.
- Le processus de fusion : représente les étapes de fusion à suivre et/ou l'algorithme de fusion à appliquer.
- Le mapping **M** : c'est l'ensemble de toutes les correspondances d'éléments et les règles qui définissent la nature de la relation entre les modèles A et B. Le mapping représente une partie de tout le moteur de fusion.

6.3.2. Principes de la fusion

L'opération de fusion nécessite la définition de quelques notions élémentaires [Pottinger, 2003]:

6.3.2.1. Représentation des modèles

Les modèles peuvent être représentés selon la terminologie conventionnelle des niveaux d'abstraction de la MDA à savoir :

- le modèle,
- le méta-modèle et
- le méta-méta-modèle.

A ce stade de l'étude, nous opérons au niveau d'abstraction M1, c'est-à-dire le niveau « modèle ».

6.3.2.2. Les entrées de la fusion

Les éléments nécessaires au déroulement d'une opération de fusion sont :

- Un modèle métier noté A qui regroupe toutes les fonctionnalités du système à étudier,
- Un modèle contextuel noté B qui formalise toutes les informations relatives au contexte d'utilisation,
- Une désignation optionnelle pour choisir le modèle privilégié (A ou B) en cas de correspondance parfaite et pour résoudre les conflits.
- Un mapping M, qui se compose de :
  - Les éléments de mapping : expriment les correspondances simples entre les éléments des deux modèles en entrée. Ces correspondances peuvent être de type «égalité» ou de type «similitude». l'«égalité» est définie lorsque le même élément origine peut être en relation simple avec deux éléments destinataires distincts mais égaux. Si cette relation est complexe, nous disons qu'il y a une «similitude».
  - Les relations : expriment les liaisons complexes qui existent entre les éléments ne présentant pas de correspondances simples.
  - Les éléments de non-mapping : ces éléments n'ont pas de représentation réelle ni dans A ni dans B mais qui peuvent aider à décrire la relation des éléments de A et B.

6.3.2.3. Le résultat de la fusion

Le modèle résultat C obtenu par fusion des deux modèles A et B doit satisfaire les conditions suivantes :

(C1) : Préserver les éléments : chaque élément en entrée possède un élément correspondant dans C.

(C2) : Préserver l'égalité : les éléments en entrée sont transformés vers le même élément de C si et seulement si ils sont de types ″égalité″ dans le mapping.

(C3) : Préserver les relations : chaque relation en entrée existe d'une façon explicite ou implicite dans C.

(C4) : Préserver la similitude : les éléments déclarés similaires (mais non égaux) dans le mapping conservent leur état de similitude dans C.

(C5) : Satisfaire la contrainte 'méta-méta-modèle' : le modèle C doit satisfaire toutes les contraintes du méta-méta-modèle en introduisant les éléments et les relations nécessaires.

(C6) : Outres les éléments et les relations cités ci-dessus, aucun élément ou relation supplémentaire n'existe dans C.

(C7) : Préserver les propriétés : chaque propriété d'un élément du modèle C doit figurer comme propriété d'un élément de l'un des modèles en entrée.

(C8) : Préférence de valeur : la valeur d'une propriété d'un élément de C est choisie parmi les éléments du mapping correspondant à cet élément, sinon à partir du modèle privilégié, sinon à partir d'un élément qui lui correspond.

6.3.3. Gestion des conflits

6.3.3.1. Types de conflits

- Conflits de représentation : ce sont des conflits qui apparaissent au niveau « modèle » et sont causés par les différentes représentations du même concept du monde réel. Par exemple, deux modèles qui décrivent le même concept par différentes façons.
- Conflits de méta-modèle : ces conflits apparaissent au niveau « méta-modèle » et ils se présentent lorsque les spécifications de ce méta-modèle ne sont pas respectées par le résultat de la fusion.
- Conflits de fond (principe) : ces conflits apparaissent au niveau « méta-méta-modèle » et sont causés par la violation de contraintes dans ce méta-méta-modèle. Par exemple, les conflits causés par la restriction du type unique ou la limitation (min, max) des cardinalités des relations.

6.3.3.2. Résolution de conflits

- Pour les conflits de représentation : la résolution nécessite une intervention manuelle de l'utilisateur en connectant les éléments des modèles provoquant le conflit vers le même élément du mapping dont le type est « égalité ».

Soit a un élément du modèle A, soit b un élément du modèle B. Si a et b décrivent un même concept, alors nous aurons un conflit de représentation. Pour résoudre ce conflit, nous devons chercher une correspondance (a=b) dans le mapping M telle que : l'élément a peut être connecté à l'élément a' et l'élément b peut être connecté à l'élément b'.

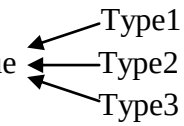
Exemples : a = identité , b = nom & prénom , identité = nom & prénom.
a = adresse , b = localité , adresse = localité.

- Pour les conflits de méta-modèle : la solution est de créer un nouvel opérateur qui doit obliger (forcer) le modèle à respecter l'ensemble des contraintes. Cet opérateur sera déterminé et choisi à partir des spécifications du méta-modèle.

Exemple : écrire une règle de contrôle qui vérifie le résultat de la fusion.

- Pour les conflits de fond (principe) : pour résoudre le conflit du type unique, il faut créer un nouveau type qui va s'intercaler entre l'élément est ses types. Le nouveau type va hériter des types déclarés d'un côté, et va replacer une liaison avec l'élément d'un autre côté.

Exemple : a : Type1
a : Type2 devient a : TypeUnique avec TypeUnique
a : Type3



6.3.4. Algorithme de fusion

Cet algorithme est inspiré de l'étude menée par Pottinger et présentée dans [Pottinger, 2003]. C'est un algorithme qui permet de fusionner deux modèles A et B vers un modèle résultat C suivant les règles et les correspondances contenues dans le mapping M.

Pour accomplir cette fusion, l'algorithme démarre par une opération de comparaison « matching » qui permet d'identifier toutes les relations existantes entre les éléments de A et B et de chercher les éléments (correspondances) du mapping M qui justifient ces relations. L'existence de ces relations et de ces correspondances permet de produire les nouveaux éléments du modèle résultat en combinant les éléments liés par une correspondance du mapping. Le modèle ainsi obtenu va subir une certaine validation par la vérification des propriétés des nouveaux éléments, la vérification des relations et la résolution de conflits.

Soient les hypothèses suivantes :

- Soit A un modèle métier,
- Soit B un modèle contextuel,
- Soit le mapping M,
- Soient les éléments a, b, c et m tels que : a est un élément de A, b est un élément de B, c est un élément de C et m est un élément de M.
- Le modèle C est le résultat de la fusion des deux modèles A et B suivant le mapping M.

Les étapes de l'algorithme de fusion peuvent être résumées comme suit :

1. **Initialisation :** mettre le modèle résultat C à **Ø**

2. **Matching des éléments :** déterminer une relation d'équivalence en groupant les éléments de A, B et M. Initialement, chaque élément se trouve dans son propre groupe. Ensuite :
   - **a.** Si une relation existe entre un élément de A ou de B avec un élément de type « égalité » appartenant à M, alors combiner les groupes correspondants.
   - **b.** Répéter l'étape (a) jusqu'à l'arrêt ensuite créer, pour chaque groupe, un nouveau élément dans C.

3. **Propriétés de l'élément :** soit c un élément de C relativement au groupe G1.
   - **a.** Les propriétés de c (sauf la propriété 'type') sont formées à partir de la réunion de toutes les propriétés des éléments appartenant au groupe G1.
   - **b.** La propriété 'type' d'un élément de C est définie si et seulement si cet élément appartient au mapping, et sa valeur est déterminée par la condition C8 (préférence de valeur).

4. **Relations**
   - **a.** Si un élément c de C correspond à un élément du mapping m de type «similitude» alors remplacer chaque relation R dont l'origine est m par une relation qui a comme origine c et comme destination l'élément de C qui correspond au groupe de la destination de R.
   - **b.** Les relations provenant d'un élément sont classées dans l'ordre suivant :

- celles qui correspondent aux relations dans le mapping,
- ensuite celles qui correspondent aux relations dans le modèle privilégié mais pas dans le mapping,
- et enfin toutes les autres relations.

c. Enlever les relations implicites du modèle C.

5. **Résolution de conflits :** la résolution d'un conflit peut, d'une part, créer un nouveau conflit, et d'autre part, elle peut s'interférer avec une autre résolution.
   Pour chaque conflit :
   a. Définir une stratégie spécifique de résolution
   b. Appliquer la stratégie choisie
   c. Sinon, appliquer la stratégie prédéfinie par défaut.

6.4. Processus d'intégration

Le processus d'intégration (Figure 28) du modèle contextuel dans l'approche MDA se base essentiellement sur l'opération de fusion de modèles comme décrite dans la section précédente. Cette opération permet de combiner les éléments du modèle métier (PIM) avec ceux du modèle contextuel (CM).

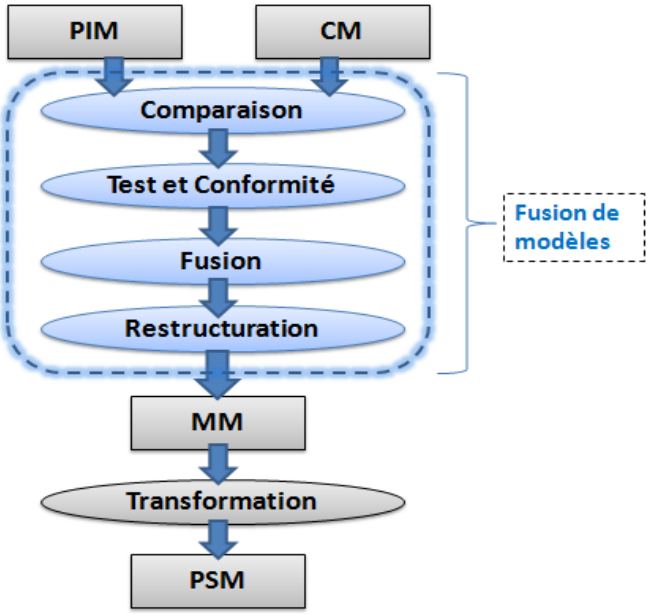


Figure 28. Etapes du processus d'intégration [Benselim, 2011a].

La fusion peut être décomposée en quatre phases : comparaison, test de conformité, fusion et restructuration [Kolovos, 2006]. Le résultat de l'opération de fusion sera un modèle fusionné (MM) qui va subir, à son tour, une transformation vers un modèle spécifique (PSM) suivant le mapping de la plateforme choisie.

6.4.1. Phase de comparaison

C'est une phase de «matching» des éléments par paires, c'est-à-dire l'identification de toutes les correspondances (ou accords) qui peuvent existées entre les éléments équivalents des

modèles en entrée. Ceci permet d'éliminer la redondance des éléments dans le modèle résultat. Les approches de comparaison peuvent être manuelles ou automatiques. Le principe de matching repose sur l'exécution de règles appelées « match-rules ». Chaque « match-rule » peut comparer des paires d'éléments appartenant à deux modèles différents et donner, ensuite, une décision sur le résultat de cette comparaison. Cette décision, de type logique, montre si les deux éléments comparés sont assortis (correspondants) entre eux ou non.

La comparaison porte sur tous les éléments qui constituent le modèle à fusionner. C'est-à-dire que chaque élément du modèle PIM sera comparé avec un autre élément du modèle contextuel CM à condition que ces deux éléments soient du même type.
Par exemple, UML propose l'utilisation des notions de classe, opération, attribut, etc. Donc, pour comparer deux modèles UML, il faut appliquer l'opération de «matching» sur tous leurs éléments constituants comme les classes, les opérations, les attributs, etc. Ceci nous amène à écrire plusieurs règles de comparaison (match-rules) dont chacune permet de décrire les éventuelles correspondances existantes entre les paires d'éléments de même type.

La syntaxe générale d'une règle de comparaison est de la forme :

rule *<nom_de_la_règle>*
match *PIM : UML!<élément_PIM>*
with *CM : UML!<élément_CM>*
extends *<portée>*
compare *<expression de comparaison>*

Cette règle permet de comparer un élément du PIM avec un autre élément du CM où les deux modèles (PIM et CM) sont conformes au même méta-modèle (UML). Le champ d'application (portée) de cette règle est défini par la partie « **extends** ». La partie « **compare** » comprend une expression de comparaison entre les deux éléments à comparer et doit retourner une décision de type logique qui affirme ou non la correspondance de ces deux éléments. Donc, si « **compare** » retourne la valeur 'vrai' alors les deux éléments comparés sont dits correspondants.

6.4.2. Phase de vérification de la conformité

Pendant cette phase, tous les éléments identifiés lors de la phase de comparaison (matching) sont testés et vérifiés. La vérification porte sur la conformité des éléments les uns par rapport aux autres, et permet l'identification d'éventuels conflits qui peuvent rendre la fusion non réalisable.

De même que pour l'opération de comparaison, la vérification de la conformité des éléments comparés doit être appliquée sur tous les éléments constituants le modèle. La conformité est, aussi, réalisée à l'aide de « match-rules » qui décident si les deux éléments comparés sont conformes ou non. Sauf que cette conformité soit définie dans une nouvelle partie appelée « **conform** ».

Donc, une règle de comparaison et de vérification de conformité sera de la forme :

rule *<nom_de_la_règle>*
match *PIM : UML!<élément_PIM>*
with *CM : UML!<élément_CM>*
extends *<portée>*
compare *<expression de comparaison>*
conform *<tests de conformité>*

La partie « **conform** » comprend des instructions décrivant les conditions à remplir pour vérifier la conformité des éléments à comparer. Cette partie doit retourner une décision de type logique (vrai ou faux) qui nous indique si ces les éléments comparés sont bien conformes ou non.

6.4.3. Phase de fusion

C'est la phase de l'union et de la combinaison des éléments issus des deux phases précédentes pour avoir les nouveaux éléments du modèle résultat. Dans cette phase, il existe deux démarches possibles, et ce, suivant le résultat du matching. En effet, si les éléments comparés sont jugés « correspondants », alors nous procédons à leur fusion dans le modèle cible MM ; et si ces éléments comparés sont jugés « non correspondants », alors nous appliquons une transformation vers le modèle MM. La correspondance des éléments comparés est définie sur la base des décisions logiques retournées par les parties « **compare** » et « **conform** ».

Pour exécuter ces opérations, deux types de règles sont utilisées : les règles de fusion « merge-rules » ou les règles de transformation « transform-rules ».

Ainsi, si « **compare** » retourne 'vrai' et si « **conform** » retourne 'vrai', alors nous pouvons conclure que les deux éléments comparés sont correspondants (égaux et conformes). Dans ce cas, nous devons appliquer une règle de type « merge-rule » qui va permettre de fusionner (assembler) ces éléments correspondants dans un même élément du modèle cible. Cette combinaison permet d'éliminer la redondance des mêmes éléments dans le modèle cible MM.
La syntaxe générale utilisée pour décrire une règle de fusion « merge-rule » est la suivante :

rule *<nom_de_la_règle>*
merge *PIM : UML!<élément_PIM>*
with *CM : UML!<élément_CM>*
into *MM : UML!<élément_MM>*
{
< instructions de fusion>
}

Sinon, si l'une ou l'autre des deux parties « compare » et « conforme » retourne 'faux', alors nous déduisons que les éléments comparés ne sont pas complètement correspondants. Dans ce

cas, nous devons déplacer chacun de ces éléments vers le modèle cible à l'aide de règles de type « transform-rules ». La syntaxe de ce type de règles est de la forme :

Si l'élément appartient au modèle source PIM,

```
rule <nom_de_la_règle>
    transform PIM : UML!<élément_PIM>
    to MM : UML!<élément_MM>
    {
     < instructions>
    }
```

Ou bien, si l'élément appartient au modèle source CM,

```
rule <nom_de_la_règle>
    transform CM : UML!<élément_CM>
    to MM : UML!<élément_MM>
    {
     < instructions>
    }
```

Pour garder la trace et l'origine des éléments fusionnés ou transformés, nous étiquetons chaque élément du modèle cible par une mention de type « stéréotype ».

Les stéréotypes sont définis préalablement dans la partie « **pre** » de la façon suivante :

```
Pre
{
def mergedStereotype : new MM !stereotype ;
mergedStereotype.name:='merged';
def PIMStereotype : new MM !stereotype ;
PIMStereotype.name:='PIM';
def CMStereotype : new MM !stereotype ;
CMStereotype.name:='CM';
}
```

6.4.4. Phase de nettoyage et de restructuration

Le modèle résultat, obtenu après la phase de fusion, peut contenir des inconsistances ou des anomalies. Pour le rendre fini et valide, il doit subir une opération de nettoyage et/ou une opération de restructuration afin de remettre l'ordre dans sa structure finale.
Cette opération est surtout applicable lorsque les modèles à fusionner ne sont pas conformes au même méta-modèle, ce qui n'est pas le cas pour nos modèles en entrée (PIM et CM).

6.4.5. Phase de transformation

La dernière étape du processus d'intégration du modèle contextuel consiste à transformer le modèle fusionné vers un modèle spécifique en utilisant les spécifications techniques de la plateforme choisie.

6.5. Architecture proposée de l'approche MDA

Pour récapituler cette partie de notre contribution, nous présentons ici les détails de l'approche proposée pour la prise en compte du contexte et des préférences des utilisateurs par les systèmes d'information ubiquitaires.

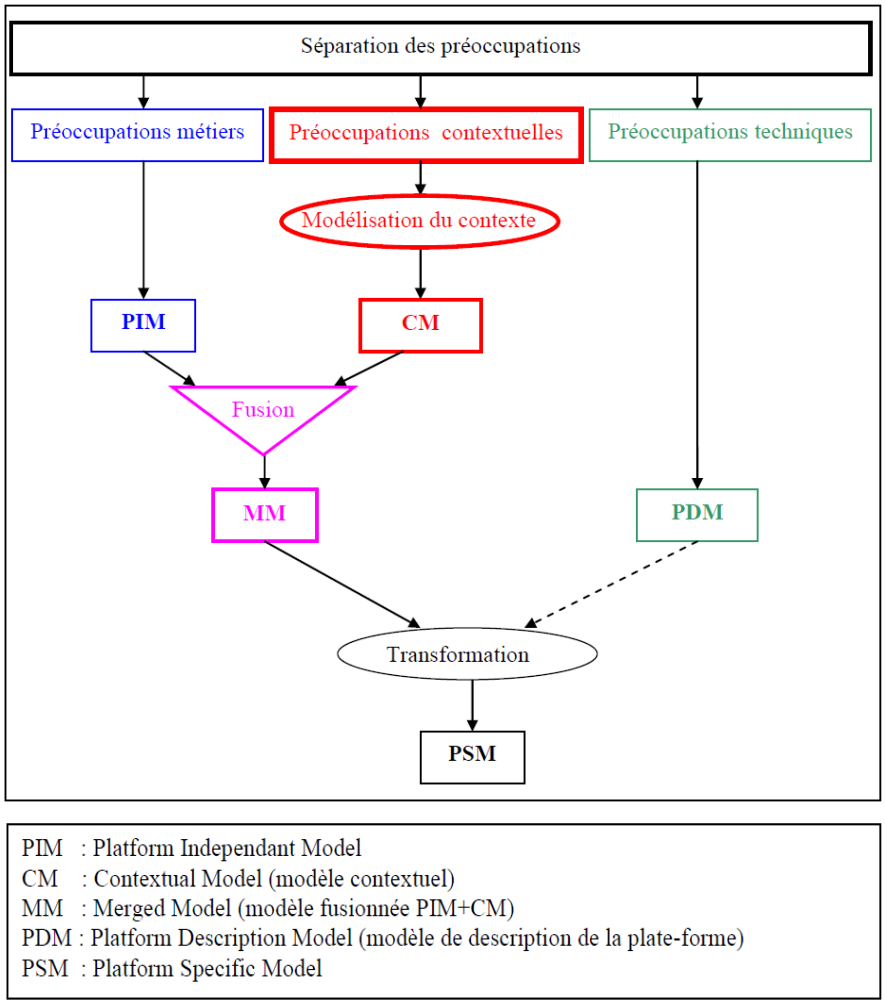


Figure 29. Architecture proposée pour l'amélioration de la MDA

L'approche proposée peut être résumée en quatre principales étapes :

a. Séparation des préoccupations contextuelles
b. Modélisation du contexte
c. Intégration du modèle contextuel
d. Transformation

La figure 29 illustre l'enchainement de ces étapes et leur disposition par rapport à l'architecture initiale de l'approche MDA.

Cette approche proposée [Benselim, 2011a] se base essentiellement sur les principes de la MDA comme la séparation des préoccupations et plus particulièrement la séparation des

préoccupations contextuelles. Ainsi, cette séparation va permettre d'étudier les contraintes contextuelles d'une façon indépendante des autres contraintes imposées au système d'information (contraintes fonctionnelles et contraintes technologiques). La notion d'ubiquité nécessite fortement que le contexte d'utilisation soit étudié séparément pour mieux prendre en charge l'évolution et les changements du contexte d'une part, et d'autre part pour ne pas remettre en cause tout le processus de développement à cause de ces changements perpétuels.
Après séparation des contraintes et propriétés contextuelles, celles-ci seront représentées à l'aide d'une approche graphique se basant sur le langage de modélisation UML. Cette représentation permet de construire un modèle particulier englobant toutes les informations contextuelles du système. Ensuite, le modèle contextuel obtenu (CM) sera intégré dans le cycle de vie de l'approche MDA en utilisant l'opération de fusion de modèles. La fusion de modèles permettra de regrouper les préoccupations métiers propres de système (PIM) avec celles relatives au contexte d'utilisation. Enfin, le modèle fusionné ainsi obtenu (MM) sera transformé vers un ou plusieurs modèles spécifiques (PSM) selon les spécifications technologiques de la plateforme à utiliser.

Conclusion

L'approche dirigée par les modèles (MDA) est une spécification basée sur un ensemble de standards tels que MOF (Meta Object Facility), UML (Unified Modeling Language), XMI (XML Metadata Interchange) et CWM (Common Warehouse Metamodel). La MDA a été conçue pour élaborer, visualiser, échanger, transformer et stocker des modèles de logiciels compréhensibles par la machine, développés indépendamment des technologies d'implémentation, et séparant les contraintes techniques des contraintes fonctionnelles. Mais du côté du contexte d'utilisation, nous avons remarqué que la MDA ne prévoit pas, de façon claire, explicite et indépendante, la prise en compte des variations subies par le contexte d'utilisation lors de son processus de développement. Au cours de ce chapitre, nous avons essayé d'augmenter le cycle de vie de la MDA pour le rendre capable de prendre en charge explicitement les éléments du contexte d'utilisation, et ce, par l'intégration d'un modèle contextuel dans son processus en utilisant une opération de fusion de modèle. Ceci permettra d'assurer un certain degré d'adaptation contextuelle des applications développées.

PARTIE III

EXPÉRIMENTATION

CHAPITRE 7

EXPÉRIMENTATION : « PROFIL UML »

Introduction

UML est devenu le standard des langages de modélisation logicielle en proposant une grande variété de concepts et de notations nécessaires à la modélisation d'applications. Mais, étant un langage général qui a été conçu pour modéliser la majorité des domaines, ce langage demeure limité quant à la modélisation de quelques domaines particuliers qui présentent des propriétés très spécifiques. Pour remédier à cette insuffisance, le langage UML offre des mécanismes d'extension qui lui permettent de s'auto-adapter avec les domaines particuliers. Ces mécanismes consistent à étendre les notations UML existantes pour obtenir de nouvelles notations plus adéquates et plus appropriées à la modélisation des caractéristiques spécifiques. Dans notre étude, nous avons profité de cette opportunité pour prendre en charge toutes les contraintes spécifiques que peuvent présenter les systèmes sensibles au contexte. Dans ce sens, nous avons proposé de nouveaux éléments de modélisation (issus par extension d'éléments UML existants) comme les stéréotypes, les contraintes et les étiquettes. Pour l'implémentation des concepts proposés, nous avons choisi la plateforme 'StarUML' qui est une plateforme extensible et supportant les notations du langage UML. Cette plateforme, destinée à la modélisation des logiciels, offre plusieurs avantages d'extensibilité, de flexibilité et de personnalisation [StarUML, 2005]. L'implémentation de notre profil proposé « UML Context-Aware » aidera et assistera les concepteurs à modéliser les besoins contextuels d'un système et à développer des applications sensibles au contexte pour les systèmes d'information ubiquitaires.

Dans ce chapitre d'expérimentation, nous exposerons, dans un premier temps, les étapes d'implémentation des concepts proposés comme :

- Préparation du fichier global du profil
- Création des stéréotypes
- Création des contraintes
- Création des étiquettes

Ensuite, nous essayerons de démontrer la faisabilité de notre proposition (relative au Profil UML) à travers un exemple illustratif mené dans le domaine médical.

7.1. Implémentation

7.1.1. Préparation du fichier global

Un profil UML est un ensemble de mécanismes d'extension (stéréotypes, contraintes et étiquettes) qui sont nécessaires pour le développement d'une plateforme spécifique ou pour la modélisation des concepts d'un domaine particulier. La première étape de cette implémentation est de préparer un fichier du profil qui sera défini au format XML (eXtended Markup Language). Ce fichier contient tous les composants du profil « UML Context-Aware » et montre comment les sous profils « ClassUML », « UseCaseUML », « SequenceUML » et « ActivityUML » sont créés. Le fichier XML du profil proposé est composé de deux principales parties : l'entête du fichier et le corps du fichier.
Ces deux parties sont précédées d'une partie déclaration dans laquelle nous spécifions la version de XML, le type de codage et la version du profil. Ceci est exprimé au format XML comme suit :

```
<?xml version="1.0" encoding="UTF-8" ?>
<PROFILE version="1.0">
```

La partie de l'entête (HEADER) comprend toutes les informations qui identifient le profil comme : le nom du profil, le titre du profil, une description du profil, etc. L'entête du fichier se présente comme suit :

```
<HEADER>
<NAME> … </NAME>
<DISPLAYNAME> … </DISPLAYNAME>
<DESCRIPTION> … </DESCRIPTION>
<AUTOINCLUDE> … </AUTOINCLUDE>
</HEADER>
```

La partie de corps du fichier (BODY) comprend la définition de tous les éléments qui composent le profil. La définition de chaque élément est délimité par deux balises début et fin. La structure générale du fichier global est la suivante :

```
<?xml version="1.0" encoding="UTF-8" ?>
<PROFILE version="1.0">

<HEADER>
    . . .
</HEADER>

<BODY>
<STEREOTYPELIST>
<STEREOTYPE> . . . </STEREOTYPE>
</STEREOTYPELIST>
<TAGDEFINITIONLIST>
```

<TAGDEFINITION> . . . </TAGDEFINITION>
</TAGDEFINITIONLIST>
<DATATYPELIST>
<DATATYPE> . . . </DATATYPE>
</DATATYPELIST>
<DIAGRAMTYPELIST> … </DISPLAYNAME>
<DIAGRAMTYPE> … </DIAGRAMTYPE>
</ DIAGRAMTYPELIST> …
</BODY>

</PROFILE>

7.1.2. Création des stéréotypes

Tous les stéréotypes du profil proposé « UML Context-Aware » sont introduits dans le fichier global XML en indiquant, pour chacun d'eux, le nom, la classe de base UML et une brève description. D'une façon générale, la définition d'un stéréotype au format XML se fait comme suit :

```
<STEREOTYPE>
    <NAME> … </NAME>
    <DESCRIPTION> … </DESCRIPTION>
      <BASECLASSES>
<BASECLASS> … </BASECLASS>
      </BASECLASSES>
</STEREOTYPE>
```

Si nous prenons l'exemple de trois stéréotypes <<ContextClass>>, <<ContextActor>> et <<ContextObject>>, la séquence XML de leur création sera la suivante :

```
<STEREOTYPE>
    <NAME>ContextClass</NAME>
    <DESCRIPTION>stéréotype étendu à partir de la méta-classe UML 'Class'
    </DESCRIPTION>
      <BASECLASSES>
<BASECLASS>UMLClass</BASECLASS>
      </BASECLASSES>
</STEREOTYPE>

<STEREOTYPE>
    <NAME>ContextActor</NAME>
    <DESCRIPTION> stéréotype étendu à partir de la méta-classe UML 'Actor'
    </DESCRIPTION>
      <BASECLASSES>
<BASECLASS>UMLActor</BASECLASS>
```

```
        </BASECLASSES>
</STEREOTYPE>

  <STEREOTYPE>
        <NAME>ContextObject</NAME>
        <DESCRIPTION> stéréotype étendu à partir de la méta-classe UML 'Object'
        </DESCRIPTION>
         <BASECLASSES>
<BASECLASS>UMLObject</BASECLASS>
        </BASECLASSES>
</STEREOTYPE>
```

7.1.3. Création des contraintes

Les contraintes proposées peuvent être introduites en utilisant l'éditeur de contraintes (Constraint Editor) disponible dans le menu système de la plateforme StarUML. Pour chaque contrainte, nous spécifions un nom et un corps de contrainte qui contiendra la définition de cette contrainte (en langage naturel ou en langage OCL) (Figure 30).

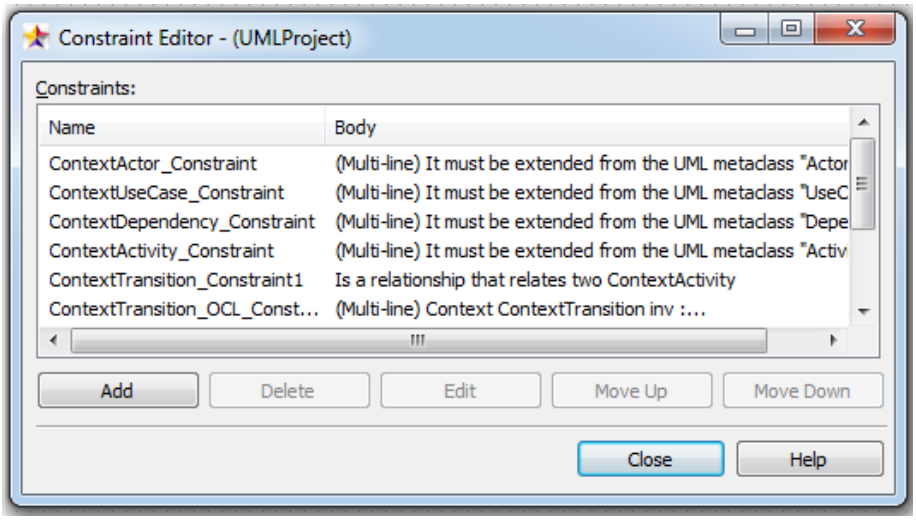


Figure 30. Edition des contraintes proposées.

7.1.4. Création des étiquettes

Les étiquettes représentent les attributs des stéréotypes proposés. Chaque étiquette est introduite dans le fichier global XML en précisant le nom de l'attribut, les valeurs possibles de cet attribut et la valeur par défaut. Ceci est réalisé selon le code suivant :

```
<TAGDEFINITION>
<NAME> … </NAME>
<TAGTYPE> … </TAGTYPE>
<DEFAULTDATAVALUE> … </DEFAULTDATAVALUE>
<LITERALS><LITERAL> … </LITERAL>
        <LITERAL> … </LITERAL>
```

<LITERAL> … </LITERAL>
<LITERAL> … </LITERAL>
</LITERALS>
</TAGDEFINITION>

L'implémentation des étiquettes proposées peut être vérifiée ou modifiée à l'aide de l'éditeur d'étiquettes (Tagged Value Editor) qui affiche toutes les informations relatives à chaque étiquette (nom, valeurs possibles, valeur par défaut) comme le montre la figure 31.

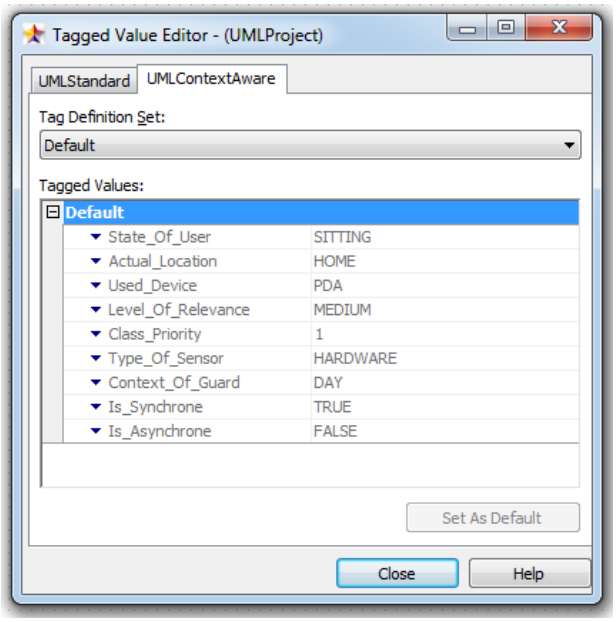


Figure 31. Edition des étiquettes proposées.

Pour plus de clarté de ce travail, nous donnerons quelques exemples d'implémentation des stéréotypes, des contraintes et des étiquettes qui composent ces sous profils (Table 22).

Part of profile file	**Example of expression (Use Case Diagram)**	**Example of expression (Activity Diagram)**	**Example of expression (Sequence Diagram)**
File header	<?xml version="1.0" encoding="UTF-8" ?> <PROFILE version="1.0"> <HEADER> <NAME>UsecaseUMLContextAware</NAME> <DISPLAYNAME>Usecase UML Context-Aware profile </DISPLAYNAME> <DESCRIPTION>Use case UML Profile for Context-Awareness Domain</DESCRIPTION> <AUTOINCLUDE>false</AUTOINCLUDE>	<?xml version="1.0" encoding="UTF-8" ?> <PROFILE version="1.0"> <HEADER> <NAME>ActivityUMLContextAware</NAME> <DISPLAYNAME>Activity UML Context-Aware profile </DISPLAYNAME> <DESCRIPTION>Activity UML Profile for Context-Awareness Domain</DESCRIPTION> <AUTOINCLUDE>false</AUTOINCLUDE> </HEADER>	<?xml version="1.0" encoding="UTF-8" ?> <PROFILE version="1.0"> <HEADER> <NAME>SequenceUMLContextAware</NAME> <DISPLAYNAME>Sequence UML Context-Aware profile </DISPLAYNAME> <DESCRIPTION>Sequence UML Profile for Context-Awareness Domain</DESCRIPTION> <AUTOINCLUDE>false</AUTOINCLUDE> </HEADER>
Stere-otypes	<STEREOTYPE> <NAME>ContextActor</NAME> <DESCRIPTION>stereotype extended from UML metaclass 'Actor' </DESCRIPTION> <BASECLASSES> <BASECLASS>UMLActor</BASECLASS> </BASECLASSES> </STEREOTYPE>	<STEREOTYPE> <NAME>ContextActivity</NAME> <DESCRIPTION>stereotype extended from UML metaclass 'Activity' </DESCRIPTION> <BASECLASSES> <BASECLASS>UMLActivity</BASECLASS> </BASECLASSES> </STEREOTYPE>	<STEREOTYPE> <NAME>ContextObject</NAME> <DESCRIPTION>stereotype extended from UML metaclass 'Object' </DESCRIPTION> <BASECLASSES> <BASECLASS>UMLObject</BASECLASS> </BASECLASSES> </STEREOTYPE>

Const-raints	*Context ContextActor inv: Attributes -> forAll(Attr1,Attr2 | Attr1<>Attr2 implies Attr1.NameOfAttribute<> Attr2.NameOfAttribute)*	*Context ContextTransition inv: Transition ->forAll (Tr| self.Activity -> includesAll(Tr.Activity))*	*Context ContextMessage inv: Operation -> forAll(Op1,Op2 | Op1<>Op2 implies Op1.NameOfAttribute<> Op2.NameOfAttribute)*
Tagged values	<TAGDEFINITION> <NAME>Used_Device</NAME> <TAGTYPE>Enumeration</TAGTYPE> <DEFAULTDATAVALUE>PDA</DEFAULTDATAVALUE> <LITERALS><LITERAL>PDA</LITERAL> <LITERAL>PC</LITERAL> <LITERAL>LAPTOP</LITERAL>	<TAGDEFINITION> <NAME>Context_of_Guard</NAME> <TAGTYPE>Enumeration</TAGTYPE> <DEFAULTDATAVALUE>DAY</DEFAULTDATAVALUE> <LITERALS><LITERAL>DAY</LITERAL> <LITERAL>NIGHT</LITERAL> <LITERAL>INDOOR</LITERAL>	<TAGDEFINITION> <NAME>Is_Synchrone</NAME> <TAGTYPE>Enumeration</TAGTYPE> <DEFAULTDATAVALUE>FALSE</DEFAULTDATAVALUE> <LITERALS><LITERAL>FALSE</LITERAL> <LITERAL>TRUE</LITERAL> </LITERALS> </TAGDEFINITION>

Table 22. Extraits du fichier XML [Benselim, 2017].

7.2. Etude de cas (Exemple illustratif)

7.2.1. Description et besoins du système

Pour démontrer la faisabilité du profil proposé, nous considérons un système décrit par un simple exemple qui illustre l'utilisation des concepts et notations introduites. Dans cet exemple, nous supposons qu'un même utilisateur essaye d'accéder à la même application « utilisation des médicaments » mais dans différentes situations. Cette variation de situations est due aux changements continus que peut subir l'utilisateur et qui affectent directement le contexte d'utilisation incluant le temps, l'endroit, l'état de l'utilisateur, les ressources disponibles, les personnes avoisinantes, etc. Pour notre exemple, l'utilisateur peut être un patient, un médecin, une infirmière, un pharmacien, un professeur, un étudiant, etc.

L'application concernée offre plusieurs services à distance comme : la consultation de la liste des médicaments disponibles, l'achat de médicaments, consulter les notices d'utilisation, connaitre les précautions de prise des médicaments, chercher un médecin, chercher une infirmière, etc. Nous supposons aussi, dans cet exemple, que l'utilisateur est en voyage de son pays d'origine (A) vers un pays de destination (B) (Figure 32). Pendant son voyage, il tente d'utiliser l'application « utilisation des médicaments » en espérant d'avoir des informations concernant les médicaments qu'il prend.

A partir de cette brève description, nous remarquons que notre système est caractérisé par trois principaux endroits (ou lieus):

- Lieu de départ (pays d'origine A)
- Lieu d'arrivée (pays de destination B)
- Lieu des frontières (entre les pays A et B)

Bien sûr, chaque endroit possède ses propres caractéristiques et ses propres règlements ; par conséquent, le déplacement d'un endroit à un autre provoque le changement d'une ou de plusieurs de ces caractéristiques entraînant, ainsi, une modification des éléments qui composent le contexte d'utilisation de la situation courante. Dans ce cas la situation courante a changé de contexte et nous disons que le système a transité vers une nouvelle situation avec de nouveaux paramètres du contexte d'utilisation.

Il faut noter qu'une telle situation peut être principalement influencée par plusieurs facteurs et propriétés telles que :

- Propriétés liées à l'utilisateur (identité, profil particulier, comportement, préférences, etc.)
- Propriétés liées à l'application (matériel, logiciel, réseaux, mode d'interaction, etc.)
- Propriétés liées à l'environnement (temps, localisation, personnes avoisinantes, objets proches, ressources disponibles, paramètres régionaux, etc.)

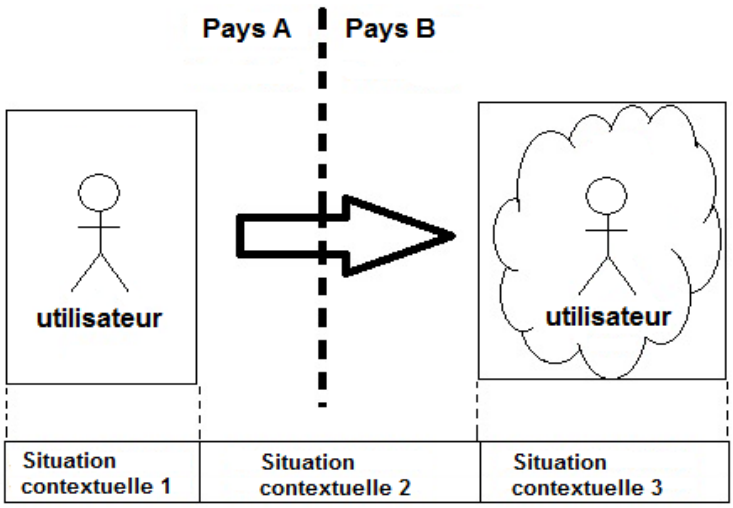


Figure 32. Vue générale de l'exemple illustratif [Benselim, 2017].

Chacune de ces propriétés doit être représentée par une notation adéquate du langage UML.

Les changements continus déclenchent de nouvelles situations contextuelles, mais ceci ne doit ni inquiéter l'utilisateur ni déstabiliser le bon fonctionnement de l'application parce que cette dernière est conçu pour être sensible au contexte et pour être adaptée à supporter la principale exigence ubiquitaire (disponibilité de l'information n'importe où et n'importe quand).

Dans la table 23, nous mettons en évidence trois situations contextuelles pour lesquelles l'utilisateur (supposé être un patient) doit faire face. Chaque situation est caractérisée par une multitude d'instances (ou de valeurs) des éléments contextuels qui définissent le contexte d'utilisation de cette situation. Dans ce cas, nous allons considérer plusieurs scénarios pour notre système parce que la majorité de ses propriétés sont continuellement changeables. Les différents scénarios peuvent être obtenus en appliquant toutes les combinaisons possibles sur les éléments contextuels de la table 23.

Eléments contextuels	**Situation contextuelle 1** ***(Pays A)***	**Situation contextuelle 2** ***(Frontières A-B)***	**Situation contextuelle 3** ***(Pays B)***
Utilisateur	Patient	Patient	Patient
Etat de l'utilisateur	Assis, Dormant	Marchant, Déplaçant	Assis, Dormant

Endroit	domicile, Hôpital	Bureau des frontières	Pharmacie, Hôtel
Langage	Arabe, Français	Arabe, Français, Anglais	Anglais, Français
Matériel utilisé	PC, PDA	PDA, Téléphone mobile	PDA, Téléphone mobile
Temps	Jour	Jour	Nuit
Devises utilisées	Euro	Euro, Dollar	Dollar
Réseaux existants	Internet, LAN	Internet	Internet
Personnes avoisinantes	Médecin, Infirmière, Parent	Parent, Ami	Médecin, Ami Pharmacien
Ressources disponibles	Téléphone, Webcam, Imprimante, smart TV	Téléphone, Fax	Fax, Imprimante
Autres objets proches	Matériel médical	Voiture, bus	Table, Chaise

Table 23. Comparaison de trois situations contextuelles différentes [Benselim, 2013].

7.2.2. Implémentation des notations de l'exemple

Pour modéliser cet exemple, nous allons dessiner les diagrammes UML qui illustrent les étapes du processus de développement. Ces digrammes seront capables de prendre en charge toutes les informations contextuelles du système, et ce, en utilisant les concepts et notations du profil UML proposé « UML Context-Aware ».

Après avoir achevé le processus d'implémentation qui comprend l'écriture des fichiers XML et l'extension du StarUML, les nouvelles notations du profil proposé sont prêtes à être utiliser afin de pouvoir modéliser le système de notre exemple.

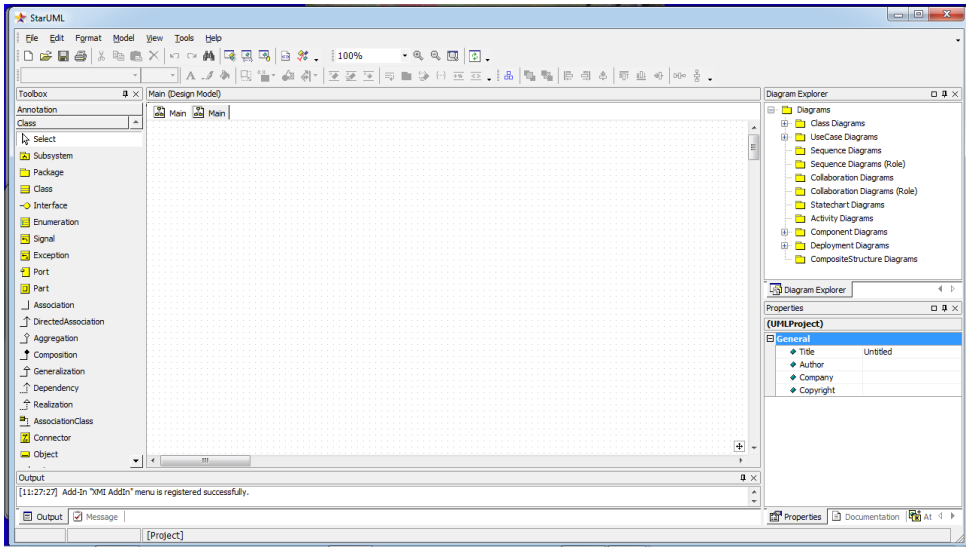


Figure 33. Menu principal de la plateforme StarUML.

Premièrement, nous lançons le menu général de la plateforme StarUML (Figure 33) ; ensuite, nous choisissons l'option du gestionnaire de profils (profile manager) en suivant la séquence d'options suivante : Model > Profile > 'profil manager'

Nous obtenons la liste des profils disponibles et prêts à être utiliser comme (Figure 34):

- C# Profile,
- C++ Profile,
- EJB Profile,
- JAVA Profile,
- UML Context-Aware Profile (notre profil proposé)

Il faut noter, ici, que le profil « UML Standard Profile » est déjà inclus comme étant le profil par défaut.

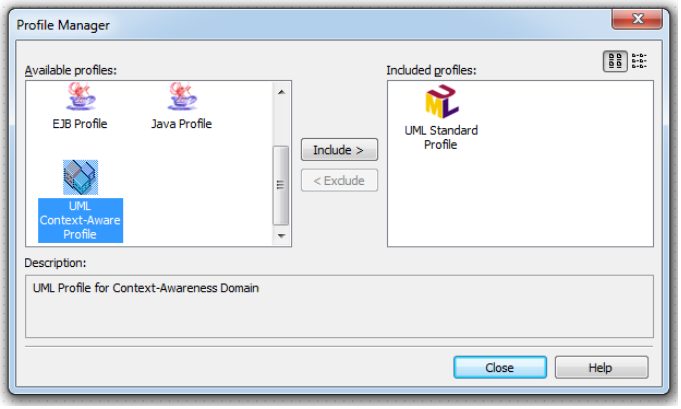


Figure 34. Le gestionnaire de profils de la plateforme StarUML.

Ici, nous choisissons le profil désiré (pour notre cas, c'est le profil « UML Context-Aware ») et nous cliquons sur le bouton « Include ». Cette opération permettra de charger tous les concepts et notations (stéréotypes, contraintes et étiquettes) liées au profil choisi. Après ce chargement, le profil est complètement prêt à être utilisé.

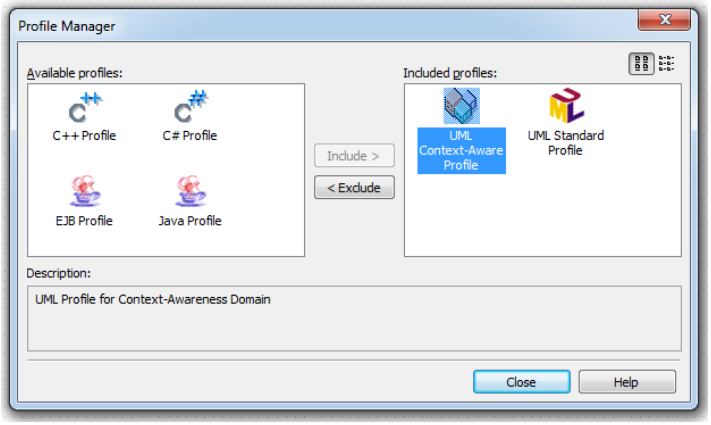


Figure 35. Chargement du profil « UML Context-Aware ».

La figure 35 montre le gestionnaire de profil ‘profile manager’ après chargement du nouveau profil « UML Context-Aware ». Ensuite, nous choisissons le diagramme UML voulu à partir du volet ‘model explorer’ afin de charger tous les composants relatifs à ce diagramme (y compris les nouvelles notations proposées.

Enfin, nous commençons à construire nos diagrammes qui vont représenter le système à modéliser, et ce, en utilisant les notations UML contenues dans le profil proposé « UML Context-Aware ».

7.2.3. Construction des diagrammes en utilisant le profil « UML Context-Aware »

Dans cette section, nous exposons les diagrammes UML construits à partir des nouvelles notations incluses dans le profil UML proposé « UML Context-Aware ». Il faut rappeler que ce profil propose les notations pour représenter quatre diagrammes UML :

- Les diagrammes de classes
- Les diagrammes de cas d’utilisation
- Les diagrammes de séquences
- Les diagrammes d’activités

Le choix de ces diagrammes UML parmi tous les autres est très justifié parce qu’ils peuvent couvrir la modélisation des différentes vues d’un système comme la vue fonctionnelle, la vue statique et la vue dynamique. Les nouvelles notations permettent d’adapter et de personnaliser les notations existantes du langage UML (initialement plus générales) au domaine particulier de la sensibilité au contexte (Context-Awareness Domain).

Ayant la liste des éléments qui composent le contexte d’utilisation comme illustrés dans la table 23, nous pouvant identifier toutes les classes de ce système. La figure 36 illustre un diagramme de classes qui modélise le contexte d’utilisation en utilisant les notations proposées. Il faut noter que ce diagramme met en évidence, à titre indicatif seulement, les principales classes et associations du système sous formes de stéréotypes. Ces derniers sont capables de représenter les classes particulières et les associations spécifiques de l’exemple à étudier. La définition de chaque stéréotype <<ContextClass>> est limitée par des conditions spécifiques (les contraintes) et peut utiliser des attributs appropriés (les étiquettes) suivant le changement du contexte d’utilisation. Les associations entre classes sont des stéréotypes <<ContextAssociation>> qui sont capables de relier les stéréotypes de classes <<ContextClass>>.

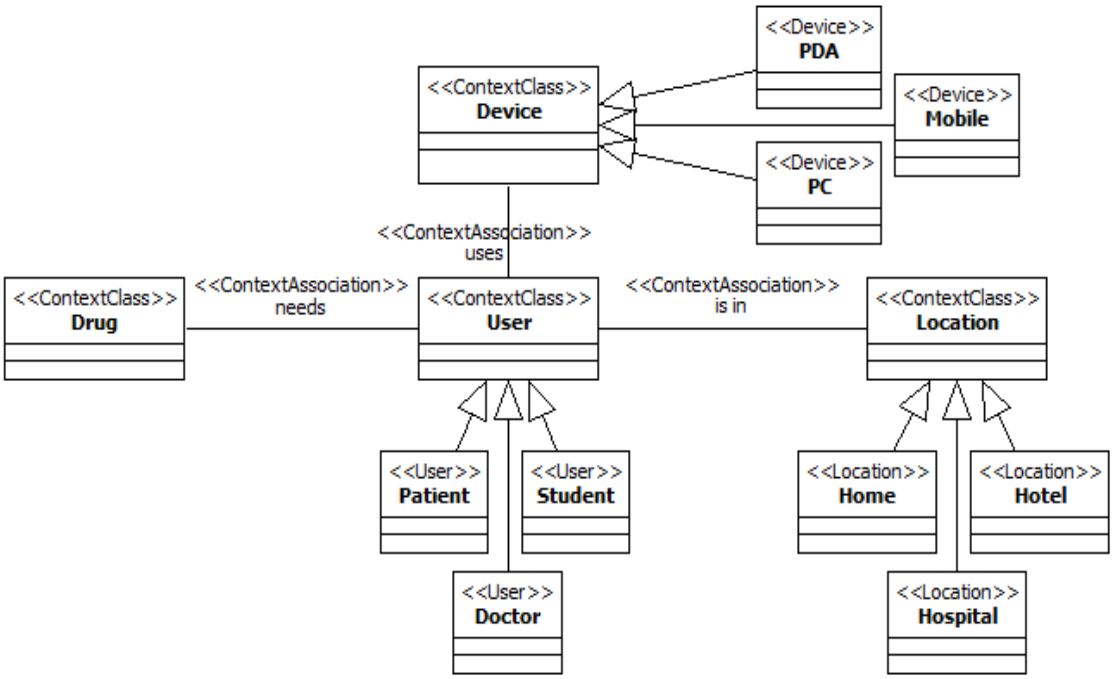


Figure 36. Diagramme de classes de l'exemple illustratif [Benselim, 2013].

Suivant la description du système, nous pouvons construire un diagramme de cas d'utilisation qui montre comment un utilisateur nomade ou un acteur mobile peut être représenté en utilisant le stéréotype proposé <<ContextActor>>. Ce stéréotype est spécifique pour les acteurs qui possèdent des propriétés variables. Dans notre exemple, les acteurs comme le patient, le médecin et le pharmacien peuvent être représentés à l'aide du stéréotype <<ContextActor>> parce que leurs caractéristiques variables (état, temps, endroit, matériel, etc.) entrainent des situations complètement différentes lors de l'exécution de l'application.

Dans le diagramme proposé de la figure 37, nous illustrons deux cas d'utilisation "To consult" et "To buy drugs" utilisant le stéréotype <<ContextUseCase>>.

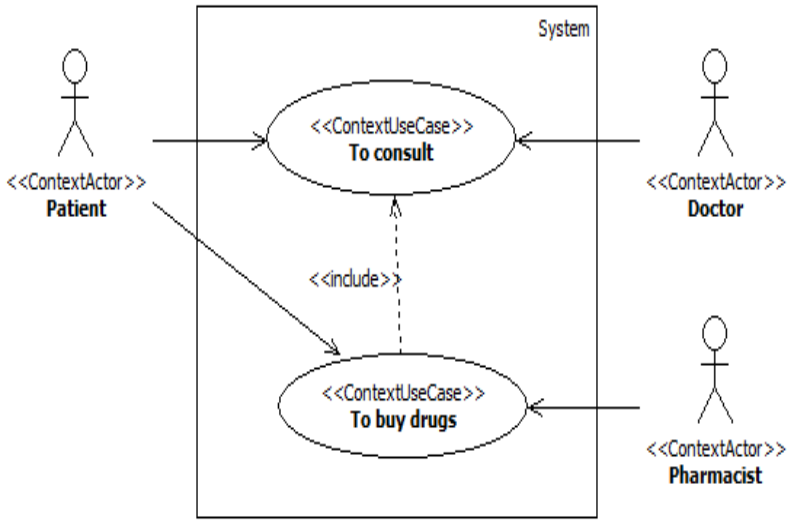


Figure 37. Diagramme de cas d'utilisation de l'exemple illustratif [Benselim, 2017].

Dans la figure 38, nous présentons un diagramme de séquences qui modélise une partie de notre exemple. Les principales notations de ce diagramme sont des stéréotypes issus du profil proposé « UML Context-Aware ». Les objets comme le patient, le médecin et le médicament sont représentés par des stéréotypes <<ContextObject>> qui possèdent leurs propres lignes de vie désignées par les stéréotypes <<ContextLifeLine>>.

Chacun de ces objets spécifiques peut envoyer et recevoir des messages contextuels représentés par des stéréotypes <<ContextMessage>>. Pour les stéréotypes <<ContextMessage>>, le mode de synchronisation est pris en charge par les étiquettes 'Is_Synchrone' et 'Is_Asynchrone' attachées à ces stéréotypes et qui indiquent si les messages sont synchrone ou asynchrone.

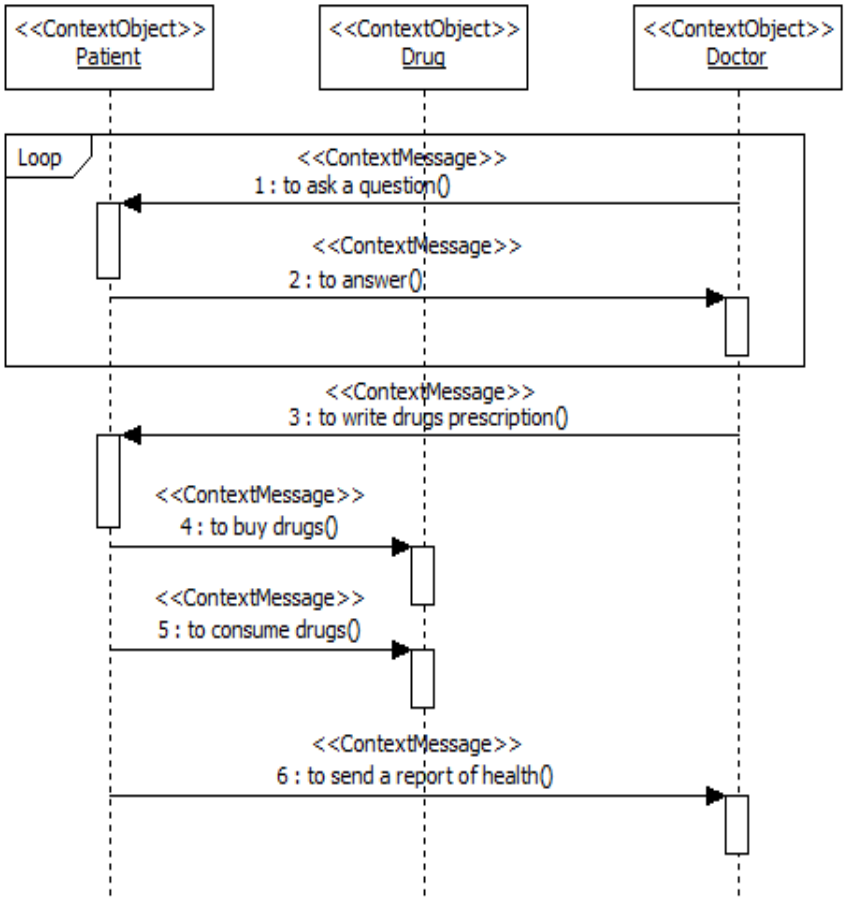


Figure 38. Diagramme de séquences de l'exemple illustratif [Benselim, 2017].

La figure 39 présente un diagramme d'activités dans lequel nous avons utilisé les nouvelles notations du profil proposé pour représenter un scénario de notre exemple.

Les stéréotypes <<ContextActivity>> sont utilisés pour modéliser les actions contextuelles qui correspondent aux opérations d'un stéréotype <<ContextActor>>. Le passage d'une

activité vers une autre est assuré par le stéréotype <<ContextTransition>> qui peut être restreinte par utilisation des étiquettes proposées suivant le contexte de cette transition.

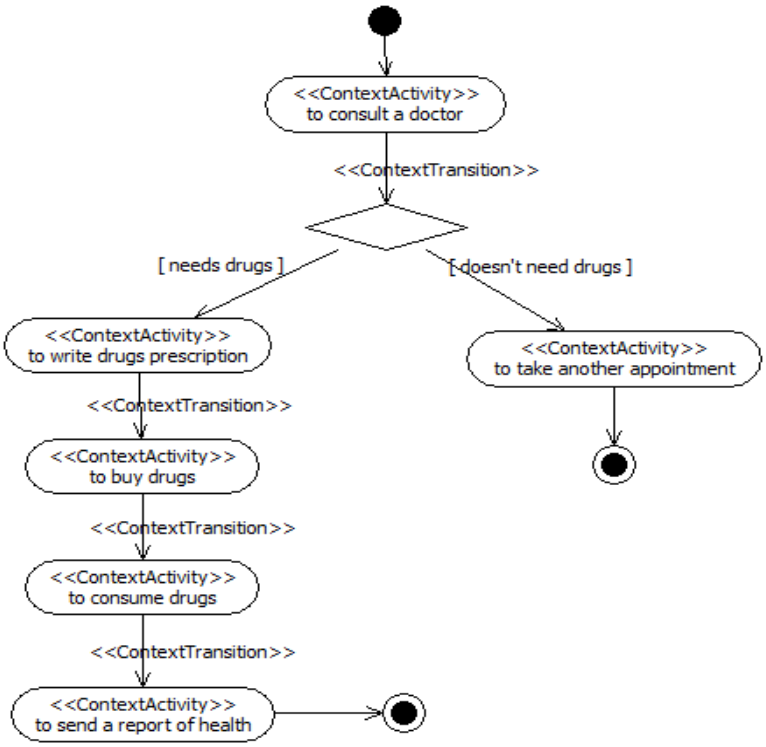


Figure 39. Diagramme d'activités de l'exemple illustratif [Benselim, 2017].

Conclusion

L'adaptation contextuelle des applications et la personnalisation des informations peuvent être garantis si nous profitons de toutes les opportunités offertes par la notion de « profil UML ». Dans ce chapitre, nous avons présenté toutes les étapes nécessaires à l'implémentation du profil UML proposé. Ces étapes décrivent la création des nouvelles notations comme les stéréotypes, les contraintes et les étiquettes. L'implémentation a été réalisée sous la plateforme StarUML. Ensuite, nous avons appliqué tous les concepts proposés sur un exemple illustratif en les utilisant à construire plusieurs diagrammes UML (classes, cas d'utilisation, séquences et activités).

CONCLUSION GÉNÉRALE ET PERSPECTIVES

Conclusion générale

Dans le champ de l'informatique ubiquitaire, les constituants du contexte d'utilisation possèdent une caractéristique commune à savoir la variation et le changement. Aussi, chaque modification (de l'état, de la position, du nombre, etc.) des objets proches peut affecter directement l'état de la situation courante de l'utilisateur. D'autre part, chaque utilisateur peut présenter des préférences particulières concernant le contenu, la présentation ou l'affichage des informations. La prise en compte des informations contextuelles pendant la conception des applications nous permet de cerner toutes les variations du contexte d'utilisation qui influencent l'exécution d'une tache par un utilisateur nomade et, aussi, de prévoir les conditions contextuelles de plusieurs situations jusque là inconnues. En effet, par la modélisation des données contextuelles et la formalisation d'un modèle purement contextuel, nous pouvons assurer la prise en considération du contexte d'utilisation d'une application à travers les transformations subies par ce nouveau modèle.

Dans cette étude, nous avons cherché à proposer des moyens simples et pratiques qui peuvent assurer la prise en charge du contexte d'utilisation dans les systèmes d'information ubiquitaires. Ce qui permettra de concevoir et réaliser des systèmes d'information adaptés et personnalisés pouvant, d'une part, vérifier les nouvelles caractéristiques de l'informatique ubiquitaire et, d'autre part, satisfaire les exigences des utilisateurs nomades.

Notre objectif principal est de garantir l'adaptation contextuelle des applications et de fournir des informations pertinentes et personnalisées suivant les préférences des utilisateurs. Cet objectif peut être atteint à travers notre proposition qui est définie par trois points essentiels. Premièrement, la séparation des préoccupations contextuelles par rapport aux autres préoccupations (métiers et techniques). Cette séparation s'appuie sur une nouvelle notion introduite, à savoir le processus 3TUP. Deuxièmement, la personnalisation du langage UML en proposant un profil UML spécialisé au domaine de la sensibilité au contexte. Le troisième point concerne la proposition d'une nouvelle vision de l'approche MDA qui se distingue par la possibilité de prendre en charge les contraintes contextuelles en insérant (par opération de fusion de modèles) un modèle contextuel dans son processus de développement.

Ce travail nous a permis de mieux comprendre l'importance de la prise en compte du contexte d'utilisation dans les systèmes modernes et de contribuer, ainsi, à l'adaptation des applications et à la personnalisation des informations pour les systèmes d'information ubiquitaires. Cette importance découle des changements continuels subis par la situation courante de l'utilisateur de plus en plus mobile et exigeant.

Enfin, nous espérons que ce modeste travail sera un point de départ et une première brique pour d'autres études qui cherchent à satisfaire les besoins et les préférences de l'utilisateur en lui fournissant l'information qu'il désire en temps, en espace et en qualité.

Perspectives

Notre vision s'étend jusqu'à la réalisation de systèmes d'informations « contextuellement paramétrables », c'est-à-dire que les applications conçues doivent renfermer des variables représentant toutes les informations contextuelles possibles et qui seront introduites selon les contraintes et les profils des utilisateurs. Les systèmes d'information ainsi obtenus pourront satisfaire, encore plus, les conditions d'adaptation et seront capables de fournir des informations personnalisées et pertinentes. Ce qui permet à l'utilisateur de s'adapter lui-même (son profil et ses préférences) et adapter l'application au contexte d'utilisation (environnement, utilisateurs voisins,…).

Nous prévoyons de poursuivre cette étude par la construction d'un « mapping contextuel ». Celui-ci comprendra de nouvelles règles de transformation (relatives au contexte d'utilisation) qui guideront le passage d'un modèle PIM vers un modèle PSM.

Aussi, nous envisageons de construire un « méta-modèle contextuel ». Ce méta-modèle permettra aux concepteurs de s'y référencer pour produire des modèles contextuels plus conformes et plus structurellement homogènes.

D'un autre côté, nous comptons se diriger vers la gestion des profils dans les environnements ubiquitaires pour essayer d'introduire les notions d'adaptation et de personnalisation proposées lors de cette thèse et démonter, ainsi, leur importance.

BIBLIOGRAPHIE

Références

[Abbas, 2008] Abbas K., « Système d'Accès Personnalisé à l'Information : Application au domaine médical ». Thèse en informatique, Institut national de sciences appliquées de Lyon, 2008.

[Abowd, 1997] Abowd G.D., Dey A.K., Orr R., Brotherton J. « Context-Awareness in Wearable and Ubiquitous Computing ». 1st International Symposium on Wearable Computers (1997), pp 179-180, 1997.

[Aldawud, 2003] Aldawud O., Elrad T., Bader A. "UML profile for aspect-oriented software development", *The Third International Workshop on Aspect Oriented Modelling*, Boston, USA, 2003.

[Alonso, 2004] Alonzo G., Casati F., Kuno H., Machiraju V. « Web Services: Concepts, Architectures and Applications ». Springer-Verlag, 1rst édition, 2004.

[Amirat, 2009] Amirat A., Oussalah M., "Towards an UML profile for the description of software architecture", *Proceedings of International Conference on Applied Informatics (ICAI'09)*, Bordj BouAreridj, Algeria, pp. 226-232, 2009.

[Ayed, 2007] Ayed D., Delanote D., Berbers Y., "MDD approach for the development of context-aware applications," In: Kokinov, B., Richardson, D.C., Roth-Berghofer, T.R., Vieu, L. (eds.) CONTEXT 2007. LNCS (LNAI), vol. 4635, pp. 15–28. Springer, Heidelberg, 2007.

[Baclawski, 2001] Baclawski K., Kokar M.K, Kogut P.A, Hart L., Smith J., Holmes W.S., Letkowski J., Aronson M.L., "Extending UML to Support Ontology Engineering for the Semantic Web," In: Gogolla, M., Kobryn, C. (eds.) UML 2001.LNCS, vol. 2185, pp. 342–360. Springer, Heidelberg 2001.

[Banavar, 2000] Banavar G., Beck J., Gluzberg E., Munson J., Sussman J., Zukowski D. «Challenges :an application model for pervasive computing». In proceedings of the 6th annual international conference on mobile computing and networking, pp 266-274, ACM Press, 2000.

[Banavar, 2002] Banavar G., Bernstein A. « Software infrastructure and design challenges for ubiquitous computing applications» commun. ACM, 45(12) pp 92-96, 2002.

[Baresi, 2001] Baresi L., Garzotto F., Paolini P., "Extending UML for modeling web applications," In Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS 2001, Hawaii, USA), pp 1285 -1294, 2001.

[Bauer, 2003] Bauer J. «Identification and modeling of contexts for different information Scenarios in air traffic», PhD thesis. March. 2003.

[Benselim, 2009a] Benselim M.S., Seridi-Bouchelaghem H. «Contextual adaptation of ubiquitous information systems», In Proceedings of the 2nd International Conference on Multimedia Computing and Systems (ICMCS'09) Morocco, April, 2009.

[Benselim, 2009b] Benselim M.S., Seridi-Bouchelaghem H. «Une approche pour le développement d'applications sensibles au contexte», Congrès INFORSID 2009, Toulouse, France, Mai 2009.

[Benselim, 2009c] Benselim M.S., Seridi-Bouchelaghem H. «Développement d'applications sensibles au contexte en utilisant l'approche Model Driven Architecture», ICAI (International Conference on Applied Informatics) 2009, Bordj-BouArreridj, Algérie, Novembre 2009.

[Benselim, 2011a] Benselim M.S., Seridi-Bouchelaghem H. "Development of context-aware applications in ubiquitous information systems", *In Proc. of the 13th International Conference on Enterprise Information Systems, (ICEIS' 2011)*, Beijing, China, Vol. 3, pp 223-228, 2011.

[Benselim, 2011b] Benselim M.S., Seridi-Bouchelaghem H. "Modelling context with extended UML",In proc. WCIT'2011 (World Conference on Information Technology), 23-27 November 2011Antalya, Turquie.

[Benselim, 2012a] Benselim M.S., Seridi-Bouchelaghem H. "Extended UML for the Development of Context-Aware Applications" , *The 4th international conference on Networked Digital Technologies NDT'2012* ,CCIS Vol. 293, pp 33-43. Springer-Verlag Berlin Heidelberg, 2012.

[Benselim, 2012b] Benselim M.S., Seridi-Bouchelaghem H. "Modelling context with extended UML", AWERProcedia Information Technology and Computer Science Journal (1), Vol. 01, pp 566-571, 2012.

[Benselim, 2013] Benselim M.S., Seridi-Bouchelaghem H. "Extending UML Class Diagram Notation for the Development of Context-aware Applications" *Journal of Emerging Technologies in Web Intelligence, (JETWI),* Vol. 5, No 1, pp 35-44, 2013.

[Benselim, 2017] Benselim M.S., Seridi-Bouchelaghem H. "Towards A UML Profile for Context-Awareness Domain" International Arab *Journal of Information and Technology, (IAJIT),* Vol. 14, No 3, May 2017, In Press (First Online on May 2016).

[Bouzeghoub, 2005a] Bouzeghoub M., Kostadinov D., « Personnalisation de l'information : Aperçu de l'état de l'art et définition d'un modèle flexible de définition de profils », Actes de la 2nde Conférence en Recherche d'Information et Applications, CORIA, Grenoble, France, pp. 201-218, 2005.

[Bouzeghoub, 2005b] Bouzeghoub M., « Action spécifique sur la personnalisation de l'information », CNRS-AS98 / RTP9, Université de Versailles, France, 2005.

[Bouzeghoub, 2014] Bouzeghoub M., Kostadinaov D., « Personnalisation de l'information : aperçu de l'état de l'art et définition d'un modèle flexible de profils utilisateurs ». Séminaire Marketing Digital, 05 juin 2014.

[Boystov, 2011] Boystov A. "Context Reasoning, Context Prediction and Proactive Adaptation in Pervasive Computing Systems". These en informatique, université Lulea, Suède, 2011.

[Brockmans, 2006] Brockmans S., Haase P., Hitzler P., Studer R., "A Metamodel and UML Profile for Rule-Extended OWL DL Ontologies," In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 303–316. Springer, Heidelberg 2006.

[Brown, 1997] Brown P.J., Bovey J.D., Chen X. «Context-Aware Applications: From the Laboratory to the Marketplace». IEEE Personal Communications, 4(5) , pp. 58-64, 1997.

[Brown, 1998] Brown P.J. « Triggering Information by Context ». Personal Technologies, 2(1) (1998), pp. 1-9, 1998.

[Canals, 2002] Canals G., Nigay L., Pucheral P. « Mobilité : accès aux données et interaction homme-machine » Actes des deuxièmes assises nationales du GdR I3, Information – Interaction – Intelligence. Nancy, France, Décembre 2002. Cépadues Éditions, 2002.

[Chaari, 2005] Chaari T., Laforest F., « L'adaptation dans les systèmes d'information sensibles au contexte d'utilisation: approche et modèles. Le projet SECAS : Simple Environment for Context Aware Systems », Cinquièmes journées scientifiques des jeunes chercheurs en génie électrique et informatique (GEI 2005), Sousse, Tunisie, 2005.

[Chaari, 2007] Chaari T., « Adaptation d'applications pervasives dans des environnements multi-contextes ». Thèse en informatique, Institut national de sciences appliquées de Lyon, 2007.

[Chassot, 2006] Chassot C., Guennoun K., Drira K., Exposito E., LonzesA., « Towards autonomous management of QoS through model-driven adaptability in communication-centric systems ». International Transactions on systems science and applications (ITSSA), special issue on self-organizing communications, Vol. 02, N° 03, pp 255-264, September 2006.

[Chen, 2000] Chen G., Kotz D., « A survey of context-aware mobile computing research ». Dartmouth Computer Science Technical Report TR2000-381, 2000.

[Da Silva, 2004] Da Silva V.T., De Lucena C.J. P., "Extending UML to Model Multi-Agent Systems ", PUC-Rio Inf.MCC 08/04, March, 2004.

[Davies, 1998] Davies N., Mitchell K., Cheverst K., Blair G. « Developing a Context Sensitive Tourist Guide ». 1st Workshop on Human Computer Interaction with Mobile Devices, GIST Technical Report G98-1, 1998.

[De Castro, 2004] De Castro V., Marcos E. and Vela, B. « Representing WSDL with Extended UML » Revista Columbiana de Computation, vol. 5, 2004.

[Dey, 1997] Dey A.K., Abowd G.D. « CyberDesk: The Use of Perception in Context-Aware Computing ». 1st Workshop on Perceptual User Interfaces (1997), pp. 26-27, 1997.

[Dey, 1998] Dey A.K. «Context-Aware Computing: The CyberDesk Project». AAAI 1998 Spring Symposium on Intelligent Environments, Technical Report SS-98-02, pp. 51-54, 1998.

[Dey, 1999a] Dey A.K., Abowd G.D., Wood A. «CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services». Knowledge-Based Systems, 11, pp. 3-13, 1999.

[Dey, 1999b] Dey A.K., Abowd G.D. «Towards a Better Understanding of context and context-awareness». Technical Report GIT-GVU-99-22, Georgia Institute of Technology, College of Computing, June 1999.

[Dey, 2001] Dey A.K., Abowd G.D., Salber D. «A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications». Human-Computer Interaction vol.16, 2001, pp. 97-166, 2001.

[Djuric, 2004] Djurić D., Gašević D., Devedžić V., Damjanović V. "A UML Profile for OWL Ontologies*", in Proceedings of the Workshop on Model Driven Architecture: Foundations and Applications*, Linköping, Sweden, 2004.

[Donsez, 1999] Donsez D., Jean S., Lecomte S. « Evolution du rôle de la carte à microprocesseur dans les systèmes d'information distribués ». Technical Report 99-11, LIFL, Université Lille 1, France, Juin 1999.

[Frankel, 2003] Frankel D.S. «Model Driven Architecture: Applying MDA to Enterprise Computing». Wiley and OMG Press, 2003.

[Fuentes, 2008] Fuentes L., Gamez N., Sanchez P. "Aspect-Oriented Executable UML Models for Context-Aware Pervasive Applications", *5th International Workshop on Model-based Methodologies for Pervasive and Embedded Software*, MOMPES 2008, pp. 34-43, Budapest , Hungary, 2008.

[Gensel, 2008] Gensel J., Villanove-Olivier M., Kirsch-Pinheiro M., « Modèles de contexte pour l'adaptation à l'utilisateur dans des Systèmes d'Information Web Collaboratifs ». 8èmes Journées Francophones, Extraction et Gestion des Connaissances (Modélisation utilisateur et personnalisation d'interfaces Web), Sophia Antipolis, 29 janvier 2008.

[Gherbi, 2006] Gherbi A., Khendek F. "UML Profiles for Real-Time Systems and their Applications", in Journal of Object Technology, Vol. 5, No. 4, pp. 149–169, 2006.

[Grassi, 2004] Grassi V., Mirandola R., Sabetta A., "A UML Profile to Model Mobile Systems", LNCS Vol. 3273, pp 128-142, 2004.Springer-Verlag Berlin Heidelberg, 2004.

[Hadj-Kacem, 2005] Hadj-Kacem M., Milady MN. "Towards a UML profile for the description of dynamic software architectures", *The Conference on Component-Oriented Enterprise Applications, COEA 2005*, Augsburg, Germany, pp. 25-39, 2005.

[Hansmann, 2003] Hansmann U. «Pervasive Computing: The Mobile World», Springer Professional Computing, 2003.

[Heckel, 2003] Heckel R., Lohmann M., Thone S. "Towards a UML Profile for Service-Oriented Architectures", *In: Proc. of Workshop on Model Driven Architecture: Foundations and Applications (MDAFA), CTIT Technical Report TR-CTIT-03-27*. University of Twente Enschede, The Netherlands, 2003.

[Hsu, 2011] Hsu I.C., "An Architecture of Mobile Web 2.0 Context-aware Applications in Ubiquitous Web", *Journal of Software*, Vol. 6, No 4, pp. 705-715, April 2011.

[Hsu, 2012] Hsu I.C., "Extending UML to model Web 2.0-based context-aware applications", *Journal of Software*, Vol. 42, issue 10, pp. 1211-1227,October 2012.

[Hsu, 2013] Hsu I.C., "Visual modelling for Web 2.0 applications using model driven architecture approach", *Journal of Simulation Modelling Practice and Theory*, Vol. 31, pp. 63-76, February 2013.

[Hull, 1997] Hull R., Neaves P., Bedford-Roberts J. «Towards situated computing». In Proceedings of the First International Symposium on Wearable Computers, Cambridge, Massachusetts, October 1997. IEEE Computer Society Press. 1997.

[Hussein, 2010] Hussein M., Han J., Colman A., "Integrated Modelling of Context-Aware Adaptive Software Systems", Swinburne University of Technology (Australia), TR C3-516_02, 2010.

[Indulska, 2003] Indulska J., Robinson R., Rakotonirainy A., Henricksen K. «Experiences in Using CC/PP in context-Aware Systems ». In Proceedings 4th International Conference On Mobile Data Management, Melbourne. Australia, pp. 247-261, January 2003.

[Jonston, 2004] Johnston S., "Rational UML Profile for Business Modeling", Rational software, IBM, March 2004.

[Jrad, 2008] Jrad Z., Anli A., Aufaure M.A., « Modèle contextuel pour la personnalisation ». 8èmes Journées Francophones, Extraction et Gestion des Connaissances (Modélisation utilisateur et personnalisation d'interfaces Web), Sophia Antipolis, 29 janvier 2008.

[Kandé, 2000] Kandé MM., Strohmeier A., "Towards a UML profile for software architecture descriptions", LNCS Vol. 1939, pp. 513-527, Springer-Verlag Berlin Heidelberg, 2000.

[Ketfi, 2002] Ketfi M., Belkhatir N., Cunnin P.Y., «Automatic adaptation of component-based software, issues and experiences». SERP'2002, Las Vegas, Nevada, USA, 2002.

[Kirsch-Pinheiro, 2006] Kirsch-Pinheiro M., « Adaptation Contextuelle et Personnalisée de l'Information de Conscience de Groupe au sein des Systèmes d'Information Coopératifs ». Thèse en informatique. Université Joseph-Fourier, Grenoble I, 2006.

[Kolovos, 2006] Kolovos D.S, Paige R.F., Polack F.A.C. «Merging Models with the Epsilon Merging Language (EML) ». In Proc. ACM/IEEE 9th International Conference on MDE Languages and Systems (Models/UML 2006), Italy, 2006.

[Korherr, 2006] Korherr B., List B., "Extending the UML 2 Activity Diagram with Business Process Goals and Performance Measures and the Mapping to BPEL", LNCS Vol. 4231, pp. 7-18, Springer-Verlag Berlin Heidelberg, 2006.

[Korteum, 1998] Korteum G., Segall Z., Bauer M. « Context-Aware, Adaptive Wearable Computers as Remote Interfaces to "Intelligent" Environments ». 2nd International Symposium on Wearable Computers (1998) pp. 58-65, 1998.

[Kostadinaov, 2014] Kostadinaov D., « Personnalisation de l'information et gestion des profils utilisateurs ». Séminaire Marketing Digital, 05 juin 2014.

[Kouadri, 2008] Kouadri S. « Advances in Ubiquitous Computing: Future Paradigms and Directions ». IGI Publishing, 2008.

[**Koutrika, 2004]** Koutrika G. "Personnalization of Queries in Database Systems", USA, April, 2004.

[Laassiri, 2013] Laassiri J. « Cours de La Modélisation Conceptuelle De Système D'information », Université Ibn Tofail, Faculté des Sciences, Département d'Informatique, Kenitra, 2013.

[List, 2005] List B., Korherr B. "A UML 2 profile for business process modelling", LNCS Vol. 3770, pp. 85-96, Springer-Verlag Berlin Heidelberg, 2005.

[Lombardi, 2014] Lombardi S. « Context-awareness and context modeling». Ubiquitous Computing Seminar FS2014, Institute for Pervasive Computing, Zurich, June 10th 2014.

[Lopez-Sanz, 2007] López-Sanz M., Acuña CJ., Cuesta CE., Marcos E. "UML Profile for the Platform Independent Modelling of Service-Oriented Architectures", LNCS Vol. 4758, pp. 304-307, Springer-Verlag Berlin Heidelberg, 2007.

[Lujan-Mora, 2006] Luján-Mora S., Trujillo J., Song IY. "A UML profile for multidimensional modelling in data warehouses", *Data & Knowledge Engineering journal*, Vol. 59, Issue 3, pp. 725–769, 2006.

[McCarty, 1993] McCarty J. «Notes on formalizing contexts». In Proc. of the 13 international joint conference on artificial intelligence, Bajcsy Ed. M.Kaufmann, pp. 555-560, California, 1993.

[Mcheick, 2014] Mcheick H., «Modeling Context Aware Features for Pervasive Computing». Procedia Computer Science, 37 (2014) pp. 135-142, 2014.

[Mignot, 2014] Mignot H., « Personnalisation : enjeux, stratégies et outils ». Séminaire Marketing Digital, 05 juin 2014.

[OCL, 2005] Object Constraint Language (OCL, OMG): OCL 2.0 Specification, Version 2.0, ptc/2005-06-06, June 2005, http://www.omg.org/cgi-bin/doc?ptc/2005-06-06.

[Odell, 2000] Odell J., Parunak H.V.D., Bauer B., "Extending UML for Agents," In Proceedings of AOIS Workshop at AAAI,
2000.

[OMG, 2003] OMG. «MDA Guide V1.0.1», June 2003.

[Ou, 2006] Ou S., Georgalas N., Azmoodeh M., Yang K., Sun X. «A Model Driven Integration Architecture for Ontology-Based Context Modelling and CAA Development» . A. Rensink and J. Warmer (Eds.): ECMDA-FA 2006, LNCS 4066, pp. 188 – 197, 2006.

[Pascoe, 1998a] Pascoe J. «Adding Generic Contextual Capabilities to Wearable Computers». 2nd International Symposium on Wearable Computers (1998), pp. 92-99, 1998.

[Pascoe, 1998b] Pascoe J., Ryan N.S., Morse D.R. «Human-Computer-Giraffe Interaction – HCI in the Field». Workshop on Human Computer Interaction with Mobile Devices. 1998.

[Perera, 2013] Perera C., Zaslavsky A., Christen P., Georgakopoulos D. "Context Aware Computing for The Internet of Things: A Survey". IEEE COMMUNICATIONS SURVEYS & TUTORIALS, Mai 2013.

[Pottinger, 2003] Pottinger R.A., Bernstein P.A. « Merging Models Based on Given Correspondences ». Proceedings of the 29th VLDB Conference, Berlin, Germany, 2003.

[Privat, 2007] Privat G., Ramparany F. « les interfaces contextuelles ». Publications France Télécoms R&D, 2007.

[Ranganathan, 2005] Ranganathan A., Al-Muhtadi J., Biehl J., Ziebart B., Campbell R., Bailey B. « Towards a pervasive computing benchmark, PerWare '05 Workshop on Support for Pervasive Computing ». Third IEEE International Conference on Pervasive Computing and Communications (PerCom 2005), pp. 194–198, 2005.

[Roques, 2003] Roques P., Vallée F. « UML en action : De l'analyse des besoins à la conception en Java ». Eyrolles, 2003.

[Ryan, 1997] Ryan N., Pascoe J., Morse D. «Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant». Gaffney,V., van Leusen, M., Exxon, S. (eds.) Computer Applications in Archaeology, 1997.

[Salber, 1998] Salber D., Dey A.K., Abowd G.D. «Ubiquitous Computing: Defining an HCI Research Agenda for an Emerging Interaction Paradigm». Georgia Tech GVU Technical Report GIT-GVU-98-01 (1998).

[Schilit, 1994a] Schilit B., Adams N., Want R. «Context-Aware Computing Applications». 1st International Workshop on Mobile Computing Systems and Applications. pp. 85-90, 1994.

[Schilit, 1994b] Schilit B., Theimer M. «Disseminating Active Map Information to Mobile Hosts». IEEE Network, 8(5) (1994) 22-32, 1994.

[Schilit, 1994c] Schilit B., Adams N., Want R. « Context-aware computing applications ». In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pp. 85-90, Santa Cruz, California, December 1994. IEEE Computer Society Press. 1994.

[Schmidt, 1998] Schmidt A., Beigl M., Gellersen H.W. «There is more to context than location». In Proceedings of Workshop on Interactive Applications of Mobile Computing (IMC'98), Rostock, Germany, November 1998. Neuer Hochschulschriftverlag. 1998.

[Seridi, 2012] Seridi H., Bouacha I., Benselim M.S. "Development of context-aware web services using the MDA approach", IJWS(International Journal of Web Science) , Vol. 1, No. 3, pp. 224–241, 2012.

[Sheng, 2005] Sheng Q.Z., Benatallah B., "ContextUML: A UMLBased Modeling Language for Model-Driven Development of Context-Aware Web Services," 4th International Conference on Mobile Business, IEEE Computer Society. Sydney, Australia, 2005.

[Simons, 2007] Simons C., "CMP: A UML Context Modeling Profile for Mobile Distributed Systems," In proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS 2007, Jan 3-6, 2007, Hawaii, USA).

[Sindico, 2009] Sindico A., Grassi V., "Model driven development of context aware software systems," International Workshop on Context-Oriented Programming (COP 2009, July 7, 2009, Genova, Italy), pp.1-5.

[Skillen, 2012] Skillen K.L, Chen L., Nugent C.D., Donnelly M.P., Burns W., Solheim I. "Ontological User Profile Modeling for Context-Aware Application Personalization", adfa, p. 1, (2012), Springer-Verlag Berlin Heidelberg, 2012.

[Sonawane, 2012] Sonawane s. « Context Aware Computing » , site internet fr.slideshare.net. 2012.

[StarUML, 2005] StarUML 5.0 developer guide, StarUML 5.0 user guide, http://staruml.sourceforge.net/docs/StarUML_5.0_Developer_Guide.pdf; http://staruml.sourceforge.net/docs/user-guide(en)/toc.html

[Strang, 2004] Strang T., Linnhoff-Popien C. «A context modelling survey» workshop on advanced context modeling, reasoning and management as part of ubicomp 2004, the 6th intl. conf. on ubiquitous computing, pp. 33-40, 2004.

[Tamine, 2005] Tamine L.L., Boughanem M., « Accès personnalisé à l'information : Approches et techniques ». Rapport interne, Institut de recherche en informatique de Toulouse, Janvier 2005.

[TEA, 1998] TEA project, Esprit project 26900: Technology for enabled awareness (TEA), 1998.

[Tiako, 2009] Tiako P.F., «Software Applications: Concepts, Methodologies, Tools, and Applications ». IGI Global, pp. 11-21, Mars 2009.

[Touzi, 2009] Touzi A., BenMessaoud M., "New Approach for Conception and Implementation of Object Oriented Expert System Using UML", *The International Arab Journal of Information Technology*, Vol. 6, No. 1, pp. 99-106, 2009.

[UML, 2004] Unified Modelling Language (UML, OMG): UML Infrastructure version 2.0, ptc/04-10-14, November 2004, http://www.omg.org/cgi-bin/doc?ptc/04-10-14

[UML, 2005] Unified Modelling Language (UML, OMG): UML Superstructure version 2.0, formal/05-07-04, August 2005, http://www.omg.org/cgi-bin/doc?formal/05-07-04

[Vale, 2008] Vale S., Hammoudi S. «Towards Context Independence in Distributed context-aware applications by the Model Driven Approach». ACM SIPE'08, Sorrento, Italy, July 7, 2008.

[Vale, 2009] Vale S., Hammoudi S., « An Architecture for the Development of Context-aware Services based on MDA and Ontologies » Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol. 1 IMECS 2009, Hong Kong, 2009

[Valée, 2005] Valée M., Ramparany F., Vercouter L., « Composition et adaptation contextuelle de services pour la communication ambiante : un état de l'art ». Ecole Nationale Supérieure des Mines de Saint-Etienne, 2005.

[Van den Bergh, 2005] Van den Bergh J., Coninx K., "Using UML 2.0 and Profiles for Modelling Context Sensitive User Interfaces", *In Proceedings of the International Workshop on Model Driven Development of Advanced User Interfaces (MDDAUI 2005)*, Vol. 159, Jamaica, 2005.

[Ward, 1997] Ward A., Jones A., Hopper A. «A New Location Technique for the Active Office». IEEE Personal Communications 4(5) (1997), pp. 42-47, 1997.

[Weiser, 1991] Weiser M. « The computer of the 21st century ». Scientific American, 265(3), pp.66–75, September 1991.

[Wagner, 2002] Wagner G., "A UML Profile for Agent-Oriented Modelling", *In Proceedings of the 3rd International Workshop on Agent-Oriented Software Engineering*, Bologna, Italy, 2002.

[Werner, 2006] Werner C., Kraatz S., Hogrefe D. "A UML Profile for Communicating Systems", *in Proceedings of the Fifth Workshop on System Analysis and Modelling (SAM 06)*, pp. 81-90, Kaiserslautern, Germany, 2006.

[Ziadi, 2004] Ziadi T., Jézéquel J.M., "Towards a UML profile for software product lines", LNCS Vol. 2014, pp. 129-139, Springer-Verlag Berlin Heidelberg, 2004.